Multitasking with minBERT

Stanford CS224N {Default} Project

Jung-Suk Lee
Department of Computer Science
Stanford University
robin09@stanford.edu

Abstract

In this project, a variant of BERT, referred to as minBERT, is developed for multiple tasks of natural language processing (NLP) . The developed minBERT can be trained for both single-task and multi-task learning. Three NLP tasks of sentiment classification, paraphrase detection and semantic textual similarity prediction are targeted with minBERT. We found that the minBERT in the setting of the multi-task learning that is trained in the finetune mode outperforms the minBERTs trained in the single-task setup for two out of the three downstream tasks. The minBERT achieved scores of 0.542 accuracy for sentiment classification, 0.827 accuracy for paraphrase detection and 0.864 Pearson correlation for similarity prediction.

1 Key Information to include

- Mentor: CAs of cs224n

2 Introduction

Bidirectional Encoder Representations from Transformers (BERT) is a framework that generates contextual embeddings of its input. Originally it was devised for natural language processing (NLP) applications but it did not take long for BERT to start influencing other domains due to its powerful pretrain-finetune concept. In this project, our goal is mainly to experience how well the concept of pretrain-finetune works for NLP tasks by implementing a variant of BERT, referred to as minBERT, and experience the procedure of pretrain-finetune for NLP downstream tasks. To this end, we first implemented minBERT and trained it for the sentiment classification task (SST) and witnessed that a pre-trained BERT for masked language modeling can be tweaked to perform (well) on a different downstream task once trained further, even with a smaller dataset, in the finetune mode. In addition, we extended our minBERT for two more downstream tasks: paraphrase detection (PARA) and semantic textual similarity prediction (STS) . We designed individual layers that are placed on top of minBERT for each extra downstream task and conducted training in the pretrain and finetune modes. We further extended minBERT to enable multi-task training with which different datasets for different downstream tasks can be used for training minBERT together, by feeding batches to minBERT from each different dataset in a round-robin manner. We observed that minBERT trained in the multi-task setup performs as good as, or better than when trained in the single-task mode for each task.

3 Related Work

In Devlin et al. (2018), BERT is proposed as a transformer Vaswani et al. (2017) based encoder that can generate 'contextual' representations of tokens given a pair of sentences as input. BERT advertised the potential of pretrain-finetune procedure by applying BERT model pretrained on the masks language model and the next sentence prediction tasks to other various NLP tasks with finetuning. Due its nature, it is natural to consider multi-task learning scheme leveraging BERT. In Liu et al. (2019) a BERT model pretrained as a language model is used as part of the multi-task learning for various NLP tasks. The pretrained BERT model is shared by additional task-specific layers for different NLP tasks and jointly finetuned with task-specific datasets. The proposed multi-task learning scheme was able to obtain state-of-the-art results on ten NLP tasks.

4 Approach

4.1 Baseline minBERT

As the baseline model, minBERT is implemented following the guideline in cs224nStaffs and the code template provided therein, which is originally designed to conduct SST task when finetuned with relevant datasets. The input to the baseline minBERT is a set of tokenized sentences. The token embedding and the position embedding of each input token are summed and fed to the encoder part of the model. BERT model parameters pretrained for the mask language model are provided with the code template. The output of the baseline minBERT is a set of contextual embedding whose element has the dimension of $M \times K$ where M is the number of input tokens and K is the size of a single contextual embedding.

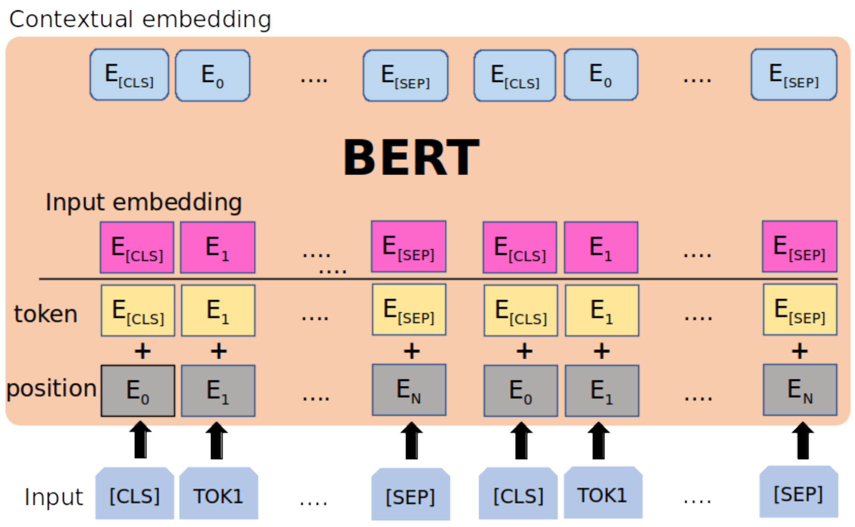

Figure 1: Baseline BERT model

4.2 Layer for sentiment classification task

For the sentiment classification task, a single sentence is passed to minBERT as input and the contextual embedding of CLS (pooler output) goes through a dropout layer and a linear layer to generate the output of dimension ($N \times C$) where N is the batch size and C is the number of classes (labels) that the groundtruth data spans over. The cross-entropy is used for training as the loss function.

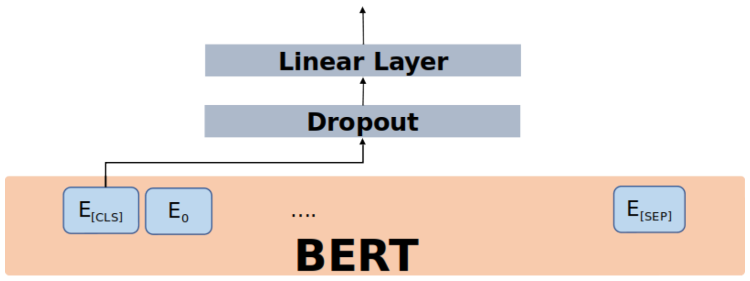

Figure 2: Layer for sentiment classification task

4.3 Layer for paraphrase detection task

For the paraphrase detection task, a pair of sentences are passed to minBERT as input. The mean of contextual embeddings is first taken separately for each sentence and the concatenation of the two means gives the intermediate variable of dimension ($N \times 2K$). After the dropout layer, the linear layer and the Sigmoid function, output of dimension ($N \times 1$) is generated. The binary cross-entropy is used as the loss function for which the output of this task-specific layer and a binary groundtruth label are taken as input.

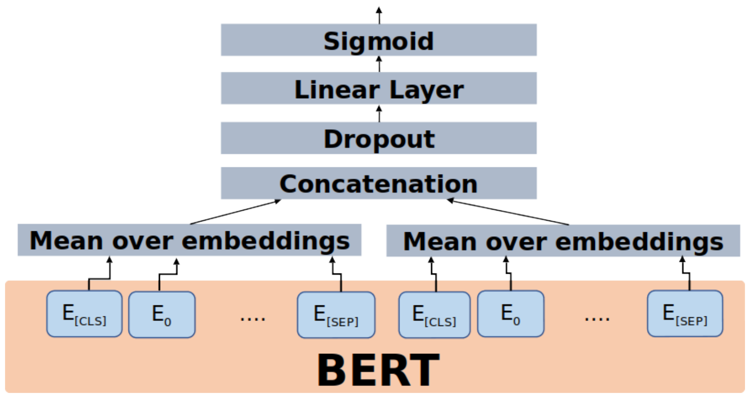

Figure 3: Layer for paraphrase detection task

4.4 Layer for semantic text similarity prediction task

For the semantic text similarity prediction task, a pair of sentences are passed to minBERT as input. The contextual embeddings of CLS from each sentence are concatenated together and passed to the dropout layer, the linear layer and the Relu function sequentially. The final output of this taks-sepcific layer is of dimension ($N \times 1$). The mean square error (MSE) between the output and the groundtruth label is then computed as the loss for training Liu et al. (2019).

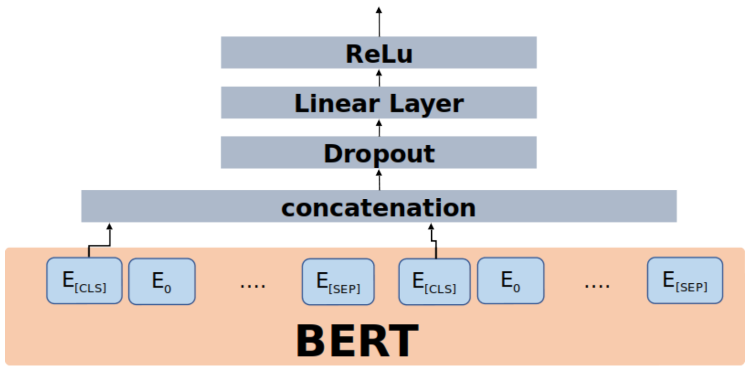

Figure 4: Layer for semantic text similarity prediction task

4.5 Multi-task training scheme

In order to facilitate multi-task learning with minBERT for three downstream tasks of SST, PARA and STS, a multi-task training scheme that can jointly take task-specific datasets is devised and implemented following Liu et al. (2019). The $D_{SST}, D_{para}, D_{STS}$, the data sets for STS, PARA and

STS tasks, respectively, are merged as one unified dataset D as following:

$$\begin{aligned}D = [&b_{SST}(1), b_{para}(1), \cdots b_{para}(R_1), b_{STS}(1), \cdots, b_{STS}(R_2),\\ &b_{SST}(2), b_{para}(R_1+1), \cdots, b_{para}(2R_1), b_{STS}(R_2+1), \cdots, b_{STS}(2R_2)\\ \cdots\\ &b_{SST}(B_{SST}), b_{para}(B_{SST}R_1 - R_1 + 1), \cdots, b_{para}(B_{SST}R_1), b_{STS}(B_{SST}R_2 - R_2 + 1), \cdots, b_{STS}(B_{SST}R_2)]\end{aligned} \quad (1)$$

where $b_{SST}, b_{para}, b_{STS}$ denote single batches from each data set, respectively. $B_{SST}, B_{para}, B_{STS}$ are the numbers of mini batches for each task, respectively. $R_1 = floor(\frac{B_{para}}{B_{SST}})$, $R_2 = floor(\frac{B_{STS}}{B_{SST}})$[1] are the ratios of B_{para}, B_{STS} relative to B_{SST} with the assumption that B_{SST} is the smallest number among others. The way that the batches from each dataset are defined and distributed in D to address the discrepancy of minibatch numbers is proposed by us based on Liu et al. (2019). In training, all of the minibatches in D are swept for each epoch. At each iteration on minibatches, the loss function associated with the minibatch type is computed and the model parameters are updated through backpropagation. Disproportionate weightings that are inverse of R_1, R_2 are applied to loss functions to mitigate the bias arising from discrepancy in the number of minibatches. This way, different task-specific datasets can be jointly used for the multi-task training.

5 Experiments

5.1 Data

Details about the datasets for each downstream task conducted for the project can be found in Table (1).

Name	Task	size(train/dev/test)	type of groundtruth
Stanford Sentiment Treebank Dataset	SST	8544/1101/2210	Discrete 5 points
CFIMDB Dataset	SST	1701/245/488	Binary
Quora Dataset	PARA	141506/20215/40431	Binary
SemEval STS Benchmark Dataset	STS	6041/864/1726	Discrete 6 points

Table 1: Datasets for downstream tasks

5.2 Evaluation method

For SST task and PARA task, *Accuracy* is computed as the metric to evaluate the performance of the model. *Accuracy* is defined as

$$Accuracy = \frac{\sum_n^{N_{total}} \mathbb{1}(\hat{y}(n) = y_{true}(n))}{N_{total}} \quad (2)$$

where $\hat{y}$ and y_{true} denote the model prediction and the corresponding groundtruth label of the n^{th} sample. N_{total} is the total number of samples in the dataset. For STS task, the Pearson correlation (will also be referred to as the *similarity score* henceforth) between the total model predictions and the groundtruth labels is used as the metric for evaluation.

5.3 Experimental details

We have ran experiments in the single-task as well as the multi-task setup both in pretrain and finetune modes for all three downstream tasks. Hyperparameters used for each training case can be found in Table (2). For all training cases, AdamW optimizer was employed to update the model parameters. All of the cases in Table (2) were run with 10 epochs.

[1]Due to the floor operation, certain amount of data are undesirably discarded. If the ratios are small or the size of the datasets are not large enough, then this scheme could be problematic.

Task	Dataset	Training scheme	Training mode	Learning rate	Batch size
SST	CFIMDB	single-task	Pretrain	1e-3	8
SST	Sentiment Treebank	single-task	Pretrain	1e-3	64
PARA	Quora	single-task	Pretrain	1e-3	16
STS	SemEval	single-task	Pretrain	1e-3	16
SST	CFIMDB	single-task	Pretrain	1e-5	8
SST	Sentiment Treebank	single-task	Finetune	1e-5	64
PARA	Quora	single-task	Finetune	1e-5	16
STS	SemEval	single-task	Finetune	1e-5	16
SST	Sentiment Treebank	multi-task	Pretrain	1e-3	64
PARA	Quora	multi-task	Pretrain	1e-3	16
STS	SemEval	multi-task	Pretrain	1e-3	16
SST	Sentiment Treebank	multi-task	Finetune	1e-5	64
PARA	Quora	multi-task	Finetune	1e-5	16
STS	SemEval	multi-task	Finetune	1e-5	16

Table 2: Hyperparameters for training

5.4 Results

Metric scores of each training case can be found in Table (3). It is shown that training in the finetune mode yields better scores then the pretrain mode for all tasks, for all training schemes. This is somewhat expected as it has been shown by many research works that BERT-based finetuning results in better performance in many NLP tasks by thoroughly leveraging pretrained BERT core . It is also shown from the experiment results that the multi-task training generates scores as good as, or higher then the single-task training, as opposed to the expectation that the results from the signal-task training would be as good as, or higher than the multi-task training.

Task	Dataset	Training scheme	Metric Score (pretrain/finetune)
SST	CFIMDB	single-task	0.722 / **0.959**
SST	Sentiment Treebank	single-task	0.382 / 0.53
PARA	Quora	single-task	0.761 / **0.885**
STS	SemEval	single-task	0.615 / 0.852
SST	Sentiment Treebank	multi-task	0.256 / **0.542**
PARA	Quora	multi-task	0.760 / 0.827
STS	SemEval	multi-task	0.001 / **0.864**

Table 3: Scores of each training case. Scores in boldface are the best scores for each dataset.

6 Analysis

Overall, it was possible to observe that finetuning improves the performances of all downstream tasks, regardless of training scheme. It is encouraging to see that the performance of the multi-task setup is as good as, or better than the performance of the single-task setup in the finetune mode. However in the pretrain mode, this was not necessarily the case. The scores obtained from the single-task setup are higher then those from the multi-task setup. It would mean that conducting training only on the tasks-specific layers is not good for the multi-task setup. Once parameter update is allowed for the BERT layers as well as the task-specific layers in the finetune mode, the benefit of multi-task setup starts appearing. We also noticed that in the multi-task setup, STS prediction results in scores that are quite off from those from the single-task setup when trained in the pretrain mode. The best similarity score we had obtained was 0.001 out of 10 epochs while for some epochs the similarity scores were negative values and for some epochs the scores were NaN (Not a Number). Especially in the case of NaN score values, we noticed that the predicted values from the model were all zeros, causing division-by-zero when computing the Pearson correlation. However when in the finetune mode, these undesirable STS scores were not observed in all epochs.

7 Conclusion

We have implemented minBERT and extended it to conduct multiple NLP downstream tasks. A multi-task training scheme is also proposed and implemented, with which different task-specific datasets can be jointly trained. Our model can carry out three different NLP tasks both in the single-task and the multi-task training setup and the results from the multi-task training setup are shown to be as good as, or better than those from the single-task training setup. Our model revealed its limit when the model is trained in the pretrain mode for multi-task learning, especially a noticeable failure in conducting STS task. However in the finetune mode, all three downstream tasks are successfully conducted in the multi-task setup.

References

cs224nStaffs. Cs 224n: Default final project: minbert and downstream tasks.

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

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.