BERT

BERT - Bidirectional Encoder Representations from Transformers

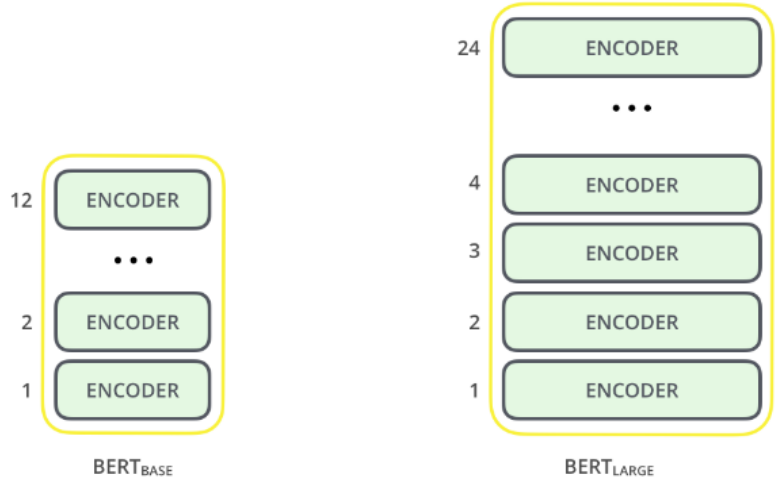
- Transformer architecture
- pre-trained on a large corpus of unlabelled text including the entire Wikipedia (2,500 million words) and Book Corpus (800 million words)
- ‘deeply bidirectional’ model

  Bidirectional means that BERT learns information from both the left and the right side of a token’s context during the training phase

➢ BERT’s Architecture

Two variants available:

- 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



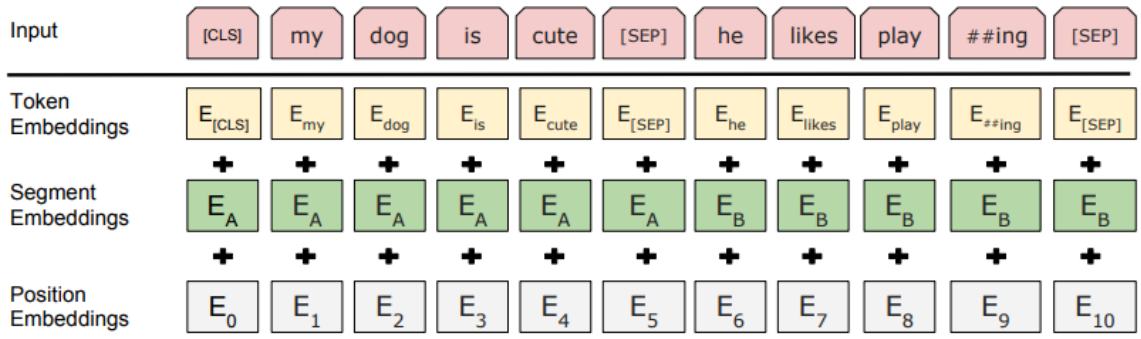
➢ Text Preprocessing

BERT has a specific set of rules to represent the input text for the model.

Every input embedding is a combination of 3 embeddings:

1. 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
2. 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)
3. 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 Tasks

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)
```