Extending BERT for General Task Applicability

Stanford CS224N Default Project

Hyun Soo Jeon
Department of Computer Science
Stanford University
hsjeon@stanford.edu

Abstract

While Bidirectional Encoder Representation from Transformers (BERT) provides a widely applicable baseline for natural language processing tasks, there still remains room for improvement in performance on various downstream tasks through further fine-tuning and extension. Specifically, while it is possible to achieve good performance by fine-tuning the model on a specific task, it may be harder to fine-tune the weights onto multiple tasks simultaneously in a way that the updated weights would be generally applicable to more than one tasks. In this project, we have explored a diverse set of approaches to extend an implementation of BERT to achieve better performance in sentiment analysis, paraphrase detection, and semantic textual similarity (STS) evaluation at once. We suggest that fine-tuning the model with a combination of previously proposed methods like "gradient surgery" with small changes to the output of the model can raise the model performance to a sufficient level for multiple tasks.

1 Key Information to include

- Mentor: David Huang
- External Collaborators (if you have any): None
- Sharing project: None

2 Introduction

The popular interest in language models and the range of its capabilities have recently been even more heightened as large language models such as ChatGPT made the headlines across the world. Many of these models fundamentally have their roots in transformers, which are a class of models that allow for making use of various attention mechanisms internally and hence are able to better consider the overall context of the input text as compared to previously used models. Bidirectional Encoder Representations from Transformers (BERT), then, is a model introduced in 2018 that stems from aggregating multiple transformer layers to provide a representation ("embeddings") of words from pretraining.

The importance of BERT, among many things, lies in that it provides a baseline representation of words in the language that it has learned from the pretraining process. This is often a process that requires a large amount of time and resources to complete; however, with BERT and its pretrained weights, many tasks in the natural language processing (NLP) field can be approached by further fine-tuning BERT on each specific task.

This project aims to fine-tune the BERT model (more specifically, the minBERT model as used in the course) to simultaneously perform well on three well-defined tasks in the field of NLP: sentiment analysis, paraphrase detection, and semantic textual similarity (STS) evaluation. Sentiment analysis evaluates if a sentence carries a positive or negative correlation, determined on a scale; paraphrase detection and STS evaluation both attemps to determine whether a pair of sentences are similar to

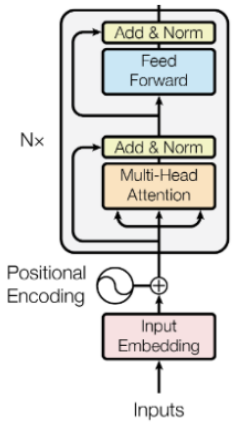

Figure 1: Diagram of a transformer layer.

each other in terms of meaning, but paraphrase detection provides a binary true/false decision whereas STS evaluation provides a numeric value on a more fine-grained scale. As these are three separate tasks, fine-tuning the weights of the BERT model to those that works well on all three tasks may be a challenging goal to achieve.

3 Related Work

The BERT model is built based on a series of Encoder Transformer layers that allow for the use of self-attention mechanism. This transformer layers are as defined in Vaswani et al. (2017), as shown in Figure 1 (taken from the aforementioned paper).

Given the structure, there always are ongoing discussions about how to best tune the parameters for a model. Some of the proposed methods are related to the specific nature of the task—for example, Henderson et al. (2017) proposes that the Multiple Negatives Ranking Loss can be used as the loss function when dealing with paired text inputs, as the function can guide the model to minimize the distance (in embedding space) between paired—or otherwise similar/related—inputs, while simultaneously maximizing the distance between non-pairs, or otherwise unrelated inputs. This, for example, may be useful for the purposes of paraphrase detection or STS evaluation, as they entail inputs of paired sentences.

On the other hand, some proposed methods consider the nature of having multiple tasks at once. For example, Yu et al. (2020) proposes "gradient surgery" to account for having multiple gradients from different tasks at a single step. Briefly, this mechanism stems from the possibility that, when training on multiple tasks at once, the gradients for each task may contradict one or more of the others, which may prevent the model from proceeding in a direction that is beneficial to all of the tasks. They propose that the mathematical projections of these gradients be used, in a way that they would not conflict with each other.

Training an implementation of BERT on multiple tasks at once can be a process where both types of methods can be useful. In this project, we aim to explore with different methods to see which one (or which combination of approaches) works best across all of the three tasks that we focus on.

4 Approach

As a preliminary step to fine-tuning, we have previously completed the implementation of minBERT as outlined in the course project handout (CS224N, Winter 2023): in this model, the sentence inputs are first tokenized into individual words (or parts of words), converted into embeddings, and then fed through Encoder Transformer layers. The model was then briefly evaluated on the Stanford Sentiment Treebank dataset from Socher et al. (2013) as well as polar movie reviews from IMDb for its performance on sentiment analysis, where the model achieved reasonable accuracies.

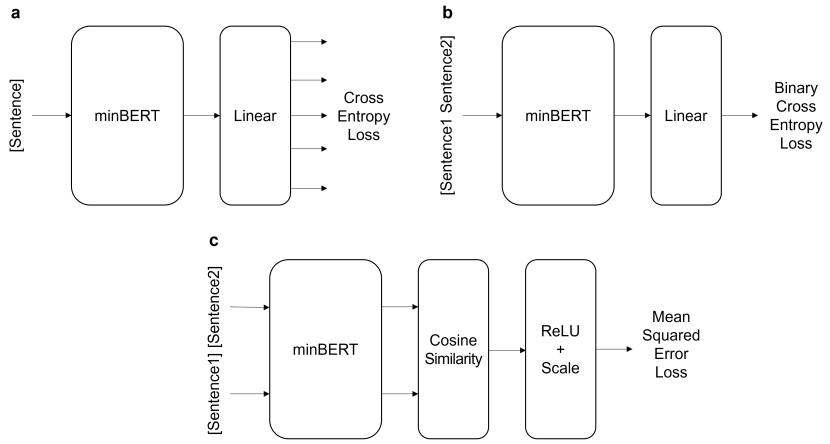

Figure 2: Diagram of header architectures for each downstream task. **a**. Sentiment analysis using a single linear layer. **b**. Paraphrase detection using concatenated paired sentences with a single linear layer. **c**. Semantic textual similarity (STS) evaluation using scaled cosine similarity.

The model architecture for each of the specific tasks is as follows; please see Figure 2 for a diagram of each.

1. Sentiment analysis
   We use a single linear layer with dropout, going from the hidden layer of the BERT model to 5 possible sentiment categories. As this is a categorical classification task, the cross entropy between the model output and the true labels is used as the loss function.

2. Paraphrase detection
   We first concatenate the provided pair of sentences, and then feed the BERT embedding of the concatenated sentence into a linear layer, resulting in a single number. This is a True/False classification task, so the binary cross entropy loss function is used (specifically, we use the `binary_cross_entropy_with_logits` function, which internally applies the sigmoid function on the output of the model).
   It should be noted that this specific approach was adapted from how paired-sentence classification was performed in Devlin et al. (2018), which outlined the original BERT model. We chose to proceed with this option after experimenting with multiple directions such as using separate linear layers or aggregating the pair of sentences in different ways.

3. STS evaluation
   We first compute the cosine similarity between the embeddings of paired sentence input, which will be a number between -1 and 1. with the assumption that, in terms of measuring similarity, the difference between a cosine similarity of 0 (not related) and -1 (opposite) are not as relevant, we then feed the value through a ReLU layer, followed by a scaling factor, to force the range to be between 0 and 5 as required by the dataset (to be further discussed below). As the similarity can be any real number inside a range, the mean-squared-error (MSE) loss function is used.

When simultaneously training the model on all three tasks, we use the "gradient surgery" (more specifically, Projecting Conflicting Gradients (PCGrad)) mechanism as proposed by Yu et al. (2020). In this approach, we first compute the cosine similarity between pair(s) of gradients from different tasks, and if they conflict (have a negative cosine similarity), it projects one of the conflicting gradients to the normal plane of the other, shown with the equation below where $\mathbf{g}_i$ and $\mathbf{g}_j$ are conflicting gradients:

$$\mathbf{g}_i = \mathbf{g}_i - \frac{\mathbf{g}_i \cdot \mathbf{g}_j}{||\mathbf{g}_j||^2} \mathbf{g}_j$$

The implementation of this method was used from Tseng (2020).

Table 1: Model performance when separately trained on each task.

	Sentiment Analysis	Paraphrase Detection	STS Evaluation
Training Set Performance	0.975	0.901	0.920
Dev Set Performance	0.511	0.871	0.704

5 Experiments

5.1 Data

For each of the three downstream tasks, we will be using a different dataset as each task requires a different set of inputs and outputs.

1. Sentiment analysis
   We used the Stanford Sentiment Treebank (SST) dataset from Socher et al. (2013), which consists of 11,855 single-sentence movie reviews with integer labels ranging from 0 (Negative) to 4 (Positive).
2. Paraphrase detection
   We used a subset of the Quora question dataset, which consists of 202,152 pairs of questions as well as a label denoting whether the sentences in each pair is a paraphrase of one another. As this dataset was much bigger in size than the datasets for the other two tasks, we used a further subsetted version of the dataset when training due to practical reasons.
3. STS evaluation
   We used the SemEval STS Benchmark Dataset, which consists of 8,628 sentence pairs and a real number similarity score on a scale of 0 (unrelated) to 5 (equivalent).

All three datasets were used as provided in the default project (CS224N, Winter 2023), with each partitioned into a train:dev:test ratio of 7:1:2.

5.2 Evaluation method

Because the sentiment analysis and paraphrase detection tasks have categorical labels (integer or True/False), we used the accuracy of the model output against the true labels when evaluating the model performance. However, the SemEval dataset used for STS evaluation provides a real number similarity between each pair of sentences. In this case, we used the Pearson correlation between the model output and the true similarity for evaluation.

5.3 Experimental details

All experiments were run with the `finetune` option as the objective was to further fine-tune the model weights. Each was run for 10 epochs, with a learning rate of 10^{-5} and a dropout probability of 0.3 where applicable. Batch size was determined to ensure there were no memory issues—for sentiment analysis and paraphrase detection, we used a batch size of 8, and for STS evaluation, the batch size was slightly enlarged to make sure all data were being used while iterating over the same number of batches as the other two tasks. Training generally took 5-6 minutes per epoch, resulting in about 1 hour for each run. All metrics reported are from the dev set, unless otherwise noted.

5.4 Results

Prior to training the model on all three tasks simultaneously, we first trained the model separately on each of the task using the corresponding architecture outlined in the Approach section, not only to confirm that it is learning the parameters correctly but also to gauge what the performance of the model is when trained on a single task. This is not necessarily a baseline model as multitasking would be a separate problem, but it would serve as a useful relative reference to how high the performance could be (Table 1).

Then, we proceeded to train the model simultaneously on the three tasks. The first attempt was to simply add the losses from one batch of each task at each iteration and use it as a loss function,

Table 2: Model performance when simultaneously trained on multiple tasks across different conditions. The accuracy/correlation on the dev set is reported. Instances of the best performance are highlighted in bold. Sentiment = Sentiment Analysis, Paraphrase = Paraphrase Detection, STS = STS Evaluation.

	Sentiment	Paraphrase	STS	Average
Addition of Losses	**0.516**	0.707	0.683	0.636
Gradient Surgery	0.475	0.778	0.702	0.652
Gradient Surgery + Average of Hidden States	0.501	**0.779**	**0.730**	**0.670**

as done in many such multitask fine-tuning procedures like in Bi et al. (2022). While this worked reasonably well, there still seemed to be room for improvement in terms of accuracy/correlation across all three tasks. We also tried using gradient surgery as previously introduced in the Approach section, as well as using an average of all hidden states as the output of minBERT's `forward()` function instead of only the `[CLS]` token. A summary of the model performance across these settings are provided in Table 2.

We can see that while the differences are not large, using a combination of gradient surgery and an alternative output of the BERT model gives the best performance in general. While the performance is certainly not as good as when trained separately on each task, it seems that the performance is still in a reasonable range considering that the weights now need to take all three tasks into account. It is also notable that there seems to be some fluctuation in the performance of sentiment analysis, and we can see that it is doing worse overall than the other two tasks.

With further explorations with hyperparameters, we achieved the following scores on the Test Set leaderboard. The only hyperparameter that resulted in a change was the number of epochs, where this was trained on 9 epochs and the batch size of 16.

	Sentiment Analysis	Paraphrase Detection	STS Evaluation	Average
Test Set Performance	0.506	0.799	0.713	0.673

6 Analysis

The lower overall performance in sentiment analysis may be something that can be addressed regarding the model architecture for the sentiment analysis task, rather than the multitasking part—we can see that even in Table 1, when the model was trained separately for each task, its performance on the training set for sentiment analysis is much higher than on the dev set for the same task, signaling that the model might be overfitting in terms of sentiment analysis. While there is a dropout layer present with the linear layer for sentiment analysis, it may be the case that sentence sentiments are harder to be captured as it generally needs to take into account the entirety of the sentence, and therefore are subject to more fluctuation with a larger diversity in expressing thoughts. Paraphrase detection and STS evaluation may suffer less in such aspects, as these tasks are sometimes subject to some key words that are present in both sentences.

Furthermore, it may be the case that tasks using paired sentences are benefitting from how the weights were pretrained for BERT. One of the tasks that BERT was pretrained on was next sentence prediction, which involves understanding the relationship between multiple consecutive sentences in a given context. This may have aided in paraphrase detection and STS evaluation, as these tasks also fundamentally involve identifying the relationship between paired sentences. This is more apparent considering how the model architecture was designed for paraphrase detection in this project—while there was no layer or term that directly aimed to capture the similarity between the paired sentences (such as cosine similarity used in STS evaluation, or other types of embedding aggregation methods), BERT still performed well in this task only given the concatenation of these paired sentences, given a few iterations of fine-tuning.

7 Conclusion

This project contributes to extending the BERT model for a wider application in various tasks in the field of NLP. While it may be possible to achieve a good performance fine-tuning the pretrained

weights to a specific task of interest, it is a more complex problem to find a way to improve the weights themselves so that they can be generally more beneficial to multiple different tasks. We have found that combining previously proposed methods with little changes to the model can synergize to give a better result across different tasks.

However, there certainly is much more room for improvement. Given that the performance on each task is much higher when trained separately, there would be a way to raise the performance of multitask-fine-tuned models to similar levels as well. I regret that while we attempted to implement a version of multiple negatives ranking loss, which we are sure would be helpful in further improving performance on paired inputs, we were not able to generate a successful implementation. In general, taking the nature of each task into account and finding the appropriate methods to fine-tune the weights while making it reusable for other tasks seems to be a field with much potential.

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 of the Association for Computational Linguistics: ACL 2022*.

CS224N, Winter 2023. CS 224N: Default final project: minBERT and downstream tasks.

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

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Wei-Cheng Tseng. 2020. Weichengtseng/pytorch-pcgrad.

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*.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*.