Fine-tuning minBERT for Various Downstream Tasks

Stanford CS224N Default Project

Longling Tian
Institute of Computational and Mathematical Engineering
Stanford University
`longling@stanford.edu`

Siqi Wang
Department of Statistics
Stanford University
`siqiwang@stanford.edu`

Abstract

Bidirectional Encoder Representations from Transformers (BERT) is a transformer-based model that generates contextual word representations. It uses a transformer-based architecture to pretrain on large amounts of textual data and can be fine-tuned for a wide range of natural language processing tasks. In this study, beyond implementing the minBERT model to perform semantic analysis, we investigate in various fine-tuning strategies of BERT model to make it simultaneously perform well on multiple downstream tasks, including sentiment analysis, paraphrase detection, and semantic textual similarity. We explore various extensions on BERT including gradient surgery, tuning model architecture and hyperparameters, incorporating additional datasets, and fine-tuning individual tasks. Our model achieved 0.640 overall accuracy on the given test set.

1 Introduction

While BERT has been shown to perform well on individual NLP tasks, it is crucial to make it perform well across multiple tasks. For one thing, from an efficiency perspective, we can save time and computational resources by using the same pretrained embeddings for various tasks instead of training different models for each task [1]; for another thing, with a pretrained BERT model that can perform well across multiple tasks, we can quickly adapt to new tasks and domains by fine-tuning the model on task-specific datasets, rather than starting from scratch [2].

Nevertheless, it remains a challenge to make a BERT model perform well in multiple tasks with the same embedding. There are several possible reasons behind this. First of all, BERT is pretrained on a large corpus of text using an unsupervised learning approach. However, to make the model perform well on a specific downstream task, it needs to be finetuned using task-specific labeled data. This fine-tuning process involves updating the model's weights to optimize its performance on the specific task. Since different tasks may require different architectures and hyperparameters, the same fine-tuning may not work well across different tasks [3]. Additionally, different downstream tasks may have conflicting objectives, making it difficult to optimize the model's performance for all tasks simultaneously [4]. For example, optimizing a BERT model for paraphrase detection may require the model to ignore certain semantic nuances, which may be crucial for semantic textual similarity.

In our study, we first implement the minBERT model to perform semantic analysis. Based on this, we extend the BERT model to create sentence embeddings that can perform well in a wide range of downstream tasks, including sentiment analysis, paraphrase detection, and semantic text similarity. To address the previously mentioned challenges and enable BERT to achieve better performance, we explore a variety of extensions, including using gradient surgery to deal with gradient conflict across tasks, tuning the model architecture and hyperparameters to obtain the best average results for the three tasks, incorporating additional datasets for each task, and applying additional fine-tuning (cosine similarity) to the semantic text similarity task. The best multitask model achieved 0.640 overall accuracy on test set.

2 Related Work

While BERT set a new state-of the-art performance on many downstream tasks, there remains much potential for improvement. The extensions that we implemented were inspired by a wide range of research studies.

Multi-task learning is a useful technique for leveraging the knowledge gained from multiple related supervised tasks. By combining the training of several tasks, this approach can lead to improved performance and greater efficiency in learning. However, one challenge with regard to performing multi-task learning is that gradient directions of different tasks may conflict with one another, which may lead to significantly degraded performance. Yu et al.[4] therefore recommend a straightforward approach to resolve conflicting gradients during optimization, namely Project Conflicting Gradients (PCGrad). In cases where the cosine similarity between the gradients of two tasks is negative, the algorithm projects the gradient of each task onto the normal plane of the other task's gradient. This results in the elimination of the conflicting component of the gradient for each task, reducing the level of harmful interference between tasks. Therefore, the performance on different tasks could decrease by a lot compared to training the specific task only.

While multi-task learning is an effective approach to share the knowledge obtained from several related supervised-tasks [5], achieving more resilient embeddings for BERT models involves a common approach of pre-training on a vast corpus of text, followed by fine-tuning for downstream tasks. Nevertheless, the availability of limited data resources poses a challenge in this regard [6]. As a result, we were motivated to incorporate additional datasets for each downstream task to achieve more robust embeddings.

Learning rate warm-up is another way to improve the performance of fine-tuning as discussed in many research studies [7] [8] . According to Liu et al, incorporating a learning rate warm-up strategy can significantly improve the stability of training, speed up the convergence process, and enhance the generalization capability of adaptive stochastic optimization algorithms such as Adam [8]. This approach ensures that the model weights are trained evenly across all training batches at the beginning of the training process, leading to a remarkable increase in the model's robustness.

3 Approach

The preliminary goal of this study is to implement the minBERT model, whose skeleton code has already been given by CS224N staff. To approach this, we implemented bert.py, classifier.py and optimizer.py. For the bulk of this part, we implemented BERT model layer structures, the attention mechanism, and embeddings.

The second and more important part of this study is to improve on training three tasks simultaneously. The multitask model imports weight from pretrained BERT model, calls the forward function of BERT model to yield pooler outputs for sentences in every task, and use those to make predictions. The model architecture is shown in figure 1 below.

For sentiment classification task, the model takes in only one sentence, gets one embedding, and uses that to feed a softmax with 5 classes. For the other two tasks, the model takes in two sentences and investigates in the relationship of the two embeddings. For sentence similarity task, we calculated the absolute value of difference in two pooler output tensors. We came up with the idea to concatenate this difference along with the two pooler outputs so that the model can learn sentence similarity that is primarily determined by the distance between embeddings.

Besides this, we performed 4 ways of improving multitask fine-tuning: using cosine similarity for training sentence similarity task, taking in more data from other sources to train the same set of tasks, applying gradient surgery to backpropagate three tasks at once, and tuning hyperparameter & model architecture. Details on implementation of these stretagies will be discussed in the next section.

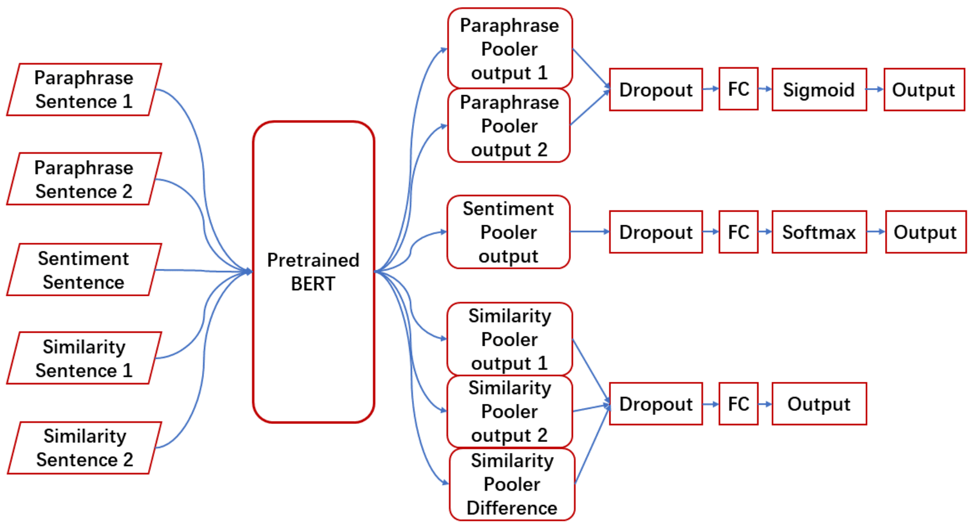

Figure 1: Multitask model architecture

4 Experiments

4.1 Data

Two datasets we used to train the minBERT model for sentiment analysis are Stanford Sentiment Treebank (SST) [9] and CFIMDB. The former dataset consists of 11,855 single sentences from movie reviews and are parsed into 215,154 unique phrases. Each phrase is classified as negative, somewhat negative, neutral, somewhat neutral, or positive. The latter dataset consists of 2,434 highly polar reviews classified as only negative or positive. To better finetune our BERT model so that they can simultaneously perform downstream tasks, we used SST dataset for sentiment analysis, Quora dataset for paraphrase detection, and SemEval STS Benchmark Dataset for semantic textual analysis. Specifically, Quora dataset consists of 400,000 questions pairs with binary labels indicating whether particular instances are paraphrases of one another, and the STS dataset consists of 8,628 different sentence pairs of varying similarity on a continuous scale from 0 (unrelated) to 5 (equivalent meaning).

To further finetune our BERT model, we used three additional datasets , namely Yelp Reviews dataset [10] for sentiment analysis, Adversarial Paraphrasing Task dataset [11] for paraphrase detection, and Sentences Involving Compositional Knowledge (SICK) dataset [12] for semantic textual analysis. The Yelp dataset consists of 10,000 reviews, each corresponding to a sentiment labeled as an integer from 0 (negative) to 5 (positive). The Adversarial Paraphrasing dataset consists of 3,667 pairs of human-generated phrases with binary labels indicating whether one instance is the semantically equivalent but lexically and syntactically disparate paraphrase of the other. The SICK dataset consists of 4500 pairs of sentences that capture similarities on purely language and common knowledge level, similarity labels ranging from 1 to 5.

4.2 Evaluation method

For sentiment classification and paraphrase detection, we used accuracy as our evaluation metric. For semantic textual similarity, we used Pearson Correlation Coefficient as evaluation metric.

4.3 Experimental details

We carried out each extention individually (i.e., testing viability of one approach at a time). Generally, we inherited the settings from the given skeleton code by CS224N staff, and used a hidden size of

768, batch size of 8, and random seed of 11711. The number of epochs was set to be 15 for most tasks.

As specified in figure 1, for the sentiment classification task, since the output has 5 classes labelled 0 to 4, softmax activation function was used, and cross entropy was chosen to be the activation function. For paraphrase detection task, since the output is yes or no, sigmoid activation function and binary cross entropy loss function was chosen. For sentence similarity task, no activation function was used as the output is a real number from 0 to 5, and Mean Squared Error (MSE) loss was used during training.

We tuned learning rate with KerasTuner [13] and ended up with $5 * 10^{-5}$ to be the optimal value. (For all experiments using gradient surgery, since learning rate scheduler is incompatible, we used the default $1 * 10^{-5}$ learning rate.) For this study, we only used the finetune mode, so that the BERT pooler outputs can be task-specific. Besides, we applied learning rate warmup as suggested by Liu et.al to our Adam optimizer. [8] We set the learning rate to logarithmically rise up to $5 * 10^{-5}$ on the first 20% steps, and linearly decay to zero on the later 80% steps.

As stated earlier, we implemented 4 finetuning strategies in this study. The experimental details of these strategies follow below.

- **Cosine similarity**. We used the cosine similarity of two embeddings along with their concatenation to train the final FC and sigmoid layers.
- **Additional Data**. We concatenated the additional dataset mention in part 4.1 along with the given training dataset to train the multitask model.
- **Gradient Surgery**. We followed the idea of projecting the gradients of a specific task onto the normal plane of another conflicting task as recommended by Yu et.al. [4] To perform gradient surgery, we need to get three loss values, one for each task, upon a batch. Therefore, the number of batches has to be the same for three tasks. Since the given dataset are of different sizes, we expanded the smaller two datasets by 5 times, and used different batch sizes for different tasks.
- **Model Architecture**. As mentioned above, we have performed basic tuning on learning rate. In addition, we tried adding in more layers to the similarity task, as it performed worst among all tasks in the beginning. After concatenating two pooler outputs and their difference, we applied two dense layers each followed by a dropout layer instead of a single FC layer as mentioned in figure 1.

All trainings were done on the Amazon AWS EC2 platform. Training generally takes about 20 to 25 minutes on a single epoch, and evaluating on training and development set takes about 5 to 10 minutes. One complete set of training takes about 8 hours in total.

4.4 Results

Finetuning Technique	Overall	Sentiment Classification	Paraphrase	Similarity
minBERT	/	0.528	/	/
Baseline	0.511	0.479	0.781	0.274
Cosine Similarity	0.546	0.503	0.722	0.412
Diff	0.615	**0.520**	**0.795**	0.529
Diff + 2-dense-layer	0.605	0.497	0.787	0.531
Diff + additional data	0.582	0.462	0.782	0.502
Grad-surg + diff	**0.648**	0.513	0.776	**0.655**
Grad-surg + diff + add-data	0.638	0.498	0.777	0.639
Test set	0.640	0.510	0.775	0.634

Table 1: Performance on Development and Test Set Using Different Finetuning Techniques

Table 1 shows the multitask model performance on three tasks and overall on the development set. The last row consists the best model performance on test set. For comparison, the single-tasked classifier achieved 0.528 accuracy using finetune mode on the SST dataset. The best model achieved 0.648 overall accuracy on development set and 0.640 on test set.

The difference between models from first four rows of the table are on the layer architecture. For the baseline model, we simply applied a linear layer to map two pooler outputs to one number for both paraphrase and similarity tasks. Surprisingly the paraphrase detection accuracy was high even for the baseline model. And as expected, the baseline model failed on sentence similarity task with only a 0.274 correlation coefficient. To solve this, we presented three approaches to improve on the sentence similarity task. All modifications below were done on the sentence similarity part only.

The first one is to replace the single linear layer with cosine similarity of the two pooler outputs, and return this number as logit. The second approach is to also concatenate the $|p_1 - p_2|$, where p_1 and p_2 are pooler outputs of the two sentences. The third approach is to have two dense layers each followed by a dropout layer after concatenating the difference in. Among those approaches the second one performed the best on almost all tasks. Therefore, the difference in pooler output approach was adapted in later experiments.

After this, the model still only got around 0.5 accuracy on sentence similarity task. We also tried gradient surgery [4] to prevent gradient conflicts. the model had a boost on sentence similarity task and overall accuracy compared to above, but had a slight decrease on the other two tasks.

To further improve on this, we fed the model with additional datasets from part 4.1 both with and without gradient surgery. However, accuracy on both experiments were worse on every task compared to their counterparts without the additional datasets.

5 Analysis

Our multitask model uses the pooler output from pretrained BERT model for all three tasks. Upon finetuning, backpropagation through the pretrained BERT modifies the pooler output to fit all three tasks at the same time. Because of this modification, models all performed worse than the minBERT model implemented in part 1. Model performance on the paraphrase detection tasks was universally higher than the other two, even for the baseline. This is because paraphrase detection is a binary classification task, and it is relatively easy for model to learn the yes/no answer from bulk amount of training data (the Quora dataset is about 20 times larger than the datasets for other two tasks).

5.1 Hyperparameter tuning

Hyperparameter tuning had limited effect on model performance for this study compared to changing model architecture. According to our tuner [13], models with different learning rates had overall performance varied by only 2%. And after implementing learning rate warmup & decay, there was another 2% improvement. One reason of this is the AdamW optimizer already has its adaptive learning rates. For gradient surgery, we also tried running the model with different learning rates, but the results were almost unchanged, so we decided to use the default $1 * 10^{-5}$.

On top of this, changing batch size had no effect on model performance, but larger batch size significantly sped up the training process. Given the limited memory of our AWS EC2 instance, we decided to use batch size of 16 for non-gradient-surgery tasks. For gradient surgery, the dataset size for three tasks after expanding the smaller two were 80K, 140K, and 60K. Therefore, we used varying batch sizes of 8, 28, and 6 for three tasks and run for 5,000 iterations for each epoch. This was to ensure the number of batches were the same for three tasks in order to carry out gradient surgery.

5.2 Diff layer vs. Cosine similarity

For almost all experiments for this study, we focused on improving the multitask model on semantic textual similarity tasks. First of all, the burst (0.274 $\rightarrow$ 0.529) in validation accuracy after concatenating the pooler output difference corroborated our idea that explicitly adding in this term helped the model understand the task. In other words, since the task was to know the difference between sentences (which is similar to knowing the difference between sentence embeddings), after adding in this difference of pooler output, the model then knew the correlation between difference in sentence embeddings and sentence similarity.

But surprisingly, using cosine similarity did not have as good effect as using the difference. In fact, this even held when we used cosine similarity along with the difference. This indicates that adding cosine similarity to our dense layers have a negative impact on this task. One possible explanation of this is that cosine similarity of two vectors results in only one number, and this number could significantly underestimate the similarity (here we mean as a human being, the way we define similarity) of two sentences. Specifically, Zhou et.al suggested that cosine similarity leads to an underestimation of sentence similarity by misinterpreting the relation between frequent words and other synonyms across texts. [14] Therefore, in practice, we did not use cosine similarity in model training during later experiments.

5.3 Gradient surgery

As introduced in part 4.4, gradient surgery boosted the performance on semantic textual similarity task by about 12%, but at the same time, led to a drop in accuracy on the other two tasks by around 2%. The bright side of gradient surgery (ie. improvement on similarity task) was an evidence that the other two tasks has conflicting gradients for similarity task. In fact, we also tried running the multitask model to learn only the sentence similarity task. The result on validation set was 0.75. Compared to sentiment classification task, which had a drop of less than 1% when training multitask compared to training minBERT on only itself, the similarity task suffered more on this conflicting gradient issue.

On the other hand, gradient surgery performed worse for the other two tasks. We propose that this is because the projection of gradient onto the plane formed by gradients from the other two tasks may still contain some useful information for the task being learnt. After gradient surgery, this part of information was subtracted from the gradient and forgotten. Since the conflicting gradient issue was very mild on sentiment classification and paraphrase detection task, this effect was also small compared to sentence similarity task. Therefore, a possible extension to this study is to perform gradient surgery only on the similarity task.

5.4 Additional dataset

Our approach of training on additional datasets failed for both the original model and gradient surgery version. The main reason could be the difference in distribution of the given and additional dataset. After learning on new data and accordingly modifying BERT parameters to adapt to the new datasets, the multitask model may had over-fitted to new dataset. For example, the original data on sentiment classification is on movie review, while the new dataset is on Yelp review. For the other two tasks, it's even more challenging to find datasets on the very same task.

6 Conclusion

We implemented minBERT and used that to build multitask model to predict on sentiment classification, paraphrase detection, and semantic textual similarity at the same time. We investigated in incorporating multiple fine-tuning strategies, and achieved 0.640 overall accuracy on the test set. Specifically, adding in embedding difference and performing gradient surgery each boosted sentence similarity task by 10 to 15%.

However, the approach of adding cosine similarity and using additional datasets failed to have any improvement. That's to say, there are several areas worth future studying into: to perform gradient surgery on similarity task only, to modify loss function and penalize on similarity task more, to find additional datasets with more similar contents, or to use the additional datasets on pretraining BERT rather than fine-tuning it.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[2] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[3] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[4] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning, 2020.

[5] Yu Zhang and Qiang Yang. An overview of multi-task learning. *National Science Review*, 5(1):30–43, 09 2017.

[6] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020.

[7] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour, 2018.

[8] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

[9] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[10] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. pages 649–657. Advances in Neural Information Processing Systems, 2015.

[11] Animesh Nighojkar and John Licato. Improving paraphrase detection with the adversarial paraphrasing task. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7106–7116, Online, August 2021. Association for Computational Linguistics.

[12] Li Zhang, Steven R. Wilson, and Rada Mihalcea. Multi-label transfer learning for multi-relational semantic similarity, Apr 2019.

[13] Tom O'Malley, Elie Bursztein, James Long, François Chollet, Haifeng Jin, Luca Invernizzi, et al. Kerastuner. `https://github.com/keras-team/keras-tuner`, 2019.

[14] Kaitlyn Zhou, Kawin Ethayarajh, Dallas Card, and Dan Jurafsky. Problems with Cosine as a Measure of Embedding Similarity for High Frequency Words. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 401–423, Dublin, Ireland, 2022. Association for Computational Linguistics.