Use Siamese BERT-Networks to fine-tune minBERT with downstream tasks

Stanford CS224N Default Project

Zihan Yi

Stanford University
`yizihan@stanford.edu`

Abstract

Our goal is to investigate how to optimize a pre-established BERT model with contextualized embeddings and pre-trained weights to perform well simultaneously on multiple tasks involving sentences, such as analyzing sentiment, detecting paraphrases, and measuring semantic textual similarity. To accomplish this, we fine-tuned the model using a Siamese network architecture and experimented with different objective functions. As a result, we were able to improve upon the performance of the baseline model, achieving a percentile increase of 58%. Furthermore, we refined the hyperparameters and achieved an average score of 0.52, with 0.525 in sentiment analysis, 0.591 in paraphrasing, and 0.443 in semantic textual understanding.

1 Introduction

The Bidirectional Encoder Representations from Transformer (BERT) (Jacob Devlin, 2018) algorithm has achieved exceptional performance in many natural language processing (NLP) tasks. However, it is better suited for discrete label sentence classification tasks like sentiment classification and paraphrase detection, rather than pair regression tasks where there are too many possible combinations, which is the case in one of downstream tasks defined in the project. For example, evaluating the semantic textual similarity dataset needs to involve evaluating the similarity of sentences on a scale of 0 to 5 using Pearson correlation.

In this paper, We incorporated a siamese network (Reimers and Gurevych, 2019) using similarity measures such as cosine-similarity or Manhattan/Euclidean distance into the existing pretrained BERT architecture. Specifically, we implemented cosine similarity for the semantic textual similarity task and experimented with various concatenations between sentence embeddings. For the paraphrase detection task, we explored different loss functions like MSE loss, L1 loss, and cross entropy loss. Our multitask classifier training approach involved training all batches of a particular task at once, updating the model weights, and moving on to the next task with the same procedure.

2 Related Work

Language model pre-training has been the key to achieving recent top-level performance in NLP. Google's BERT (Jacob Devlin, 2018) model, which is built on the Transformer (Ashish Vaswani, 2017) architecture, is one of the most notable examples.

Despite its effectiveness, BERT still has some constraints in generating semantically significant sentence embeddings. The paper "Sentence-BERT: Sentence Embeddings Using Siamese BERT Networks" (Reimers and Gurevych, 2019) is trying to address the issue of generating high-quality fixed-length vector representations (i.e., embeddings) for variable-length sentences. To achieve this, they propose a novel approach that uses a Siamese neural network architecture with a pre-trained

BERT model to encode two sentences and compute their similarity based on the cosine distance between their embeddings. This approach is intended to overcome the limitations of traditional methods that rely on simple averaging or concatenation of word embeddings to represent sentences, which often fail to capture the complexity and nuance of natural language.

3 Approach

We trained baselines with sentiment analysis dataset (both SST and CFIMDB) and then finetuned using several techniques to better merge and incorporate the two downstream tasks (paraphrase detection and semantic textual similarity) into the existing pretrained BERT with sentiment analysis dataset.

3.1 minBERT setup and architecture

Our experiment commences by utilizing the code base of the minBERT project. The model comprises a standard tokenizing mechanism, a positional embedding layer, and a BERT transformer layer. The BERT transformer layer consists of a multiheaded self-attention sublayer, a position-wise feed-forward sublayer, and multiple normalization sublayers, as depicted in Figure 1 and Figure 2.

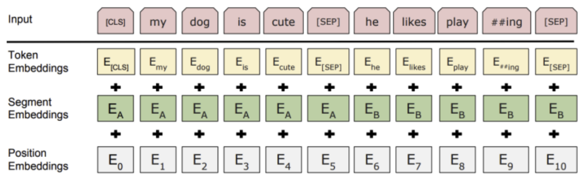


Figure 1: The BERT embedding layer generates input embeddings, which are created by adding the token embeddings, segmentation embeddings, and position embeddings. These input embeddings are then used in subsequent stages of the model. Figure from Jacob Devlin (2018)

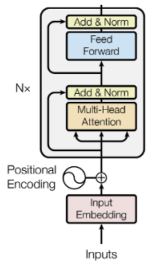


Figure 2: Encoder Layer of Transformer used in BERT. Figure from Ashish Vaswani (2017)

Prior to this implementation, BERT underwent training using two unsupervised tasks, namely masked token prediction and next sentence prediction, which were carried out on Wikipedia articles. Jacob Devlin (2018)

3.2 Baselines

Baselines are established by pretraining sentiment analysis task. The we finetune the sentiment classifier along with the paraphrase detection and the semantic similarity task. The way we set up the two sentence-pairing prediction functions is by simply just adding a dropout layer after getting the results returned by the `forward()` function of each sentence, and put the two sentences dot products together, then applying a linear layer projection to `dim=1` and return the logits back. We use softmax as the loss function for all three of the tasks.

3.3 Fine-tuning paraphrase detection task

According to the SBERT paper, the classification objective function can be written as follows:

$$o = softmax(W_t(u, v, |u - v|)) \tag{1}$$

In Figure 3, we illustrate the process in which we combine the sentence embeddings u and v. We first take the element-wise difference $|u - v|$ and concatenate it with u and v. The resulting vector is then multiplied by a trainable weight $W_t \in \mathbb{R}^{3n \times k}$, with n representing the dimension of the sentence embeddings and k being the number of labels. We optimize the cross-entropy loss in this structure.

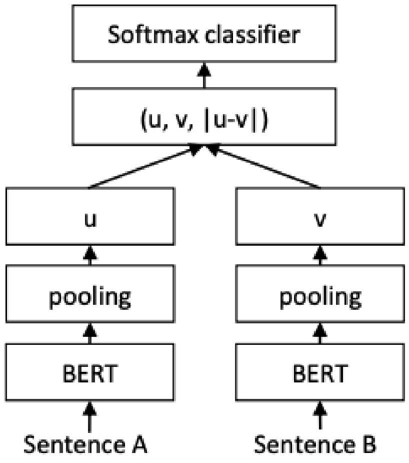
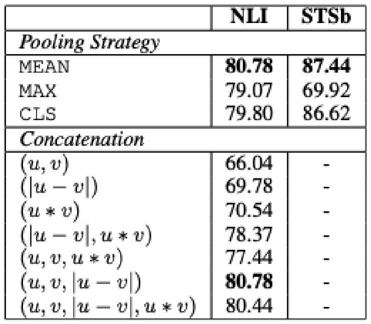


Figure 3: SBERT architecture with classification objective function. Figure from Reimers and Gurevych (2019)

Keeping the pooling strategy by default `MEAN`, we evaluated multiple methods of combining sentences, including dot product $(u \cdot v)$, concatenations of sentence embeddings with dot product $(u, v, u \cdot v)$, etc. Eventually, we found that using the element-wise difference along with sentence embeddings produced the highest F1 accuracy score for paraphrase detection. The finding also matches the paper's sentence pairing task result with different concatenation strategies as figure 4 shown below.



	NLI	STSb
Pooling Strategy		
MEAN	**80.78**	**87.44**
MAX	79.07	69.92
CLS	79.80	86.62
Concatenation		
(u, v)	66.04	-
$(\|u - v\|)$	69.78	-
$(u * v)$	70.54	-
$(\|u - v\|, u * v)$	78.37	-
$(u, v, u * v)$	77.44	-
$(u, v, \|u - v\|)$	**80.78**	-
$(u, v, \|u - v\|, u * v)$	80.44	-

Figure 4: SBERT trained on NLI data with the classification objective function. Figure from Reimers and Gurevych (2019)

3.4 Fine-tuning semantic textual similarity task

In the paper, for regression tasks where there are infinite possible outputs (in our case it's the STS task), it is recommended to calculate the cosine similarity between sentence embeddings u and v (as shown in Figure 5) and to use the mean squared-error loss as the objective function. We have also investigated alternative loss functions, such as L1 loss and hinge loss.

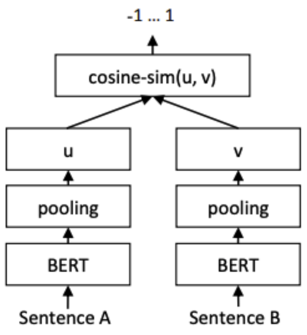


Figure 5: The SBERT architecture can be utilized during inference to calculate similarity scores, and it can also be applied with a regression objective function. Figure from Reimers and Gurevych (2019)

4 Experiments

4.1 Data

We used the standard dataset provided by the CS224n project.

For sentiment analysis, we use the Stanford Sentiment Treebank (SST) dataset and split the 11855 single sentences into train/dev/test as 8,544/1,101/2,210.

For paraphrase detection, we use the Quora dataset which consists of 400,000 question pairs with labels indicating whether particular instances are paraphrases of one another. The data is split into train/dev/test as the number 141,506/20,215/40,131.

For semantic textual similarity task, we use the SemEval STS Benchmark dataset which consists of 8,628 different sentence pairs of varying similarity on a scale from 0 to 5. The exact number for train /dev/test are as follows: 6,041/864/1,726.

4.2 Evaluation method

Given the discrete labels of both SST and Quora Dataset, the metric that we utilize to test those two datasets are simply accuracy. For SemEval dataset, we calculate the Pearson correlation of the true similarity values against the predicted similarity values across the test dataset. The correlation will be normalized to a value between 0 and 1. Finally, we average out all three evaluation values from the three tasks to get the overall accuracy.

4.3 Experimental details

All fune-tuning was done on the given dataset above, with one exception that we found the Quora dataset 141,506 to be way heavy to train as it would take around 5 hours to run through the training set for just one epoch, thus we use pytorch's `WeightRandomSampler` to selectively train with 1/4 of the entire training set, we also halve the dev set (around 10,000) data to match the training and dev ratio. The way we calculate the weight is simply iterating over the entire dataset, get each label's count and put them into the weight array.

The model was trained for 5 epochs using the uncut SST and SemEval datasets, along with a 1/4 portion of the Quora dataset. During each run, which also involved evaluation, it took approximately 3 hours on an Nvidia A10G Tensor Core GPU. The default project settings were initially used for other parameters, including a learning rate of 1e-3 with an exponential weight rate starting at 0.0, a batch size of 8, and a dropout rate of 0.5.

4.4 Results

We first obtained baselines by performing the above training procedure on pre-trained BERT. Then we finetuned by running 3 epochs on multitask classifier with a naive softmax loss function for each task. The results are shown in Table 1.

	SST	Quora	STS	Avg.
training accuracy / Pearson correlation	0.627	0.461	0.124	0.404
dev accuracy / Pearson correlation	0.52	0.378	0.089	0.329

Table 1: minBERT multitasks baseline results

Next, we examined different sub-approaches for each individual approach mentioned and chose the one that can best enhance the performance of a single task as the strategy for multitask training.

	$(u \cdot v)$	$(u, v, u \cdot v)$	$(u, v, \|u - v\|)$
training acc.	0.52	0.66	0.71
dev acc.	0.43	0.58	0.63

Table 2: Comparing paraphrase detection task's various concatenations strategies, u and v represents two input sentences' embeddings. F1/EM score calculated based on single task performance running after 5 epochs

	MSE loss	L1 loss	Cross-Entropy
training Pearson sim.	0.78	0.66	0.3
dev Pearson sim.	0.62	0.57	0.24

Table 3: Comparing STS task's various loss functions performance

Next, we fine-tuned the BERT by summing up all the optimized strategies that have been applied on each task, after running x epochs, this is the final result we reached (the result in test leaderboard)

	SST	Quora	STS	Avg.
training accuracy / Pearson correlation	0.866	0.629	0.860	0.785
dev accuracy / Pearson correlation	0.525	0.591	0.443	0.520

Table 4: submitted final minBERT multitasks fine-tuned results

5 Analysis

5.1 Qualitative Evaluation

Based on the baseline results, the performance of Quora and STS in terms of accuracy was not optimal. Particularly, when examining the STS dataset, the development accuracy was only 0.08, indicating that the model was not learning efficiently. This was because the baseline STS used the Cross-Entropy loss function, which is suitable for binary/multiple choices classification but not for a regression task like the STS dataset, which is scored continuously between 0 and 5. Additionally, even though L1 loss and L2 loss showed comparable performance, L2 loss was selected because it could provide more non-linearity.

In terms of the concatenation strategy for the paraphrase detection task, both the dot product and absolute distance methods of combining the sentence embeddings demonstrated equally strong performance. This could be attributed to the fact that the tuple contains more information than a singleton, which may be the underlying reason for their similarity in performance.

The final accuracy result shows a slight decrease in accuracy when compared to training the model on a single task, which was expected because the model needed to find a balance between all three tasks. We also observed that when trained on the full Quora dataset, the model not only took an incredibly long time to train, but it also showed a bias towards paraphrase detection. This resulted in an imbalance and a significant decrease in the accuracy of the other two tasks. To address this issue, we decided to train the model on only 1/5 of the original data, which led to a more balanced and better overall averaging accuracy.

Finally, we observed that the accuracy of the sentiment analysis and paraphrase detection tasks was oscillating during training when using the default dropout rate for the classifier layer. We believe that this issue may be due to the neural network's learning, given the existing relatively small learning rate of 1e-5. To prevent early overfitting, we increased the dropout rate from 0.3 to 0.5.

6 Conclusion and Future Work

To sum up, we utilized various techniques outlined in the Siamese networks paper to fine-tune the multitask model, resulting in an overall F1 and EM score that is 58% outperforming the baseline model. Additionally, we were able to maintain the accuracy of sentiment analysis compared to the pretrained model that solely focused on the sentiment analysis task.

Future work might include extending the amount of time spent on fine-tuning or tweaking the hyper-parameters, which we limited in this project due to time constraints. Given the existing large Quora dataset and the fact that both paraphrase detection and STS task need to leverage the similarity between sentence pairs, we can also potentially speed up the training time by sharing the layers and networks of two together. This would allow the STS prediction model to take advantage of the vast Quora dataset.

References

Niki Parmar Jakob Uszkoreit Llion Jones Aidan N. Gomez Lukasz Kaiser Illia Polosukhin Ashish Vaswani, Noam Shazeer. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

Kenton Lee Kristina Toutanova Jacob Devlin, Ming-Wei Chang. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *arXiv preprint arXiv:1810.04805*.

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 Languages Processing and the 9th International Joint Conference on Natural Languages Processing (EMNLP-IJCNLP)*.