BERT and Multitask Learning

Stanford CS224N Default Project

Sureen Heer
Department of Computer Science
Stanford University
`sureen@stanford.edu`

Collin Jung
Department of Computer Science
Stanford University
`collinj2@stanford.edu`

Adrian Molofsky
Department of Computer Science
Stanford University
`molofsky@stanford.edu`

Abstract

Transformer models play a crucial role in detecting and understanding the contextual relationships between words, enabling them to answer complex queries, predict text, analyze sentiment, and summarize large corpus of text. BERT (Bidirectional Encoder Representations from Transformers), a deep learning model developed by Google and reliant on a transformer architecture, is highly performant and efficient at tackling a wide range of natural language processing (NLP) tasks, including question answering, analyzing sentiment, and language translation. We propose evaluating the performance and accuracy of minBERT a smaller, less complex version of the original BERT model on a broad range of NLP tasks. We aim to fine tune the performance of minBERT by applying pretraining, gradient surgery, cosine similarity, and other techniques. We intend to use sentiment analysis, se- mantic similarity and paraphrase detection datasets to evaluate our multitask minBERT model and compare it against the baseline minBERT model. By applying different optimizations and training methods on a wide spectrum of tasks, we hope to assess the impact of each optimization and ex- pose the performance improvements of multiple optimization techniques in tandem. Moreover, we want to understand how we can train smaller transformer models in resource constrained environments to preform optimally and take advantage of their smaller size which will help with the development of more powerful, minimal transformer architectures capable of processing and comprehending complex human language.

1 Key Information to include

- TA mentor: Johnny Chang
- External collaborators (if no, indicate "No"): No
- External mentor (if no, indicate "No"): No
- Sharing project (if no, indicate "No"): No

Team contributions:

- Sureen: Implemented all of the baseline minBERT model, cosine similarity extension, and contrastive learning extension. Helped write the final report and milestone.
- Adrian: Gradient surgery, project proposal, project milestone, final report
- Collin implementations: Baseline multitask model trained on single task, `predict_sentiment` function, `predict_paraphrase` function, `predict_similarity`

function, Mean pooling, `process_batch` function with loss calculation for pretraining/finetuning, task-specific dataloaders for `train_multitask`, individual pretraining for each task, multitask finetuning, round robin and annealed sampling scheduling for finetuning, conducted hyperparameter testing. Helped write milestone, helped write final paper, helped implement Cosine-Similarity Siamese structure, and unsuccessfully attempted implementation of Projected Attention Layers.

2 Introduction

For humans, the idea of "multitasking" has quite a different definition than multitasking in the context of language models. In our daily life, we perform actions simultaneously without much thought such as reading body language, determining the real meaning behind people's words, summarizing texts and thoughts, and responding to people's questions. For us, we see multitasking as the ability to perform multiple unrelated tasks simultaneously to a successful degree. However, in the context of language models, multitask has a different meaning. Early language models usually had limited functionality, with early chatbots only being able to speak based on a context-free grammar or models only being able to perform single tasks based on a series of rules. However, with the introduction of the Bidirectional Encoder Representations from Transformers, or BERT, model, the potential of language models to perform multiple tasks simultaneously became an attainable goal. Using a series of classifier layers and pretraining/finetuning, BERT is able to be trained to complete multiple tasks.

For the scope of our project, we explored the potential of BERT in being able to perform sentiment analysis, paraphrase detection, and semantic text similarity. First, we implement a base BERT model that is able to output pooled embeddings. Then, using an array of implementations such as annealed sampling, cosine-similarity loss, different pooling strategies, and more, we see how these new methodologies can improve upon a minimal implementation of BERT.

3 Related Work

We implemented the baseline minBERT model using the details described in the Devlin et al. (2019) paper, and implemented the transformer architecture of minBERT using the equations described in Vaswani et al. (2023) and used the Adam optimization algorithm from Kingma and Ba (2017). We use the annealed sampling methodology as described in the Stickland and Murray (2019) paper to improve upon the existing round robin scheduling algorithm. We used the unsupervised contrastive learning approach described in the Gao et al. (2022) paper, which takes as input a sentence and predicts itself, with dropout as the noise between the sentences in the positive pair. The SimCSE model has shown improvement on Semantic Textual Similarity (STS) tasks, which is why we employ it as an extension to leverage its benefits. In addition, to improve performance on STS, we also use employ cosine similarity finetuning as described in the Reimers and Gurevych (2019) paper.

4 Approach

4.1 Baseline

For our baseline, we used our implementation of minBERT trained on the SST dataset. For sentiment classification, we simply passed in the BERT embeddings into a dropout and classifier layer. For paraphrase detection, we concatenated the two input ids with a $[SEP]$ token before getting the embeddings from the minBERT implementation. We then put these embeddings through a separate dropout and classification layer. Similarly, for semantic text similarity, we concatenated the inputs, used the minBERT embeddings with a dropout and classifier layer, then put the final input into a RELU layer multiplied by 5 to account for the necessary outputs. See Figure 1 for visual representations of our baseline architecture.

For sentiment analysis, we used cross entropy loss, for paraphrase similarity we used binary cross entropy loss, and for semantic text similarity, we used MSE loss. For our baseline minBERT, we used $[CLS]$ pooling.

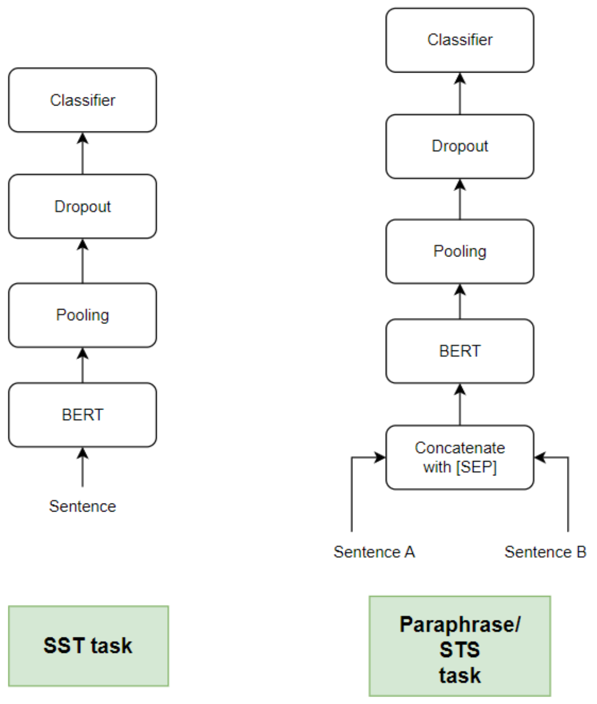

Figure 1: Architectures for each task

4.2 Pooling Strategy

Our baseline model utilized $[CLS]$ pooling, where the CLS embedding is used to represent the entire sentence. For one of our improvements, we experimented with **Mean Pooling** in which the mean of embeddings across the layer is taken and used. We hypothesized that this would improve performance for multitask as it would be able to adapt better for each specific task.

4.3 Individual Pretraining

In order to allow our model to be better at each of the downstream tasks we utilized the SST dataset for sentiment analysis, the Quora dataset for paraphrase detection, and the SemEval dataset for semantic text similarity. We individually pretrained each classifier layer over 3 epochs. Because the individual layers were not dependent on each other, we were able to pretrain each layer in parallel, cycling through each task for every training epoch. This method resulted in a marked improvement in performance. We wrote a function called `process_batch` which takes in a batch from training, then applies the predict function based on the task. With the results from the prediction, we used the respective loss function and stepped through the optimizer. Throughout the training, we saved the best version of the classifier model. At the end of training, we loaded each of the best models for a final evaluation.

4.4 Finetuning and scheduling

To further improve our model, we implemented finetuning, in which we unfroze all layers and allowed the BERT parameters to be updated. For our first trial, we used the standard Round Robin scheduling algorithm where each task was cycled through. However, we realized that with the uneven sizes of the datasets (With the Quora dataset being several times larger), this may result in an uneven distribution of training and subsequent task-level performance. Namely, the SST dataset had 8544 training examples and 1101 dev examples. The Quora dataset had 283010 training examples and 40429 dev examples. The SemEval dataset had 6040 training examples and 863 dev examples.

To mitigate this, we used a scheduling algorithm called "Annealed Sampling" as described in the Stickland and Murray (2019) paper. In this paper, the authors define the expression:

$$p_i \propto N_i^{\alpha}$$

where you choose a batch of examples from task i with probability p_i at each training step, and set p_i proportional to N_i or the number of training examples for task i. The value of $alpha$ is determined with the following expression:

$$\alpha = 1 - 0.8 \frac{e-1}{E-1}$$

where e is the current epoch and E is the total number of epochs. Using these expressions, weight each dataset with a probability, and have a sampler choose a task at random accordingly to train a batch. This allows the training to favor larger datasets less heavily and provide a more equal training opportunity across datasets.

4.5 Cosine Similarity

To improve performance on semantic textual similarity (STS) tasks, we implemented cosine similarity during finetuning when evaluating on the SemEval task. We did this by first getting the pooled embeddings of each sentence in the batch during the `predict_similarity` function. Then, we computed the cosine similarity between these two embeddings and return this value to the batch processing function. In our batch processing function, we scale this to fit the 0-5 scale the logits by doing `logits = 2.5 * (logits + 1)`, since cosine similarity returns a value between -1 and 1. Finally, we use the MSE loss function for calculating the train loss.

4.6 Contrastive Learning

When the SIMSCE flag is turned on, we calculate the contrastive loss for each task. We employ unsupervised SimCSE. When processing the batch, each sentence is passed through the minBERT model twice, which dropout acting as the noise between the positve pair. Then, we calculate the cosine similarity matrix and do matrix manipulations to calculate the contrastive loss according to the loss described in the Gao et al. (2022) paper.

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} {}_{[k \neq i]} \exp(\text{sim}(z_i, z_k)/\tau)} \tag{1}$$

for a mini-batch of N sentences where z_i and z_j form a positive pair.

4.7 Hyperparameters

Although we did not go into in depth testing with hyperparameters, we were able to find values for each argument that improved performance. For hidden dropout probability, we found that using a value of 0.2 was better than the default of 0.3. For epochs, we used 3 epochs for pretraining and 10 epochs for finetuning. For batch size, we raised it to 128 for pretraining and 32 for finetuning (The smaller size for finetuning was also due to the conda memory limitations).

4.8 Gradient Surgery

To improve training of our model, we incorporated gradient surgery, which divides and normalizes the gradient based on the formula below:

$$g_i = g_i - \frac{g_i \cdot g_j}{\|g_j\|^2} \cdot g_j$$

By doing so, we observed faster, smoother convergence that optimized the learning process and improved the overall training outcomes for our model.

5 Experiments

This section contains the following.

5.1 Data

We used the datasets provided in the default project handout. Sentiment analysis: SST dataset. Paraphrase detection: Quora dataset. Semantic Text Similarity: SemEval dataset.

5.2 Evaluation method

We used the evaluation metrics as described in the default project handout. Sentiment classification: Classification accuracy. Paragraph detection: Binary prediction accuracy. Semantic textual similarity: Pearson correlation value.

5.3 Experimental details

For our optimizer, we used the AdamW model with the same learning rate as the training method (1e-3 for pretraining, 1e-5 for finetuning). For our pretraining, we used a hidden dropout probability of 0.2, a learning rate of 1e-3, trained over 3 epochs, and used a batch size of 128. For finetuning, we used a hidden dropout probability of 0.2, a learning rate of 1e-5, finetuned over 10 epochs, and used a batch size of 32.

5.4 Results

For our part 1 implementation of minBERT, we were able to achieve results that were generally better than the reference accuracies (with the exception of the SST finetuned task).

Part 1 Accuracies

Model	SST pretrain	CFIMDB pretrain	SST finetune	CFIMDB finetune
minBERT	**0.409**	**0.788**	0.513	**0.971**
Reference	0.390	0.780	**0.515**	0.966

For our part 2 results, we first recorded the accuracies and pearson correlation scores achieved on the dev set. Our initial thoughts when running the baseline were that the paraphrase detection accuracy was higher than the other two, but we attributed this to the difference in dataset. Naturally, as the model is trained on more examples of paraphrase detection, it was able to do better.

Evaluation Accuracies on Dev set			
Model	**SST acc**	**Para acc**	**STS Pearson corr**
(Baseline) BERT + CLS pooling + Single task training	0.360	0.632	-0.129
BERT + CLS pooling + Individual pretraining	0.346	0.694	0.400
BERT + Mean pooling + Individual pretraining	0.460	0.755	0.651
BERT + Mean pooling + Individual pretraining + finetuning + Contrastive Learning (1 epoch only)	0.415	0.753	0.327
BERT + Mean pooling + Individual pretraining + finetuning + Gradient Surgery	0.461	0.760	0.499
BERT + Mean pooling + Individual pretraining + finetuning + Cosine Similarity	0.496	0.795	0.698
Evaluation Accuracies on Test set			
BERT + Mean pooling + Individual pretraining + finetuning (Round robin)	0.478	**0.831**	0.763
BERT + Mean pooling + Individual pretraining + finetuning (Annealed sampling)	**0.521**	0.803	**0.857**

We noticed that there was a vast increase in performance after implementing the Mean pooling. Between the CLS pooling and the Mean pooling models, there was an increase from 0.346 to 0.460 for SST accuracy, an increase from 0.694 to 0.755 for paraphrase detection, and an increase from 0.400 to 0.651 for the STS Pearson correlation score. We believe this means that the CLS pooling may have made the model ignore pieces of information from the rest of the pooled embeddings, at which the mean pooling would be able to combine the information in a better way. This also reflects that using Mean pooling allowed our model to be more generalizable across tasks. The two models that performed the best were the ones individually pretrained and finetuned. The model with the best paraphrase detection accuracy was the model finetuned using a Round robin scheduler. The model with the best performance for the other two tasks was the model finetuned using the annealed sampling scheduler. This made sense to us because the round robin sampling meant that the paraphrase detection dataset would get disproportionately more attention, while in the annealed sampling, all three datasets would be more even.

Across all tasks, the task with the most improved score was for semantic text similarity. The paraphrase detection tasks had the most consistent accuracy, likely due to its disproportionately large dataset. The SST task had a steady increase across each additional approach we used.

6 Analysis

We compared our model at each step against the baseline and achieved significant performance gains across multiple tasks. We approached the problem of improving the accuracy of our model by applying pre-training, fine-tuning, and implementing a variety of pooling strategies. Significant performance improvements came from implementing Mean Pooling, which outperformed CLS Pooling, on the task of semantic text similarity, an indication that Mean Pooling may capture more nuanced meaning than CLS Pooling. We also noticed through Annealed Sampling scheduling how the proportionality of each task and the size of the dataset influenced the performance of our model. Through carefully adjusting the model to fit the specific tasks, we were able to better balance learning and effectively prevented overfitting removing bias from training. Finally, through applying pre-training the model was better able to understand sentence structure and the specifics of each task before fine-tuning, which contributed to the detailed improvements in task handling which was absent from the baseline minBERT model. We significantly improved our model's ability to generalize well across datasets by including pre-training, whereas our fine-tuning approaches were more valuable for further improving accuracy for specific tasks.

An interesting result we noticed was that gradient surgery reduced the training time, but did not yield as positive results compared to other methods. Contrastive learning was only able to be run for 1 epoch due to memory and time limits but it showed results comparable to the gradient surgery model. Out of our dev results, we found that cosine similarity had the best performance but it did not improve upon our models that did not use cosine similarity.

7 Conclusion

We successfully developed and optimized the performance of our minBERT model achieving higher accuracy on a number of different tasks. Significant performance improvements on the SST dataset came from implementing the baseline configurations with single-task training. Further enhancements were achieved by pre-training, Mean Pooling, and Annealed Sampling scheduling. These techniques improved our model's ability to generalize and preform well on specific tasks, including sentiment analysis, semantic similarity, and paraphrase detection. Incorporating pre-training was particularly useful since it allowed our model to better understand the contextual relationships between words and to outperform the baseline minBERT during task specific testing.

Our results highlight the advantage of training a lightweight model in a resource constrained environment to perform optimally and efficiently. The significant performance improvements in accuracy, especially on sentiment analysis, showed the effectiveness of methods such as Mean Pooling and Annealed sampling scheduling compared to the minBERT model. While our minBERT model did perform well on sentiment analysis using significant lower computational resources and data, larger models tend to outperform in this task and would be ideal for situations in which the amount of compute is unconstrained.

Future work and improvements to our model, can come from incorporating larger and more diverse data into the pre-training phase to improve the model's ability to generalize and its understanding of sentence structures. In addition, including the latest pooling strategies and scheduling approaches might yield better accuracy and results during our task specific testing. Furthermore, we might include other measurements to compare the performance benefits from our extensions and to understand the specific advantages of applying one technique versus another.

Considering the harm a model like minBERT may cause, we might investigate how to ensure fairness by detecting and removing biases from the training datasets. By doing so, we would need to ensure the model is still retaining relevant information and not incorrectly classifying data, which would require a robust bias qualification criterion.

8 Ethics Statement

One ethical consideration regarding our project is the inherent bias that may be reflected in the predictions due to biases in the training set. Especially for datasets that are sourced from users online such as the Quora dataset, there may be implicit bias related to demographic information present. For example, if the examples in the data refer to or imply that certain occupations or roles should be performed by a certain gender, this may be reflected in the predictions made when asked about this topic. Additionally, since we do not do any preprocessing related to the actual information present, any biased examples may go unnoticed until manual evaluation afterwards. This issue is significant in particular to our model because as a multitask model, it must be able to perform well across a series of tasks, and if there is bias present, the predictions made by the model as a whole may be inaccurate or biased. In order to mitigate this, we propose an additional layer of preprocessing texts in which we mask or abstract away demographic information, such that it is not a factor in the embeddings and subsequent outputs. By abstracting away the demographic information, we could potentially remove the risk of predictions being made based on that data.

Another ethical consideration for our project is that none of the results are verified with additional sources. Since our model directly outputs a prediction for any inputs given, the user cannot see why the model gave the results that it did. This is an issue with our model because if someone provides an input to the model and gets an inaccurate solution, the user would not be able to tell whether the answer is a hallucination or accurate answer, and even if they were able to determine its inaccuracy, it would be impossible to determine why the model answered the way it did. In order to mitigate this issue from arising, we could implement a process similar to Retrieval-Augmented Generation, which refers to past training examples and can provide them along with the prediction. This means that the user could see the "source" of the solution and have a better idea of the accuracy of the solution. Of course, this method also relies on the accuracy of the training data and could potentially further exacerbate the issue by providing misleading source documents.

References

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

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2022. Simcse: Simple contrastive learning of sentence embeddings.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks.

Stickland and Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In *Association for Computational Linguistics (ACL)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need.