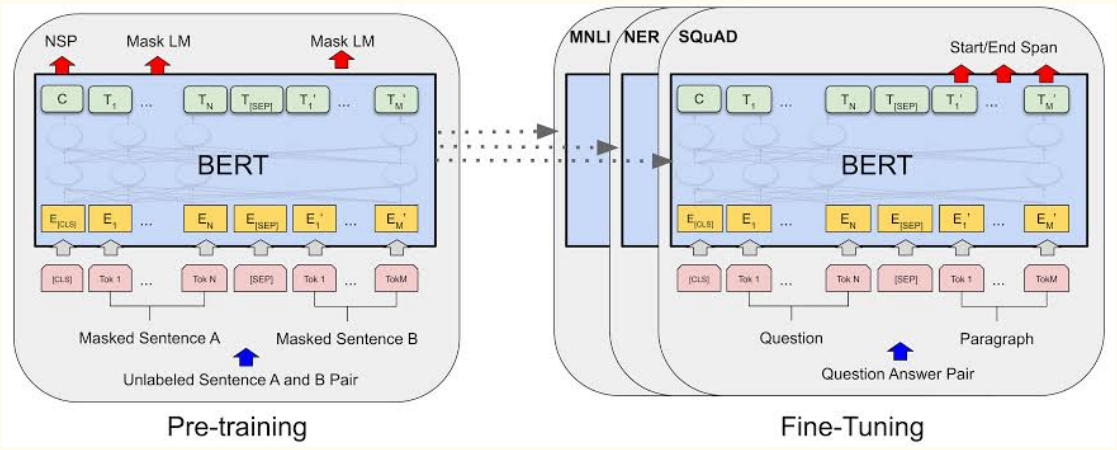
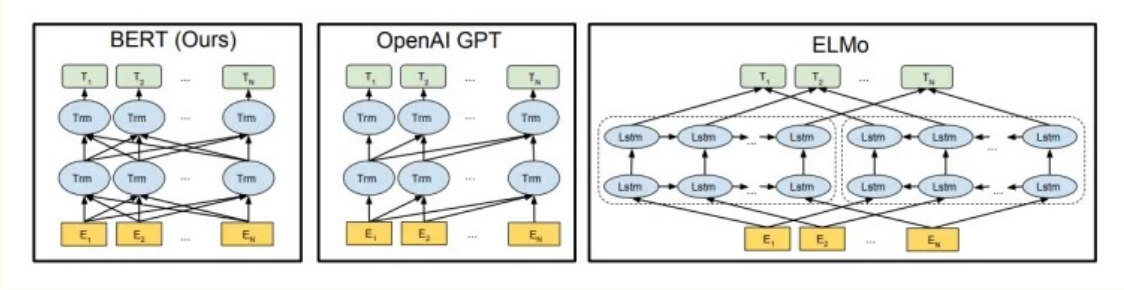
BERT:





→ BERT uses Encoder part of the Transformer

→ key part ⇒ Bidirectional training of Transformer

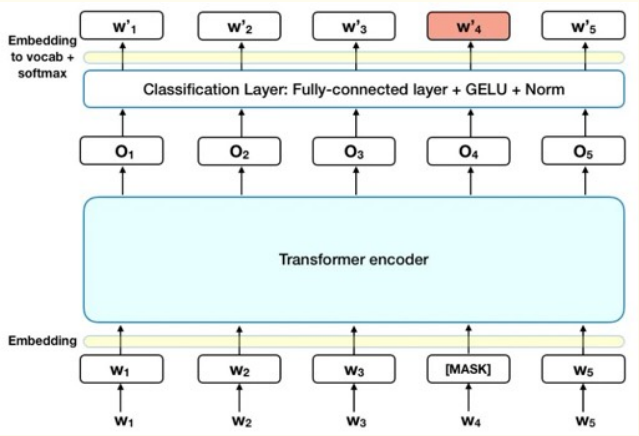
↳ get deeper understanding of words inside a sentence.

→ encoder reads entire sequence at once

↳ Bidirectional

→ vanilla BERT → MLM
→ NSP

MLM:

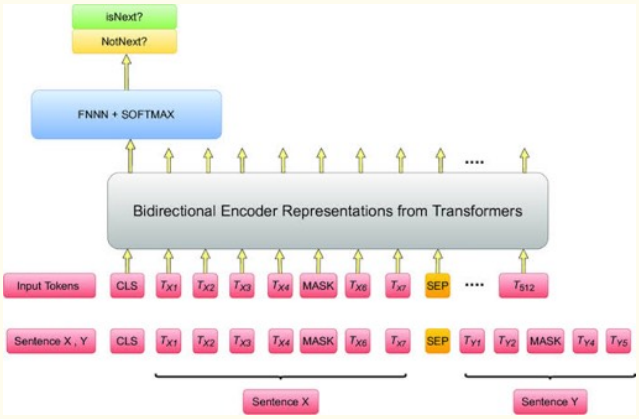


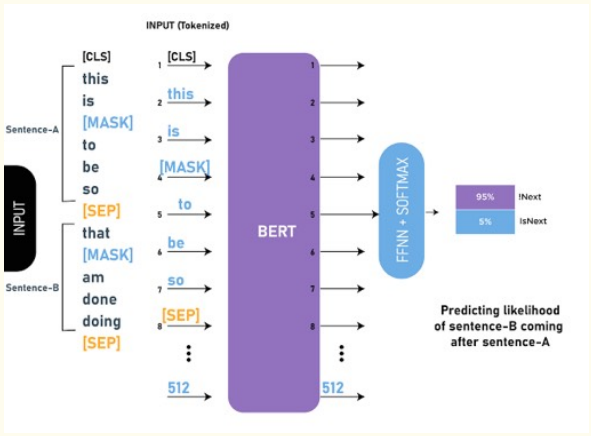
→ 15% of words sequence are replaced with [MASK] token

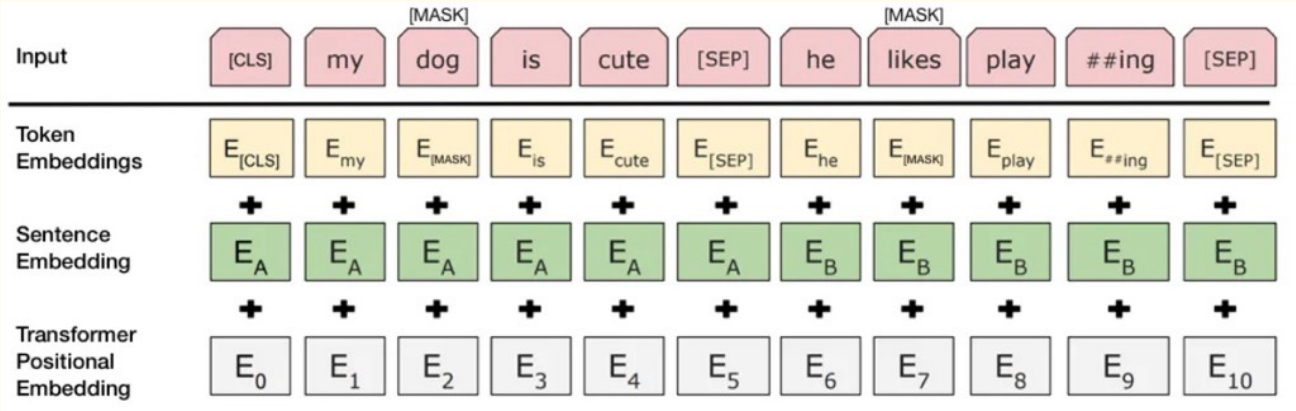
Training the language model in BERT is done by predicting 15% of the tokens in the input, that were randomly picked. These tokens are pre-processed as follows — 80% are replaced with a "[MASK]" token, 10% with a random word, and 10% use the original word. The intuition that led the authors to pick this approach is as follows (Thanks to Jacob Devlin from Google for the insight):

The BERT loss function takes into consideration only the prediction of the masked values and ignores the prediction of the non-masked words. As a consequence, the model converges slower than directional models, a characteristic which is offset by its increased context awareness (see Takeaways #3).

NSP:







Application:

How to use BERT (Fine-tuning)

Using BERT for a specific task is relatively straightforward:

BERT can be used for a wide variety of language tasks, while only adding a small layer to the core model:

1. Classification tasks such as sentiment analysis are done similarly to Next Sentence classification, by adding a classification layer on top of the Transformer output for the [CLS] token.
2. In Question Answering tasks (e.g. SQuAD v1.1), the software receives a question regarding a text sequence and is required to mark the answer in the sequence. Using BERT, a Q&A model can be trained by learning two extra vectors that mark the beginning and the end of the answer.
3. In Named Entity Recognition (NER), the software receives a text sequence and is required to mark the various types of entities (Person, Organization, Date, etc) that appear in the text. Using BERT, a NER model can be trained by feeding the output vector of each token into a classification layer that predicts the NER label.

Architecture

There are four types of pre-trained versions of BERT depending on the scale of the model architecture:

`BERT-Base`: 12-layer, 768-hidden-nodes, 12-attention-heads, 110M parameters
`BERT-Large`: 24-layer, 1024-hidden-nodes, 16-attention-heads, 340M parameters

Fun fact: BERT-Base was trained on 4 cloud TPUs for 4 days and BERT-Large was trained on 16 TPUs for 4 days!

For details on the hyperparameter and more on the architecture and results breakdown, I recommend you to go through the original paper.