BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

1 Citation

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

`https://arxiv.org/pdf/1810.04805.pdf`

2 Abstract

Bidirectional Encoder Representations from Transformers (BERT) a pretrained bidirectional transformer language model that takes in a sequence of tokens and outputs an embedding vector. You can stack another layer on top of it and fine-tune it for your task. We set state of the art in 11 natural language processing (NLP) tasks.

3 Introduction

To leverage pretrained language models for NLP, you can use them as features (e.g. ELMo) or fine-tune them (e.g. OpenAI GPT).

The problem with these models is that they are unidirectional or use recurrent neural nets. We fix both issues with bidirectional transformers.

We train on the masked language model (MLM) pretraining task, where we randomly mask a token in the sentence and try to predict it from the other tokens. We also consider a next sentence prediction pretraining task.

4 Related Work

Pretrained word (or paragraph) embeddings are a big part of NLP now. ELMo makes the embedding a function of the sentence that the word is in.

OpenAI GPT provides a pretrained model that you can fine-tune for your task.

In computer vision, we have many models that have been pretrained on the ImageNet dataset.

5 BERT

We basically stack L bi-directional transformers (see the Attention is All You Need paper) on top of each other. Each produces hidden state of dimensionality H and has A self attention heads.

We produce a base model (with roughly same size as OpenAI GPT) and a large model.

An input token is represented as the sum of its token (WordPiece embeddings), segment (this is either the sentence A embedding or sentence B embedding, dependeing on whether we are doing the next sentence prediction task), and positional (learned) embedding. Every segment starts with CLS and segments are separated with SEP. The output of the final transformer is the output embedding vector we use for the token.

To pretrain our model, we use the MLM task or the next sentence prediction task.

For the MLM task, we pick 15% of the tokens to mask. For 80% of these tokens, we replace them with the MASK token. 10% of the time, we replace them with a random word. 10% of the time, we do not change the word at all. Then we train our model to predict the chosen word.

For the next sentence prediction task, we feed in a sentence pair. 50% of the time, the second sentence follows the first and 50% of the time it's just a random sentence from the corpus. The model aims to predict which of the two situations above is at play.

We train on the union of the BooksCorpus and Wikipedia because they have long contiguous segments on the same topic.

We sample sentence pairs (for next sentence prediction) and assign them the sentence A (and sentence B, in the case of next sentence prediction) embeddings. Together the sentences have under 512 tokens. Our batch size is 256 sequences. We train for 1 million steps, or 40 epochs. We use ADAM with learning rate 10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, L2 weight decay 0.01, learning rate warmup for 10 thousand steps with a linear decay thereafter, dropout of 0.1, and the gelu activation function. The objective function is the sum of the MLM likelihood and next sentence likelihood.

How do you use BERT for your task? Well, if you are doing sequence classification, use the final transformer output for the CLS token of your sequence (call this C). Then predict the distribution over the classes as $P = \text{softmax}(CW^T)$ where you fine-tune to learn W and the adjusted BERT parameters. Use the same hyperparameters from training, except use a batch size of 16 or 32, learning rate of 5e-5 or 3e-5 or 2e-5, and 3 to 4 epochs.

The key reason that BERT beats OpenAI GPT is because it is bidirectional (which we can do because we have the MLM and next sentence tasks).

6 Experiments

The General Language Understanding Evaluation (GLUE) benchmark is a bunch of NLP tasks.

We used the fine tuning approach described before. Fine tuning the large model was hard, so we did some random restarts and picked the best model. We set state of the art on all the tasks. The large model beats the base model.

For the question answering task in GLUE, we're given a question (sentence A) and context (sentence B) and we need to output a span in the context that contains the answer. We learn start and end embedding vectors, denoted S and E respectively. Then, given that T_i is final transformer output for token i, we can define the probability that token i is the start of the span as $P_i = \frac{\exp(S \cdot T_i)}{\sum_j \exp(S \cdot T_j)}$. The end of the span is handled similarly. At inference time, we require that end of the span come after the start of the span.

For named entity recognition, we simply take each token representation T_i and pass it through a classification layer to determine whether it is a person, organization, location, or other.

For the SWAG task, we are given a sentence with four possible continuations and we need to pick the right one. We consider each continuation separately, treating the input sentence as sentence A and the continuation as sentence B. This gives us C_1, C_2, C_3, C_4. We learn a vector V and compute the score for continuation C_i as $P_i = \frac{\exp(V \cdot C_i)}{\sum_j \exp(V \cdot C_j)}$.

7 Ablation Studies

If you remove the next sentence prediction task, the model does worse. If you use the usual left-to-right language model task (instead of MLM), the model does worse.

We varied the number of layers, hidden layer size, and number of attention heads. Larger models do better.

Pretraining for 1 million steps helps does a good deal better than training for 500 thousand steps. So, we do indeed need to pretrain for a long time.

What if you want to feed BERT into a complex neural net, rather than just through a matrix multiplication and softmax? We tried this and found that it does give gains.

If you use an ELMo-like approach, where you take a weighted (the weights are learned) sum of all the transformer outputs, rather than just taking the last transformer output, you get some gains.

8 Conclusion

BERT is a bidirectional transformer model trained on the MLM and next sentence prediction task that you can use for various NLP tasks. Just run BERT on your input sentence, take the last transformer output for CLS as the sentence embedding, and just feed it through a matrix multiply and softmax.