Multi-Task Learning With a BERT-y Good Model

Stanford CS224N Default Project

Nabil Ahmed
Department of Statistics
Stanford University
nabilah@stanford.edu

David Karamardian
Department of Statistics
Stanford University
dk11@stanford.edu

Abstract

We explore methods of fine-tuning Bidirectional Encoder Representations from Transformers (BERT) embeddings in order to optimize performance for a variety of downstream tasks, including sentiment analysis, paraphrase detection, and semantic textual similarity. After implementing a minimalist, working version of BERT, dubbed miniBERT, we layer in different fine-tuning tactics, like multitask loss and gradient surgery, to improve performance across all three tasks.

1 Introduction

The arrival of large, pre-trained language models has given birth to the potential for break-throughs in performance across a spectrum of natural langauge processing tasks. Similar to Sun et al. (2019), the background and motivation for our paper is to build upon the amazing success of BERT's language understanding capabilities and explore its full potential for a variety of downstream tasks by investigating different fine-tuning methods. Sun et al. (2019) note in their introduction that for Natural Language Processing, text representation is a critical intermediate step before engaging in further downstream tasks like text classification. Pre-trained models like OpenAI GPT and BERT have shown many improvements in this area and offer the benefit of not having to train a model from scratch. However, the problem we explore is how to optimize BERT for the specific task, or multiple tasks, at hand.

Optimizing a single network to perform well across multiple tasks presents challenges that we tackle by experimenting with different fine-tuning protocols. The tasks we seek to optimize for are sentiment analysis, paraphrase detection, and semantic textual similarity. We begin by implementing key components of a minimalist version of BERT for just the task of sentiment classification, and our working version of miniBERT, both with and without fine-tuning, beats baseline performance measures given in the default project guidelines 1. We then pivot to designing our network architecture to be able to accommodate all three tasks in parallel, and we fine-tune our network to optimize performance for all three tasks, leveraging tools like multitask loss (Bi et al., 2022) and gradient surgery (Yu et al., 2020). Ultimately, after experimenting with the aformentioned techniques, our final iteration of the model achieves validation results of 0.484 for sentiment classification accuracy, 0.803 for paraphrase detection accuracy, and 0.746 for semantic textual similarity correlation, as shown in Figure 2. These results place our team at a respectable level on the CS 224N leaderboard.

2 Related Work

Our research was influenced by a number of other works that exist in the context of fine-tuning large language models for multitask performance. The initial motivation for our paper came from Sun et al. (2019), who outlined their interest to build upon the amazing success of BERT's language understanding capabilities and explore its full potential for text classification by investigating different fine-tuning methods. For fine-tuning, the authors keep in mind that different layers of the network capture different levels of information, so the layers should be fine-tuned accordingly depending on the target task. Different optimization algorithms and learning rates can be leveraged to address this.

Beyond these micro-level strategies, the macro-level strategies they suggest are further pre-training BERT on the data from the target task distribution, and doing multitask fine-tuning. We found this paper to be a good starting point for understanding the broader context of our work and establishing our direction to approach our tasks.

Regarding multitask performance, Bi et al. (2022) propose a framework for multitask learning that takes into consideration multi-field information into BERT. To train on multiple tasks at once, they sum up the losses from the individual tasks to get a combined loss. Their approach heavily influenced our approach for training our model, since we also needed to optimize BERT to perform well across three separate tasks.

Bi et al. (2022) also reference Yu et al. (2020)'s novel form of gradient surgery, named projected conflicting gradients (PCGrad). In an effort to optimize performance across a variety of training tasks, PCGrad aims to deconflict conflicting gradients from the individual tasks during optimization; for conflicting gradients with negative cosine similarity between two tasks, the gradient of each task is projected onto the normal plane of the other task's gradient (Yu et al., 2020). To illustrate, we would replace the gradient g_i from task T_i with its projection onto the normal plane of gradient g_j as follows:

$$T_j : g_i = g_i - \frac{g_i \cdot g_j}{||g_j||^2} g_j$$

We leveraged the PCGrad repository from (Yu et al., 2020) for easy and efficient implementation of gradient surgery.

3 Approach

In accordance with Sun et al. (2019) and their efforts to fine-tune BERT, our primary aim for this project was to try a variety of approaches to see what fine-tuning methods on BERT work best for achieving high performance across all three tasks of sentiment analysis, paraphrase detection, and semantic textual similarity. To start, in accordance with the guidelines for the default final project, we implemented some of the core components of the miniBERT model, including the multi-headed self-attention layer, other sections to realize the full BERT transformer layer, the sentiment classifier, and the step function for the Adam Optimizer.

Once we finished implementing a working version of the miniBERT model, we conducted sentiment analysis on two datasets. For the task of sentiment classification, since we are only classifying the sentiment of one sentence, we run the sentence through miniBERT and generate a CLS token embedding that we then run through the sentiment classifier head, a 2-layer neural network that acts as a binary classifier. We generated two sets of results, one evaluating our model with just pre-trained weights, and another evaluating our model with embeddings that we fine-tuned on this task of sentiment analysis. For training purposes here, we used Cross Entropy Loss. The results compared to baseline accuracy values provided in the project guidelines can be found below in Table 1.

After implementing the core components of the miniBERT model and evaluating its performance on the task of sentiment classification, we pivoted to adjusting our network architecture to accommodate the additional tasks of paraphrase detection and semantic textual similarity, as shown in Figure 1. Since the tasks of paraphrase detection and semantic textual similarity compare two sentences, for both tasks we run the two sentences being compared to each other separately through the model, generating a separate CLS token embedding for each sentence. Then, for paraphrase detection, we concatenate the two CLS token embeddings and run this output through a 2-layer neural network classifier head, with the final layer acting as a binary classifier. For training purposes here, we used Cross Entropy Loss. For semantic textual similarity, we run both CLS tokens separately through the 2-layer neural network classifier head, then take the cosine similarity between the two final layers that are generated. We then multiply this scalar by 5 to re-scale for our task, which has a numeric continuous label from 0 to 5, and use the Mean Square Error Loss for training purposes.

To fine-tune our model, we leveraged a combination of training on tasks individually, as well as training on all three tasks simultaneously by computing multitask loss in accordance with Bi et al. (2022). For sentiment classification and semantic textual similarity, we trained each of these tasks

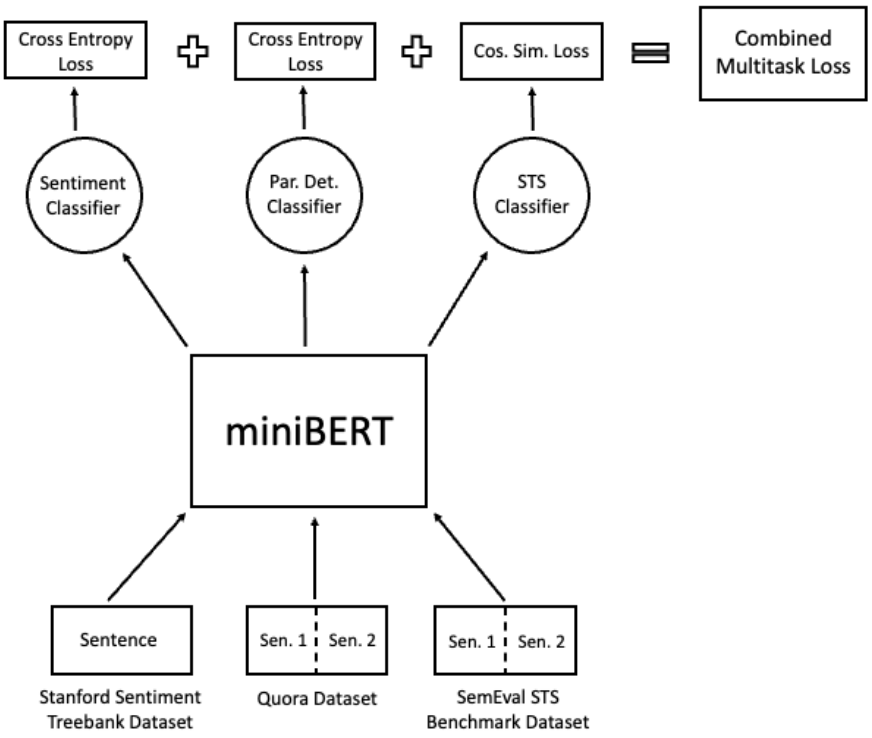

Figure 1: In order to calculate multitask loss during multitask training, we leverage the above architecture to sum up losses across all three tasks.

individually, backpropagating through the entire network. After fine-tuning on those two tasks individually, we then fine-tuned on all three tasks concurrently, summing up their three losses to create a combined multitask loss at the end of each batch, and backpropagating through the entire network. Since the datasets sizes differed across the three tasks and datasets, we aligned them to be able to run in parallel by repeating batches in the smaller datasets to match length of the biggest dataset. During backpropagation, we experimented with leveraging PCGrad gradient surgery (Yu et al., 2020) as well in order to optimize performance across all three tasks by resolving any conflicting gradients.

Details for these tasks, their corresponding datasets, evaluation methods, and experiment details can be found below in Section 2. Results for these tasks can be found in Table 2.

4 Experiments

4.1 Data

For Part 1 of our project, we are using the Stanford Sentiment Treebank (SST) dataset SA (2013) and the CFIMDB dataset for the task of sentiment analysis. The SST dataset consists of phrases labeled as negative, somewhat negative, neutral, somewhat positive, or positive. The CFIMDB dataset consists of movie reviews labeled as positive or negative.

For Part 2 of our project, we are using the Quora dataset Quo for the task of paraphrase detection, as well as the SemEval STS Benchmark dataset STS for the task of semantic textual similarity. The Quora dataset consists of 400,000 question pairs with labels indicating whether particular instances are paraphrases of one another, and the SemEval STS Benchmark dataset consists of 8,628 different sentence pairs of varying similarity on a scale from 0 (unrelated) to 5 (equivalent meaning).

All datasets come with training, dev, and test splits, and all evaluation metrics in this milestone report are computed on the dev datasets.

4.2 Evaluation Method

For the task of sentiment classification, we used the evaluation metric of accuracy, which measures whether the model correctly classified a movie review as positive of negative for the CFIMDB dataset, or as one of the five labels for the STS dataset.

For the task of paraphrase detection, which uses a binary label of 'Yes' or 'No' depending on if the sentences are paraphrases of each other, we also use the metric of accuracy.

For the task of semantic textual similarity, which can produces values ranging from 0 to 5 depending on the similarity of the sentences within a pair, we compute the Pearson correlation of the true similarity values against the predicted similarity values.

4.3 Experimental Details

As shown in Figure 2, the Baseline metrics come from our milestone update earlier in the quarter, when we only trained the three classifier heads individually without incorporating multitask loss training. Iteration 1 was our first pass at incorporating multitask loss; however, for the semantic textual similarity task, we did not use the version of Mean Square Error loss that incorporated cosine similarity between embeddings, and we did not incorporate gradient surgery. For Iteration 2, we incorporated the cosine similarity loss for semantic textual similarity, and for iteration 3, we incorporated the cosine similarity loss and gradient surgery. The Baseline and Iteration 1 trained on two epochs of data, and all subsequent iterations trained on 5 epochs of data individually for the tasks of sentiment classification and semantic textual similarity for phase 1, as well as 5 epochs of data for all three tasks simultaneously with multitask loss.

4.4 Results

Sentiment Classification Acc.	**Baseline**	**Results**
SST Dataset		
Pretraining	0.390	0.399
Finetuning	0.515	0.525
CFIMDB Dataset		
Pretraining	0.780	0.796
Finetuning	0.966	0.967

Table 1: Part 1 Results

As shown in Table 1, for the task of sentiment classification for the SST and CFIMDB datasets, our pretrained and finetuned models all outperform baseline accuracy results provided in Part 1 of the default project guidelines.

Evaluation Metric	**Baseline**	**Iteration 1**	**Iteration 2**	**Iteration 3**
Sentiment Classification Acc.	0.522	0.514	0.499	0.484
Paraphrase Detection Acc.	0.468	0.669	0.791	0.803
Semantic Textual Similarity Corr.	0.007	0.331	0.728	0.746
Overall Score	0.332	0.504	0.673	0.678

Table 2: Part 2 Results

As shown in Table 2, our performance across paraphrase detection and semantic textual similarity tasks increases with each iteration of experiments we conduct to improve the model, but our performance for sentiment classification decreases with each iteration. This likely due to the increased emphasis that our subsequent techniques placed on the former two tasks. We also see that while cosine similarity loss for the semantic textual similarity task enabled a massive lift in performance on that task, gradient surgery via PCGrad yielded only a marginal gain in performance on the paraphrase task. The former result makes sense given the mathematical relationship between the cosine similarity and the Pearson correlation of two vectors (correlation is the cosine similarity between centered versions of the vectors). The latter result was slightly disappointing, but it did indicate that larger and more radical changes were needed to achieve the next level of dev set accuracy.

5 Analysis

We present here analysis pertaining to our final and best overall model (model iteration 3). The basic takeaways are also true for model iteration 2, as gradient surgery had a minimal impact on the results.

We begin by examining the accuracy plots for the first phase of model training when the sentiment and similarity tasks were trained individually for five epochs. While the model achieves respectable performance on the sentiment and similarity tasks, it is certainly over-fitting, as the training on the dev sets reaches its peak after two epochs for the sentiment task and four epochs for the similarity task. We can also see that the model has yet to converge for these tasks, as the training accuracies are still increasing as of epoch five. This is acceptable, as the model will continue to improve on these tasks during the multi-task training phase.

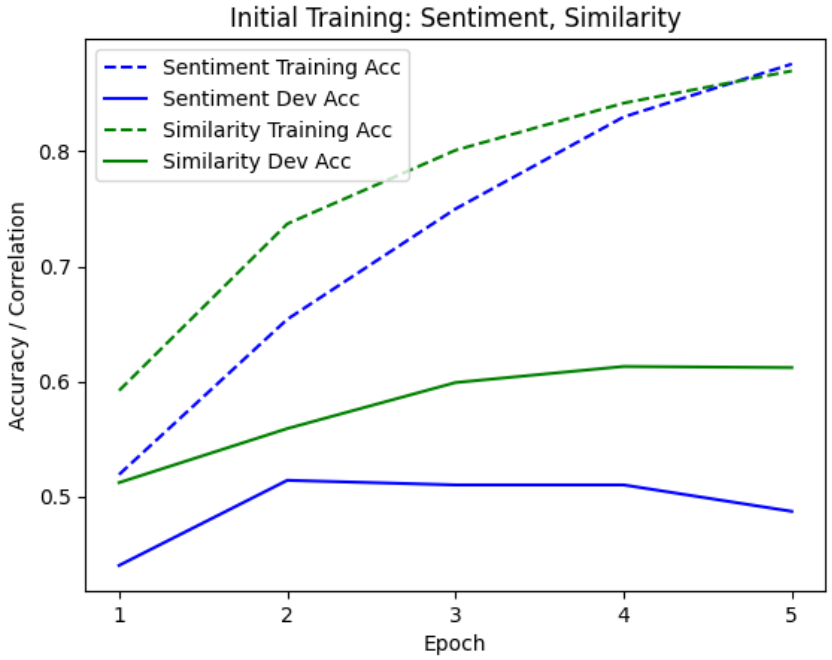

Figure 2: Training Phase 1 - Individual Training on Sentiment, Similarity Tasks

In the second phase of training, we iterate through the three training datasets simultaneously in batches of size 32 and sum their losses in the manner prescribed by the gradient surgery algorithm. The paraphrase training dataset is significantly larger than the other two, so we repeat the smaller datasets to match the size of the paraphrase set. Thus, each epoch in the plot represents one pass through all observations in the paraphrase training dataset, and within this pass there were simultaneously multiple passes completed through the other two datasets.

We see that the sentiment and similarity training accuracies increase to nearly perfect within the first epoch, while paraphrase training accuracy reaches near perfection in the fifth and final epoch.

Paraphrase dev accuracy peaks in epoch four, while the other two tasks see little change in their dev accuracies after the first epoch.

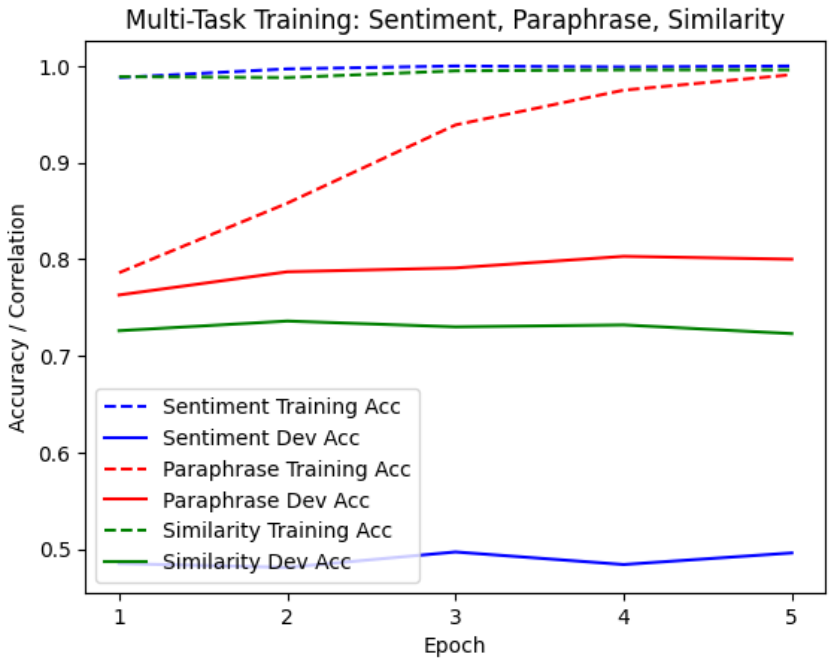

Figure 3: Training Phase 2 - Simultaneous Multi-Task Training

These plots are highly indicative of overfitting. The model is achieving near perfect training accuracies on each task (which could indicate that it has memorized many of the training examples) while the dev accuracies are stabilizing at lower levels. The problem is particularly severe on the sentiment classification task, which is conspicuously the smallest dataset among the three in terms of number of examples, and least severe on the paraphrase dataset. This leads to ideas for improving model generalization, including:

- Augmenting the training dataset(s) with additional human-labeled examples
- Augmenting the training dataset with synthetic examples generated by GPT-4
- Sharing model weights between the paraphrase task and the others
- Regularization of model weights

6 Conclusion

Working on this project illuminated the incredible power that large language models yield today, but it also shed light on the limitations that we face in order to implement them for specific use cases. Creating a model that performs well across a variety of language tasks presented a unique set of challenges, and the main limitation we believe we faced was a lack of training data. Especially for the tasks of sentiment classification and semantic textual similarity, we believe that bolstering our training datasets for these tasks could have led to considerable gains in performance. Future work could look at augmenting these training datasets with synthetic data examples generated from another large language model, such as GPT-4. Stanford's Alpaca 7B model follows just such an approach.

Despite the data constraints, we were able to improve performance across the tasks of paraphrase detection and semantic textual similarity by layering in a variety of different techniques. Increasing the number of training epochs from two to five improved performance, as well as incorporating multitask loss, rather than just training on all of the tasks individually. Furthermore, implementing cosine similarity prior to computing Mean Square Error loss for semantic textual similarity led to improved performance, and incorporating gradient surgery with PCGrad boosted performance as well.

Ultimately, we realized the value in conducting a lot of experiments that layer in different approaches in order to come upon an optimal strategy.

References

First quora dataset release question pairs. https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs.

Semantic textual similarity. https://aclanthology.org/S131004.pdf.

2013. Stanford sentiment treebank. https://nlp.stanford.edu/sentiment/treebank.html.

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*.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune BERT for text classification? *CoRR*, abs/1905.05583.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *CoRR*, abs/2001.06782.