# Finetuning minBERT Model for Multiple Downstream Tasks

Stanford CS224N Default Project

**Yuan Wang**
Department of Computer Science
Stanford University
ywang09@stanford.edu

## Abstract

Pre-trained Large Language Models, such as BERT and GPT, contain rich token embeddings that are useful for various downstream tasks. Instead of building a separate model for each individual task, it could be more resource-efficient to build one model that could perform multiple tasks. This paper presents the author's findings in extending and finetuning a minBERT model to perform multiple downstream tasks. Improvements against the baseline are achieved through training on additional task data, implementing a round robin multi-task training algorithm, performing additional finetuning with minBERT parameters fixed, and finding optimal hyperparameters.

## 1 Key Information to include

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

## 2 Introduction

The emergence of powerful pretrained large language models (LLMs) such as BERT and GPT has radically changed the landscape of Natural Language Processing Research. In the past, research mostly focused on developing individual models that focuses on specific language tasks from scratch, and little knowledge sharing occurs across different models and different tasks. However, large attention-based language models that are heavily trained on simple tasks over a huge corpus of text have proven to produce very powerful token embeddings that significantly benefit almost every major downstream language task. As a result, researchers began to utilize these pretrained models as a starting point to build state-of-the-art models that tackle different downstream language tasks.

Since the rich token embeddings produced by LLMs contain useful information for various tasks, it is naturally possible to utilize the same set of embeddings for multiple downstream tasks. This is the objective of multitask language models. Since token embeddings often constitute the largest portion of a language model, utilizing the same set of embeddings for different tasks potentially offers a resource-efficient solution for multiple tasks. In this project, the author attempts to extend and finetune a minBERT model to perform three different downstream tasks: sentiment analysis, paraphrase detection, and semantic textual similarity analysis. For simplicity, these tasks will be referred to as SST, PARA, and STS.

## 3 Related Work

When LLMs become available, a lot of research and experimentation has been done on extending them to achieve state-of-the-art performance various downstream language tasks. The rich research in this realm offer a lot of interesting and promising ideas for building and improving any model that utilizes a pretrained LLM as its starting point. For example, Chi Sun *et al.*[1] performed systematic experimentation on ways in which the BERT model could be finetuned to perform various downstream tasks. In addition, there is a lot of interesting research in the realm of multitask learning. For example, Qiwei Bi *et al.*[2] used multi-task learning to improve a model's performance in news encoding and comprehension.

## 4 Approach

### Model

The model itself has a pretty simple structure. The minBERT layers provide the sentence embeddings, which are fed into one of three downstream sub-models, depending on the task being performed. As shown in Figure 1, after the sentence embedding(s) are generated from the minBERT layers, they are fed towards different task-specific downstream layers. The SST task downstream layers consist of a single linear layer, and both the PARA and STS task downstream layers consist of two linear layers for the two input sentences – whose outputs are multiplied together – and an interact linear layer. For all tasks, dropout is applied before the first linear layers. The SST task is trained using Cross Entropy Loss, and the other two tasks are trained using L1 Loss.

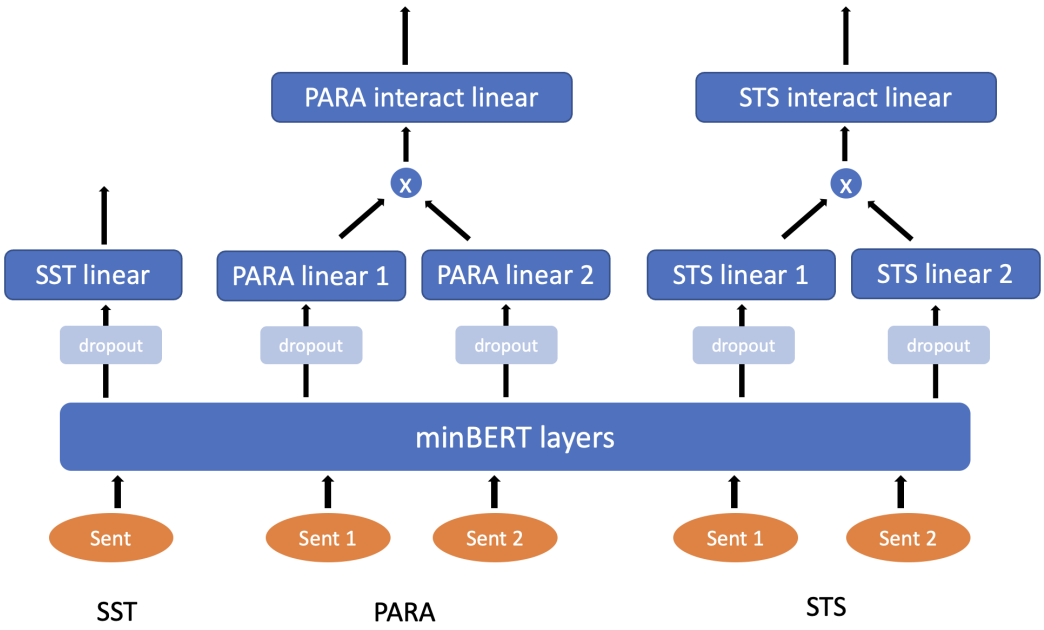


Figure 1: Model Structure

## Baseline

For the baseline method, I use task-by-task sequential training to train the model on all three datasets, as shown in Figure 2. Within each epoch, the model is first trained on the SST datset, and next on the PARA dataset, and finally on the STS dataset. Therefore, each epoch involves three rounds of parameter updates, with each round dedicated to a specific task. The model is trained for 10 epochs, with a dropout rate of 0.3 and a learning rate of 1e-5. The batch size is 32.

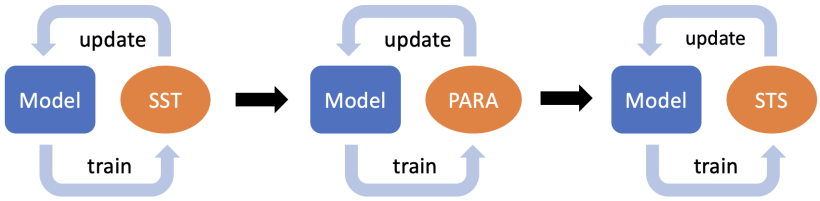


Figure 2: Baseline: Task-by-Task Training

## Round Robin multitask training

Other than the baseline training method, I also implemented Round Robin multi-task training, as shown in Figure 3. Within each epoch, multiple iterations of training are performed. Within each iteration, the model is trained on some batches of data from each of the three datasets, and the parameters are updated. The amount of data trained on is proportional to the total size of the dataset, so that at the end of each epoch, all data in the datasets have been trained on at least once, and most of the data have been trained on at most once. Since each round of parameter update is based on training losses from all three datasets, this method of training is designed to improve the overall performance of the model on all three tasks during each iteration of model update.

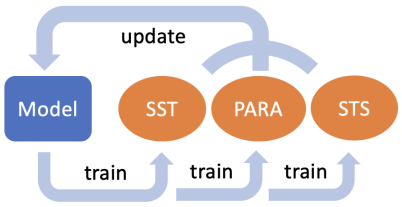


Figure 3: Round Robin Multitask Training

## Layer sharing

In the baseline model, the structure between the PARA downstream layers and the STS downstream layers is identical. The two tasks are also similar in that they require the model to compare the meaning between two input sentences. Therefore, it is possbile that training shared linear layers that extract meaning from input sentences could produce useful output for both the PARA task and the STS task. To test that hypothesis, I implemented a version of the model where the first linear layers for the PARA task and the STS task are shared, as shown in Figure 4

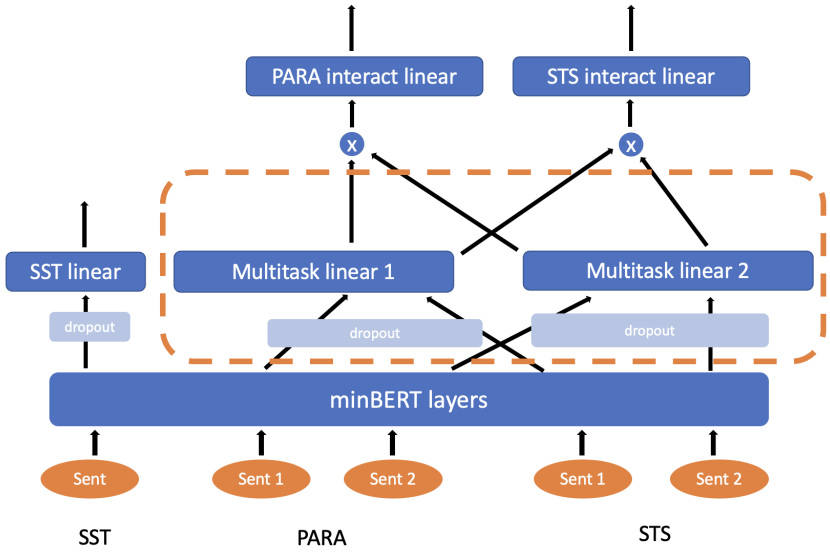


Figure 4: Layer Sharing

## Additional Training Data

While the provided training datasets are probably most useful for model training, additional related datasets could also improve performance. Since the provided training datasets for SST and STS are much smaller than the training dataset for PARA, I adapted outside datasets and added them to the provided datasets. Please see the **Data** section for more details.

## Additional training with minBERT parameters fixed

Updates to the minBERT layers affect the model's performance on all three tasks, whereas updates to a downstream layer only affects the model's performance on an individual task. While updating the minBERT layers is very important for improving model performance, it could also improve the performance on a single task at the expense of another task. Therefore, I implemented additional training with the minBERT parameters fixed, as shown in Figure 5. For each epoch, the model is trained on only one task dataset, and it was trained for a total of 9 epochs.

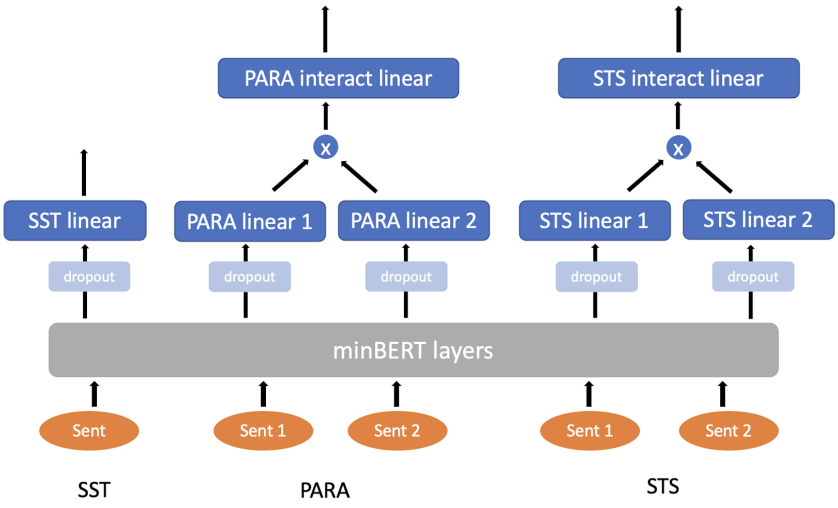


Figure 5: Additional Training with minBERT Parameters Fixed

## Other implementations and experiments

In addition to the extensions and experiments described above, there are a few additional experiments that I have done. I tried using only a portion of the PARA dataset in the training process, so

that it is similar in size to the other two datasets (see **Data** section). I also searched for optimal hyperparameters: I tried increasing the number of epochs to train, varying the learning rate and the batch size.

# 5 Experiments

## 5.1 Data

For model training, we are provided with three datasets. For SST, we are provided with a subset of the Stanford Sentiment Treebank dataset, which contains about 8.5k rows. For PARA, we are provided with a subset of the Quora dataset, which contains about 141.5k rows. For STS, we are provided with a subset of the SemEval STS dataset, which contains about 6.0k rows. The Quora dataset is more than ten times larger than the other two datasets.

In additional to the provided data, I also utilized two additional datasets. For SST, I utilized the train and dev subsets of the CFIMDB dataset, which contains about 1.9k rows of data in total. For STS, I utilized the SICK2014 dataset, which contains about 10k rows of data in total [3]. The SICK2014 dataset consists of sentence pairs and includes a sentence relatedness score, which measures how close the two sentences the pair are related to each other.

Unlike the SST dataset, which measures sentence sentiment in five discrete levels (0 to 4), the CFIMDB dataset measures sentence sentiment as positive or negative. The sentences in the dataset are said to be highly polar. To make the CFIMDB data compatible with the SST dataset, I created two versions of the dataset. In one (polar) version, I assigned a sentiment score of 0 to negative and 4 to positive; in the other (not polar) version, I assigned a score of 1 to negative and 3 to positive.

The STS data uses a continuous score between 0 and 5 to measure the similarity between two sentences. The SICK2014 dataset uses a continuous score between 1 and 5 to measure relatedness between sentences. To make the latter compatible with the former, I proportionally re-scaled the scores from $[1, 5]$ to $[0, 5]$ using the formula $s_{new} = (s_{old} - 1) \cdot 1.25$, where $s_{new}$ is the rescaled score, and $s_{old}$ is the original score.

## 5.2 Evaluation Method

I used the model's accuracy on the provided dev datasets as the primary evaluation method.

## 5.3 Experimental Results

| | | SST Accuracy | PARA Accuracy | STS Accuracy | Average Accuracy |
|---|---|---|---|---|---|
| [0] | baseline | 0.50 | 0.75 | 0.44 | 0.56 |
| [1] | 20 epochs | 0.50 | 0.77 | 0.48 | 0.59 |
| [2] | batch size 8 | 0.45 | 0.72 | 0.43 | 0.54 |
| [3] | share layers | 0.50 | 0.73 | 0.42 | 0.55 |
| [4] | cut down on PARA training | 0.50 | 0.63 | 0.38 | 0.50 |
| [5] | additional SST data (polar) | 0.50 | 0.76 | 0.48 | 0.58 |
| [6] | additional SST data (not polar) | 0.47 | 0.75 | 0.47 | 0.56 |
| [7] | additional STS data | 0.50 | 0.75 | 0.48 | 0.58 |
| [8] | additional SST data (polar) and STS data | 0.50 | 0.75 | 0.52 | 0.59 |
| [9] | additional SST data (not polar) and STS data | 0.50 | 0.75 | 0.52 | 0.59 |
| [10] | round robin multitask (batch size 8) | 0.53 | 0.76 | 0.46 | 0.58 |
| [11] | round robin multitask normalized | 0.51 | 0.73 | 0.40 | 0.55 |
| [12] | additional training with minBERT parameters fixed | 0.52 | 0.75 | 0.46 | 0.58 |

Figure 6: Experiment Results

Figure 6 shows the results I obtained from running the experiments described in the **Approach** section. The blue numbers represent improvements against the baseline, whereas the red numbers represent deterioration in performance. The list of changes that improved performance includes increasing number of epochs trained [2], training on additional datasets ([5]-[9]), using *un-normalized*

round robin multitask training[1] [10], and performing additional training with minBERT parameters fixed [12]. On the other hand, the list of changes that did not lead to improvement are decreasing batch size to 8 [2], sharing linear layers between PARA and STS [3], cut down on PARA training[2] [4], and performing normalized round robin multitask training[3] [11].

### 5.4 Final Results

In the final version, I ensembled all the changes that improved the performance of the model against the baseline version, and was able to achieve non-trivial improvement over the baseline, as shown in Figure 7. The final version was trained using the Round Robin multitask method on both the provided training datasets and the additional CFIMDB and SICK2014 (polar version) datasets. It was trained for 20 epochs with the minBERT parameters actively updated; next, the currently best parameters were loaded, and the model was further trained for 9 epochs with the minBERT parameters fixed, with a learning rate of 1e-6. In each of the 9 epochs, the model is trained on only one task dataset.

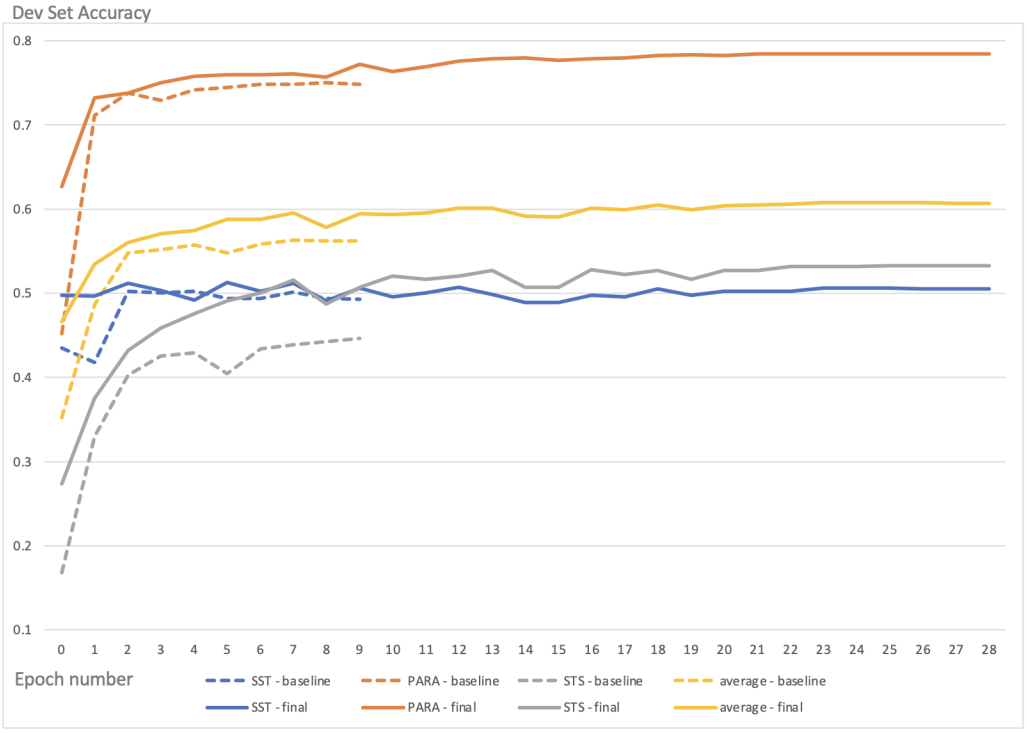


Figure 7: Final Results

My submission to the test set leaderboard yielded an SST accuracy of 0.510, a PARA accuracy of 0.788, an STS accuracy of 0.531, and an overall average accuracy of 0.610. While there is non-trivial improvement from the baseline version to the final version, the improvement in accuracy isn't as significant as I expected. There are likely many unexplored areas which could further improve model performance.

## 6 Analysis

One interesting phenomenon is that the training of the model on the high-volume Quora dataset seems to benefit model performance on the other two tasks through producing more powerful and pertinent minBERT embeddings. This could be reflected when comparing the baseline [0] and the

[1]Due to limitation in computation power, all round robin multitask training is run with a batch size of 8.
[2]Only 8000 rows of the Quora dev dataset is used for training.
[3]Within each iteration, gradients are normalized by the amount of task data trained on.

version where PARA training was cut [4], and when comparing the normalized and unnormalized versions of round robin multitask training [10] [11]. This is surprising, because it was anticipated that balancing model training towards the SST and STS task and against the PARA task - which has a much larger training dataset, regardless of whether additional training data is added - might improve the performance on the two former tasks, at the potentially slight expense of the PARA task. However, this balancing strategy actually tends to decrease model accuracy on all three datasets, as illustrated in Figure 8, for example. It seems that training on the Quora dataset has a significant enriching effect on the minBERT parameters that improves performance on the other two tasks, especially the STS task.

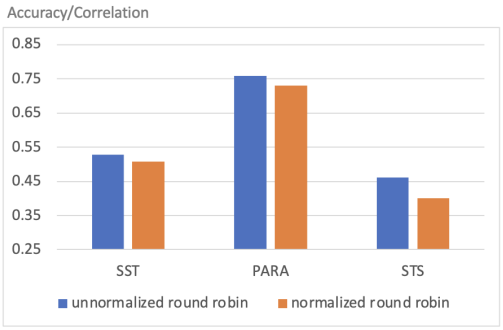


Figure 8: Round Robin Multitask Training: a Comparison

Another interesting phenomenon is that there seems to be "diminishing return" when ensembling multiple methods that help improve performance against the baseline. The ensembled model is more accurate than any model that implements an individual measure, yet the increase in accuracy is far smaller than the numerical sum of accuracy increase of all the individual models (see Figure 9). To a certain extent, this is necessarily the case, because otherwise it would not be very challenging to repeatedly bundle up helpful measures until a perfect accuracy is achieved. However, it would be interesting to research and study what types of measures tend to lead to "redundant" improvements when ensembled, and what types of measures tend to "stack up" improvements when combined.

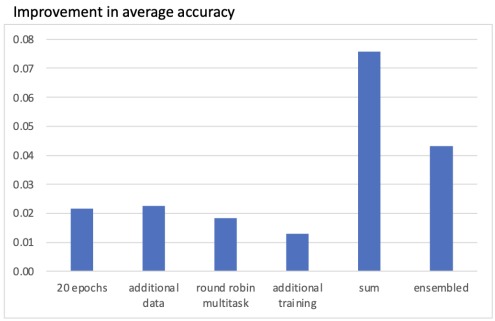


Figure 9: The Diminishing Returns of Ensembling

## 7 Conclusion

This project discovered a few successful measures that help improve performance of the multitask minBERT model against the baseline. These measures include increasing number of epochs trained, training on additional datasets, using round robin multitask training as opposed to task-by-task training, and performing additional training with minBERT parameters fixed. Ensembling these changes has improved the overall performance of the model by close to 5% across all three tasks. It was also discovered that training on the high-volume PARA Quora dataset helps improve model performance on the two other tasks.

If more time and resources are available, it would be helpful to perform more experimentation with different model designs, including altering the downstream layers and the loss functions.

## References


[1] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer, 2019.

[2] Lifeng Shang Xin Jiang Qun Liu Qiwei Bi, Jian Li and Hanfang Yang. Mtrec: Multi-task learning over bert for news recommendation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2663–2669, 2022.

[3] Marco Marelli; Stefano Menini; Marco Baroni; Luisa Bentivogli; Raffaella Bernardi; Roberto Zamparelli. The sick (sentences involving compositional knowledge) dataset for relatedness and entailment. In *Association for Computational Linguistics (ACL)*, 2014.