YourBERT: Tailoring BERT for Precision

Stanford CS224N Default Project

Paolo Tayag
Department of Symbolic Systems
Stanford University
`mptayag@stanford.edu`

Jack Walter
Department of Computer Science
Stanford University
`jtwalter@stanford.edu`

Abstract

We enhance BERT for various NLP tasks through domain-specific fine-tuning, after finding negligible improvements from domain-specific pre-training. Initially, we established a working minBERT model and validated its performance on tasks like sentiment classification, paraphrase detection, and semantic textual similarity. We then compared different fine-tuning methods, developing four models: three fine-tuned on individual datasets, and one on all three datasets simultaneously. The best performance was achieved with task-specific fine-tuning: SST-5 (0.511 dev accuracy), QQP (0.804 dev accuracy), and STS (0.382 dev correlation). Our findings suggest task-specific fine-tuning may be more effective than multi-task fine-tuning for specialized NLP applications.

1 Key Information to include

- Mentor: Josh Singh
- External Collaborators (if you have any): None
- Sharing project: No
- Team contributions: Paolo focused on testing methods for multi-task fine-tuning and performing hyperparameter tuning to optimize the model's performance across different tasks. Jack focused on implementing the minBERT model and refining the STS loss to enhance its accuracy and robustness.

2 Introduction

Pre-training on language models has proven highly effective for enhancing various NLP tasks (Dai and Le (2015); Peters et al. (2018); Howard and Ruder (2018)). BERT (Bidirectional Encoder Representations from Transformers), introduced in 2019, advanced this approach by enabling deep bidirectional representations from unlabeled text and demonstrating strong performance across multiple tasks with minimal task-specific modifications (Devlin et al. (2019)).

Despite its success, BERT's evaluation has primarily focused on specific datasets, suggesting potential for broader optimization. We initially investigated domain-specific pre-training, hypothesizing that additional training on domain-relevant corpora would yield performance gains. However, our initial experiments indicated that this approach was computationally intensive and provided limited improvements, prompting us to shift our focus to more efficient fine-tuning techniques.

Our project aims to enhance BERT's performance on sentiment classification, paraphrase detection, and semantic textual similarity through targeted fine-tuning methods. We established a baseline using a minimal BERT model (minBERT) and validated its performance. To determine the most effective fine-tuning strategy, we compared models fine-tuned on individual datasets with those trained on multiple datasets simultaneously. Our findings demonstrated that task-specific fine-tuning generally outperformed multi-task fine-tuning, delivering superior results across the targeted NLP tasks.

3 Related Work

Our work is influenced by key studies on fine-tuning pre-trained models and skill localization within these models. The foundation for our approach is the paper "How to Fine-Tune BERT for Text Classification?" by Sun et al. (2019), which provides a structured methodology for fine-tuning BERT, including domain-specific pre-training and various fine-tuning techniques. Using pre-trained transformer networks for downstream NLP tasks was initially explored by Devlin et al. (2019), who introduced BERT using masked language modeling and next sentence prediction. "Task-Specific Skill Localization in Fine-Tuned Language Models" by Panigrahi et al. (2023) also significantly influenced our approach. Their study shed light on the concept of skill localization, also explored by Sun et al. (2019), which identifies specific regions within a model responsible for task performance.

4 Approach

Our approach to enhancing BERT's performance on specific NLP tasks involved several key steps: establishing a baseline model, implementing fine-tuning techniques, and experimenting with both task-specific and multi-task fine-tuning.

We began by re-implementing the core components of BERT. Leveraging the starter code and a minimalist implementation of BERT (minBERT), we focused on enhancing the model's foundational elements. Specifically, we completed the implementation of the minBERT model, incorporating Multi-head Self-attention, the Transformer Layer, and the Adam Optimizer. Additionally, we implemented Sentiment Classification using BERT embeddings. With these foundational implementations in place, we moved on to our primary objective of enhancing the model's robustness and generalization capabilities broadening its applicability across various tasks and datasets. In the following sections, we detail methodologies we applied to achieve these enhancements.

4.1 Adjusting hyperparameters

Our first step was to conduct a hyperparameter search to identify the best settings for our implemented minBERT model. Using our completed model with both pre-trained and fine-tuned embeddings, we carried out extensive experiments on selected datasets, testing various hyperparameters. The model's performance was evaluated using accuracy metrics across different datasets. This is discussed in more detail in section 5.4.

4.2 Multi-task fine-tuning

We tackled two different approaches to fine-tuning our classifier to be able to handle multiple tasks. Our first approach was to create one model that can adapt to multiple tasks at once. To do this, we employed multi-task learning to fine-tune BERT simultaneously, based on the loss functions of three different tasks (sentiment classification, paraphrase detection, and semantic textual similarity). For the sentiment analysis task, the labels for each input are 5 class categorical variables, and subsequently the loss function is the cross entropy loss. For the paraphrase detection task, the labels are binary categorical variables (yes or no) and the loss function is the binary cross entropy loss. For the semantic textual similarity task, we used mean squared error between the predicted similarity scores and ground-truth scores.

In this multi-task setup, we integrated the loss functions for each task into a combined objective function, ensuring that the model learns from all tasks concurrently. The combined loss function was calculated as follows:

$$Loss_{multi-task} = Loss_{sentiment} + Loss_{paraphrase} + Loss_{similarity}$$

Before settling on this loss function, to balance the contributions of each task, we experimented with different weighting strategies to ensure that each task influenced the training process appropriately.

Our second approach focused on task-specific fine-tuning. Here, we fine-tuned separate models for each individual task. Each model was trained exclusively on its corresponding dataset, allowing the fine-tuning process to specialize in the nuances of a single task. This method aimed to optimize the model's performance for each specific task without the potential interference from other tasks.

For instance, in enhancing the performance of semantic textual similarity we adapted a model from Reimers and Gurevych (2019) that instead used the the cosine similarity of two sentences' BERT embeddings to evaluate semantic similarity. We then followed their usage of the mean squared error between the prediction and ground truth as a loss function. We reshaped the cosine similarity scale to match with the ground truth labels as well.

5 Experiments

5.1 Data

To fine-tune and validate our BERT model across various NLP tasks, we utilized several datasets provided in the default project handout. We used the Stanford Sentiment Treebank (SST-5) and CFIMDB datasets to test our implementation of minBERT through its performance on the task of sentiment analysis. Both datasets consist of movie reviews labeled with sentiment scores ranging from 0 (very negative) to 4 (very positive). We also used the Quora Question Pairs (QQP) and SemEval STS Benchmark to further fine-tune and validate our BERT model. QQP was used to test paraphrase detection, and SemEval STS Benchmark was used to test semantic textual similarity. Each dataset contains train, dev, and test splits, which are provided to us by the CS224N staff.

5.2 Evaluation method

We evaluated our model's performance through a number of baselines. In terms of overall performance, we used the formula used by the leaderboard to quantify the multi-task performance of our models.

$$f(\hat{y}) = \frac{1}{3}\left(acc_{sentiment} + acc_{paraphrase} + \frac{corr_{similarity} + 1}{2}\right)$$

To assess the task-specific performance of our models, we drew baselines from scores reported in foundational research literature, including previously reported accuracy scores for sentiment classification and paraphrase detection, as well as Pearson Correlation values for semantic textual similarity. Specifically, Munikar et al. (2019) achieved an accuracy of 0.532 on the Stanford Sentiment Treebank (SST-5) dataset using the $\text{BERT}_{\text{BASE}}$ model. Devlin et al. (2019) achieved an accuracy of 0.893 on the Quora Question Pairs (QQP) dataset and a Pearson Correlation of 0.876 on the SemEval STS-B dataset using the $\text{BERT}_{\text{LARGE}}$ model, although no corresponding results were shared for the $\text{BERT}_{\text{BASE}}$ model.

5.3 Experimental details

With our implemented version of minBERT, we moved on to finding the ideal hyperparameters for our model as well as experimenting with the two aforementioned architectures for a model capable of multi-task classification.

First, we evaluated the performance of a fine-tuned minBERT with different hyperparameters.

Then, using the default hyperparameters provided in the starter code, we created four separate models. Three models are each separately fine-tuned on one of the three provided datasets (SST, QQP, STS-B). The final model is simultanously fine-tuned on all three datasets using the aforementioned combined loss function.

The experiments in this paper were carried out on Google Cloud Platform, where we utilized the Compute Engine API Service. We used a single NVIDIA T4 GPU VM located in the us-west3-b zone with the image pytorch-2-0-gpu-v20231105-debian-11-py310.

5.4 Results

In our search for the most effective hyperparameters, we tested an array of hyperparameters independent of each other, as shown in table 1.

We found that while certain hyperparameters performed better than others, there was minimal variance in performance when compared to the default hyperparameters. Only when using the 1.00e-03 learning rate, which was suggested for use only when fine-tuning the last linear layer, were

Dev Accuracy	**SST (Finetune)**	**CFIMDB (Finetune)**
Default	0.530	0.967
batch=16	0.525	0.968
batch=32	0.517	0.962
seed=33333	0.508	0.976
seed=100	0.528	0.953
P dropout=0.1	0.524	0.975
P dropout=0.7	0.526	0.967
lr = 1.00E-03	0.262	0.502

Table 1: Sentiment Analysis on default and different hyperparameters

there significant losses in the accuracy scores for both datasets. Therefore, for our following models, we chose to stick with the default hyperparameters when fine-tuning in following experiments.

Our minimal and multi-task implementations of BERT both met/exceeded each of the baselines provided in the default project handout for sentiment analysis. We tested the performance of fine-tuning the full model as well as only fine-tuning the last linear layer, as pictured in figure 1.

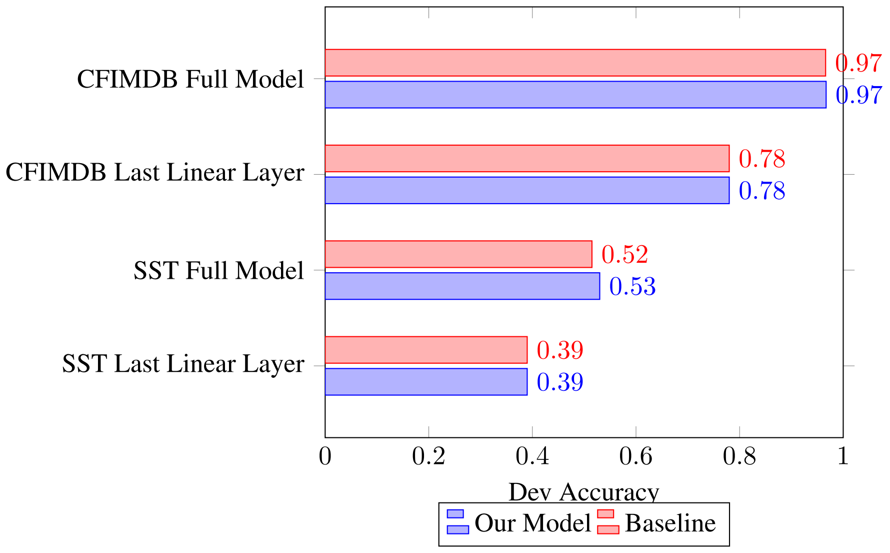

Figure 1: Sentiment analysis accuracies of the models compared to the baselines

However, our multi-task models mostly did not perform to the level of our baselines. As shown in table 2, while our individually fine-tuned models reached comparable accuracy scores to the baselines we provided for ourselves, only the sentiment analysis score on the leaderboard test set (also performed by our individually fine-tuned models) reached/exceeded these baselines. Furthermore, our multi-task fine-tuned model performed below expectations across several metrics.

The multi-task model struggled particularly with overfitting, likely due to the increased complexity and the smaller effective training data for each task. The shared representation learned by the multi-task model did not generalize as well as we hoped, leading to lower accuracy scores in all three tasks.

6 Analysis

In this section, we delve into a qualitative evaluation of our models to gain deeper insights into their performance, understand their behavior, and identify areas for improvement. We employ various qualitative evaluation techniques, including comparing the behaviors of different fine-tuning approaches.

Task	**Individually Fine-tuned (dev)**	**Multi-task Fine-tuned (dev)**	**Baseline**	**Individually Fine-tuned (test)**
Sentiment Analysis (SST)	0.511	0.306	0.532	0.538
Paraphrase Detection (QQP)	0.804	0.655	0.893	0.805
Semantic Textual Similarity (STS-B)	0.729	0.507	0.876	0.731

Table 2: Comparison of scores produced by individually fine-tuned models, multi-task fine-tuned model, baselines, and leaderboard scores

6.1 Comparison of Fine-tuning Approaches

Comparing the behaviors of our individually fine-tuned models and the multi-task fine-tuned model revealed interesting differences. While individually fine-tuned models exhibited task-specific performance improvements, the multi-task model struggled with overfitting and failed to generalize effectively across tasks. This suggests that task-specific fine-tuning may be more suitable for optimizing performance on specialized tasks, whereas multi-task fine-tuning may require additional regularization techniques or architectural modifications to enhance generalization.

For instance, in our experiments with fine-tuning for semantic textual similarity, we observed a decrease in accuracy across the other tasks. This can be largely attributed to the changes enforced in the BERT embeddings during the fine-tuning process for semantic textual similarity. As the model adapts to focus more on capturing semantic relationships between sentences, it may lose some of its ability to accurately represent features relevant to other tasks. This trade-off highlights the importance of carefully designing fine-tuning strategies to balance performance across multiple tasks.

6.2 Error Analysis

Error analysis provides valuable insights into the limitations of our models and potential areas for improvement. By analyzing misclassifications, we identified common error patterns. For instance, in paraphrase detection, our model occasionally failed to recognize subtle semantic differences between sentence pairs, leading to incorrect predictions. This indicates the need for more nuanced feature representations and finer-grained modeling of semantic similarity.

6.3 Selected Examples

Examining selected examples helps us understand how our models perform on specific instances. We manually inspected predictions made by our models across different tasks to identify patterns and instances of success or failure. For example, in sentiment analysis, our model often accurately classified straightforward reviews with strong sentiment expressions. However, it struggled with ambiguous or sarcastic statements, where context played a significant role in determining sentiment polarity. This provided us with ideas for potential future preprocessing on inputs to help erase ambiguity in the analysis, such as the manipulation of word tokens to account for negation and other complexities.

7 Conclusion

In this project, we set out to enhance the performance of BERT on specific NLP tasks through targeted fine-tuning techniques. We began by exploring domain-specific pre-training as a means of improving BERT's performance, but found it to be computationally intensive with limited gains. Instead, we shifted our focus to fine-tuning strategies, where we experimented with both task-specific and multi-task fine-tuning approaches.

Through our experiments, we found that task-specific fine-tuning generally outperformed multi-task fine-tuning, delivering superior results across sentiment classification, paraphrase detection, and semantic textual similarity tasks. Individually fine-tuned models exhibited task-specific performance improvements, while the multi-task fine-tuned model struggled with overfitting and failed to generalize effectively across tasks.

Our analysis revealed insights into the behavior and limitations of our models. Error analysis highlighted areas for improvement, such as the need for more nuanced feature representations in paraphrase detection. We also observed trade-offs in performance across tasks, particularly when fine-tuning for semantic textual similarity, which emphasized the importance of carefully designing fine-tuning strategies to balance performance.

Overall, our project contributes to the growing body of research on fine-tuning pre-trained language models for specialized NLP tasks. Our findings underscore the importance of task-specific optimization and the need for continued research into effective fine-tuning strategies to maximize performance and generalization across diverse NLP applications.

7.1 Future Work

Moving forward, several avenues of research could further enhance the effectiveness and applicability of fine-tuning techniques for pre-trained language models. One direction is to explore more sophisticated multi-task learning approaches that mitigate overfitting and improve generalization across tasks. In the context of our own experiments, we could explore multi-task fine-tuning with a portion of the tasks (2 out of 3 tasks), rather than only fine-tuning on 1 task or all of the tasks at once. Additionally, investigating novel fine-tuning strategies, such as curriculum learning or adaptive regularization, could lead to better performance and robustness on diverse NLP tasks.

8 Ethics Statement

Both the pre-training and fine-tuning of language models bring about important ethical considerations as we go about examining them. One of our initial goals, implementing additional domain-specific pre-training for the tasks at hand, proved to be computationally expensive while not yielding the results we had hoped for. This became immediately evident with only three tasks at hand. On a much larger scale, as mentioned by Strubell et al. (2019), running additional pretraining for the many NLP tasks that exist in the world can have detrimental effects on the environment, along with the fiscal issues that may arise from allocating the necessary funds to a risky pretraining and possibly taking away advanced research opportunities from more financially restricted researchers. To mitigate these issues, one possible strategy is to create a policy that ensures researchers prioritize efficiency in their training techniques. There should be a board that analyzes the risk and reward associated with high-cost training endeavors. This board could establish guidelines for the responsible use of computational resources, encouraging the adoption of energy-efficient practices and promoting the sharing of pre-trained models to reduce redundant efforts. Additionally, investing in and promoting research into more efficient algorithms and hardware can help mitigate the environmental impact and lower the financial barriers to advanced research.

Additionally, as we add layers and create alternative processes for our model to fine-tune to a specific task, we must also consider the opacity of the models we develop. While tasks like categorizing movie ratings and question pairs may seem trivial at first glance, it is not unlikely that similar models will be deployed in more critical or sensitive applications (healthcare, finance, law enforcement, etc.). It is important that we provide transparency as to how our models make decisions, as human oversight is essential for ensuring accountability and trustworthiness in AI systems. To achieve this, we suggest that developers and researchers prioritize model interpretability features as we deploy these models. For example, feature importance methods can help identify which input variables have the most influence on the model's predictions, helping to demystify how models arrive at their decisions.

References

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *Proceedings of the 29th International Conference on Neural Information Processing Systems*, pages 3079–3087. MIT Press.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339. Association for Computational Linguistics.

Manish Munikar, Sushil Shakya, and Aakash Shrestha. 2019. Fine-grained sentiment classification using bert.

Abhishek Panigrahi, Nikunj Saunshi, Haoyu Zhao, and Sanjeev Arora. 2023. Task-specific skill localization in fine-tuned language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, page 3982–3992.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650. Association for Computational Linguistics.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune BERT for text classification? In *Chinese Computational Linguistics: 18th China National Conference*.