BERT: Pre-Training of Transformers for Language Understanding

Understanding Transformer-Based Self-Supervised Architectures

Rohan Jagtap Follow
Jun 28 · 8 min read ★

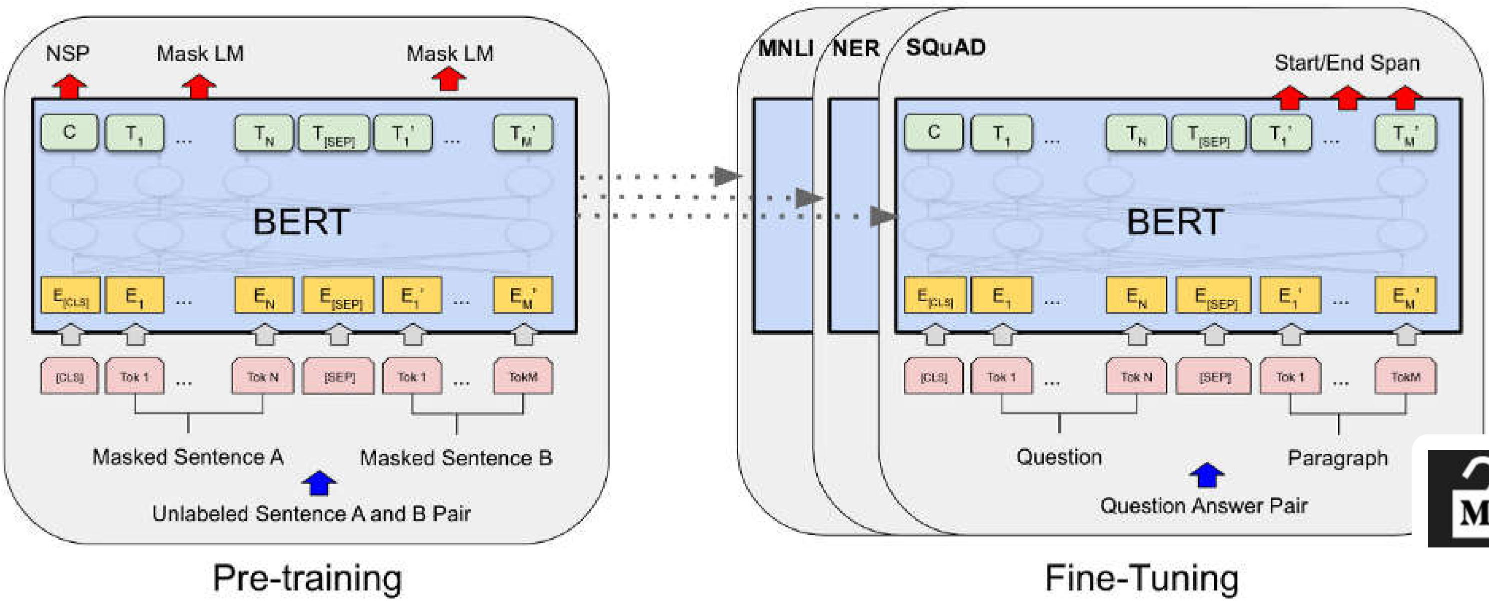
Photo by Brett Jordan on Unsplash

Pre-training Language Models has taken over a majority of tasks in NLP. The 2017 paper, "Attention Is All You Need", which proposed the Transformer architecture, changed the course of NLP. Based on that, several architectures like BERT, Open AI GPT evolved by leveraging self-supervised learning.

In this article, we discuss **BERT** : **B**idirectional **E**ncoder **R**epresentations from **T**ransformers; which was proposed by Google AI in the paper, "BERT: Pre-training of

Deep Bidirectional Transformers for Language Understanding”. This is one of the groundbreaking models that has achieved the state of the art in many downstream tasks and is widely used.

Overview



BERT Pre-training and Fine-Tuning Tasks from the Paper **(We will cover the architecture and specifications in the coming sections. Just observe that the same architecture is transferred for the fine-tuning tasks with minimal changes in the parameters).**

BERT leverages a fine-tuning based approach for applying pre-trained language models; i.e. a common architecture is trained for a relatively generic task, and then, it is fine-tuned on specific downstream tasks that are more or less similar to the pre-training task.

To achieve this, the BERT paper proposes **2 pre-training tasks**:

1. **Masked Language Modeling (MLM)**
2. **Next Sentence Prediction (NSP)**

and **fine-tuning on downstream tasks** such as:

1. **Sequence Classification**
2. **Named Entity Recognition (NER)**
3. **Natural Language Inference (NLI) or Textual Entailment**
4. **Grounded Common Sense Inference**
5. **Question Answering (QnA)**

We will discuss these in-depth in the coming sections of this article.

BERT Architecture

> *BERT's model architecture is a multi-layer bidirectional Transformer encoder based on the original implementation in Vaswani et al.*
>
> *— BERT Paper*

I have already covered the Transformer architecture **in this post**. Consider giving it a read if you are interested in knowing about the Transformer.

To elaborate on the BERT-specific architecture, **we will compare the encoder and the decoder of the Transformer**:

- **The Transformer Encoder** is essentially a Bidirectional Self-Attentive Model, that uses all the tokens in a sequence to attend each token in that sequence

> *i.e. for a given word, the attention is computed using all the words in the sentence and not just the words preceding the given word in one of the left-to-right or right-to-left traversal order.*

Mathematically:

$$\max_{\theta} \quad \log p_{\theta}(\bar{\mathbf{x}} \mid \hat{\mathbf{x}}) \approx \sum_{t=1}^{T} m_t \log p_{\theta}(x_t \mid \hat{\mathbf{x}})$$

For a Sequence of Tokens x_hat, we Maximize the Probability of a Token x_t to Occur at the 't'th Position, Given all the Tokens in that Sequence x_hat. Equation from XLNet Paper.

- While the **Transformer Decoder**, is a Unidirectional Self-Attentive Model, that uses only the tokens preceding a given token in the sequence to attend that token

> *i.e. for a given word, the attention is computed using only the words preceding the given word in that sentence according to the traversal order, left-to-right or right-to left.*

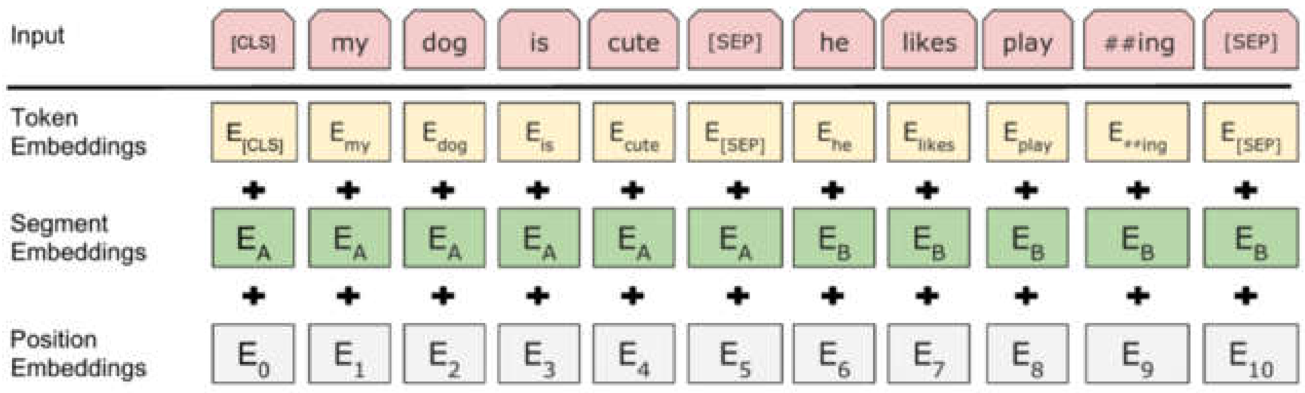
Mathematically,

$$\max_{\theta} \quad \log p_{\theta}(\mathbf{x}) = \sum_{t=1}^{T} \log p_{\theta}(x_t \mid \mathbf{x}_{<t})$$

For a Sequence of Tokens, we Maximize the Probability of a Token x_t to Occur at the 't'th Position, Given all the Tokens Preceding that Token in the Sequence. Equation from XLNet Paper.

Therefore, the **Transformer Encoder gives BERT its Bidirectional Nature**, as it uses tokens from both the directions to attend a given token. We will elaborate on this even further when we'll discuss the **MLM task**.

BERT Representations


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}

BERT Input/Output Representations from the Paper

Before moving on to the pre-training methodology, we'll have a quick glance through the input/output representations used in BERT.

1. **[CLS]:** This token is called as the '*classification*' token. It is used at the beginning of a sequence.
2. **[SEP]:** This token indicates the separation of 2 sequences i.e. it acts as a delimiter.
3. **[MASK]:** Used to indicate masked token in the MLM task.
4. **Segment Embeddings** are used to indicate the sequence to which a token belongs i.e. if there are multiple sequences separated by a [SEP] token in the input, then along with the Positional Embeddings(from Transformers), these are added to the original Word Embeddings for the Model to identify the sequence of the token.
5. The tokens fed to the BERT model are tokenized using **WordPiece embeddings**. The working of this tokenization algorithm is out of scope for this discussion, however, it basically is a tokenization technique between purely character-level encoding and complete word-level encoding to increase the coverage for most of the words in the vocabulary with an appreciable reduction in the vocabulary size (i.e. most of the out of vocabulary or OOV words are covered). Check **this blog** out for WordPiece Embeddings.

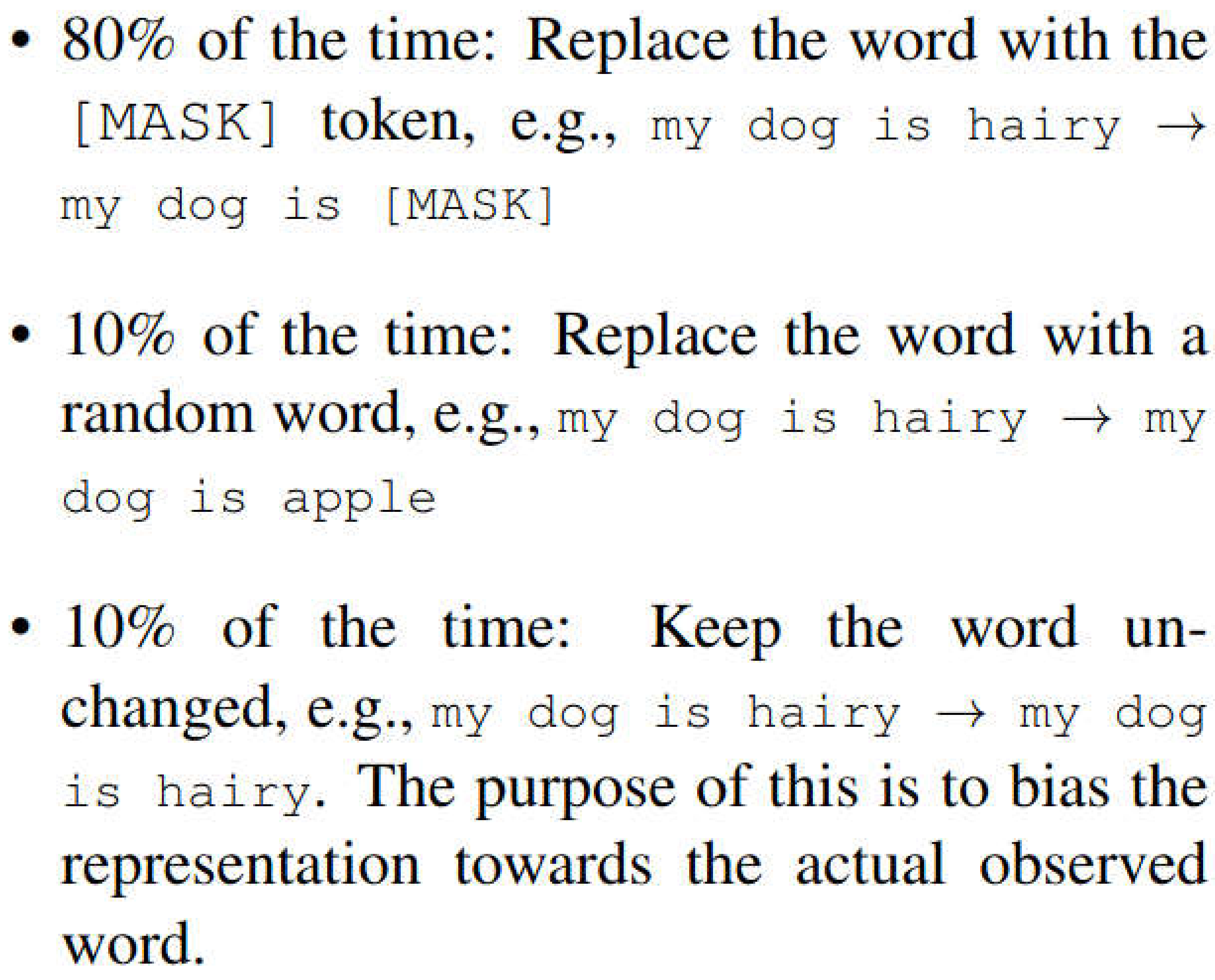
BERT Pre-Training

As mentioned previously, BERT is trained for 2 pre-training tasks:

1. Masked Language Model (MLM)

In this task, 15% of the tokens from each sequence are randomly masked (replaced with the token **[MASK]**). The model is trained to predict these tokens using all the other tokens of the sequence.

However, the fine-tuning task is in no way going to see the [MASK] token in its input. So, for the model to adapt these cases, 80% of the time, the 15% tokens are masked; 10% of the time, 15% tokens are replaced with random tokens; and 10% of the time, they are kept as it is i.e. untouched.



- 80% of the time: Replace the word with the `[MASK]` token, e.g., `my dog is hairy` → `my dog is [MASK]`
- 10% of the time: Replace the word with a random word, e.g., `my dog is hairy` → `my dog is apple`
- 10% of the time: Keep the word unchanged, e.g., `my dog is hairy` → `my dog is hairy`. The purpose of this is to bias the representation towards the actual observed word.

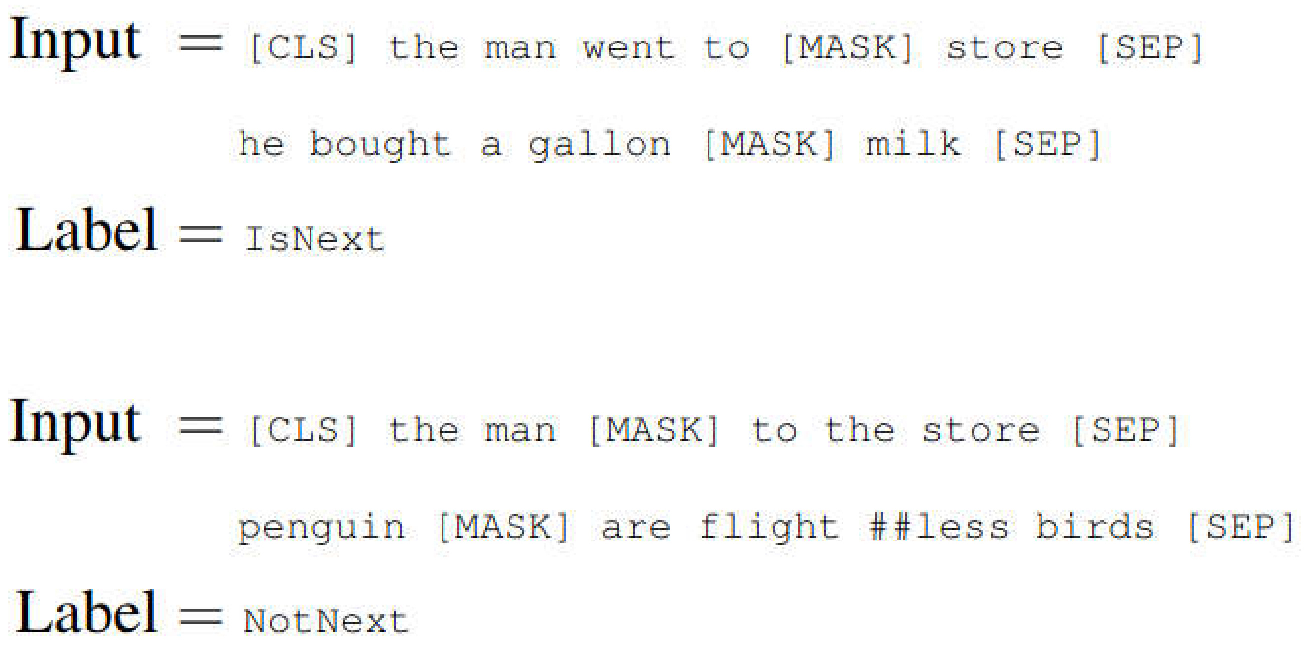
Masked LM Example from the Paper

This is a type of Denoising Autoencoding and is also known as Cloze Task.

2. Next Sentence Prediction (NSP)

This task is somewhat similar to the Textual Entailment task. There are two input sequences (separated using [SEP] token, and **Segment Embeddings** are used). It is a binary classification task involving prediction to tell if the second sentence succeeds the first sentence in the corpus.

For this, 50% of the time, the next sentence is correctly used as the next sentence, and 50% of the time, a random sentence is taken from the corpus for training. This ensures that the model adapts to training on multiple sequences (for tasks like question answering and natural language inference).



Input = [CLS] the man went to [MASK] store [SEP]
he bought a gallon [MASK] milk [SEP]
Label = IsNext

Input = [CLS] the man [MASK] to the store [SEP]
penguin [MASK] are flight ##less birds [SEP]
Label = NotNext

Next Sentence Prediction Example from the Paper

Fine-Tuning BERT

BERT achieved the state of the art on 11 GLUE (General Language Understanding Evaluation) benchmark tasks. We will discuss modeling on a few of them in this section:

1. **Sequence Classification:** The pre-trained model is trained on a supervised dataset to predict the class of a given sequence. Since the **output of the BERT (Transformer Encoder) model is the hidden state for all the tokens in the sequence**, the output needs to be pooled to obtain only one label. The Classification token ([CLS] token) is used here. The output of this token is considered as the classifier pooled output and it is further put into a fully-connected classification layer for obtaining the labeled output.

2. **Named Entity Recognition (NER):** The hidden state outputs are directly put into a classifier layer with the number of tags as the output units for each of the token. Then these logits are used to obtain the predicted class of each token using *argmax*.

3. **Natural Language Inference (NLI) or Textual Entailment:** We train BERT the same as in the **NSP** task, with both the sentences i.e. the **text** and the **hypothesis** separated using [SEP] token, and are identified using the Segment Embeddings. The [CLS] token is used to obtain the classification result as explained in the Sequence Classification part.

4. **Grounded Common Sense Inference:** Given a sentence, the task is to choose the most plausible continuation among four choices. We take 4 input sequences, each containing the original sentence and the corresponding choice concatenated to it. Here too, we use the [CLS] token state and feed it to a fully-connected layer to obtain the scores for each of the choices which are then normalized using a softmax layer.

5. **Question Answering:** A paragraph is used as one sequence and the question is used as another. **There are 2 cases in this task:**

- In the first case, the answer is expected to be found within the paragraph. Here, we intend to find the **Start** and the **End** token of the answer from the paragraph. For this, we take the dot product of **each token *T_i*** and the **start token *S*** to obtain the **probability of the token *i* to be the start of the answer**. Similarly, we obtain the probability of the end token *j*. The score of a candidate span from position *i* to position *j* is defined as ***S.T_i + E.T_j***, and the maximum scoring span where $j \geq i$ is used as a prediction.

- In the second case, **we consider the possibility that there may be no short answer to the question present in the paragraph**, which is a more realistic case. For this, we consider the probability scores for the **start and end of the answer at the [CLS] token itself**. We call this as *s_null*. We compare this *s_null* with the max score obtained at the best candidate span (i.e. the best score for the first case). If this obtained score is greater than *s_null* by a sufficient threshold **τ**, then we use the candidate best score as the answer. The threshold **τ** can be tuned to maximize the dev set F1-score.

Conclusion

We took a deep dive into the BERT's architecture, pretraining specification and fine-tuning specification.

The main takeaway from this post is that BERT is one of the most fundamental Transformer-based language models, and a similar pattern is followed for improvement of this architecture and tasks in the future models that have surpassed the state of the art.

The official BERT code is open source and can be found here.

HuggingFace Transformers is a repository that implements Transformer-based architectures and provides an API for these as well as the pre-trained weights to use.

I am planning to cover the major Transformer-based Models that have achieved the state of the art in the coming posts. BERT being the most fundamental of these had to be the first. Stay Tuned:)

References

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

We introduce a new language representation model called BERT, which stands for Bidirectional Encoder Representations...

arxiv.org

XLNet: Generalized Autoregressive Pretraining for Language Understanding

With the capability of modeling bidirectional contexts, denoising autoencoding based pretraining like BERT achieves...

arxiv.org

Attention Is All You Need

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks in an...

arxiv.org

Transformers Explained

An exhaustive explanation of Google's Transformer model; from theory to implementation

towardsdatascience.com