BERT Extension Using Sentence-BERT for Sentence Embedding

Stanford CS224N Default Project

Anicet Dushime Wa Mungu
Department of Computer Science
Stanford University
dushime@stanford.edu

Abstract

The aim of this project is to implement extensions to improve the performance of the BERT model in the three downstream tasks. To do this, I first implemented a round-robin approach that combines the losses of the three tasks so that the model trains and improves on all three at the same time: sentiment analysis, paraphrase detection and semantic textual similarity. Afterwards, I targeted improving the underlying representations of the sentence embeddings which are the backbones used in the three tasks, by implementing a siamese network. The idea is that, since these tasks depend on the embeddings produced by BERT, improving these embeddings would improve the performance on these tasks, with other modifications. The project specifically targeted the semantic text similarity task, although as we will come to see, I generally got an improvement in all three tasks.

1 Key Information to include

- Mentor: Olivia Lee

2 Introduction

The rise of Large Language Models has been revolutionary in improving the performance of NLP models in a variety of tasks. Specifically, BERT is a pretrained transformer-based language model that has set new state of the art results in various fields like question answering, sentence classification as well as sentence-pair regression. One way to apply it to sentence pair regression tasks like semantic text similarity, is to pass the two input sentences separated by the special [SEP] token and applying multihead attention, then applying a regression function to obtain the label. While this approach did achieve impressive results, the big drawback was time, since you would have to pass every pair of sentences to the model. One solution to this is coming up with sentence embeddings, which is motivated by existing approaches like GloVE embeddings, which are vector representations of tokens. Having vector representations of sentences would make the comparisons very fast and efficient through approaches like Cosine Similarity. This has had success, through the use of the special [CLS] token, for instance, but has generally performed worse than alternative methods like averaging GloVE embeddings. This is the main motivation for this project; fine tuning the BERT model to be able to generate better sentence embeddings that would perform much better than simply using the [CLS] tokens or averaging GloVE embeddings. While this project focuses specifically on BERT and BERT embeddings, we will see that BERT alternatives like ALBERT and RoBERTa can also be used as the underlying models for the implementation of the siamese network in this project, and would also achieve better sentence embeddings for downstream tasks.

3 Related Work

In this section, we give the background or related work for the extensions we use in our project. Specifically, we provide a brief introduction to BERT, and sentence-BERT that is based on cosine-similarity.

3.1 BERT

BERT is a breakthrough pre-trained language model introduced by Devlin et al. in their seminal paper "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" in 2018. The model architecture is based on the transformer architecture, comprising multiple layers (12-24) of self-attention mechanisms. Unlike previous models, BERT learns bidirectional representations of text by pre-training on large corpora with masked language modeling (MLM) and next sentence prediction (NSP) objectives. By fine-tuning BERT on downstream tasks, it achieves state-of-the-art results across various natural language processing tasks due to its ability to capture deep contextual embeddings and understand the semantics of text more effectively.

3.2 Sentence-BERT

Sentence-BERT is an implementation on top of BERT, introduced by (Reimers and Gurevych, 2019) in their paper "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". The model architecture involves creating a siamese network using BERT and finetuning the model to enable it to come up with meaningful sentence embeddings so that methods like cosine similarity can be applied. Cosine similarity itself can capture the semantic similarity of two sentence embeddings by getting their directional similarity. Therefore, using cosine similarity in this siamese network trains the model to come up with close or similar embeddings for sentences that bear similar semantic meaning.

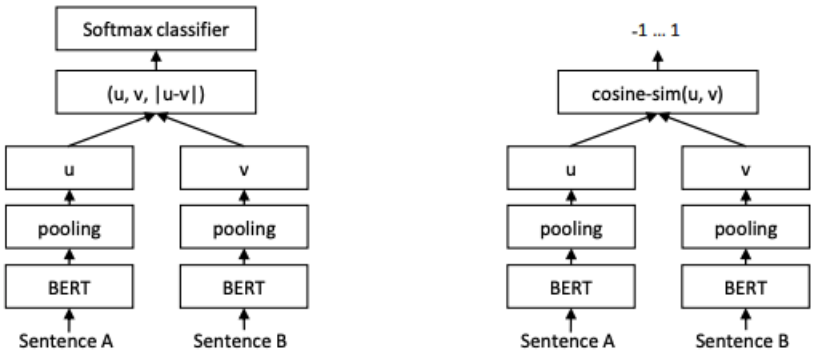

Figure 1: Structure of SBERT (Reimers and Gurevych, 2019)

4 Approach

I implemented the BERT model with the following additions: a round-robin multitask training, and Sentence-BERT siamese network in the Semantic Text Similarity section (above). I describe the different layers before as well as the different approaches I tried for the layers, and will discuss the results under Experiments.

4.1 Forward layer

For sentiment analysis, the forward layer simply involved extracting the BERT embeddings of the [CLS] token, a dropout layer and a linear layer.
For paraphrase detection, the forward layer involved extracting the BERT embeddings of the [CLS]

token, a dropout layer and a linear layer for each of the two input sentences, followed by a dot product to calculate similarity ie: for $output1$ and $output2$,
$similarity = torch.diagonal(torch.mm(output1, output2.t()))$. For Semantic Text Analysis, I took a few different approaches, whose results I will discuss below:
extracting the BERT embeddings of the [CLS] token, a dropout layer and a linear layer for each of the two input sentences, followed by:

1. Dot product to calculate similarity
2. Cosine similarity to calculate similarity
3. Classification objective function: Concat($W_t(u, v, |u - v|)$) as described in (Reimers and Gurevych, 2019). This added a linear layer W_t.
4. Regression Objective Function : cosine_similarity(u, v) followed by Mean Squared Error Loss.

4.2 Training

Round-robin approach, which trains all the three downstream tasks simultaneously by accumulating their individual losses in each step and doing the backward step on the accumulated loss from the three tasks. This is important because the alternative of training one tasks followed by the other lead to dismal performance, potentially due to the model forgetting the task being trained first when training the next one.

5 Experiments

5.1 Data

I used the following datasets for training and evaluation:

1. SST which comprised a sentence and a sentiment score between 1-5
2. Quora which was composed of two sentences and a paraphrase score of 0 or 1
3. STS dataset which was composed of two sentences and a similarity score between 0 and 5

5.2 Evaluation method

For evaluation, I used prediction accuracy for Sentiment analysis and Paraphrase detection, and used Pearson correlation for Semantic Text Similarity. I also included the Spearman rank correlation as suggested in (Reimers and Gurevych, 2019). However, it ended up being consistent with the Pearson correlation scores.

5.3 Experimental details

I had the following variations in my models:

1. Pretrain-only without Round-Robin
2. Pretrain-only with Round-Robin
3. Finetuned with Round-Robin
4. Finetuned with Classification objective function
5. Finetuned with Classification OF and scaled logit (0-5)
6. Finetuned with Cosine similarity as feature of Linear Layer and dropout after Cosine similarity
7. Finetuned with MEAN pooling
8. Finetuned with MEAN pooling without dropout and mapping cosine to (0-1)

9. Finetuned with Regressive Objective Function

As a baseline, I pretrained for the three downstream tasks one after another (without Round Robin). Pretraining would also not modify the frozen BERT parameters and therefore use preexisting embeddings.
As another baseline, I implemented Round Robin for the tasks but without changes to the architecture so the embeddings did not change here either..
I then went on to experiment on a couple of different finetuning appraoches, like using the classification objective function, the regressive objective function, using MEAN pooling instead of the [CLS] token, scaling the logits to match the range of the labels, moving the dropout layer to after the cosine similarity operation, using the cosine similarity value as a feature in the linear layer, and removing the dropout layer entirely.
The best result came from using the cosine similarity value as an input to the linear layer while applying the dropout after calculating the cosine similarity, all with scaled logits to match the labels.

5.4 Results

Method	Sentiment	Paraphrase	STS	Overall
Pretrain (No Round-Robin)	0.144	0.685	0.097	-
Pretrain (Round-Robin)	0.316	0.657	0.125	0.512
Finetuned (Round-Robin)	0.452	0.717	0.293	0.605
Classification Obj. Fn (W_t)	0.504	0.740	0.276	0.627
Scaled Logits to match labels	0.521	0.737	0.357	0.646
Cosine similarity as feature of W_t	0.512	0.729	0.466	0.657
MEAN pooling instead of [CLS]	0.510	0.733	0.397	0.647
No dropout and 0-1 Cosine range	0.518	0.738	0.417	0.655
Regression Obj. Fn	0.510	0.739	0.372	0.645

Table 1: Experimental Results

Across the board, the accuracy and STS correlation increases with the implementation of Round-Robin training. Furthermore, the inclusion of cosine similarity as a feature of the linear layer (W_t) also increases the STS correlation score, as expected.
The performance is generally as I expected in comparison to the chosen baselines.
The one unexpected observation was that using MEAN pooling in place of the [CLS] token performed worse.

6 Analysis

6.1 Hypotheses

Going into the project, I hypothesized that:

1. The general performance in Semantic Text Similarity would improve drastically.
2. I did not expect to see that big of a difference in terms of training for one task at a time versus Round Robin.

6.2 Results

Siamese network implementation on BERT is specifically targetted at producing meaningful sentence embeddings. Therefore, even though in (Reimers and Gurevych, 2019), the emphasis is on STS tasks, since the underlying embeddings are what are used by the three downstream tasks, it makes sense that improving these embedding representations through finetuning would generally lead to an increase in performance in all three tasks. It also appears that the no one task was penalized or rewarded excessively, since from the baselines, the underlying model seemed to perform better in Paraphrase detection, followed by Sentiment, and performed the worst in Semantic Text Similarity. After the implementation of the Siamese network, even though all three tasks improved, the performance

in paraphrase detection was still higher, followed by Sentiment Analysis and then Semantic Text Similarity.

I also observed a significant gain in performance between the two baselines just from implementing the Round Robin training. This also makes sense with the view that training one task at a time gets very good at that task, but when the model moves to train on the next task, it in a way ovewrites the learned parameters of the previous task. Combining the losses and doing the backward step for the three tasks simultaneously expectedly optimizes the performance of all three at the same time.

Your report should include *qualitative evaluation*. That is, try to understand your system (e.g., how it works, when it succeeds and when it fails) by inspecting key characteristics or outputs of your model.

7 Conclusion

I was able to implement Sentence-BERT to the BERT model and observe significant performance improvement in three tasks: paraphrase detection, sentiment analysis and Semantic Text Similarity due to the improvement of the underlying sentence embeddings.

7.1 Future Work

In my implementation, I used the learning rate $1e-5$ for pretraining and $1e-3$ for finetuning, 10 epochs, a batch size of 8 consistently and a 0.3 dropout probabillity. There is room to experiment with different hyperparameters to achieve higher improvements. Another area of work would be to integrate more data for the finetuning tasks since the 3 datasets used in this project were possibly not sufficient to finetune the parameters to the best extent possible.

References

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 and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.