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

Sun

April 14, 2020

1 Notes:

1.1 Summary:

1. BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers.
2. By using a masked-language-modeling (MLM) based objective, BERT allows the pretraining of bidirectional transformers without the fallbacks of unidirectionality.
3. The architecture is extremely forgiving in terms of tasks. You don’t need to adhere to a strict format. Doable unsupervised as well.
4. At the time of publishing, it got SOTA on 11 tasks.

1.2 High level:

History: It all started in 1953 when a teaching objective, called the “Cloze procedure”, was introduced as a way to teach kids language. Random words are deleted from a given passage, and replaced by a blank token. The passage is then presented to the students, who would try to replace the words as they read the passage.

As is intuitive, this task requires context from both directions. For example, the sentence, “She tied her _____” read from left-to-right would likely result in “shoes”, but if the right-hand context of “‘_____ into a bun” is used, the obvious answer is “hair”.

Transferring to NLP is a direct switch. Just replace the selected tokens with a special token, `[MASK]`. The models’ task is then to predict the original token from the entire vocabulary of words.

The reason why most training objectives force unidirectionality is the ease of identity mapping. If your task is to predict the next word in a sentence, and you let the model “see” the next part of the sentence, the prediction is trivial (in my experience, less than 200 samples is required to map identity flawlessly).

By masking the token entirely, you allow the model to gather context from both directions without any issues.

Additionally, you train the model to discriminate between pairs of sequential sentences and two sentences that are not sequential. This allows the model to generalize better to downstream tasks.

1.3 Medium level:

BERT is a simple encoder-only transformer, with the most common size being 12 blocks, hidden dimension of 768, and 12 attention heads. This sums to 110 million parameters. One of the major goals of BERT is being a unified architecture, so the input/output representation format is chosen to be extremely open.

To do this, they developed a way to input two sentences at once, or one sentence, without ambiguity. The first token of a given sequence will always be `[CLS]`, standing for 'classification". Then, if your input sequence contains more than one sentence, separate them using the special `[SEP]` token.

One major part of many NLP tasks is embeddings. As was mentioned in the previous presentation, a given embedding of a word will change depending on what position it is via (traditionally) either a sinusoidal embedding of the position, or a learned embedding of the position. This is not unique to positions. We can add a learned embedding that will change depending on whatever we want. In BERT, they use a learned embedding that is added to a given token depending on what sentence it belongs to (before or after `[SEP]`). For a given token, the representation of it is the sum of positional, sentence, and word, embeddings.

Definition 1 *A* ***language model*** *(LM) is a probability distribution over a sequence of tokens. Given a sequence of length m, it assigns a probability $P(w_1, \ldots, w_m)$.*

It is important to note BERT is *NOT* a language model by this definition.

1.3.1 Pretraining BERT:

Task 1: MLM As mentioned previously, one of the major tasks is the prediction of masked words given a sequence. This alone can develop a strong understanding of the language, especially when combined with a powerful model. The downside of this is a large gap between pretraining and fine-tuning, assuming the end task is not a MLM objective (the `[MASK]` token will never appear in the end task). Instead, we randomly select $\sim 15\%$ of tokens to be modified. 80% of them are replaced with `[MASK]`, 10% of the time they are replaced by a random token, and 10% of the time they are untouched.

The objective is to minimize the cross-entropy-loss between the predicted token and the ground truth.

Task 2: Next Sentence Prediction (NSP) Relationship between sentences is, by definition, not explicitly captured by language modeling. Considering many downstream tasks are related to question-answering (QA) and natural language inference (NLI) that has explicit context provided in previous sentences, a lone MLM objective is not ideal.

In order to train a model that understands this sentence relationship, the second training objective is *next sentence prediction.* It is a trivial addition.

Assuming you have a monolingual corpus of paired sequential sentences, $\mathcal{A}$ and $\mathcal{B}$, merely randomly replace $\mathcal{B}$ with a random sentence 50% of the time. Then, train BERT to discriminate between the two possibilities: matching or

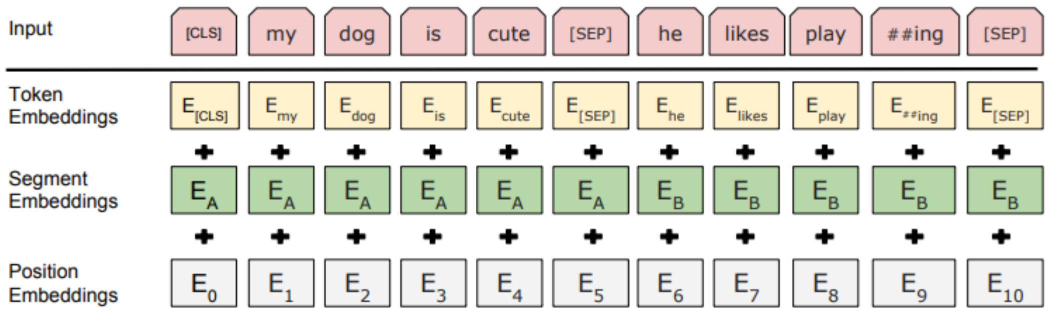
Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8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

Figure 1: Example input into BERT and token representation

non-matching. Despite the simplicity, this improves QA and NLI a significant amount.

1.3.2 Fine-tuning BERT:

Fine-tuning is very straightforward, considering the two minimalistic pretraining tasks already generalize to both the format and content of many downstream tasks. For whatever downstream task, simply plug in the inputs and outputs in the wanted format and fine-tune all of BERTs parameters. For example, sentence $\mathcal{A}$ and $\mathcal{B}$ in pretraining can correspond to: sentence pairs in paraphrasing, hypothesis-premise pairs, question-answer pairs, etc....

Arguably the most important part is the cheapness of fine-tuning. Starting with a pretrained model, one can fine-tune in under 1 hour for most tasks on a TPU, or a few hours on a GPU.

1.4 Empirical results:

Extremely good. SOTA on many tasks, computationally inexpensive to fine-tune, and very very general. The paper runs ablation studies on the model and find that task 1 & 2 are ideal.