Mihir Raul (D013)

Bidirectional Encoder Representations from Transformers (BERT)

What's BERT?

BERT, which stands for Bidirectional Encoder Representations from Transformers, is based on Transformers, a deep learning model in which every output element is connected to every input element, and the weightings between them are dynamically calculated based upon their connection.

Historically, language models could only read text input sequentially -- either left-to-right or right-to-left -- but couldn't do both at the same time. BERT is different because it is designed to read in both directions at once. This capability, enabled by the introduction of Transformers, is known as bidirectionality.

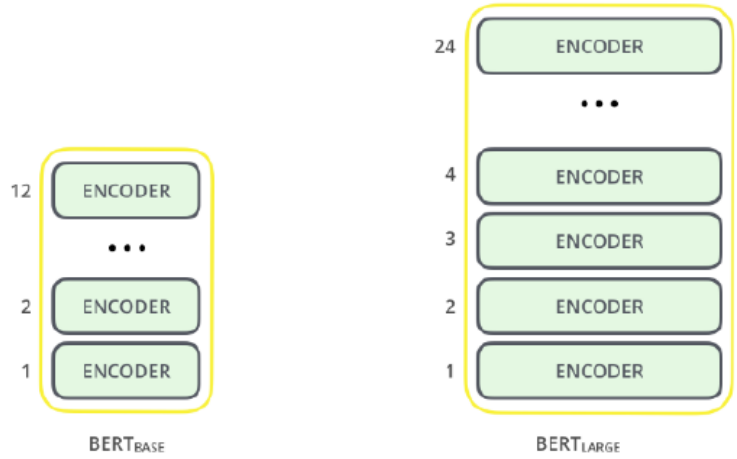
Using this bidirectional capability, BERT is pre-trained on two different, but related, NLP tasks: Masked Language Modeling and Next Sentence Prediction.

BERT's Architecture

There are two variants of BERT:

- *BERT Base:* 12 layers (transformer blocks), 12 attention heads, and 110 million parameters

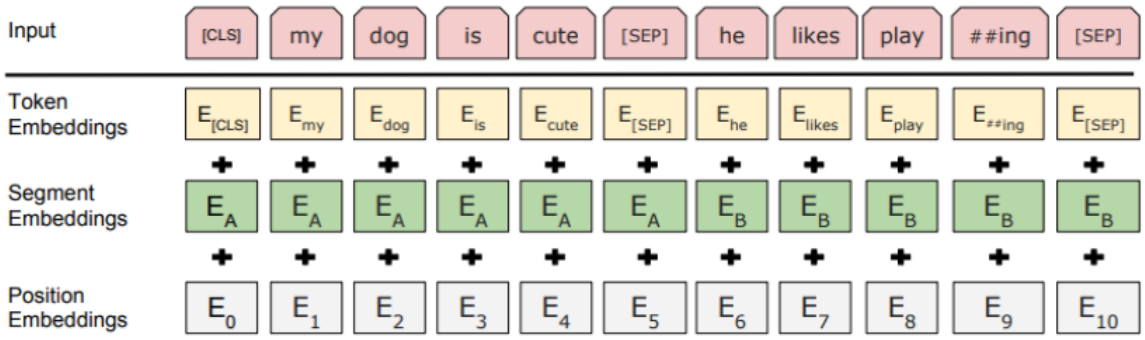
- *BERT Large:* 24 layers (transformer blocks), 16 attention heads and, 340 million parameters



Text Pre-Processing

BERT has a specific set of rules to represent the input text for the model.
Every input embedding is a combination of 3 embeddings:

- ***Position Embeddings*:** BERT learns and uses positional embeddings to express the position of words in a sentence. These are added to overcome the limitation of Transformer which, unlike an RNN, is not able to capture "sequence" or "order" information
- ***Segment Embeddings*:** BERT can also take sentence pairs as inputs for tasks (Question-Answering). That's why it learns a unique embedding for the first and the second sentences to help the model distinguish between them. In the above example, all the tokens marked as EA belong to sentence A (and similarly for EB)
- ***Token Embeddings*:** These are the embeddings learned for the specific token from the WordPiece token vocabulary For a given token, its input representation is constructed by summing the corresponding token, segment, and position embeddings. Such a comprehensive embedding scheme contains a lot of useful information for the model.



Pre-Training Task:

BERT is pre-trained on two NLP tasks:

- Masked Language Modeling
- Next Sentence Prediction

Code

```
import tensorflow as tf
import tensorflow_hub as hub
!pip install tensorflow-text
import tensorflow_text as text
```

 # Download the BERT preprocessor and encoder for generating the model.

```
bert_preprocess =
hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3")

bert_encoder =
hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-
768_A-12/4")
```

 # Bert layers

```
text_input = tf.keras.layers.Input(shape=(), dtype=tf.string, name='text')

preprocessed_text = bert_preprocess(text_input)

outputs = bert_encoder(preprocessed_text)
```

 # Neural network layers (binary text classification)

```
l = tf.keras.layers.Dropout(0.1, name="dropout")(outputs['pooled_output'])
```

```
l = tf.keras.layers.Dense(1, activation='sigmoid', name="output")(l)

# Use inputs and outputs to construct a final model

model = tf.keras.Model(inputs=[text_input], outputs = [l])

# Compile and fit model

model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])


model.fit(X, y, epochs=epochs)
```