Improving BERT – Lessons from RoBERTa

Stanford CS224N Default Project

Channing Lee, Hannah Prausnitz-Weinbaum, Haoming Song
Department of Computer Science
Stanford University
lee18@stanford.edu, hannahpw@stanford.edu, song4016@stanford.edu

Abstract

We implemented BERT and finetuned it on three downstream tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. We used RoBERTa, the first extension to BERT, as an inspiration and guide in our finetuning. To extend this model, we implemented a variety of changes including changes to hyperparameters, loss, and inputs, additional datasets and pretraining, dropout, and multitask fine-tuning. Following implementation of extensions, we achieved accuracies significantly above baselines.

1 Key Information to include

We have no external collaborators or external mentor, and we are not sharing this project.

Team contributions: All team members contributed to all parts of the project. Hannah primarily implemented minBERT, analyzed related work, provided and cleaned additional datasets, implemented multiple negatives ranking loss, debugged/tweaked hyperparameters, and worked on report writeup. Haoming assisted in implementing the initial train functions, adding in new datasets and converting them into new dataloaders for training, worked on Cosine Embedding Loss, and assisted in the report writeup. Channing worked on the train functions, assisted both Hannah and Haoming on implementation, helped with getting GCP and Google Colab set up, focused on integrating both MNR loss and Gradient surgery into the model, performed hyperparameter tuning, included more layers, dropout , trained the model, generated graphs for the writeup, and worked on the report writeup.

2 Introduction

Natural Language Processing has made a plethora of monumental advancements in the past few decades, and much of it is thanks to Google's introduction of the Bidirectional Encoder Representations from Transformers (BERT) in 2018 (Devlin et al, 2019). Commonly referred to as the Michael Jordan of NLP, BERT implemented new, state-of-the-art implementations during its time, such as bidirectional encoding and contextualized word representations, and its open-source technology skyrocketed language modeling to new leaps and bounds, significantly improving accuracy in eleven NLP tasks. However, with all its glory came many limitations that BERT possessed. Particularly, the bare bones minBERT model was not finetuned for specific tasks, making its usage comparable to that of a swiss army knife, being compatible in a variety of different tasks, but never the optimal choice for any.

Our objective for combating minBERT's jack-of-all-trades dilemma is to expand upon the model and finetune it upon three different tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. We took inspiration from previous approaches towards expanding BERT, such as RoBERTa (Liu et al, 2019), enforcing a massive expansion on pretraining through doubling the pretraining data, adding the CFIMDB dataset for sentiment analysis, the MRPC and SE datasets for paraphrase detection, and the SICK dataset for semantic textual similarity. Additionally, we implemented changes in finetuning such as using more optimal loss functions, applying dropout to prevent overfitting, and

manually adjusting hyperparameters and learning rate for optimal performance, sharpening the swiss army knife into a finely honed machete in the three tasks.

After evaluating the model, we found our model exhibited significant improvement above BERT's baseline model, achieving an average score of 0.653 in the dev set over all three tasks.

3 Related Work

We use a BERT model as our starting point, and add additional capabilities primarily inspired by the RoBERTa study (Liu et al., 2019).

BERT. When Devlin et al. (2019) introduced BERT, it was state-of-the-art across many language processing tasks. However, it was unclear which aspects of the model were responsible for the jump in performance. It is difficult to thoroughly understand the impact of various architecture changes in large lanugage models due to the difficulty of training these models, since doing many tests is both time-intensive and computationally expensive.

BERT (Bidirectional Encoder Representations from Transformers) was developed by Google scientists in 2018. It takes as input a pair of language segments, and uses a transformer architecture to perform a wide range of language classification tasks. The model is pretrained on a large corpus of unlabeled text and finetuned for each task using task-specific labeled data. During pretraining, BERT optimizes both a Masked Language Model (MLM) and Next Sentence Prediction (NSP) objective. For the MLM objective, random masking is applied to all training data before training begins, in order for the model to predict the masked words. The NSP objective requires the model to separate out positive examples of sentences that appear together in the corpus, from negative examples that do not. BERT is trained using the Adam optimizer with a warmed-up and then decayed learning rate.

Figure 1 below details the BERT architecture and training paradigm. The same pre-trained model is used as a starting point for multiple specific tasks. In encoding text, the symbol [CLS] is used at the start of each input and [SEP] is used to separate parts of an input.

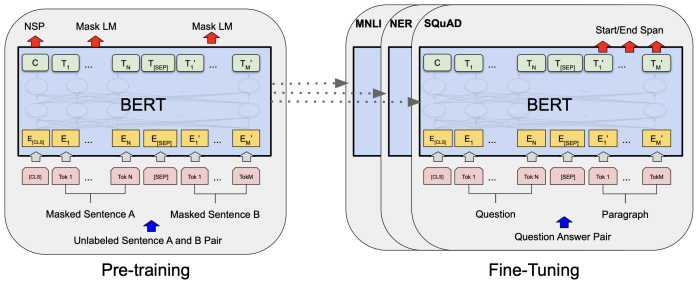

Figure 1: Elements of BERT Pre-training and Fine-Tuning architecture. Obtained from Devlin et al. (2019)

RoBERTa. The RoBERTa study (Liu et al., 2019) aimed to carry out a methodical approach to better understand the parameters most useful in developing BERT, with the goal of improving it. RoBERTa examined some of BERT's hyperparameters and training details in more detail, in order to find the best variation of it. RoBERTa includes four major changes that it finds to improve upon the previous version.

1. RoBERTa has a longer pretraining, using bigger batches for each step of training. It also trains on a larger corpus of data.
2. RoBERTa removes the NSP objective, and trains only on the MLM objective.
3. RoBERTa uses longer sentences instead of segments.
4. RoBERTa performs masking during training, so that the model does not see the same masking pattern twice.

The largest contribution of RoBERTa was the emphasis of training on performance – longer training times and more data led to greater performance gains than any hyperparameter changes. The authors found that after training BERT for 5x longer, it performed better with no signs of overfitting.

RoBERTa also introduced a methodical way of improving upon BERT, in finetuning batch size, learning rate, and working with ensembles. RoBERTa was able to achieve these improvements using only single-task finetuning, whereas many recent models emphasize multitask finetuning.

Discussion. Although RoBERTa achieved state-of-the-art results on many benchmark tests, its implementation has limitations. RoBERTa trained on five datasources – three more than the original BERT. However, these sources seem to be chosen due to convenience, and not because they provide the best balance of training data. These additional data sources may add to the diversity of training, but they do not balance it strategically for language tasks. Additionally, the corpus is solely English-language text, limiting the capability of the model.

RoBERTa includes the stated goal of tuning hyperparameters methodically in order to improve upon BERT. It does this somewhat, but ends up emphasizing training time and data far more. This is an important point, but not as scientifically interesting as understanding in detail the parameters that improve the model. The hyperparameter sweep included in the paper is somewhat limited and includes only some parameters. A broader sweep would be more useful in ensuring that this improvement upon BERT is truly the best one.

Finally, RoBERTa includes test results on a number of benchmarks where it is compared to other large language models. On each of GLUE, RACE, and SQuAD, it achieves state-of-the-art performance in some categories but not others. While the authors include some discussion of differing model choices, it is overall not clear why this model should succeed in only some categories. It would be helpful to see conjectured or actual limitations in the model compared to others in the areas where it did not achieve highest performance.

Despite these limitations, the fact remains that RoBERTa improved upon BERT in many categories, and achieved state-of-the-art performance across all three benchmark tests. While the findings could benefit from further justification, the model results are convincing.

Wider research context. BERT created a disturbance in the large-language model community, setting the stage for additional models to move it forward and increase capabilities in this space. RoBERTa started the process of helping build better representations of language. Through more training and data, we understand what a model needs to be able to build a strong representation. The removal of the NSP objective helps us understand what models really care about when completing language modeling tasks. In changing the masking pattern, we help the model learn the true language instead of the artificial masks. These changes are widely applicable in language representation and across artificial intelligence. Though RoBERTa doesn't make all possible changes to improve BERT, it gives us a reasonable idea of where to go next with this model.

4 Approach

Our goal in this project was to grasp an intuitive understanding of the BERT model through implementation and apply it to downstream subtasks. Using RoBERTa as an inspiration and guide, we implemented BERT with additional changes to improve its performance on three downstream tasks.

The first subtask we tackled was sentiment analysis on the SST and CFIMDB datasets. We first implemented BERT, described in the Related Work section, as a baseline and trained it on each dataset. After achieving reasonable accuracy with this minimal BERT model, we extended it to perform highly on three subtasks: sentiment analysis, paraphrase detection, and semantic textual similarity. We added capabilities to the model that improved all three subtasks and other capabilities designed to improve a single subtask. Many of these extensions are described in detail below.

Additional datasets. Inspired by RoBERTa, our first step was to increase the amount of training data for the model. We expected that adding to the quantity and diversity of the training corpus, using task-specific data, would improve our performance on each task. We sourced additional datasets for each of the three tasks and used them in training.

Adjustment of learning rate. We tested multiple learning rates throughout the process of updating our architecture. We initially observed that the model was not learning quickly enough with the initial learning rate of 10^{-5}, so we increased the learning rate up to a maximum of 10^{-3}. However, this led to a large amount of fluctuation between epochs, indicating the parameter was too large. We then used other methods described here to help our model learn more quickly, and decreased the learning rate to be 10^{-6}.

Multiple negatives ranking loss. For the paraphrase detection task, we observed that each pair of positive examples also produces a pair of negative examples, by matching up one sentence from each example. By exploiting this property of the training data, we could create a more powerful model. Multiple negatives ranking loss (Reimers and Gurevych, 2019) takes a batch of positive examples, and calculates each loss based on that positive example and the negative examples created by pairing it up with each other example in the batch. The calculation of this loss is exhibited in equation 1, for K positive pairs of sentences x, y and model predictions $P(x, y)$. We implemented this loss function in the training loop for paraphrase detection.

$$L(x, y, P(x, y)) = -\frac{1}{K}\sum_{i=1}^{K}\left[P(x_i, y_i) - \log\sum_{j=1}^{K} e^{P(x_i, y_j)}\right] \tag{1}$$

Dynamic masking. Following RoBERTa as a guide, we changed the masking pattern to be implemented during training instead of in advance for each training example. This ensures that randomization in masking cannot be learned by the model.

Cosine similarity as additional input feature. For the semantic textual similarity task, we noted that the naive approach of transforming the concatenated inputs did not take advantage of the problem structure. Therefore, we calculated the cosine similarity of the two encoded sentence inputs. We initially used this as our output, but found that it was not powerful enough to fully encode the semantic textual similarity. We opted to include it as an additional input feature to the model, following a recommendation from classmate Matthias Heubi in Ed post #1815. The equation for cosine similarity of encodings x, y is shown in equation 2.

$$cossim(x, y) = \frac{x \cdot y}{||x||||y||} \tag{2}$$

Dropout. After adding some extensions described above, we saw our model exhibit serious overfitting on the sentiment analysis and paraphrase detection tasks. On both tasks, it achieved a training accuracy near 100% while dev set accuracies remained far lower. To combat this problem, we added dropout layers throughout the model, with dropout parameters of 0.4 to 0.5, depending on the severity of overfitting in a given task.

Additional pretraining. Instead of carrying out finetuning only on our added layers for each of the three downstream tasks, we decided to improve our model further by finetuning the pretrained BERT weights using our domain-specific data.

Multitask fine-tuning. Instead of training our model separately on the three downstream tasks, we created a multitask model that performed well on all three tasks. We were worried that the model would forget a task as it trains on other datasets that perform a different task. To achieve this, we interspersed batches for all three tasks evenly during the training loop. Specifically, we adjusted the proportion of each dataset trained for each batch depending on its size. If a dataset was large, we grab more portions of the dataset during each batch than a smaller dataset. We expected this would allow our model to learn all tasks simultaneously.

Gradient surgery. Because we performed multitask fine-tuning, we trained on batches from separate tasks throughout the same training loop. This gave loss values associated with different tasks. To combine these losses for backpropagation, we included gradient surgery (Yu et al., 2020) to make the losses orthogonal before adding them together. The idea is that this prevents backpropagation on losses that point in contradictory directions, as different tasks have different objectives. Equation 3 demonstrates the gradient used for backpropagation in this method, following surgery.

$$g_i = g_i - \frac{g_i \cdot g_j}{||g_j||^2} \dot{g}_j \tag{3}$$

5 Experiments

5.1 Data

Sentiment classification data consists of a sentence with a negative or positive sentiment and a score describing that sentiment. We used two datasets for this task: the Stanford Sentiment Treebank (SST) and the CFIMDB dataset. The SST dataset consists of 11,855 single sentences, with each sentence labeled using a score 0 to 5. A score of 0 corresponds to an entirely negative sentiment, while a score of 5 corresponds to an entirely positive sentiment. The CFIMDB dataset consists of 2,434 highly polar movie reviews, with binary labels of negative or positive. This totals 14,289 examples.

Paraphrase detection data consists of a pair of sentences that may be paraphrases of each other, and a binary label indicating whether they are paraphrases. We use three datasets to train on this task: Quora, Microsoft Research Paraphrase Corpus (MRPC), and Stack Exchange. The Quora dataset consists of 400,000 question pairs; the MRPC (Dolan and Brockett, 2005) consists of 5,801 sentence pairs collected from newswire articles; and the Stack Exchange dataset (StackExchange) consists of 169,438 pairs of similar questions from Stack Exchange. This totals 575,239 examples.

Semantic textual similarity data consists of sentence pairs marked with a 0 through 5, precise to the nearest tenth, based on how similar the sentences are. A score of 0 corresponds to no similarity, while a score of 5 corresponds to complete similarity. We use two datasets for this task: SemEval and the Sentences Involving Compositional Knowledge (SICK) dataset (Marelli et al., 2014). The SemEval dataset consists of 8,628 sentence pairs and the SICK dataset consists of 4,502 sentence pairs. This totals 13,130 examples.

5.2 Evaluation method

Our first step was to implement minBERT with only pretraining, and no finetuning for individual tasks. As given by the assignment spec, our implementation has accuracy baselines of 0.390 for SST and 0.780 for CFIMDB. After finetuning using data specific to each task, we expect a higher accuracy. As given by the assignment spec, our finetuned implementation will have accuracy baselines of 0.515 for SST and 0.966 for CFIMDB. We achieved accuracies slightly higher than these baselines in our minBERT implementation alone.

After finetuning on the three downstream tasks, we evaluate separately for each on train and dev datasets. For sentiment classification and paraphrase detection, our accuracy score is the percent of examples classified correctly. For semantic textual similarity, our accuracy score is the Pearson correlation coefficient between the predictions and labels of all data. We expect that these accuracies will at a minimum outperform baselines for random guess. This corresponds to $\frac{1}{6} \approx 0.17$ for sentiment classification, 0.5 for paraphrase detection, and 0 for semantic textual similarity.

Because the default final project includes a leaderboard, we also compare to performance on the leaderboard as a metric. We aim to perform comparably to other leaderboard responses to judge our model as successful in comparison to similar implementations.

5.3 Experimental details

We used the model architecture described in the Approach section to run all experiments, varying learning rate as described. We ran each iteration of our model for 10 epochs, which took ≈ 11 hours.

5.4 Results

Table 1 displays the results of our final model, where SST and Paraphrase contain accuracy percentages and STS contains correlations. The final model performed significantly above baseline expectations on all datasets.

Table 1: Highest Accuracy Results

Dataset	SST	Paraphrase	STS	Overall Score
Test	0.508	0.764	0.364	0.651
Dev	0.503	0.758	0.397	0.653
Random Guess	0.17	0.5	0	

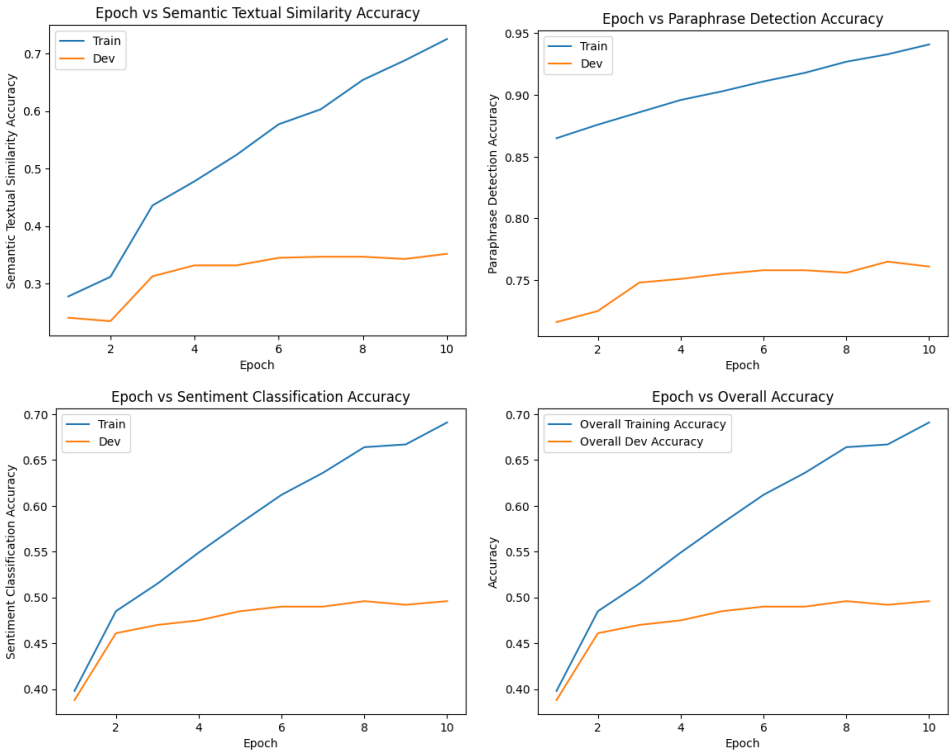

Figure 2: Performance in each task and overall over 10 epochs

We saw improvement over the course of the ten epochs. Results for each of the three tasks and overall performance are shown in figure 2. There is evidence of overfitting in all three tasks, even in our final run.

While we were impressed by our results, they were not as high as we expected to achieve. This is because we implemented a large number of improvements to the starting BERT model, but did not see all of the increases that we expected. This is especially true on the STS dataset. However, it is still clear that we created an improvement to minBERT on all three tasks.

6 Analysis

Our model is most accurate in the paraphrase detection task. This could be due to two possible factors. First, the training dataset is significantly larger for paraphrase detection than the other two tasks. This allows the model to learn the paraphrase detection task well during training. Second, baseline for this task is higher than the others, making it easier to achieve a higher score.

The remainder of this section analyzes how each model change affected performance, with a discussion of the performance changes in each case.

Additional datasets. Adding additional datasets improved our model, with larger improvement dependent on the size of dataset added. This is expected from our analysis of RoBERTa. Additional datasets allowed the model to learn a larger variety of data.

Adjustment of learning rate. We chose to decrease the learning rate because we saw large oscillations in accuracy between epochs. After decreasing it to 10^{-6}, the oscillations decreased dramatically. However, the model also trained more slowly, so accuracy did not increase after ten epochs with the change to learning rate.

Multiple negatives ranking loss. Multiple negatives ranking loss increased the training time required by a factor of three. This is because the model now needed to check all negative pairs for each positive pair. Due to the increased training time, we could not run the model to completion.

Dynamic masking. Because our implementation did not include all features of the full BERT model, the default masking implementation was not randomized. Therefore dynamic masking did not change the performance of the initial model. We opted not to add randomized masking due to time constraints.

Cosine similarity as additional input feature. Replacing the previous semantic textual similarity layer with cosine similarity was not effective on its own. Adding cosine similarity as an additional input led to an increase in training accuracy, but this did not correspond to an increase in test accuracy. It may be that cosine similarity is not as strong of an indicator as we expected.

Dropout. After increasing dropout, we did note some decrease in overfitting. Although the training accuracies remained higher than dev accuracies, they were not significantly higher. However, this did not lead to higher dev accuracies overall. This could mean either that overfitting was benefitting our model or that the model with increased dropout learned more slowly, causing it to lag behind the previous version.

Additional pretraining. Additional pretraining improved all accuracies, as expected from our analysis of RoBERTa. This allowed our model weights to work for task-specific data.

Multitask fine-tuning. Multitask fine-tuning also worked well. We trained a single model and saw improvements significantly above baseline on all three tasks. This suggests that the model was able to concurrently learn all three tasks.

Gradient surgery. We decided to organize our multitask fine-tuning setup such that batches for all three tasks completed at the same time. Since the amount of training data was not even across all three tasks, this meant we would run many batches of one task before switching to another. However, gradient surgery requires us to backpropogate on losses from all three tasks at the same time. This would require us to wait to backpropogate until at least one batch from each task had completed, which was over 100 batches total. Therefore, we could not incorporate both multitask fine-tuning and gradient surgery in the way that we wanted, and opted to prioritize our multitask fine-tuning approach.

7 Conclusion

While some of our implementations, such as following RoBERTa's tracks on expanding pretraining data, succeeded in having a formidable impact on performance, the overall inability to achieve as high of a score as we were expecting gave us a harsh reality check that we still have much to go in terms of understanding the intricate enigma of the digital universe of language modeling.

One limitation we encountered was the amount of time it took to train our new model. Multiple extensions we made, such as multiple negatives ranking loss and additional pretraining, made the training time of the model so huge that we were not able to finishing running the model without sacrificing training performance through augmenting the learning rate. We ultimately had to make difficult decisions about how to use our time most effectively during a quarter-long course, at the expense of testing all configurations that we wanted.

However, despite the model not achieving as proficient of a performance as we have hoped, it still accomplished the objective of significantly improving upon baseline BERT scores.

Given more time to pursue this project, we would have liked to fully test our model performance with multiple negatives ranking loss and gradient surgery. In addition, we would have liked to add additional dropout and regularization, as our model continued to exhibit significant overfitting. Finally, we would have liked to run our model for more than 10 epochs to encourage further improvement. We leave these tasks as future work for other students working on improvements to BERT.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Robert Zamparelli. 2014. The sick (sentences involving compositional knowledge) dataset for relatedness and entailment.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

StackExchange. Stack exchange duplicate questions dataset. `https://public.ukp.informatik.tu-darmstadt.de/reimers/sentence-transformers/datasets/paraphrases/`. Accessed: 2024-03-15.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 5824–5836. Curran Associates, Inc.