CS 224N written report:

Member: Kaushik Sampath, kau2002@stanford.edu

Project: Default

Title: BERT downstream task training utilizing different pooling methods

Abstract

The Bidirectional Encoder Representations from Transformer model (BERT) is a masked language model that uses masking in order to learn latent representations to perform downstream tasks such as next sentence prediction, paraphrase detection, or text classification. In this project, I attempt to implement the BERT model while making additional finetuning in order to create a transformer that performs better on a set of downstream tasks like semantic textual similarity, paraphrase detection, and sentimental classification than the regular BERT model. To improve the expressiveness of the BERT embeddings, I attempt to utilize different layer pooling techniques, along with different task-specific loss functions, in down-stream task predictions in order to improve training and test accuracies on novel data.

Introduction

As explained before, BERT is one of the pre-eminent language transformer models in all of Natural Language processing. BERT's architecture contains a tokenizer, embedding layer, transformer layer, and multi-headed self attention. The output of Bert consists of a contextualized embedding for each wordpiece from the last BERT Layer and a CLS token embedding, which represents a sentence-level embedding. In order to obtain new embeddings after each layer, we run our previous layer embeddings through another transformer that consists of a multi-headed attention layer, a normalization layer, a feed-forward layer, and then once again in an add-norm layer. This is shown below in the following figure.

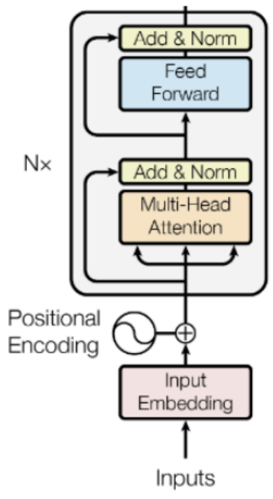
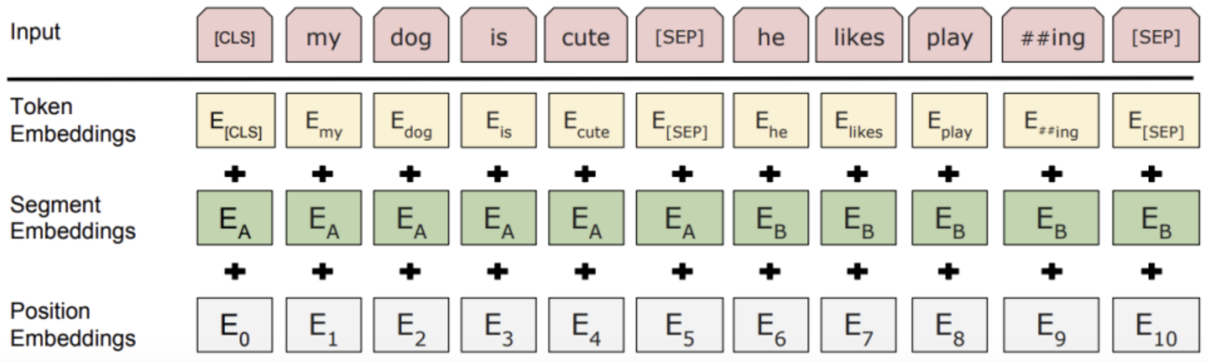


Figure 2: Encoder Layer of Transformer used in BERT. Figure from [5]

The multi-head attention layer is used to calculate the attention score vector in order to map a query between a set of key-pair values.

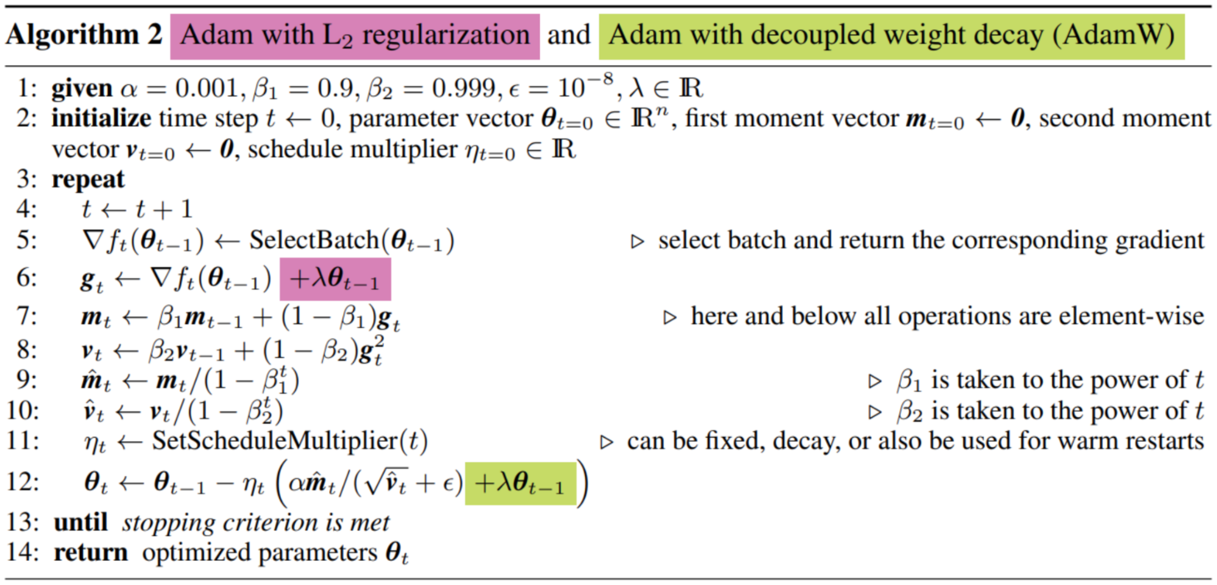

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Our inputs also use token, segment, and position embeddings in order to get a total embedding that gives expressiveness in order to differentiate amongst different sentences. While the BERT model gives a solid baseline for sentence representation and embeddings, it struggles for high accuracy on certain downstream tasks like sentiment analysis, paraphrase detection, and semantic textual similarity. In order to fix this problem, I investigate different ways in which sentence embeddings can be changed in order to perform better on the aforementioned tasks.

Using techniques from graph neural networks, in particular mean and max layer-pooling, I attempt to find a better way to numerically express

Approach

Our baseline model was the default BERT Model, which has baselines of 39% and 78% for pre-training on the SST and CFIMDB dataset and baselines of 51.5% and 96.6% for the respective datasets when fine tuning. Using the layers stored in the BERTLayer method, I averaged the layer representations and also maxed them element-wise in order to see which technique provides the better embedding. I trained and tested them on the SST, STS, and PARA dataset. I utilized the pre-set parameters on the AdamW optimizer. AdamW is an optimizer that attempts to modify Adam's implementation of weight decay by utilizing first and second order momentum to update the weight function. Below is a diagram that illustrates Adam optimizer's training method.



Algorithm 2 Adam with L_2 regularization and Adam with decoupled weight decay (AdamW)

```
1: given $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, \lambda \in \mathbb{R}$
2: initialize time step $t \leftarrow 0$, parameter vector $\boldsymbol{\theta}_{t=0} \in \mathbb{R}^n$, first moment vector $\boldsymbol{m}_{t=0} \leftarrow \boldsymbol{0}$, second moment
   vector $\boldsymbol{v}_{t=0} \leftarrow \boldsymbol{0}$, schedule multiplier $\eta_{t=0} \in \mathbb{R}$
3: repeat
4:     $t \leftarrow t + 1$
5:     $\nabla f_t(\boldsymbol{\theta}_{t-1}) \leftarrow \text{SelectBatch}(\boldsymbol{\theta}_{t-1})$    ▷ select batch and return the corresponding gradient
6:     $\boldsymbol{g}_t \leftarrow \nabla f_t(\boldsymbol{\theta}_{t-1})$ $+\lambda\boldsymbol{\theta}_{t-1}$
7:     $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\boldsymbol{g}_t$    ▷ here and below all operations are element-wise
8:     $\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1 - \beta_2)\boldsymbol{g}_t^2$
9:     $\hat{\boldsymbol{m}}_t \leftarrow \boldsymbol{m}_t/(1 - \beta_1^t)$    ▷ $\beta_1$ is taken to the power of $t$
10:    $\hat{\boldsymbol{v}}_t \leftarrow \boldsymbol{v}_t/(1 - \beta_2^t)$    ▷ $\beta_2$ is taken to the power of $t$
11:    $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$    ▷ can be fixed, decay, or also be used for warm restarts
12:    $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_t \left(\alpha \hat{\boldsymbol{m}}_t/(\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon) +\lambda\boldsymbol{\theta}_{t-1}\right)$
13: until stopping criterion is met
14: return optimized parameters $\boldsymbol{\theta}_t$
```

Experiments:

I used the SST, STS, and PARA datasets in order to train and test my hypothesis. I used a learning rate of 1e-5, beta1 and beta2 of 0.9 and 0.999 respectively and eps of 1e-6. I used two epochs that utilized 8544 sst training examples, 141498 quora training examples, and 6040 training examples. We also use 1101 sst-dev training examples, 20212 quora-dev training examples, and 863 sts-dev training examples. We used cosine similarity for the evaluation metric for paraphrase detection and predict_sentiment.

Results and Analysis

Our data shows that our model seems to work better than the baseline More hyperparameter tuning in the AdamW optimizer, with regards to learning rate, the amount of epochs, and other parameters should be conducted in future experiments.

Paraphrase detection	Sentiment Classification	Semantic Textual Similiarty
37.5%	49.8%	55.6%

Conclusion

We find that given similar training conditions and hyperparameters, mean pooling and max pooling layers from previous encoding inputs ends up giving us more expressive embeddings that train better on semantic textual embeddings, paraphrase detection, and sentiment classification. While more hyperparameter tuning in the AdamW optimizer, and other training hyperparameters should be conducted in future experiments, our study provides promising results and basis for a new type of embedding.

References

R. Ying, J. You, C. Morris, et al. Hierarchical Graph Representation Learning with Differentiable Pooling. arXiv:1806.08804v4 [cs.LG], 20 Feb 2019.

J. Devlin. M. Chang, K. Lee, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805v1 [cs.CL] 11 Oct 2018.