BERT's Mean Teacher and Multitask Fine-Tuning

Stanford CS224N Default Project

Anthony Qin
Department of Computer Science
Stanford University
antqin27@stanford.edu

Kevin Tran
Department of Computer Science
Stanford University
ktran25@stanford.edu

Abstract

This project explores different approaches for improving the performance of BERT on three downstream tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. We start by implementing a minimalist version of BERT as our baseline, and then fine-tuning and extending the model to create sentence embeddings that perform well across all three tasks simultaneously. We explore approaches such as balanced round-robin fine tuning, mean teacher algorithm, and adjusting hyperparameters such as loss, batch size, and learning rate decay. Our experimental results show a significant performance increase over the baseline, demonstrating the effectiveness of our approach and its potential applications in various natural language processing tasks.

1 Key Information to include

- Mentor: Candice Penelton
- External Collaborators (if you have any): n/a
- Sharing project: n/a

2 Introduction

In this project, we tackle the NLP problem of building a model to simultaneously perform well on not just one but multiple sentence-level tasks, namely sentiment analysis, paraphrase detection, and semantic textual similarity. We achieved this by implementing important components the BERT model (Devlin et al., 2018) and extending upon it to achieve improved results across all three tasks.

For sentiment analysis, our goal was to classify a text's polarity on a range from 0 (Negative) to 4 (Positive). For paraphrase detection, our goal was to find paraphrases of texts in a large corpus of passages, specifically whether two questions were paraphrases of each other. For semantic textual similarity, our goal was to capture the notion that some texts are more similar than others by evaluating it on a scale from 5 (same meaning) to 0 (not at all related).

Using pre-trained weights loaded into our BERT model, we first performed sentence classification on the Stanford Sentiment Treebank (SST) dataset and the CFIMDB dataset. Then, we experimented with multi-task finetuning BERT's contextualized embeddings to simultaneously perform well on the three aforementioned multiple sentence-level tasks using the SST, Quora, and SemEval STS Benchmark Datasets. We experimented with in-domain and cross-domain fine-tuning by training on each dataset individually then in an unbalanced as well as balanced round-robin style. We experimented with different loss functions on each task and with varying hyperparameters. We also implemented the mean teacher algorithm to address overfitting in our multitask BERT model, where the teacher model's parameters are an exponential moving average of the student model's weights, and the consistency loss is calculated using Bregman Proximal Point Optimization with either mean squared error or KL divergence. (Tarvainen and Valpola, 2018). Among all these approaches, we found a batch size of 8, balanced round-robin style with Mean Teacher, cross-entropy

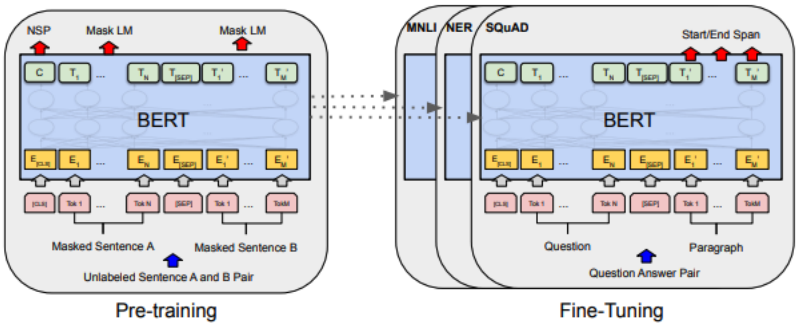

Figure 1: Overall pre-training and fine-tuning procedures for BERT. Figure from (Devlin et al., 2018)

loss on sentiment analysis and paraphrase detection, and a MSE loss on semantic textual similarity to significantly improve model performance across the three downstream tasks.

3 Related Work

Bidirectional Encoder Representations from Transformers (BERT) is a transformer-based model that generates contextual word representations (Devlin et al., 2018). Released in 2018, BERT took a large leap forward for contextual word embeddings, large language models, and foundational models by making use of deeply bidirectional word representations and a transformer backbone. Unlike previous language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers, allowing the model to be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of downstream tasks, three of which we explored in this project: sentiment analysis, paraphrase detection, and semantic textual similarity. It uses a masked language model (MLM) pre-training objective that enables the representation to fuse the left and the right context as well as a next sentence prediction task that jointly pretrains text-pair representations.

Our work is also inspired by Sun et al. (pages 194–206. Springer, 2019) and Jiang et al. (2019). Sun et al. (pages 194–206. Springer, 2019) investigated how to maximize the utilization of BERT for the text classification task and explored several ways of fine-tuning BERT to enhance its performance, from which we drew inspiration. Tarvainen and Valpola (2018) proposes a new method for semi-supervised deep learning called Mean Teacher, which improves upon a previous method called Temporal Ensembling. Mean Teacher uses two models: a student model that is trained on both labeled and unlabeled data, and a teacher model that is an exponential moving average of the student model's weights. The paper shows that using the teacher model as a target for consistency regularization improves the performance of the student model, especially when there are few labeled examples available.

4 Approach

4.1 minBERT

Our baseline is Devlin et al. (2018)'s BERT, much of it provided by the CS224N staff. We implemented the Multi-head Self-Attention, the Transformer Layer, and the Adam Optimizer. For more details on the architecture, we refer the reader to the default project handout. We trained minBERT on the SST and the CFIMDB datasets using both pre-trained and fine-tuned embeddings for sentiment classification. Our results are as follows:

- Pretraining for SST: Dev Accuracy: 0.392
- Pretraining for CFIMDB: Dev Accuracy: 0.788
- Finetuning for SST: Dev Accuracy: 0.511
- Finetuning for CFIMDB: Dev Accuracy: 0.976

4.2 Single-task Fine-Tuning

As a first attempt to improve our BERT baseline on multiple downstream tasks, we first tried fine-tuning our model on each dataset individually: SST, Quora, and SemEval.

For fine-tuning on the SST and Quora datasets, we used Cross Entropy loss as sentiment analysis and paraphrase detection are classification problems. We chose to use MSE loss for fine-tuning on the SemEval dataset because semantic textual similarity is a regression problem with a continuous variable output.

4.3 Round-robin Multi-task Fine-Tuning

Next, we implemented round-robin multitask fine-tuning to fine tune our model on all three datasets at once. We used the same losses: Cross Entropy loss for SST and Quora, and MSE loss for SemEval. Since the size of each dataset are different, we experimented with multiple different architectures.

First, we attempted to use the entirety of each dataset in our training, meaning that after one dataset is exhausted, the epoch would not complete until the largest dataset is also exhausted. We experimented with varying batch sizes for each dataset, as the batch size didn't matter since each dataset would be ran to completion.

Then, we experimented with using balanced round-robin multitask fine-tuning by setting the size of each dataset to the size of the smallest datsaset, SemEval, and taking a random sample of that size from each dataset. We used the same batch size across all three datasets in this scenario (either size 8 or 32). This way, the model is trained evenly across all three datasets and doesn't favor the bigger datasets/tasks over the others.

As we experimented with balanced round-robin multitask fine-tuning, we tried different ways of keeping track of the loss and gradient descent. Initially, we summed together all three of the task losses and took one big step afterwards for each batch. Then, after changing the STS loss from Cross Entropy to MSE, we adjusted our update to take a small step after every task, so we took three small steps instead of one big step for each batch (using a specialized loss of Cross Entropy for SST and Paraphrase, MSE loss for STS).

4.4 Mean Teacher

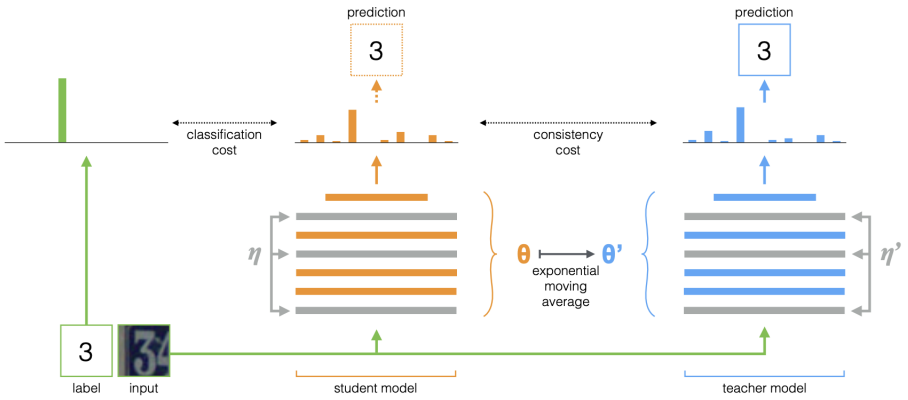

Figure 2: Illustration of the Mean Teacher Algorithm, showing both supervised classification and unsupervised consistency losses. Figure from (Tarvainen and Valpola, 2018)

In our project milestone, we faced a challenge of overfitting, where our multitask BERT model was performing well on the training data but not on the test data. To address this issue, we implemented an extension using the mean teacher algorithm.

In our mean teacher algorithm implementation, our multitask BERT model plays two roles: the teacher model and the student model. Both models share the same architecture but have different parameters. The student model is our unmodified multitask BERT model, and the teacher model's

parameters are an exponential moving average of the student model's weights. This approach aims to make the teacher model more stable and less prone to overfitting.

The idea behind the mean teacher algorithm is that models that are consistent with their past predictions are more likely to generalize well on new data. Therefore, we use the teacher model to enforce consistency on the student model's predictions. We accomplish this by using Bregman Proximal Point Optimization to calculate and minimize a penalty that penalizes the student model's Bregman divergence, which is a measure of the distance between the teacher and student model's logits.

We calculated this divergence, called consistency loss in our referenced paper, varying between mean squared error or Kullback-Leibler divergence. Mean squared error measures the distance between the student and teacher model's prediction logits, while KL divergence is a statistical measure of how much information is lost when using the student's prediction instead of the teacher's. Because KL divergence is not symmetric ($KL(Q|P) \neq KL(P|Q)$), we used the sum of both KL divergences as our consistency loss.

In the mean teacher algorithm, the student model can learn from both supervised and unsupervised data. The supervised loss is the typical loss used, either MSE or Cross Entropy loss on labeled data. In addition, consistency loss is introduced by the mean teacher algorithm, which measures how well the student model agrees with the teacher model.

By learning from a more consistent teacher model, we hope to reduce overfitting and increase accuracy while not needing as much labeled data. The mean teacher algorithm is a promising extension that improved the performance of our multitask BERT model in downstream tasks.

5 Experiments

5.1 Data

For **sentiment analysis**, we used the Stanford Sentiment Treebank (SST) dataset [1] (splits: 8,544 train, 1,101 dev, 2,210 test) and the CFIMDB dataset (splits: 1,701 train, 245 dev, 488 test). Given text, it classifies its polarity: negative, somewhat negative, neutral, somewhat positive, or positive.

For **paraphrase detection**, we used the Quora Dataset [2] (splits: 141,506 train, 20,215 dev, 40,431 test). Given a question pair, it determines whether the different questions were paraphrases of each other (classification).

For **for semantic textual analysis**, we used the SemEval STS Benchmark Dataset [3] (splits: 6,041 train, 864 dev, 1,726 test). Given two phrases, it determines on a scale from 5 (same meaning) to 0 (not at all related) if the phrases are related to each other.

5.2 Evaluation method

We evaluated our model by performing inference on the held-out dev sets from Stanford Sentiment Treebank, and Quora, and SemEval STS Benchmark Datasets. We compared the prediction accuracy to those of our baseline models as well as to each other to learn what changes we made were beneficial.

Finally, we ran our model on the test sets from Stanford Sentiment Treebank, and Quora, and SemEval STS Benchmark Datasets and evaluated on the test set accuracy and correlation scores.

5.3 Experimental details

In both single and multi-task portions of our project, the training tasks were ran for 10 epochs. We applied dropout to BERT's pooled outputs, with a probability of 0.3. We used flat learning rates of 1e-3 for pretraining, and 1e-5 for finetuning. For the single task classifier, we used a batch size of 8, while for the multi-task classifier, to accommodate memory limits we used batch sizes of 8, 32, and 64 depending on the dataset (specified in the results).

[1] `https://nlp.stanford.edu/sentiment/treebank.html`

[2] `https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs`

[3] `https://ixa2.si.ehu.eus/stswiki/index.php/STSbenchmark`

5.4 Results

We present the compiled results of our models on the development set in Table 1. The results can largely be split into two sections: **single-domain fine-tuning** on one dataset and **round-robin fine-tuning** on all three datasets. It is apparent that with single-domain fine-tuning, the model's performance largely favors the domain it was trained on, as expected. In round-robin fine-tuning, we observed a significant performance boost after switching from uneven to even fine-tuning across the three datasets. It makes sense that round-robin fine-tuning on uneven datasets resulted in bad performance of the model on STS, as the SemEval dataset is the smallest of the three. Amongst the even dataset round-robin models, we observed only small changes in performance with the exception of round-robin with mean teacher, which helped the model generalize better.

We present our best model's performance on the test set in Table 2. It is expected that our model performs slightly worse on the test set because it is completely new data. Its results mostly resemble that of the dev sets with the exception of STS.

Table 1: Performance of various models on SST, Paraphrase, and STS development sets

Model description	dev SST	dev Para	dev STS	dev average
Finetuned on only SST (baseline)	0.516	0.463	0.248	0.409
Finetuned on only SST w/ Mean Teacher	0.515	0.475	0.245	0.417
Finetuned on only Para w/ Mean Teacher	0.197	**0.823**	0.388	0.469
Finetuned on only STS w/ MSE Loss & Mean Teacher	0.203	0.594	**0.722**	0.506
Round Robin w/ Uneven Dataset & Batch Sizes	0.305	0.455	0.019	0.260
Round Robin w/ Even Dataset Sizes & Batch Size 8	0.490	0.748	0.613	0.617
Balanced RR w/ Even Dataset, Specialized Loss	0.493	0.748	0.654	0.632
Balanced RR w/ Mean Teacher, Even Dataset, Specialized Loss	**0.519**	0.746	0.727	**0.664**

Table 2: Performance of best model on SST, Paraphrase, and STS test sets

Model description	test SST	test Para	test STS	test average
Balanced RR w/ Mean Teacher, Even Dataset, Specialized Loss	**0.529**	**0.744**	**0.677**	**0.650**

6 Analysis

In this section, we offer analysis of our main extensions of BERT for performing well simultaneously on three downstream tasks: single-domain fine-tuning, balanced and unbalanced round-robin fine-tuning, different batch sizes, specialized loss, and mean teacher.

6.1 Single-domain fine-tuning

We observe that training on only one dataset can lead to improved performance on that specific task but can lead to degradation in performance on other tasks. For example, the model finetuned on only Para dataset with Mean Teacher performed well on Para dataset but poorly on SST and STS datasets. Similarly, the model finetuned on only SST dataset showed poor performance across all metrics. This suggests that training on a single dataset can lead to overfitting, which reduces the model's generalization ability to other datasets. Hence, it is crucial to use multiple datasets and appropriate finetuning techniques to improve the model's performance on multiple tasks.

6.2 Unbalanced/balanced Round-robin Fine-tuning

We observe that with round-robin fine-tuning, it performs even worse than single-domain fine-tuning due to the uneven distribution of training on the different sized datasets, leading to poor generalization. This is made especially apparent by the model's poor performance on the STS dataset as the SemEval dataset is the smallest of the three.
However, switching to training the same amount on each dataset (balanced to the size of the smallest dataset, SemEval), resulted in a significant increase in performance as the model is fine-tuned evenly across all three tasks.

6.3 Different batch sizes

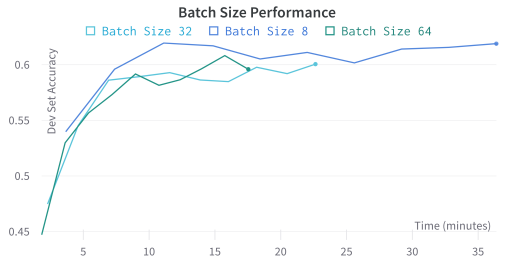

Figure 3: Experiments with varying batch size

In our experiments, we found that using a smaller batch size, such as 8, led to better performance than using a larger batch size, such as 32 or 64. This is likely due to the fact that a smaller batch size allows the model to see a more diverse set of examples in each batch, which can lead to better generalization. Additionally, using a smaller batch size can lead to a more stable training process, as it reduces the likelihood of the model getting stuck in a poor local minimum.

6.4 Specialized Loss

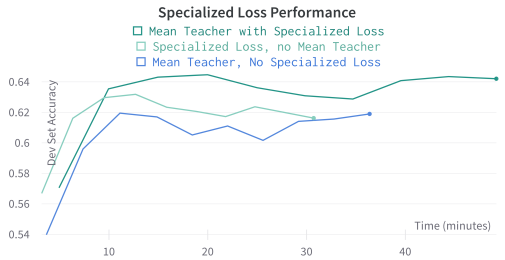

Figure 4: Experiments with varying loss function

The specialized loss function we experimented with was Mean Squared Error (MSE) loss for the STS dataset. As our baseline, we originally used cross-entropy loss for all three prediction tasks, however, using cross-entropy loss for the STS dataset treats it as a classification task, which may not be the most appropriate choice. Using MSE loss for the STS dataset, on the other hand, explicitly optimizes the model to predict continuous scores, which is a more appropriate choice for this task.

Our experiments showed that using MSE loss for the STS dataset improved the model's performance on this task, on par with the mean teacher method. Additionally, using mean teacher in conjunction with the specialized loss function further improved the model's performance on all three tasks, suggesting that using a specialized loss function can be beneficial when fine-tuning BERT on multiple tasks.

6.5 Mean teacher

In our experiments, we used Mean Teacher to improve the generalization ability of our model when fine-tuning on multiple tasks. To calculate the Bregman divergence between the student and teacher model's logits, we experimented with using two different divergence functions: KL divergence and MSE divergence.

Interestingly, we found that using KL divergence led to a significant improvement in performance over using MSE divergence. This result is contrary to the findings in the original Mean Teacher paper, where the authors reported that MSE divergence showed better performance. The exact reason for this discrepancy is unclear, but it is possible that the nature of our dataset, architecture, and prediction task favored KL divergence over MSE divergence.

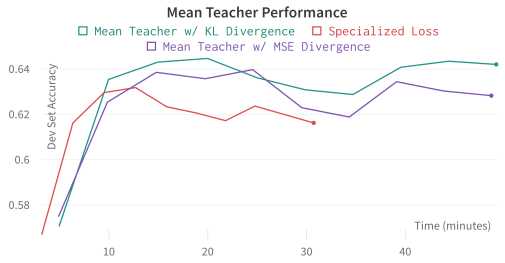

Figure 5: Experiments with mean teacher

Overall, our observations suggest that using KL divergence to calculate the Bregman divergence in Mean Teacher can lead to improved performance on multiple downstream tasks when fine-tuning BERT. In our case, using KL divergence led to better performance, but this may not be the case for all datasets and prediction tasks. It is therefore important to experiment with different divergence functions to determine the most appropriate choice for a given task.

7 Conclusion

In this project, we implemented a BERT model and extended upon it to simultaneously perform well on sentiment analysis, paraphrase detection, and semantic textual similarity. We show that a model with round-robin multitask fine-tuning on even datasets across the three domains and mean teacher with specialized losses for each task performs significantly better than the baseline on all three tasks, achieving strong results: SST/Para/STS/total 0.519/0.746/0.727/0.664 on the dev leaderboard and SST/Para/STS/total 0.529/0.744/0.677/0.650 on the test leaderboard. In future work, we are interested in experimenting with extensions such as further pre-training, perhaps on external datasets in the target domains, as well as gradient surgery multi-task fine-tuning.

References

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

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Zhao Tuo. 2019. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations for natural language understanding. page 353–355.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning.

Chi Sun, Qiu Xipeng, Yige Xu, and Xuanjing Huang. pages 194–206. Springer, 2019. How to fine-tune bert for text classification?

Antti Tarvainen and Harri Valpola. 2018. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results.

8 Appendix

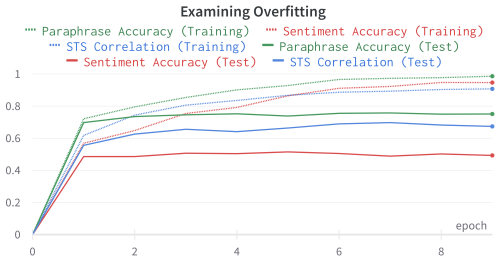

Figure 6: Comparing Category Training and Test Accuracies

To better understand the performance of our model, we examined it to determine whether the misclassifications were due to underfitting or overfitting. Underfitting occurs when a model is too simple and fails to capture the complexity of the data. As a result, it performs poorly both on the training and test data. Overfitting, on the other hand, occurs when a model is too complex and captures noise in the training data. We were concerned that our task specific classifiers were very simple, and our test error may have been due to underfitting rather than overfitting.

We examined the difference between training and test accuracy and found that while all three training accuracies were between 90 and 95%, the test accuracies were between 50% and 75%. To address the issue of overfitting, we implemented the mean teacher method, and exposed the model to more training data. Despite our best efforts to optimize our model, we found that it was still overfitting, albeit significantly less than with just pure dropout as regularization.

We also tried implementing gradient decay, which is a technique that reduces the magnitude of the gradient over time to improve model performance. We tested values of 0.01 and 0.1, but found that they had minimal effect on both training and test accuracies. Despite these challenges, we remain optimistic about our model's performance and are open to explore other optimization techniques to reduce overfitting.