BERT with LORA: Low-rank Adaptation Of Large Language Models

Stanford CS224N Default Project

Tolu Oyeniyi
Department of Computer Science
Stanford University
tolu316@stanford.edu

Abstract

Pre-trained models like the Bidirectional Encoder Representations from Transformers (BERT) model are used because of their power to generate better sentence embeddings for natural language processing (NLP) tasks, and are easy to fine-tune (Devlin et al. (2019)). In this project, we propose leveraging a pre-trained BERT encoder in the implementation of our multi-task model for three specialized downstream tasks - sentiment analysis, paraphrase detection, and semantic textual similarity. Further, we aim to improve upon our baseline by fine-tuning a pre-trained BERT encoder model with Low-Rank Adaptation (LoRA), which freezes all of the BERT weights during training except the weights of the LoRA linear layers initialized (Shen et al. (2022)). Specifically, we will first implement additional trainable linear layers for only the key, query, and value in our BERT model, then we will implement LoRA layers for all linear layers in our BERT model to observe any improvements. Lastly, we will attempt to further improve the model's performance by configuring the hyperparameters of this pre-trained BERT model that is fine-tuned with LORA; hyperparameters such as the learning rate and the rank (of the LoRA layers). As a result of these methods, we achieved an improvement in accuracy and correlation of our dev set, which overall increased by 0.153 from our baseline's overall dev score, across all three tasks. We found that increasing the rank while adding more LoRA linear layers resulted in our best model.

1 Key Information to include

- Mentor: Soumya Chatterjee
- External Collaborators (if you have any): N/A
- Sharing project: N/A

2 Introduction

Communication is at the forefront of human connection and striving to understand the nuances of communication is the difference between wars starting or alliances being made. Our ability to communicate through natural language can be expressed through numerous tasks. With the recent arrival of transformers and large language models, our ability to understand various Natural Language Processing (NLP) tasks has improved as the models have set a new standard for interpreting human language.

However, because models are pre-trained on a large scale on general domain data, as they get larger, fine-tuning, which retrains all model parameters, becomes harder to complete and computationally more expensive. This project addresses this NLP challenge of fine-tuning, specifically for three downstream tasks - sentiment analysis, paraphrase detection, and semantic textual similarity.

Our baseline in this project is a multi-task pre-trained BERT model, and our initial approach to improving upon this baseline is fine-tuning with LoRA, which freezes the pre-trained model weights and initializes new, smaller, trainable linear layers that will get added to the frozen pre-trained weights. This method succeeds because it significantly decreases the number of parameters being trained for the downstream tasks while getting similar results to fine-tuning the full pre-trained weights. We further improved this model by adding more, smaller, trainable layers beyond only the key, query, value and by increasing the rank by a little. Next, we experiment with different hyperparameter configurations to see which configuration optimizes our model. The hyperparameters we explore are the rank - the input and output size between the two additional, smaller, linear layers, and the learning rate of our model. As a result, by fine-tuning our pre-trained BERT model with LoRA, it significantly reduces the memory usage and runtime, making multi-task fine-tuning more feasible.

3 Related Work

Transformers are deep learning models that have revolutionized Natural Language Processing (NLP). Because of their many benefits, such as their ability to be parallelized, their architecture has been adopted and become the backbone of many Large Language Models (LLMs) and has achieved immense success in various foundational NLP tasks, such as generating text, question-and-answer, and machine translation. Early approaches to language modeling employed sequence-based models like Recurrent Neural Networks (RNNs) (Sutskever et al. (2011)) and Long Short-Term Memory (LSTMs). However, with the introduction of transformers, its attention mechanism has allowed models to focus on relevant parts of the input, thus enabling them to capture long-range dependencies more effectively (Vaswani et al. (2017)).

LLMs are trained on extensive data and typically billions or trillions of parameters. Some of the most influential models include BERT, introduced by Google, which uses masked language modeling (MLM) and next sentence prediction (NSP) while capturing both directions of an input word or sentence, enabling it to understand the context of the sentence better (Devlin et al. (2019)). BERT achieved state-of-the-art performance and set new benchmarks in the NLP space. Another LLM is GPT by OpenAI; ever since GPT-1, many more models have been released with increasingly more parameters it has been trained on, with GPT-4o ("o" for "omni") being the most advanced model. To elaborate, GPT-4o is a multimodal LLM with a context window of 128,000 tokens that generates text two times faster than GPT-4, it is 50% cheaper, and it performs better across non-English languages, according to openai's website.

Traditionally, in the NLP space separate models have been trained for different tasks, and sometimes transfer learning was used - applying the knowledge/weights of a model which was trained for a specific task to another task. Unfortunately, blind transfer learning for this structure are poor (Devlin et al. (2019)), and usually result in overfitting. But with the advent of Multi-task learning (MTL), we can effectively train a single model for multiple downstream tasks, on task-specific training data. (Bi et al. (2022)) propose an approach to calculate the loss for multi-task learning by summing up the losses of each task. In our project, we use this approach when pre-training our model on all downstream tasks.

Further, because language models can get large, several techniques exist to reduce the memory usage and running time required to finetune them. One technique we employ is LoRA - Low Rank Adaptation. LoRA freezes the pretrained model weights and introduces trainable rank decomposition matrices in the transformer architecture which significantly reduces the number of trainable parameters while preserving performance on downstream tasks like Semantic Textual Similarity (STS)(Shen et al. (2022)). In the paper, LoRA creates additional trainable linear layers for the key, query, and value; in our project we will explore the performance of our model when we add trainable linear layers to other parameters, as well as the performance by changing the rank of those layers.

4 Approach

4.1 MinBERT Model Architecture

Referencing the project handout and the original BERT paper (Devlin et al. (2019)), the starting point of our project was to first implement minBERT.

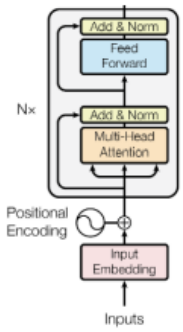

Figure 1: Encoder Layer of Transformer used in BERT. Left of Figure 1 from [Devlin et al. (2019)]

The first step of minBERT is the tokenization of the input sentence, where sentences get converted to tokens - in this case, where sentences get split into one of BERT's 30K different word pieces - using a WordPiece tokenizer; any unseen word pieces will be set as the [UNK] token and the sentences will be padded to a max_length of 512 with the [PAD] token. The word pieces are then get converted into ids which will be used in the rest of the model. Additionally, separate input sentences are represented with the [SEP] token, and the whole input sentence is represented by the [CLS] token. The embedding layer consists of token embeddings which map input ids to vector representations, and positional embeddings which encode the position of different words within the sentence; both embeddings have a dimensionality of 768.

Similar to the base BERT in the original BERT paper (Devlin et al. (2019)) which made use of the layers defined initially in the paper Attention is All You Need (Vaswani et al. (2017)), our model uses 12 Encoder Transformer layers, each layer utilizing Multi-head attention which is the dot-product attention mechanisms for the n heads. The attention mechanism is computed by taking the weighted sum of the value and the softmax of the dot products of the keys and queries for a subspace of size d/n.

We then perform normalization, a feed-forward layer, and another normalization layer, all while applying dropout to each output of the sub-layers, as well as the sums of the embeddings and the positional encodings. The final output of BERT is contexualized embeddings for each word piece of the sentence from the last BertLayer, and the [CLS] token embedding.

4.2 Baseline

The baseline for our project is the last linear layer of a pre-trained BERT model trained on the SST, Quora, and SemEval datasets. Further, we implemented the Adam Optimizer which only requires first-order gradients for stochastic optimization. It is a method that computes adaptive learning rates for different parameters by updating exponential moving averages of the gradient, the squared gradient, controlling the hyperparameters β_1, β_2 [0, 1) which control the rate of exponential decay of the averages, and performing bias correction at each step, according to the project handout.

4.3 Multi-task Learning

Our first approach to implementing a multi-task model was to train on all the tasks at the same time and not separately. This implementation consisted of iterating over the same batch size of data for all three datasets at the same time while making sure the process continues after one dataset ends as all three datasets are different in size. Further, we modified the loss function; instead of having three different loss functions being optimized separately, we combined all three loss functions - cross-entropy loss for sentiment analysis, binary cross-entropy loss for paraphrase detection, and mean squared error loss for semantic textual similarity - and we optimized the new loss now which represents all tasks.

4.4 LoRA Technique

Our main extension was fine-tuning our pre-trained BERT model using LoRA. We implemented this technique according to the LoRA paper (Shen et al. (2022)), by freezing all the BERT pre-trained

weights, training on two added linear layers A and B combined with the frozen pre-trained weights, and initialized layer B with zeroes. To elaborate, the weight matrices of the BERT layers typically have full-rank, and inspired by Aghajanyan et al.(2020), who showed that the pre-trained language models have a low "instrisic dimension" and can still learn efficiently despite a random projection to a smaller subspace, the researchers of LoRA hypothesized that the updates to the weights will also have a low "intrinsic rank" during adaptation. Specifically, constraining the update of a pre-trained weight matrix $W_0 \in R^{dxk}$ by representing it as a low-rank decomposition $W_0 + W = W_0 + BA$ (1), where $B \in R^{dxk}$, $A \in R^{rxk}$, and the rank $r \ll min(d, k)$, and where W_0 and $W = BA$ are multiplied with the same input.

$$h = W_0 x + \Delta W x = W_0 x + BAx \quad (1)$$

4.5 Hyperparameter Optimization

Our final extension consisted of configuring the hyperparameters of our pre-trained BERT model that was fine-tuned with LoRA to see if it improves the performance of any of our three downstream tasks. The hyperparameters we were interested in analyzing was the learning rate and the rank.

5 Experiments

5.1 Data

As described in the project handout, we will be using the SST dataset for sentiment analysis, the Quora dataset for paraphrase detection, and the SemEval dataset for semantic textual similarity.[table 1]

Task (label)	Dataset	Train (#)	Dev (#)	Test (#)
sentiment analysis (0-5)	SST	8,544	1,101	2,210
paraphrase detection (0/1)	Quora	283,010	40,429	80,859
semantic textual similarity (0/1)	SemEval	6,040	863	1,725

Table 1: Datasets used for our multitask pre-trained BERT model

5.2 Evaluation method

For sentiment analysis (SST) and paraphrase detection tasks, we use accuracy, and for the semantic textual similarity (STS) task, we use Pearson correlation.

5.3 Experimental details

Unless otherwise stated, we fine-tune our pre-trained BERT using the following default parameters.

default hyperparameters (on GPU): seed = 11711, epochs = 10, learning rate = $1e-5$, dropout = 0.3, fine-tune-mode = full-model, batch_size = 8, rank = 4.

- **Model 1** pre-trained BERT (without LoRA; baseline): fine-tune-mode = last-linear-layer
- **Model 2** pre-trained BERT (with LoRA - key&query&value, default)
- **Model 3** pre-trained BERT (with LoRA - key&query&value): rank = 32, learning rate = $1e-9$
- **Model 4** pre-trained BERT (with LoRA - key&query&value): rank = 48

We then experiment with adding lora layers (A&B) to one more pre-trained weight - pooler_dense in which its rank = 4 for model 5-7.

- **Model 5** pre-trained BERT (with LoRA - pooler_dense): fine-tune-mode = last-linear-layer
- **Model 6** pre-trained BERT (with LoRA - pooler_dense&key&query&value, default)

- **Model 7** pre-trained BERT (with LoRA - pooler_dense&key&query&value): rank = 32, learning rate = $1e-9$
- **Model 8** pre-trained BERT (with LoRA - pooler_dense&key&query&value): rank = 48

We then experiment with adding lora layers (A&B) to all pre-trained weights except out_dense which is a linear layer used in the normalization function.

- **Model 9** pre-trained BERT (with LoRA - all-1, default)
- **Model 10** pre-trained BERT (with LoRA - all-1): rank = 48
- **Model 11** pre-trained BERT (with LoRA - all-1): rank = 100

5.4 Results

The following tables contain the results we achieved on the test leader board, and the results we achieved on the dev leader board:

Model	SST accuracy	Para accuracy	STS correlation	Overall Dev Score
Model 9	0.236	0.632	0.002	0.456

Table 2: ev set accuracy for sentiment analysis (SST) and paraphrase detection (Para), Pearson correlation for semantic textual similarity (STS), and the overall Dev score from the test leader board for three models (referenced under section 5.3).

Model	SST accuracy	Para accuracy	STS correlation	Overall Dev Score
Model 1 (baseline)	0.144	0.369	-0.009	0.336
Model 2	0.272	0.632	0.013	0.470
Model 3	0.176	0.631	-0.047	0.428
Model 4	0.256	0.632	0.157	0.489
Model 5	0.256	0.380	0.030	0.384
Model 6	0.279	0.632	-0.037	0.464
Model 7	0.261	0.369	0.084	0.391
Model 8	0.241	0.632	0.133	0.480
Model 9	0.256	0.632	0.157	0.489
Model 10	0.257	0.632	0.021	0.466

Table 3: Dev set accuracy for sentiment analysis (SST) and paraphrase detection (Para), Pearson correlation for semantic textual similarity (STS), and the overall Dev score from gradescope for each model (referenced under section 5.3).

The results are in line with what we expected to see - that adding more linear layers and increasing the rank would yield higher performance. This tells us that fine-tuning our pre-trained BERT with LoRA does improve our baseline model and that configuring the rank of LORA to a higher number than the default (which is still significantly smaller than the original pre-trained weight matrix dimensions) can further increase the performance of our model. The best model overall is model 9, since it achieves an overall dev set score of 0.489.

6 Analysis

The implementations/modifications we noticed which improved the performance our model include increasing the rank, and training on the full model and not the last linear layer. To elaborate, a reason for why increasing the ranke improved our model is that adding more parameters that need to be trained allows our model to learn more, thus perform better. Next is training on the full model, training on a full-model ensures that we have weights that are being updated through the training process, if we only use the last-linear layer we are only limiting our model to learn the tasks based on weights that were trained on datasets whose limitations include possibly being outdated a little as times change, people's way of communication changes as well, so the frozen pre-trained weights, though rare, might not capture the new nuances within the English language when considering the three tasks we are training for. The implementations/modifications we noticed which decreased the

performance of our model include decreasing the learning rate. The reason why this was a problem is because our model was liekly taking too small of a step to learn quickly, thus, becuase of the limited time/epochs, our model is slower in learning.

When comparing our models, we noticed that all the models that were fine-tuned with LoRA, performed better than our baseline. Further, the models that used the default parameters - model 2,6, and 9 - performed better on sentiment analysis, while the models that utilized adding more linear layers and raising the rank such as model 6, 8, and 9 performed the best on paraphrase detection and semantic textual similarity. One thing to note that was interesting was that the highest accuracy that any model got on paraphrase detection was 0.632.

7 Conclusion

In this project, we implemented a pre-trained BERT model and leveraged an influential fine-tuning technique known as LoRA to fine-tune it. We investigated the performance of our baseline model on three specialized downstream tasks when (1) fine-tuning our model with LoRA, (2) changing the number of additional linear layers we add to the pre-trained weights, and (3) configuring hyperparameters such as the rank and learning rate.

We learned that training on a full model yielded higher accuracy and correlations than training on the last-linear layer which freezes all weights; that increasing the rank significantly increases performance as there are more parameters being trained which overall is still small compared to the pre-trained weight matrix dimensions; and that decreasing the learning rate fundamentally hurts our model because there is not enough time to learn with too small of a step. Further, we found that our best model was model 9 which achieved a 0.153 increase from our baseline's overall dev score. Future works would utilize better pre-trained models such as a Robustly optimized BERT approach (RoBERTa) which was introduced by Facebook AI researchers and is known to perform better than BERT because it is pretrained on a larger and more diverse dataset. As well as, training on additional datasets to get richer and more robust embeddings. for each downstream task.

8 Ethics Statement

One societal risk specific to our project is our model's ability to create bias. If our model is trained on a biased dataset in the beginning, then fine-tuning it will only make the result for the task more biased, and could be reinforcing stereotypes or encouraging discrimination. For example, if an employer is using a model for sentiment analysis to classify the responses of job applicants and the model was trained on a dataset that identified the German accent or the use of German vernacular as negative then potential German applicants could be weeded out of the hiring pool because the model deems Germans as sounding negative based on their way of speaking. One mitigation to this would be to train the model using non-biased data, or training the model on English expressions/sentences that Germans (or other groups) typically use when speaking that are not culturally seen as negative from their standpoint, so as to not label a whole demographic of people as sounding negative.

Another societal risk is that our model could end up ostracizing (leaving-out) a group in society. For example, for paraphrase detection, if a specific culture uses a different expression that holds the same meaning in one state compared to another, and model is trained on data collected from only one of the states, then the model could make individuals from the other state feel like they are being misunderstood when interacting with the model or make them feel like they are not speaking "correct English", especially if the model is being used as a standard application in different aspects of society. One way to mitigate this is to use data that captures the slang or expressions of different generations or cultures when training for tasks such as paraphrase detection; every culture or group has a way of saying the same thing and we want our model to capture this better. This mitigation will make our model more holistic and more mindful of the different demographics and cultures that exist in our world.

References

Qiwei Bi, Jian Li, Lifeng Shang, Xin Jiang, Qun Liu, and Hanfang Yang. 2022. Mtrec: Multi-task learning over bert for news recommendation. In *Findings*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations (ICLR)*.

Ilya Sutskever, James Martens, and Geoffrey Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 1017–1024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.