Proceedings of the 2023 International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/41/20230736

Spam classification based on parallel optimized BERT

Shaobo Li[1], Yanyang Li[2], Haoqin Xu[3,4]

[1]Computer Science, Sun-Yat-Sen University, Guangzhou, China
[2]Department of Digital Media Technology, Dalian University of Technology, Liaoning, China
[3]Software Institute, Nanjing University, Nanjing, China

[4]201250067@smail.nju.edu.cn

Abstract. With the popularity of email and the increase in spam, effectively filtering spam has become an urgent need. This research presents a cutting-edge approach to spam filtering, leveraging the power of BERT and innovative parallel optimization techniques, improving email security. This study proposes a spam classification method based on the BERT (Bidirectional Encoder Representations from Transformers) model, aiming to improve the accuracy and efficiency of spam filtering. The study first investigated the shortcomings of traditional classification methods and then analysed the applicability of the BERT model to classifying spam. The study also performed two parallel optimizations on BERT, DDP (Distributed Data Parallel), and Gpipe. Among them, DDP was used to accelerate the large model BERT parallelly. After conducting experiments on spam classification, it was found that the average training time of an epoch was 54 seconds, with the accuracy on the test set reaching 98%, achieving significant performance improvements.

Keywords: Large Model, Parallelism, Spam Classification.

1. Introduction

Today, although there are many other forms of communication, such as instant messaging, social media, etc., email is still one of the most used formal communication tools [1]. People use email extensively in their work, study, and personal lives in the digital age. However, with the increase in the number of emails, effective management of emails has become increasingly important, and spam is a critical issue that needs to be addressed.

As early as the beginning of the 21st century, spam has become an emerging problem. In 2008 statistics, the proportion of spam in China and India reached 20-30% [2]. In 2012, the Pew Research Center stated that more than 69% of cellular device users have been subjected to SMS spam, including promotional texts and unsolicited marketing messages [3]. Over half of all emails sent to users' inboxes are considered spam. Spam emails generally mean unsolicited, commercial, bulk electronic messages [4]. These spam emails waste users' time and energy and may cause security risks. Some companies or platforms send unwanted content, such as silent ads, viruses, and malware, to consumers, businesses, or government organizations [5]. The existence of spam not only causes trouble to individual users but also affects the email communication of enterprises and organizations, reducing efficiency. To solve this problem, developing more advanced spam classification technology is crucial.

DOI: 10.54254/2755-2721/41/20230736

After researching the currently popular methods, this paper explores their potential shortcomings [6]. For example, keyword filtering technology allows spammers to intentionally modify spelling, use synonyms, or add noise to keywords to avoid keyword filtering [7]. Rule-based systems may be too strict or lenient, resulting in a high false positive or false negative rate for spam. Spammers can continuously improve their deception techniques for natural language processing technology, making it difficult for NLP models to capture new features [8]. If machine learning algorithms are used, a large amount of labelled data is required for training [9]. If the dataset needs to be more comprehensive and balanced, it may lead to a decrease in classifier performance, and there may also be overfitting issues.

This study is based on the BERT model and delves into the spam classification problem, integrating two parallel methods, Distributed Data Parallel and Gpipe, for further optimization [10]. The research results show that the BERT model achieved a remarkable 98% classification accuracy in spam classification tasks, demonstrating its outstanding performance in this field. This highlights the potential of BERT in natural language processing tasks and demonstrates the critical role of parallelization technology in accelerating model training. This study strongly supports developing spam recognition technology and provides a valuable reference for solutions to similar problems.

2. Methods

This section focuses on the construction process of large models, emphasizing parallel optimization.

2.1. BERT

BERT (Bidirectional et al. from Transformers) is a revolutionary natural language processing (NLP) model proposed by Google in 2018 [11]. Its emergence has sparked a revolution in the field of NLP, significantly improving the performance of various text processing tasks, from text classification and named entity recognition to machine translation and question-answering systems. One of BERT's core innovations is bidirectional context modelling, which breaks the limitations of previous unidirectional language models and brings new possibilities for models to understand and process contextual relationships.

BERT's architecture is based on Transformer, a neural network structure based on a self-attention mechanism first proposed by Vaswani et al. [12]. Transformer's self-attention mechanism allows the model to simultaneously pay attention to all positions in the input sequence when processing the input sequence, thereby capturing global context information. BERT's architecture consists of multiple encoder layers, each consisting of a multi-head self-attention mechanism and a feed-forward neural network.

The workflow of BERT is divided into two main stages: pre-training and fine-tuning.

Pre-training phase: In the pre-training phase, the BERT model is first trained on a large-scale unlabelled text corpus to learn rich language representations. BERT introduces two main pre-training tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP). Masked Language Model (MLM): In this task, some words in the input text will be randomly masked, and then the model needs to predict these masked words. In this way, the model is forced to infer the meaning of the masked words from the context, thus learning to understand the contextual information better. Next Sentence Prediction (NSP): In this task, the model must judge whether two sentences are adjacent. This task helps the model understand the logical and semantic relationship between sentences, better capturing the text's coherence.

Fine-tuning stage: BERT can be fine-tuned on specific downstream tasks after pre-training. Fine-tuning refers to adjusting the parameters of a pre-trained model on a specific task using a small amount of labelled data. For example, the BERT model can be fine-tuned on tasks such as text classification, named entity recognition, and sentence relationship determination to adapt to the requirements of specific tasks.

The advent of BERT triggered a massive change in the field of NLP. Before it, the model often required specific feature engineering and model design for each task, consuming a lot of workforce

DOI: 10.54254/2755-2721/41/20230736

and time. With BERT, excellent performance on new tasks can be achieved with fine-tuning. This approach dramatically lowers the threshold for model development, enabling more researchers and practitioners to devote themselves to innovative tasks without paying too much attention to the design of the underlying model.

2.2. Distributed Data Parallel

When training large deep learning models, multiple GPUs are often utilized to speed up the training process. In distributed deep learning, each GPU can process a different subset of data and then synchronize its computed gradients to update model parameters. However, such distributed training involves complex communication and synchronization operations. To simplify this process, PyTorch introduces Distributed Data Parallel (from now on referred to as DDP), which makes it more efficient and easier to train deep learning models in a multi-GPU and multi-node distributed environment. It copies the model to multiple GPUs for training in a data-parallel manner while handling details such as gradient synchronization and data distribution. DDP can significantly increase the training speed and accelerate the model training process to achieve faster training of high-performance deep learning models on large-scale data sets. Overall, DDP makes distributed training on multiple GPUs more efficient and convenient. It enables developers to focus on model design and training strategies by automatically handling complex details such as data distribution, gradient synchronization, and parameter updates. There is no need to overthink about the details of distributed communication. This helps to accurate model training and improve training efficiency to achieve faster training of high-performance deep learning models on large-scale data. This section will detail how to use DDP to accelerate the training process of large models.

Initialize the distributed environment. Before using DDP, user need to initialize the distributed training environment. This involves selecting the communication backend, specifying the distributed environment address, and configuring the communication method between processes.

Model copying and encapsulation. In distributed training, a complete copy of the model is required on each GPU. Distributed Data Parallel automatically copies the model on each GPU and creates a separate running copy for each GPU.

Data loader settings. The data loader needs to be set appropriately according to the number of GPUs to ensure that each GPU gets the appropriate training number. Data distribution is a critical challenge in data-parallel training Distributed Sampler can be used to perform Distributed sampling.

Training loop. Each GPU independently performs forward propagation, backpropagation, and gradient calculations in each training iteration. During this time, gradients are calculated, and each GPU has gradient information. Once each GPU has completed backpropagation, Distributed Data Parallel will collect the gradients of each GPU and perform gradient synchronization. This means that each GPU updates its model parameters to match the gradient information of all other GPUs. After gradient synchronization, each GPU will update its model parameters with the updated gradients. This ensures that model parameters remain consistent across all GPUs.

Overall, DDP makes distributed training on multiple GPUs easier. It enables developers to automatically handle complex details such as data distribution, gradient synchronization, and parameter updates to increase efficiency and convenience. It is enough to focus on model design and training strategies without overthinking the details of distributed communication. This helps speed up model training. It improves training efficiency and enables faster training of high-performance deep learning models on large-scale data.

2.3. Gpipe

Gpipe introduces a pipelined parallelism approach to training large neural networks. The main idea is to split the training data into smaller micro-batches and process them pipelined. During training, the micro-batches are fed into the first accelerator, which performs the forward pass and computes the activations. Gpipe only stores the output activations at each partition during the forward pass. When

DOI: 10.54254/2755-2721/41/20230736

the gradients need to be computed in the backward pass, Gpipe recomputes the forward function. This allows efficient memory utilization and training of larger models.

The embedding layer of the BERT model was reimplemented using a custom approach in the conducted experiments. Regarding the specific binary classification task, the prediction component of the BERT model was redesigned, involving the reconstruction of the linear layer and the adaptation of the loss function.

Table 1. Training time for different partitioning methods.

Epoch	Balance by time	Balance by size
Epoch1	170.63s	158.08s
Epoch2	171.61s	160.19s
Epoch3	171.98s	158.87s
Epoch4	173.69s	158.07s
Epoch5	170.08s	166.56s
Epoch6	172.63s	160.74s
Epoch7	170.10s	158.43s
Epoch8	170.06s	158.29s
Epoch9	170.43s	159.82s
Epoch10	165.23s	158.11s

There are two methods: “balance_by_time” and “balance_by_size.” The difference lies in the criterion used for partitioning, where one considers the time taken by the layers while the other looks at the memory usage. To select the optimal partitioning strategy, it is necessary to conduct experiments and compare the effectiveness of the two partitioning methods, as shown in Table 1. After comparing the two methods, it was observed that “balance_by_size” resulted in slightly faster training time within one epoch. However, regarding training effectiveness, “balance_by_size” showed slow loss reduction. After training for ten epochs, the “balance_by_size” approach achieved an accuracy of 0.24 and a loss of 31.74 on the test set, whereas the “balance_by_time” approach achieved an accuracy of 0.28 and a loss of 28.85. Considering all factors, the “balance_by_time” method was ultimately chosen for partitioning.

Reimplementation of the Tokenizer. This study modified the original BERT model by adjusting the extraction of three dimensions: input_ids, token_type_ids, and attention_mask. The first two dimensions are sent to the embedding layer. Moreover, padding_idx is introduced to mark the sequence.

Composition of Transformer Layers. The transformer modules can be stacked using the torch library, following the approach in the BERT source code. In this case, a total of 12 layers are defined, with each layer consisting of transformers. The number of attention heads for each transformer is 12, and the embedding size is 768.

Construction of the Prediction Layer. Multiple linear layers and activation functions are combined to form a single unit, and its forward method is rewritten. This unit simulates extracting information from the last dimension of the hidden states in the BERT classification task and performs classification.

Implementation of Dataset Loading. The data loading process is modified to convert the original labelled “spam” and “ham” tags into 1 and 0, respectively, corresponding to the processing by the tokenizer. After rewriting the forward method for the layers, they can all be included in an nn.Sequential container.

DOI: 10.54254/2755-2721/41/20230736

3. Results and analysis

3.1. Datasets description

The 2007 TREC Public Spam Corpus dataset was chosen for this study. This dataset is widely used in information retrieval and designed for spam email classification and filtering tasks. It consists of an extensive collection of spam emails and non-spam emails (also known as "ham" or "legitimate" emails). The dataset aims to assist researchers and practitioners in developing and evaluating spam filtering algorithms to improve email systems' efficiency and user experience. The TREC07 dataset comprises approximately 75,419 emails, with 50,199 classified as spam and 25,220 as non-spam. The emails cover various topics and domains, including advertisements, scams, promotions, news, and more. Each email in the dataset is collected and filtered from natural email streams and stored in its original format, including email subject, sender, recipient, body content, attachments, and other relevant information. Each email has been manually labelled as either spam or non-spam during the dataset construction process. These labels serve as ground truth for training and evaluating the accuracy of models.

3.2. Experimental environment and tools

This study uses a large model optimized in parallel to classify spam. To ensure the efficiency and reliability of the algorithm, an excellent experimental environment is needed to support the implementation and operation of the algorithm. This paper chose a high-performance computer as the experimental platform, equipped with two NVIDIA GeForce RTX 3090 graphics cards with reasonable computing power and computing resources. Secondly, the experimental environment is configured with the deep learning framework PyTorch. This study configures an efficient, stable, and safe experimental environment, guaranteeing the experiment's smooth progress.

3.3. Results

This experiment employs parallel-optimized large spam email classification models and compares the accuracy and speed differences among different models.

Table 2. Results of different machine learning methods.

Method	Accuracy
SVC	0.9838
MultinomialNB	0.9883
LogisticRegression	0.9802
KNeighborsClassifier	0.9121
DecisionTreeClassifier	0.9650
RandomForestClassifier	0.9686
AdaBoostClassifier	0.9677
BaggingClassifier	0.9721
ExtraTreesClassifier	0.9802
LSTM	0.9737
GRU	0.9532

The experiment used Keras to build the LSTM and GRU models. The models will be trained for 100 epochs, and the RNN layers will use the sigmoid activation function. Machine learning classification methods utilized the built-in models from the scikit-learn library. Data pre-processing used the Tokenizer from the Keras library. The dataset used for testing RNN, and machine learning methods is Trec07, the same as used for testing BERT. The classification accuracy results are shown in Table 2.

DOI: 10.54254/2755-2721/41/20230736

Table 3. Training time for DDP.

Epoch1	00:55
Epoch2	00:54
Epoch3	00:54
Epoch4	00:54
Epoch5	00:54
Epoch6	00:54
Epoch7	00:54
Epoch8	00:54
Epoch9	00:54

In the context of this research study, this paper employs Distributed Data Parallelism (DDP) as a pivotal technique to parallelize and enhance the training speed of the formidable BERT model, renowned for its substantial size and complexity. The primary objective is to investigate the tangible effects and advantages of employing DDP during the training process. To comprehensively evaluate the efficacy of DDP, this paper has designed a series of experiments, with a particular focus on spam classification as the use case.

To delve into the specifics, this paper has meticulously set up these experiments to harness the power of distributed data parallelism by training the BERT model across two graphics processing units (GPUs). This approach stands in stark contrast to the traditional single-card training methodology. By conducting a comparative analysis between the results obtained from distributed data parallelism and single-card training, this paper aims to shed light on the substantial performance gains and improvements that can be achieved by utilizing DDP.

Based on the provided timetable, there are notable differences in training efficiency and accuracy when comparing Distributed Data Parallelism (DDP) and single-card training for the BERT model in the context of spam classification.

When employing DDP, the training time per epoch is impressively reduced to an average of 54 seconds (Table 3). Concurrently, the model achieves a remarkable accuracy of approximately 98% on the test set. In contrast, each epoch takes significantly longer during single-card training, clocking in at 104 seconds, almost twice the time required for dual-card distributed data-parallel training. This observation highlights the substantial advantage of DDP in terms of speeding up the BERT model's training process, making it a highly efficient choice for the spam classification task.

Moreover, the graphics card occupancy rates provide additional insights into resource utilization. In the case of single-card training, the graphics card operates at a 9% occupancy rate, indicating that there is still considerable untapped potential in GPU utilization. However, when DDP is employed for dual-card data parallelism, the graphics cards exhibit occupancy rates of 7% and 3%, respectively. This suggests that DDP optimizes the GPU resources more effectively, ensuring a more balanced and efficient distribution of computation across the two GPUs.

Table 4. Training time for Gpipe.

Epoch1	15:33
Epoch2	15:28
Epoch3	15:28
Epoch4	15:27
Epoch5	15:27
Epoch6	15:22

On average, Gpipe takes around 15 minutes to train one epoch after training for 6 epochs. It achieves an accuracy of approximately 70% on the test set (Table 4).

DOI: 10.54254/2755-2721/41/20230736

4. Conclusion

This study explores the effectiveness and performance of using the BERT model in conjunction with parallel methods to accelerate spam email classification. After fine-tuning the pre-trained BERT model, more ideal results can be obtained compared to traditional machine learning methods. Compared to traditional rule-based or feature engineering methods, the BERT model can better capture text semantics and contextual information, improving classification accuracy and generalization ability. After conducting experiments with two parallel methods, Distributed Data Parallel (DDP) and Gpipe, it was determined that DDP is the optimal parallel approach for this task. In Gpipe, the computation in each stage requires frequent data exchange, which can increase the computation time of each stage and reduce the overall training speed. In contrast, DDP performs computations simultaneously on multiple devices, improving training speed. Simultaneous computation on multiple devices also allows for better utilization of device computational resources, accelerating the model training process. For popular machine learning methods in spam email classification, using large-scale AI models in this study enables a better understanding of email content. It could lead to more accurate email classification in the future. Additionally, within the parallel optimization methods, further experiments can be conducted to investigate layer partitioning and batch size in Gpipe, aiming to reduce the communication cost of the pipeline.

Authors Contribution

All the authors contributed equally and their names were listed in alphabetical order.

References

[1] Mansoor R, Jayasinghe N D, Muslam M M A. A comprehensive review on email spam classification using machine learning algorithms[C]//2021 International Conference on Information Networking (ICOIN). IEEE, 2021: 327-332.

[2] Sarah Jane Delany, Mark Buckley, Derek Greene,SMS spam filtering: Methods and data,Expert Systems with Applications,Volume 39, Issue 10,2012,Pages 9899-9908,ISSN 0957-4174,

[3] Boyles, J.L., Rainie, L., 2012. Mobile Phone Problems. In: Pew Research Center's Internet & American Life, Pew Research Center, pp. 2–3.

[4] Graham, P., 2002. A plan for spam. Retrieved on the 16th June 2017. Available online at http://www.paulgraham.com/spam.html.

[5] Abayomi-Alli O, Misra S, Abayomi-Alli A, et al. A review of soft techniques for SMS spam classification: Methods, approaches and applications[J]. Engineering Applications of Artificial Intelligence, 2019, 86: 197-212.

[6] Cormack G V. Email spam filtering: A systematic review[J]. Foundations and Trends® in Information Retrieval, 2008, 1(4): 335-455.

[7] Kim S E, Jo J T, Choi S H. SMS spam filterinig using keyword frequency ratio[J]. International Journal of Security and Its Applications, 2015, 9(1): 329-336.

[8] Garg P, Girdhar N. A systematic review on spam filtering techniques based on natural language processing framework[C]//2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence). IEEE, 2021: 30-35.

[9] Crawford M, Khoshgoftaar T M, Prusa J D, et al. Survey of review spam detection using machine learning techniques[J]. Journal of Big Data, 2015, 2(1): 1-24.

[10] Huang Y , Cheng Y , Bapna A ,et al.GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism[J].arXiv, 2018.DOI:10.48550/arXiv.1811.06965.

[11] Devlin J , Chang M W , Lee K ,et al.BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[J]. 2018.

[12] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.