1008 homework 5

Shuang Gao

December 2020

1

(a) BERT stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers.

(b) BooksCorpus (800 million words) and English Wikipedia (2500 million words)

(c) $BERT_{large}$ use a multi-layer bidirectional Transformer encoder, with number of layers equals to 24, number of self-attention heads equals to 16, and hidden size equals to 768.

(d) Position embeddings are vectors used to encode the position of corresponding token in the entire sentence. We add position embedding to the token embedding (and the segment embedding) to inject information about the absolute and relative position of each token, so that the model is assisted to understand the order structure of a sequence.

(e) **MLM** is Masked Language Model. It randomly masks some percentage of the input tokens at random, and then predict those masked tokens. It mainly learns the token-level information.

NSP (Next Sentence Prediction) is a binary classification to tell whether sentence B is the next one to sentence A. It captures the relationship between two sentences. It mainly learns the sentence-level information.

(f) Suppose y_{t_1} is masked by $\hat{y}_{t_1}$, and y_{t_2} is masked by $\hat{y}_{t_2}$, where $\hat{y}_{t_1}$ and $\hat{y}_{t_2}$ could be the mask token [mask], itself, or another random word. Then the sentence is

$$\hat{y} = (y_1, ..., y_{t_1-1}, \hat{y}_{t_1}, y_{t_1+1}, ..., y_{t_2-1}, \hat{y}_{t_2}, y_{t_2+1}, ..., y_T)$$

Then the negative log-likelihood is

$$NLL = -log\, p_\theta(y_{t_1}, y_{t_2}|\hat{y}) = -(log\, p_\theta(y_{t_1}|\hat{y}) + log\, p_\theta(y_{t_2}|\hat{y}))$$

(g) According to the Table 5, if pre-trained on MLM and Left-to-Right respectively without NSP task, the model achieve obviously better performance on MRPC and SQuAD (increasing for about 10%), and slightly better performance on MNLI, QNLI and SST-2, which are sentence-level classification tasks and more-related to NSP.

According to Figure 5, though MLM model converges slower than the LTR model, it outperforms LTR model almost immediately.

(h) It took them 4 days to train $BERT_{large}$ on 16 Cloud TPUs (64 TPU chips total).

(i) BERT paper used the second way to fine-tune on GLUE tasks, training all parameters end-to-end. Compared to the other way, it can achieve better performance, having embedding adjusted to the specific task.

(j) [CLS] is a special symbol added in front of every input example. For classification task, we fed the last hidden vector of the token [CLS] to an output layer to predict the class.

(k) The pre-trained model can be leveraged with an additional RNN layer to fine-tune on MNLI. For each input example, we obtain a sequence of hidden vectors $x_1, x_2, ..., x_T$, where each vector is the hidden vector of corresponding token through the pre-trained model. Then we can use a RNN layer from left to right or a bidirectional-LSTM to extract features from the sequence hidden vectors. As a simplified expression:

$$h_t = \theta_h(W_h x_t + U_h h_{t-1} + b_t)$$

$$y_t = \theta_y(W_y h_t + b_y)$$

And we can take the last output y_T as extracted features, functioning similarly as the hidden vector of [CLS], and feed it into the output layer to do classification.

(l) BERT pretraining used Adam with learning rate of $1e-4$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $L2$ weight decay of 0.01. Learning rate was warmed up over the first 10,000 steps, and followed by linear decay as a learning rate scheduler.

2

We can integrate BERT to the encoder-decoder by using BERT to represent the source English sentences during training on the NMT task. Given an input pair of English sentence and its Burmese translation, first feed the English sentence through BERT and obtain the final hidden vectors as a representation of this sentence. Then we can introduce an additional BERT-encoder attention in each encoder layer and an additional BERT-decoder attention in each decoder layer. For both BERT-encoder and BERT-decoder, we use the representation extracted from BERT to compute query and key, while using the output of last encoder or decoder layer to compute the value. We can run through the additional BERT-encoder/decoder attention parallel with the original attention layers in transformer, and combine their results as the output of each encoder/decoder layer in the transformer.

In this way, we leverage the knowledge from BERT as well as keeping the transformer architecture. The encoder is trained by both self attention and the attention with representation via BERT, and the decoder is trained by both using output of encoder as context and using representation via BERT as context.

3

(a) In catastrophic forgetting, information learned during earlier stages of training is "overwritten". In general, catastrophic training almost removes the generalization advantage obtained from pre-train task, so it may leads to over-fitting when the downstream task is trained on a small dataset. Specifically, when trying to fine-tune on multiple tasks sequentially, if a intermediate task leads to catastrophic forgetting, the model would achieve bad performance on the following tasks due to forgetting about the information from pre-training.

(b) First, multi-task training can leverage data from different tasks, which helps to solve the challenge of data insufficiency.

Second, training on multiple tasks jointly can introduces an inductive bias, which functions as regularization and helps the model to be more generalized.

Third, multi-task training is motivated from the human learning, and a model trained on multiple tasks is able to capture the knowledge in different perspectives.

(c) Multi-task training suffers from catastrophic forgetting and catastrophic interference. If the tasks trained jointly with shared weights are too different, the performance on each of them would be harmed. When introducing a new task in, the performance on a set of original tasks may deteriorates.

4

(a) There are 3 classes in MNLI but only 2 classes in RTE. The training dataset of MNLI is comparatively large, while training data of RTE is much less.

(b) Mean is 0.5283, and standard deviation is 0.0139.

(c) Mean is 0.6582, and standard deviation is 0.0123.

(d) The number of trainable parameters in the model is 110074370.

(e) The limiting factor of training RTE from scratch is insufficiency of training data. It is addressed by fine-tuning on BERT which has been pre-trained on large corpus, learning and storing information in both token-level and sentence level. Therefore, initializing on BERT helps RTE leverage knowledge out of its own training data.

5

(b) I selected MNLI, QNLI and SICK as the 3 other task adapters, because they all tasks categorized as Natural Language Inference, capturing the relations between two sentences, so they are helpful to RTE which is a binary classification on the entailment of two sentences.

As a report of the results, the mean is 0.6715 and the standard deviation is 0.0029.