minBERT and Multiple Downstream Tasks

Stanford CS224N Default Final Project

Xianling Zhang
lilyzhng@stanford.edu

Abstract

BERT (Bidirectional Encoder Representations from Transformers) [1] has significantly advanced natural language processing (NLP) by leveraging transformer architecture to learn contextualized word representations from large amounts of text data. In this paper, we present the minBERT implementation that simultaneously performs three tasks: sentiment analysis [2], paraphrase detection [3], and semantic textual similarity (STS) [4]. Rather than fine-tuning BERT on individual tasks, we leverage multi-task learning to update BERT model, which has been shown to improve performance (Bi et al. [5]). We also apply hyperparameter optimization to further improve the model's performance. Our results demonstrate the effectiveness of minBERT in achieving compelling performance on all three tasks, while also providing insights into the optimal hyperparameters for each task.

1. Introduction

Sentiment analysis, paraphrase detection, and semantic textual similarity are important natural language processing tasks that have applications in various domains. Specifically, sentiment analysis task is aimed to understand a given text is classifying its polarity; in a large corpus of passages, paraphrase detection essentially seeks to determine whether particular words or phrase convey the same semantic meaning; The semantic textual similarity (STS) task is aimed to measure the degree of equivalence given input texts. These tasks have been traditionally addressed using separate models trained independently. However, recent advances in multitask learning have shown that jointly training models for related tasks can lead to better performance and faster convergence.

Joint training of multiple related tasks is a powerful approach to improve the performance of machine learning models. However, it is not clear whether joint training is always better than separate training, especially for tasks that have different objectives and evaluation metrics. For paraphrase detection and semantic textural similarity task, they share similar intent of seeking for semantic likelihood in given texts. What differentiate STS from paraphrasing task is that instead of providing a yes or no answer, STS rather uses degrees of similarity. Among all three tasks, sentiment analysis is the one of distinct objectives comparing to the rest. In this paper, we compare the performance of joint and separate training approaches for sentiment analysis.

2. Related Work

The BERT model has been pretrained using a large amount of unlabeled text data (sourced from various publicly available sources such as Wikipedia, Common Crawl [6], and Book Corpus [7]). The input text was tokenized into subword units using WordPiece [8] tokenization. As depicted in Figure 1, The input embeddings used in the model are the sum of the token embeddings, the segmentation embeddings, and the positional embeddings. The token embeddings map the individual input ids into vector representations. The learnable segment embeddings are utilized to decide whether a word belongs to sentence A or B. Lastly, the positional embeddings are used to encode the position of different words within the input.

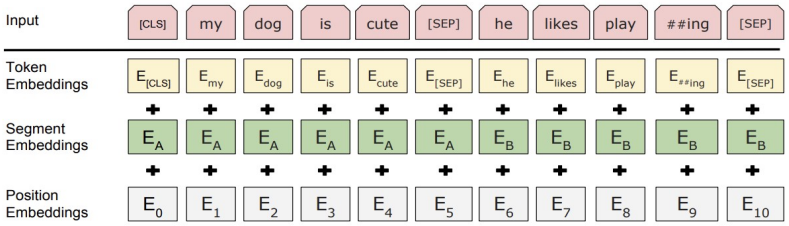


Figure 1: BERT embedding layer.

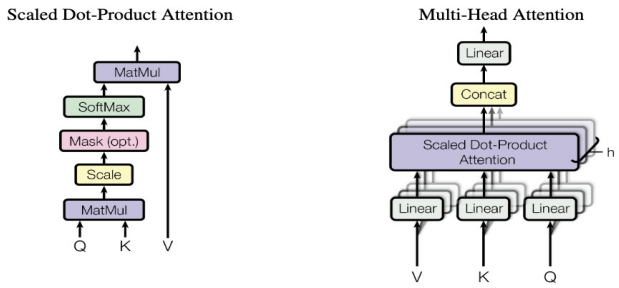


Figure 2: Scaled Dot-Product and Multi-Head Self-Attention. Figure from [1]

As seen in Figure 2, The transformer layer of the BERT transformer consists of multi-head attention, followed by an additive and normalization layer with a residual connection, a feed-forward layer, and a final additive and normalization layer with a residual connection. By attending to different parts of the input sequence, multi-head self-attention allows BERT to capture both local and global dependencies between words, while also being able to distinguish between different types of relationships. This is important for tasks such as sentiment analysis, where understanding the context and meaning of a sentence is critical to accurate predictions.

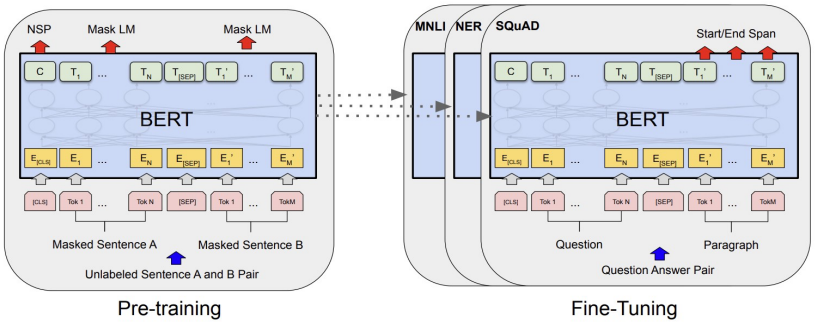


Figure 3: The original BERT model was trained on two unsupervised tasks, masked token prediction and next sentence prediction. Figure from [9].

Pre-Training: The BERT pre-training process involves training the model on a large amount of unlabeled text data in a self-supervised manner. During pre-training, BERT uses a masked language model (MLM) and next sentence prediction (NSP) tasks to learn contextualized word representations. In the MLM task, some of the input tokens are randomly masked, and the model learns to predict the original token from the masked context. In the NSP task, the model predicts whether two input sentences are consecutive or not. By training on these tasks, BERT learns to generate high-quality, context-aware representations of words and sentences.

Fine-Tuning: After pre-training, BERT can be fine-tuned on a variety of downstream NLP tasks. During fine-tuning, BERT is adapted to the specific task by training it on a labeled dataset. The weights of the pre-trained model are frozen, and only the task-specific layers are trained on the labeled data. This approach allows the model to quickly adapt to new tasks with only a small amount of task-specific training data.

3. Approach

In this section, we explain the model's architecture. We use the same tokenization and architecture as the original BERT model. We add task-specific classification layers on top of the shared BERT layers to predict sentiment labels, paraphrase pairs, and semantic texture similarity scores. We jointly train the model using a multi-task loss function that combines the individual task loss functions. We also perform hyperparameter optimization to identify the optimal learning rate, batch size, and regularization strength for each task.

3.1 Baseline Model

As indicated in the final project handout [11] and presented in the Figure 1, the baseline sentiment classifier models are reported to have accuracy of 0.390 and 0.515 for SST dataset using pretraining and finetuning. Respectively, for CFIMDB [12], the baseline models have 0.780 and 0.966 accuracy on dev dataset. Since there is no baseline model provided for the multitask BERT model for jointly training sentiment analysis, paraphrase detection, and semantic textual similarity, this paper will report the model performance using pretraining and finetuning options, with and without hyperparameters optimization.

3.2 minBERT Sentiment Classifier

After implementing the minBERT model, we performed sentiment analysis on two datasets: the Stanford Sentiment Treebank (SST) dataset and the CFIMDB dataset. The BERT model was utilized to encode the sentences to obtain contextualized representations for the sentence classification task. The final step was to fine-tune the BERT model for the downstream sentence classification task.

3.3 Adam Optimizer

In addition, the Adam optimizer has been implemented based on Decoupled Weight Decay regularization and Stochastic Optimization. The Adam optimizer is a very helpful component in model training because it calculates adaptive learning rates with different parameters such as batch size and the number of GPUs available. The learning rate or step size will be updated throughout the training. With greater updates, the model can be tuned to speed up or slow down the gradient descent process, based on the magnitude of the gradients. It can help the learning rate to give quicker updates to those of small gradients, thus contributing to quicker convergence. The adaptive learning rate is one essential differentiator between Adam and classical SDG, and the latter uses a single learning rate for all weights, and this learning rate does not change during training time. In comparison, Adam adjusts the learning rate based on the square root of the gradients in real-time.

3.4 Multitask minBERT

The BERT model has been expanded to carry out three tasks concurrently, namely sentiment analysis, paraphrase detection, and semantic textual similarity measurement, rather than being restricted to a single task. The objective of the Multitask BERT model is to optimize the use of BERT embeddings and deliver high performance across a wide variety of tasks.

Upon importing the weights of a pretrained BERT model, our approach entails predicting the sentiment score of sentences, verifying the paraphrasing of sentence pairs, and measuring the similarity between the two input texts. Notably, the experiments conducted with our model solely leverage the pooler output. In addition, considering that different tasks have different requirements for regularization, we also specialize different ratio to dropping out neurons for different tasks at training time to prevent overfitting.

During training, we cycle the data for each time, record individual task's token IDs, attention masks, and labels. The predicted output for each task is unnormalized (a logit) and undergoes normalization via a sigmoid function. The sentiment analysis task utilizes cross entropy loss[13], whereas the paraphrase detection task relies on binary cross entropy loss [14]. Textual similarity measurement employs mean squared error (squared L2 norm) [15]. Lastly, leveraging the use of multi-task learning to update BERT, we add each loss together to update the model with the averaged loss from all three tasks. To retain the best model, we use the average dev metric, only overwriting the old model with the newly saved best model if it achieves better average dev accuracy.

3.4 Hyperparameter Tuning

In order to enhance the overall performance of the model, a series of comprehensive experiments were carried out to evaluate the impact of various hyperparameters and derive meaningful comparisons. The evaluation involved exploring different values of dropout, batch size, and learning rates for both individually trained tasks and multitask BERT models. Despite the time-consuming and computationally intensive nature of this hyperparameter optimization study, significant improvements in accuracy were observed across all three tasks. These findings underscore the importance of carefully selecting and tuning hyperparameters to achieve optimal performance in complex machine learning models.

4. Experiments

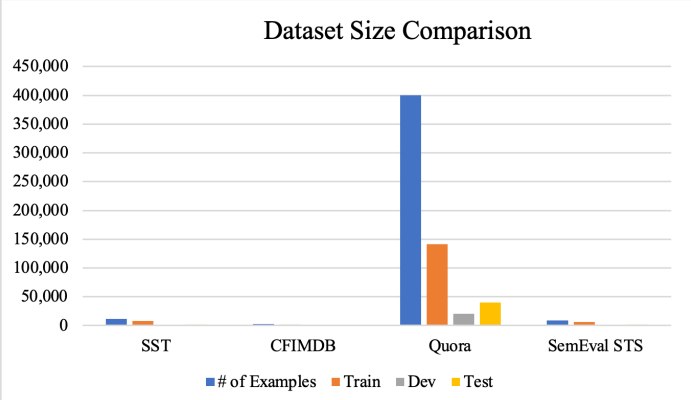
4.1 Datasets

For the separately trained sentiment analysis task, both Stanford Sentiment Treebank (SST) dataset and CFIMDB dataset have been used. However, for the multitask training, the SST dataset was employed for sentiment analysis, while the Quora dataset was used for paraphrase detection, and the SemEval STS Benchmark Dataset was utilized for semantic textual similarity measurement. As shown in Figure 4, a dataset size comparison between the 4 datasets have been presented.

- The SST dataset is comprised of 11,855 single sentences extracted from movie reviews, with a total of 215,154 unique phrases. Each phrase is labeled as negative, somewhat negative, neutral, somewhat positive, or positive. The dataset is divided into train (8,544 examples), dev (1,101 examples), and test (2,210 examples) splits.
- The CFIMDB dataset, on the other hand, contains 2,434 highly polar movie reviews with binary labels of either negative or positive. The dataset is relatively small compared to the SST dataset, with

train (1,701 examples), dev (245 examples), and test (488 examples) splits. Models trained with small-scale datasets are at a higher risk of overfitting.

- The Quora dataset[16] is a diverse collection of 400,000 question pairs with labels indicating whether the instances are paraphrases of each other. The dataset is divided into train (141,506 examples), dev (20,215 examples), and test (40,431 examples) splits.
- The SemEval STS Benchmark Dataset [17] is composed of 8,628 sentence pairs with varying degrees of similarity, ranging from 0 (unrelated) to 5 (equivalent meaning). The dataset is divided into train (6,041 examples), dev (864 examples), and test (1,726 examples) splits.



Dataset	# of Examples	Train	Dev	Test
SST	11,855	8,544	1,101	2,210
CFIMDB	2,434	1,701	245	488
Quora	400,000	141,506	20,215	40,431
SemEval STS	8,628	6,041	864	1,726

Figure 4. Dataset sizes comparison for SST, CFIMDB, Quora, STS for train, dev, and test splits.

4.2 Evaluation Metrics

The SST and CFIMDB datasets employ accuracy as the evaluation metric, which quantifies the proportion of correctly classified instances. The Quora dataset utilizes the F1 score as the evaluation metric, which assesses the harmonic mean of precision and recall. The SemEval STS Benchmark dataset utilizes the Pearson correlation coefficient as the evaluation metric, which gauges the linear association between the predicted similarity scores and the actual similarity scores. In the context of selecting the optimal model for the multitask minBERT model, the average dev/test metric is used.

4.3 Experiments Details

4.3.1 Different Dropout Ratio

minBERT Sentiment Analysis (Separately Trained), Pretraining							
SST-SST Dataset				SST-CFIMDB Dataset			
Model Name	Option	Dropout	Dev Acc	Model Name	Option	Dropout	Dev Acc
Baseline	pretrain	0.3	0.390	Baseline	pretrain	0.3	0.780
Model SSP-D0.2	pretrain	0.2	**0.424**	Model CFP-D0.2	pretrain	0.2	0.804
Model SSP-D0.3	pretrain	0.3	0.422	Model CFP-D0.3	pretrain	0.3	**0.824**
Model SSP-D0.4	pretrain	0.4	0.396	Model CFP-D0.4	pretrain	0.4	0.784
Model SSP-D0.5	pretrain	0.5	0.402	Model CFP-D0.5	pretrain	0.5	0.792
Model SSP-D0.6	pretrain	0.6	0.391	Model CFP-D0.6	pretrain	0.6	0.784

Table 1. For separately trained minBERT sentiment analysis classifier. Different dropout ratios (0.2 – 0.6) have been applied at training time using the pretraining option. Best model obtained with hidden_dropout_prob = 0.2 on SST dataset, and hidden_dropout_prob = 0.3 on CFIMDB dataset.

minBERT Sentiment Analysis (Separately Trained), Finetuning							
SST-SST Dataset				SST-CFIMDB Dataset			
Model Name	**Option**	**Dropout**	**Dev Acc**	**Model Name**	**Option**	**Dropout**	**Dev Acc**
Baseline	finetune	0.3	0.515	Baseline	finetune	0.3	0.966
Model SSF-D0.2	finetune	0.2	0.517	Model CFF-D0.2	finetune	0.2	0.963
Model SSF-D0.3	finetune	0.3	0.530	Model CFF-D0.3	finetune	0.3	0.967
Model SSF-D0.4	finetune	0.4	0.494	Model CFF-D0.4	finetune	0.4	**0.976**
Model SSF-D0.5	finetune	0.5	0.528	Model CFF-D0.5	finetune	0.5	0.963
Model SSF-D0.6	finetune	0.6	**0.534**	Model CFF-D0.6	finetune	0.6	0.967

Table 2. For separately trained minBERT sentiment analysis classifier. Different dropout ratios have been applied at training time using the finetuning option. Best model obtained with hidden_dropout_prob = 0.6 on SST dataset, and hidden_dropout_prob = 0.4 on CFIMDB dataset.

Regularization techniques play a crucial role in preventing overfitting and improving the generalization ability of machine learning models. The optimal regularization parameters, such as the dropout ratio, vary depending on the specific task at hand. Therefore, we employ task-specific dropout ratios during training to enhance model performance. In the context of sentiment analysis using minBERT, we train separate classifiers for each dataset and experiment with different dropout ratios. For pretraining, we found that the best model for the SST dataset had a hidden_dropout_prob of 0.2, while for the CFIMDB dataset, the best ratio was 0.3. On the other hand, for finetuning, the best ratio was 0.6 for the SST dataset and 0.4 for the CFIMDB dataset. These findings highlight the importance of tuning the dropout ratio according to the specific dataset and training method. By doing so, we can improve the performance and robustness of the sentiment analysis classifier, thereby enabling it to generalize better to new data.

4.3.2 Different Batch Sizes

minBERT Sentiment Analysis (Separately Trained), Pretraining							
SST-SST Dataset				SST-CFIMDB Dataset			
Model Name	**Option**	**Batch size**	**Dev Acc**	**Model Name**	**Option**	**Batch size**	**Dev Acc**
Baseline	pretrain	8	0.390	Baseline	pretrain	8	0.78
Model SSP-D0.2-B4	pretrain	4	0.423	Model CFP-D0.3-B4	pretrain	4	**0.824**
Model SSP-D0.2-B8	pretrain	8	**0.424**	Model CFP-D0.3-B8	pretrain	8	**0.824**
Model SSP-D0.2-B16	pretrain	16	0.405	Model CFP-D0.3-B16	pretrain	16	0.800
Model SSP-D0.2-B32	pretrain	24	0.403	Model CFP-D0.3-B24	pretrain	24	0.792
Model SSP-D0.2-B32	pretrain	32	0.409	Model CFP-D0.3-B32	pretrain	32	0.792
Model SSP-D0.2-B32	pretrain	64	0.411	Model CFP-D0.3-B64	pretrain	64	0.788

Table 3. With hidden_dropout_prob = 0.2 fixed on SST dataset, and hidden_dropout_prob = 0.3 fixed on CFIMDB dataset, different batch sizes (4-64) have been experimented. Best model for SST dataset obtained using default batch size 8. Best model for CFIMDB dataset obtained using default batch size 8 or smaller batch size of 4.

minBERT Sentiment Analysis (Separately Trained), Finetuning							
SST-SST Dataset				SST-CFIMDB Dataset			
Model Name	**Option**	**Batch size**	**Dev Acc**	**Model Name**	**Option**	**Batch size**	**Dev Acc**
Baseline	finetune	8	0.515	Baseline	finetune	8	0.966
Model SSP-D0.2-B4	finetune	4	0.530	Model CFP-D0.3-B4	finetune	4	0.967
Model SSP-D0.2-B8	finetune	8	**0.534**	Model CFP-D0.3-B8	finetune	8	**0.976**
Model SSP-D0.2-B16	finetune	16	0.512	Model CFP-D0.3-B16	finetune	16	0.963
Model SSP-D0.2-B32	finetune	24	0.519	Model CFP-D0.3-B24	finetune	24	0.955
Model SSP-D0.2-B32	finetune	32	0.523	Model CFP-D0.3-B32	finetune	32	0.959
Model SSP-D0.2-B32	finetune	64	0.527	Model CFP-D0.3-B64	finetune	64	0.959

Table 4. With hidden_dropout_prob = 0.6 fixed on SST dataset, and hidden_dropout_prob = 0.4 fixed on CFIMDB dataset, different batch sizes (4-64) have been experimented. Best model for SST dataset obtained using default batch size 8. Best model for CFIMDB dataset obtained using default batch size 8.

In this study, we examined the impact of varying batch sizes on the performance of the SST task during pretraining and finetuning. For pretraining, we fixed the hidden_dropout_prob to 0.2 for SST and 0.3 for CFIMDB and experimented with batch sizes ranging from 4 to 64. The best performing model for the SST dataset was obtained using the default batch size of 8, while the best model for the CFIMDB dataset was obtained using either the default batch size of 8 or a smaller batch size of 4. For finetuning, we fixed the hidden_dropout_prob to 0.6 for SST and 0.4 for CFIMDB, and again experimented with batch sizes ranging from 4 to 64. The best performing model for both SST dataset and CFIMDB was obtained using the default batch size of 8. Our results suggest that varying batch sizes during pretraining and finetuning did not have a significant impact on the model's performance on the SST task.

4.3.3 Different Learning Rate

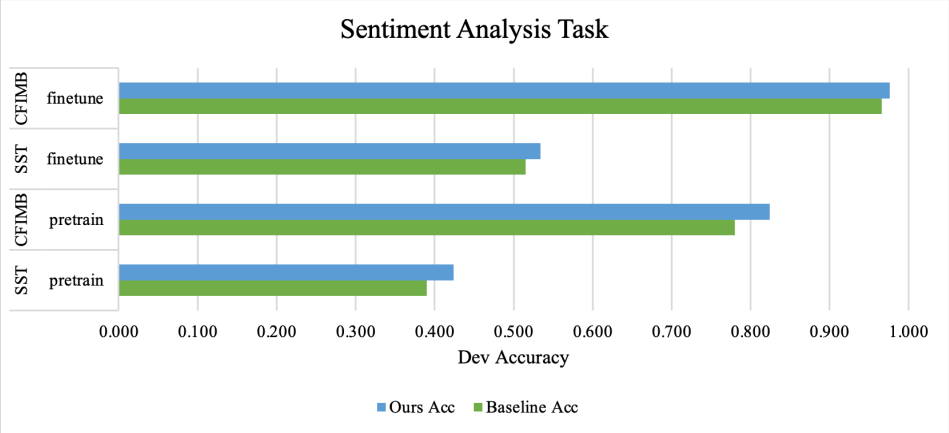
minBERT Multitask Training, Finetuning						
Model Name	**Option**	**Learning Rate**	**SST**	**PD**	**STS**	**Average**
Baseline	finetune	1.00E-05	0.514	0.859	0.847	0.740
Model MSSF-B-L1	finetune	5.00E-06	0.504	0.873	0.841	0.739
Model MSSF-B-L2	finetune	5.00E-05	0.465	0.825	0.841	0.710
Model MSSF-B-L3	finetune	5.00E-04	0.262	0.625	0.061	0.316
Model MSSF-B-L4	finetune	1.00E-04	0.253	0.625	0.076	0.318
Model MSSF-B-L5	finetune	1.00E-03	0.253	0.625	0.052	0.310

Table 5. With best sst_hidden_dropout_prob, para_ hidden_dropout_prob, and sts_ hidden_dropout_prob and batch sizes fixed for each task, different learning rate has been explored.

The study of best hypermeters have also been conducted for multitask minBERT of 3 tasks. Table 5 is one example of tuning learning rate on minBERT multitask trainig. Note that the default learning rate is 1e-05 for finetuning, and this is the rate that model achieved best performance.

5 Performance Summarization

5.1 minBERT Sentiment Analysis Best Model



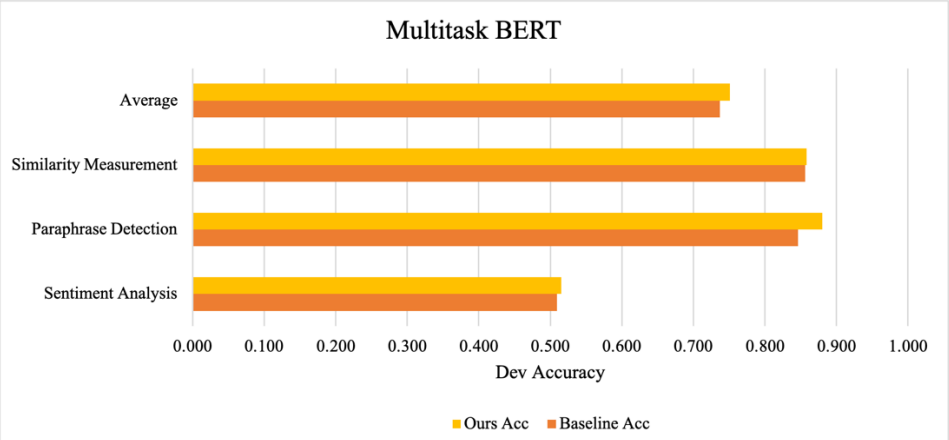
Sentiment Analysis Task			
Dataset	**Option**	**Model Name | Baseline Acc**	**Model Name | Ours Acc**
SST	pretrain	Model SSP-A | 0.390	Model SSP-B | **0.424**
CFIMB	pretrain	Model CFP-A | 0.780	Model CFP-B | **0.824**
SST	finetune	Model SSF-A | 0.515	Model SSF-B | **0.534**
CFIMB	finetune	Model CFF-A | 0.966	Model CFF-B | **0.976**

Figure 4. minBERT Sentiment Classifier model performance comparing against baseline models.

For minBERT sentiment classifier model, the baseline model results are reported in the default final project handout. Both methods are evaluated on the SST/CFIMDB dev datasets respectively. The baseline model names are denoted as SSP-A, CFP-A, SSF-A, CFF-A. A – baseline type, SSP – pretraining on SST dataset, CFP- pretraining on CFIMB dataset, SSF - finetuning on SST dataset, CFF - finetuning on CFIMB dataset. Our optimized models are denoted as SSP-B, CFP-B, SSF-B, CFF-B. B – represents our model of optimization. As highlighted in blue and our method presents better accuracy comparing against the baseline model performance (see Figure 4). Note that all models trained for 20 epochs, and the best model is using the below parameters:

- batch size 8
- learning rate 1e-05
- dropout 0.6 on SST, 0.4 on CFIMB datasets.

5.2 minBERT MultiTask Best Model on Dev



Multitask BERT				
Task	**Dataset**	**Option**	**Model Name | Baseline Acc**	**Model Name | Ours Acc**
Sentiment Analysis	SST	finetune	Model MSSF-A | 0.509	Model MSSF-B | **0.515**
Paraphrase Detection	Quora	finetune	Model MPDF-A | 0.846	Model MPDF-B | **0.88**
Similarity Measurement	SemEval STS	finetune	Model MSTF-A | 0.856	Model MSTF-B | **0.858**
Average	Average	finetune	Model MAVF-A | 0.737	Model MAVF-B | **0.751**

Figure 5. Multitask minBERT model performance with or without hypermeter optimization.

As there are no baseline model results reported, thus we report the model performance before and after regularization and hyperparameter optimizations. Without optimization, our model names are denoted as MSSF-A, MPDF-A, MSTF-A, MAVF-A. M - multitask, SS - sentiment analysis, PD-paraphrase detection, ST-textual similarity measurement, AV – average, F – finetune, A – baseline type. With optimization, our models are denoted as MSSF-B, MPDF-B, MSTF-B, MAVF-B. B represent our model of optimization. As highlighted in Figure 5, our model with optimization achieved better performance on all 3 tasks. Note that all models trained for 20 epochs, the best model is obtained using the following parameters:

- learning rate 1e-05 for all 3 tasks
- batch sizes 4 : 64 : 4 for SST, PD, STS
- dropout prob 0.5, 0.6, 0.5 for SST, PD, STS

5.3 minBERT MultiTask Dev and Test Acc Comparison

Multitask BERT					
Task	**Dataset**	**Option**	**Model Name**	**Dev Acc**	**Test Acc**
Sentiment Analysis	SST	finetune	Model MSSF-B	0.515	0.518
Paraphrase Detection	Quora	finetune	Model MPDF-B	0.880	0.875
Similarity Measurement	SemEval STS	finetune	Model MSTF-B	0.858	0.861
Average	Average	finetune	Model MAVF-B	0.751	0.752

Table 6. minBERT multitask dev and test accuracy for the best model.

The performance of a model on both the dev and test data is crucial in assessing its ability to effectively solve the given problem. The dev data is used during the model development phase for hyperparameter tuning and to evaluate the model's performance on a subset of the available data. In contrast, the test data is used to evaluate the performance of the final model on unseen data, which provides an estimate of its true generalization ability.

In this paper, we present the results of our analysis by comparing the dev and test accuracy of our model with the best performing model. Our findings demonstrate that the test accuracy achieved by our model is marginally higher than the dev accuracy. This observation is indicative of the fact that the model is able to effectively generalize and perform well when presented with previously unseen data. The model's performance on the test data is a reliable indicator of its generalization ability, as it is able to effectively adapt and perform well when presented with new data. This implies that the model is able to capture the underlying patterns in the data and is not overfitting to the dev data.

6 Limitation

It is important to note that while our optimization focused on batch size and dropout tuning, other hyperparameters, such as learning rate and optimizer choice, may also influence model performance. Further research is needed to investigate the combined effect of various hyperparameters on model performance for these datasets.

7 Conclusion

There are several potential areas for future work building on the findings of this study. One avenue for future research is to explore the effectiveness of minBERT on other NLP tasks, such as named entity recognition or machine translation. Additionally, further investigation could be conducted on how to optimize model performance for multiple tasks simultaneously in a more efficient manner with reduce compute cost.

In conclusion, this paper presents minBERT, a multi-task learning approach to perform sentiment analysis, paraphrase detection, and semantic textual similarity tasks. The study demonstrates that leveraging multi-task learning with hyperparameter optimization can improve the performance of the BERT model on multiple tasks. The results show that minBERT achieves compelling performance on all three tasks and provides insights into the optimal hyperparameters for each task. This study contributes to the growing body of research on multi-task learning in NLP and provides a useful framework for developing more efficient and effective NLP models.

References

[1] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

[2] Medhat, Walaa, Ahmed Hassan, and Hoda Korashy. "Sentiment analysis algorithms and applications: A survey." *Ain Shams engineering journal* 5.4 (2014): 1093-1113.

[3] Fernando, Samuel, and Mark Stevenson. "A semantic similarity approach to paraphrase detection." *Proceedings of the 11th annual research colloquium of the UK special interest group for computational linguistics*. 2008.

[4] Han, Lushan, et al. "UMBC_EBIQUITY-CORE: Semantic textual similarity systems." *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*. 2013.

[5] Bi, Qiwei, et al. "Mtrec: Multi-task learning over bert for news recommendation." *Findings of the Association for Computational Linguistics: ACL 2022*. 2022.

[6] Smith, Jason R., et al. "Dirt cheap web-scale parallel text from the common crawl." Association for Computational Linguistics, 2013.

[7] McEnery, Tony, Richard Xiao, and Yukio Tono. *Corpus-based language studies: An advanced resource book*. Taylor & Francis, 2006.

[8] Song, Xinying, et al. "Fast wordpiece tokenization." *arXiv preprint arXiv:2012.15524* (2020).

[9] Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

[11] Stanford CS 224n, final project hand out, 2023, https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1204/reports/default/report25.pdf

[12] Choi, Seungtaek, et al. "C2l: Causally contrastive learning for robust text classification." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. No. 10. 2022.

[13] Zhang, Zhilu, and Mert Sabuncu. "Generalized cross entropy loss for training deep neural networks with noisy labels." *Advances in neural information processing systems* 31 (2018).

[14] Ruby, Usha, and Vamsidhar Yendapalli. "Binary cross entropy with deep learning technique for image classification." *Int. J. Adv. Trends Comput. Sci. Eng* 9.10 (2020).

[15] Das, Kalyan, Jiming Jiang, and J. N. K. Rao. "Mean squared error of empirical predictor." (2004)

[16] Sharma, Lakshay, et al. "Natural language understanding with the quora question pairs dataset." *arXiv preprint arXiv:1907.01041* (2019).

[17] Shao, Yang. "Hcti at semeval-2017 task 1: Use convolutional neural network to evaluate semantic textual similarity." *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. 2017.