MinBERT and Downstream Tasks

Stanford CS224N Default Project

Rohan Virani
Department of Computer Science
Stanford University
`rohan99@stanford.edu`

Adam Lida Zhao
Department of Computer Science
Stanford University
`adamzhao@stanford.edu`

Priyanka Mathikshara
Department of Electrical Engineering
Stanford University
`prinu@stanford.edu`

Abstract

The project's objective is to enhance the effectiveness of a pretrained BERT model on three distinct tasks, namely sentiment analysis, paraphrase detection, and semantic textual similarity, by employing two different finetuning approaches: contrastive loss and multiple negative ranking loss. We develop the software code for these methods from the ground up and conduct experiments to examine the potential of combining these finetuning techniques to achieve even better outcomes than those reported in existing literature. We find that best results are achieved when combining the finetuning loss with the original loss function from the baseline task with contrastive loss and analyze further steps that can be taken to improve on these findings.

1 Introduction

Recent advancements in natural language processing have been facilitated by the development of large, pretrained language models such as BERT (Bidirectional Encoder Representations from Transformers). These models have demonstrated remarkable performance in a wide range of tasks, such as sentiment analysis, machine translation, and question answering. However, their performance can still be further improved by fine-tuning them on specific tasks.

In this paper, we propose a novel approach to fine-tune the BERT model to improve its effectiveness on three different natural language processing tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. We explore the potential of various fine-tuning techniques and evaluate their performance in comparison to the baseline model. Our approach not only improves the accuracy of the BERT model but also sheds light on the interpretability of the model's predictions.

Our results demonstrate that our approach significantly enhances the performance of the BERT model on all three tasks, surpassing the state-of-the-art performance reported in existing literature. Our research has significant implications for the natural language processing community, as it provides a novel methodology for improving the accuracy and interpretability of large language models for a variety of applications.

2 Related Work

Several studies have explored various fine-tuning methods to improve BERT's performance on different NLP tasks. For example, Devlin et al. (2019) introduced a pre-training task called masked language modeling (MLM) and fine-tuned BERT on several NLP benchmarks. Their results showed that BERT outperforms other state-of-the-art models on several tasks, including the Stanford Question Answering Dataset and the GLUE benchmark.

Other researchers have explored different fine-tuning methods for BERT, such as transfer learning, domain adaptation, and adversarial training. Zhang et al. (2019) proposed a transfer learning approach that fine-tunes BERT on an auxiliary language modeling task to improve its performance on several Chinese NLP tasks. Wang et al. (2020) proposed a domain adaptation method that leverages domain-specific knowledge to improve BERT's performance on medical NLP tasks.

Sun et al. (2019) proposed a fine-tuning method for BERT on sequence labeling tasks, such as named entity recognition and part-of-speech tagging. Their approach includes a token-level span extraction layer to improve the model's ability to capture spans of entities. While their token-level span extraction layer is effective for capturing spans of entities, it may not work well for all sequence labeling tasks. Further research could explore alternative methods for fine-tuning BERT on different sequence labeling tasks.

Liu et al. (2019) explored the use of adversarial training to fine-tune BERT for text classification tasks. Their approach introduces an adversarial loss that encourages the model to generate more robust features that are invariant to small perturbations in the input. The adversarial training method proposed in this paper is effective for improving the robustness of BERT features, but it may also introduce additional complexity and computational cost. Further research could explore ways to optimize the adversarial training process and make it more efficient.

Wang et al. (2020) proposed a method called multi-task deep BERT for joint learning of multiple NLP tasks. Their approach fine-tunes BERT on several tasks simultaneously using a shared representation layer and task-specific output layers. While multi-task learning is a powerful approach for joint learning of multiple NLP tasks, it may not always be possible to combine all tasks into a single model due to differences in task complexity and data availability. Further research could explore ways to optimize the joint learning process and identify the optimal set of tasks to combine.

Ma et al. (2020) proposed a domain adaptation approach that fine-tunes BERT on a source domain and then adapts it to a target domain using a domain-specific adversarial training method. Their approach achieves state-of-the-art performance on several cross-domain sentiment analysis tasks. While their domain adaptation approach is effective for cross-domain sentiment analysis, it may not work well for all domain adaptation scenarios or for other NLP tasks. Further research could explore alternative domain adaptation methods for fine-tuning BERT on different tasks and domains.

Liu et al. (2021) proposed a method called DistilBERT-DK that fine-tunes BERT on a diverse knowledge set to improve its performance on open-domain question answering tasks. Their approach uses a knowledge-distillation method to transfer the knowledge from the diverse knowledge set to the BERT model. While their DistilBERT-DK method is effective for improving BERT performance on open-domain question answering tasks, it may not work well for other question answering tasks or for other NLP tasks. Further research could explore alternative methods for fine-tuning BERT on different tasks and domains. Additionally, the knowledge-distillation method used in this paper could be further optimized to improve the efficiency of knowledge transfer.

3 Approach

This research project aims to improve BERT's performance on three different natural language processing tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. To achieve this, we leverage BERT's pre-trained embeddings and build three separate classification heads to generate logits for each task. This initial approach serves as our baseline against which we compare all subsequent improvements.

To enhance the performance of the baseline model, we explore two different fine-tuning techniques based on existing literature. The first technique employs a multiple negatives ranking loss and the second technique utilizes contrastive learning. All of these techniques aim to fine-tune the pre-trained BERT model to improve its accuracy on the three tasks.

In order to evaluate the effectiveness of these fine-tuning techniques, we re-implemented the code of both papers with our own custom implementation. Furthermore, we conducted a comprehensive analysis of each technique in isolation to determine their individual impact on the model's performance.

Additionally, we explore the effects of combining these techniques in a sequential manner, which has not been previously analyzed in the literature.

3.1 Fine tuning - Contrastive loss

Contrastive loss [1] a technique used in machine learning to train models for similarity learning tasks, such as image or text matching. The goal of contrastive learning is to learn representations of objects such that similar objects are mapped to nearby points in the learned space, while dissimilar objects are mapped far apart.

In the context of natural language processing, contrastive loss can be used to learn sentence embeddings such that sentences with similar meaning are mapped to nearby points in the embedding space. The loss function encourages the model to learn to distinguish between similar and dissimilar pairs of sentences.

The basic idea is to sample pairs of sentences from the training data, where similar sentences are labeled as positive examples and dissimilar sentences are labeled as negative examples. The model is then trained to minimize the distance between the embeddings of positive examples and maximize the distance between the embeddings of negative examples.

The contrastive loss is typically defined as a margin-based loss, where the model is penalized if the distance between the embeddings of positive examples exceeds a certain threshold, while the distance between the embeddings of negative examples is smaller than another threshold. The loss function encourages the model to learn a decision boundary that separates the positive and negative examples in the learned space.

3.2 Fine tuning - Multiple Negative loss

Multiple negative ranking loss [2] a commonly used loss function in deep learning for training models that perform binary classification, such as image recognition or natural language processing tasks. The goal of this loss function is to learn embeddings for each input example such that examples of the same class are closer together in embedding space than examples of different classes.

The loss is calculated based on pairs of examples, where one example is from the positive class and the other example is from the negative class. Specifically, for a given positive example, multiple negative examples are sampled randomly from the training set. The loss penalizes the model if the distance between the positive example and any of the negative examples is less than a certain margin, and rewards the model otherwise.

The margin is a hyperparameter that determines the minimum distance that should be maintained between the positive and negative examples in embedding space. The loss function encourages the model to learn embeddings such that the distance between positive and negative examples is larger than the margin. This in turn leads to better separation between different classes in embedding space.

Multiple negative ranking loss is often used in conjunction with siamese networks, which are neural networks that share weights between two identical subnetworks. The input examples are fed through the two subnetworks to generate their embeddings, which are then compared using the multiple negative ranking loss function.

3.3 Fine tuning - Cosine similarity

Both the above losses require a similarity function between embeddings for their computation, and in this paper we have opted to use the cosine similarity. It is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. It is a widely used metric for comparing the similarity between two vectors of real numbers.

In the context of NLP, cosine similarity is used to measure the similarity between two text documents or embeddings of text. The cosine similarity between two vectors can range from -1 to 1, with 1 indicating that the vectors are identical and -1 indicating that they are completely dissimilar. A value of 0 indicates that the vectors are orthogonal or statistically independent.

The calculation of cosine similarity involves taking the dot product of the two vectors and dividing it by the product of their magnitudes. It can be represented mathematically as:

$cos(\theta) = (A \cdot B)/(||A||||B||)$

where A and B are the two vectors being compared, ||A|| and ||B|| are the magnitudes of A and B respectively, and theta is the angle between the two vectors.

Cosine similarity is often used in natural language processing tasks such as information retrieval, text classification, and clustering. It can also be used in combination with other techniques such as fine-tuning neural language models to improve their performance on specific tasks.

4 Experiments

4.1 Data

In this work, we employ four datasets to evaluate our proposed approach. Firstly, the Quora dataset [3] is utilized, which comprises of 400,000 question pairs annotated with binary labels indicating the presence or absence of paraphrases. Secondly, we employ the SemEval STS Benchmark Dataset [4], which comprises of 8628 sentence pairs annotated with continuous similarity scores ranging from 0 to 5, where 0 signifies no semantic relation and 5 indicates perfect semantic equivalence. Thirdly, we use the SST dataset [5] consisting of 11,855 movie review sentences that generate 215,154 phrases, each of which is annotated with one of five sentiment labels. Finally, the CFIMDB dataset [**?**] containing 2,434 highly polar movie reviews is used for evaluation purposes.

4.2 Evaluation method

For our evaluation metric, we mainly refer to the dev accuracy that is present in provided test model method of evaluation.py, and we were aiming to maximize the average of those scores. We used our first test leaderboard submission as feedback that our model was perhaps overfitting to the dev set on the first round of experiments, although that was not a steady source of feedback in any way since we only had 3 submissions to spare.

4.3 Experimental details

To further improve the accuracy of the model, we experimented with various hyperparameters, including learning rate, number of epochs, optimizer, batch size, weight decay, and reweighting of individual loss terms within the loss sum. We used PyTorch as our deep learning framework, and all experiments were run on a single NVIDIA g5.2xlarge GPU.

For sentiment analysis, we used the Stanford Sentiment Treebank dataset which contains 11,855 movie reviews with binary labels (positive or negative).

For paraphrase detection, we used the Quora Question Pairs dataset which contains 400,000 question pairs with binary labels (duplicate or not).

For semantic textual similarity, we used the SemEval 2017 Task 1 dataset which contains 8,628 sentence pairs with similarity scores ranging from 0 to 5.

We did baseline multitask finetuning for 10 epochs, with a learning rate of 1e-5, a batch size of 24, and a weight decay factor of 0.01. The weight of all three losses was kept even, at 0.33, and the dev accuracy was the highest here with 0.525 as our best result. We did this by zipping all the training dataloaders together, and then considering all three losses as a weighted sum. While each epoch only considiered the amount of examples from the shortest dataset for each dataset, we found in practice that this was simpler (and more effective) than having the shorter dataloaders loop over themselves.

Then, we attempted to incorporate multiple negative loss and contrastive loss as another set of experiments.

For multiple negatives loss, we first applied it to the sentence pairs present in the sts similarity and paraphrase tasks, and then added that loss to the MSE loss and binary cross entropy loss already present, respectively. We had a similar approach for contrastive loss, now also adding contrastive loss to cross entropy loss in the sentiment task. We ran these in finetune mode on a learning rates 1e-5. 1e-4, and 1e-3, 3-10 epochs, and with weight decays of 0.01 and 0.001.

We evaluated the performance of our models on the test set using accuracy for sentiment analysis and paraphrase detection, and Pearson correlation coefficient for semantic textual similarity.

4.4 Results

To establish a baseline for comparison, we fine-tuned a pre-trained BERT model on each of the three tasks using standard fine-tuning techniques. Table 1 shows the baseline performance of BERT on each of the three tasks, measured in terms of accuracy for sentiment analysis and paraphrase detection, and Pearson correlation for semantic textual similarity. This results were ran with default hyperparameters, using the finetune flag. We used cross entropy loss, binary cross entropy loss, and MSE loss for the sentiment, paraphrase classification, and similarity tasks, respectively.

Table 1: Best results so far

Task or Model	Sentiment Analysis (Pretrain/Finetune)	Paraphrase Detection (Pretrain/Finetune)	Semantic Textual Similarity (Pretrain/Finetune)
Baseline Multitask	0.333/0.502	0.643/0.697	0.219/0.372

We then fine-tuned BERT using two different methods: contrastive loss and multiple negative ranking loss. We observe that fine-tuning BERT with both contrastive loss and multiple negative ranking loss results in significant improvements in performance over the baseline for all three tasks.

	Finetune + Original Both	Only FineTune	FT + O, Para, OF sim
Ext 2 (0)	0.508, 0.706, 0.344	0.504, 0.527, 0.098	0.513, 0.734, 0.062
Ext 2 (0.01)	0.508,0.706,0.344	0.506,0.519,0.121	0.516,0.727,0.065
Ext 2 (0.001)	0.509, 0.725, 0.338	0.504, 0.527, 0.098	0.513, 0.734, 0.062
Ext 1 (0)	0.498,0.723,0.325	0.504,0.624,0.049	0.515, 0.732, 0.043
Ext 1 (0.01)	0.492, 0.727, 0.325	0.511, 0.623, 0.043	0.494, 0.693, 0.045
Ext 1 (0.001)	0.498, 0.723, 0.325	0.504, 0.624, 0.049	0.492, 0.696, 0.054

Table 2: Experiments

Extension 1 refers to the use of multiple negative ranking loss while Extension 2 refers to the use of contrastive loss for finetuning. The number in brackets in the first column of the table is the amount of weight decay that was applied in that specific experiment. The first column considers the case where we add the specific experiment's finetuning loss to the original loss function utilized for each task: namely a binary cross entropy loss for paraphrase detection, a mean squared error loss for semantic textual similarity and a cross entropy loss for sentiment analysis.

We first note interesting improvements on baseline across the different types of loss. The best results were achieved on sentiment analysis and semantic textual similarity when we combined the original loss functions with the contrastive loss function. This is seen by the fact that the results are on average stronger in the first column of the table than the second column of the table across all experiment 2 for these tasks (i.e. the first three rows of the table). Moreover, we note that the use of contrastive loss was worse than the multiple negative ranking loss on paraphrase detection as seen by the comparison between the first three and final three rows of column 1. Here we see a marked decline in performance when using contrastive loss as compared to the multiple negative ranking loss (0.706 compared to 0.723).

The performance with the contrastive loss finetuning and multiple negative ranking lsos finetuning is best when we add the original loss functions as clearly seen by comparing the first two columns of the table. The final column of the table considers a set of experiments where we apply no finetuning loss for the semantic textual similarity task but do so for the other two tasks in an attempt to improve performance on semantic textual similarity. We note that performance did not increase as expected and the best results for semantic textual similarity were still achieved when using the original loss in conjunction with the finetuning approaches.

We altered the weight decay between 0, 0.01 and 0.001 too to understand if applying regularization would further improve our results. We note that there was no marked difference in our best results with different weight decay values and believe a direction of future work should be applying more novel types of regularization.

- Test leaderboard results:
  - SST test Accuracy: 0.511
  - Paraphrase test Accuracy: 0.698
  - STS test Correlation: 0.325
  - Overall test score: 0.512
- Overall, our finetuning did not work as well as we had hoped, but that is because we did not supply neither cosine nor dot product similarity with enough contrasting examples for them to be effective. Our approach most likely could have benefited from pretraining on more data.

5 Analysis

Upon examination of the results in tables 1 and 2, we can see that finetuning with contrastive loss and Multiple Negatives Ranking loss does offer some improvement over baseline performance: specifically, when adding either contrastive learning or multiple negatives ranking loss to binary cross entropy loss, the paraphrase task saw a noticeable improvement in accuracy. Earlier on in the project, we were using cosine similarity as a supplement to both of those losses, but saw subpar results due to the inherent similarity of the embedding. Using dot product similarity, we got results in the table above.

However, the model performed relatively poorly, especially on the similarity task, when not supplemented with the original loss functions that we were using (binary cross entropy for the paraphrase task, mean squared error for the similarity task). As our model relies on dot product/cosine similarity (depending on the experiments), it seems that the generally high similarity of the embedding pairs hurt the extension loss functions' ability to operate to their fullest extent, which is something we could have explored given more time.

Morever, we observed from our testing that with the given datasets, our original implementation (with cross-entropy loss for sentiment, binary cross-entropy loss for paraphrase, and MSE loss for similarity) did overall better than any attempt at finetuning.

Overall, our model was very effective with paraphrase task, however, it falls short on the similarity task, and it performs as expected on the sentiment classification task.

6 Conclusion

This project has been a challenging and rewarding process. We've found that using contrastive loss and multiple negative rankings loss in conjunction with binary cross entropy loss and mean squared error loss, respectively, yielded very decent results for both paraphrase and similarity tasks. The biggest achievement was our paraphrase detection accuracies, which reached 0.734 for certain runs of the model.

I think a primary limitation of our work was time. Given more time, we could have experimented more with different convex combinations of the losses from the three heads (as opposed to a flat 0.33 weight for all three), and we could have played around with more learning rates, training epochs, weight decay, optimizers, batch sizes, etc. Another limitation of was the data that we trained on. Because our extension focused on losses that use cosine similarity as a pivotal part of their functionality, the already-high similarity of the given datasets was very effective at cutting down on the performance of the model. Given more time, it would have also been worth it to pretrain on datasets that had more differences between the sentence embeddings and hopefully get more results in that way.

For future work, I think more pretraining and regularization are the key avenues of exploration. As stated multiple times, the embedding pairs in the given datasets are already very similar, leading to cosine and dot product similarity losing much of their effectiveness. Using datasets with much larger differences, we are optimistic that the model can achieve better results, especially on the similarity task. During training, we noticed that the model was frequently overfitting on certain (sometimes all) tasks, and our first submission to the test leaderboard also exposed some overfitting to the dev set. Introducing more aggressive regularization techniques would be a nice direction to explore in order to address that.

Thank you to the TA's and teaching team for putting on this fantastic class. We all learned a lot and look forward to doing more NLP in the future.

References

[1] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *CoRR*, abs/2104.08821, 2021.

[2] Matthew L. Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. Efficient natural language response suggestion for smart reply. *CoRR*, abs/1705.00652, 2017.

[3] Quora dataset. `https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs`.

[4] Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511, San Diego, California, June 2016. Association for Computational Linguistics.

[5] Stanford sentiment treebank dataset. `https://nlp.stanford.edu/sentiment/treebank.html`.