NLP: Natural Language Processing

BERT: Bidirectional Encoder Representations from Transformers

GTTS: Google Text to Speech

MLMs: Masked Language Models

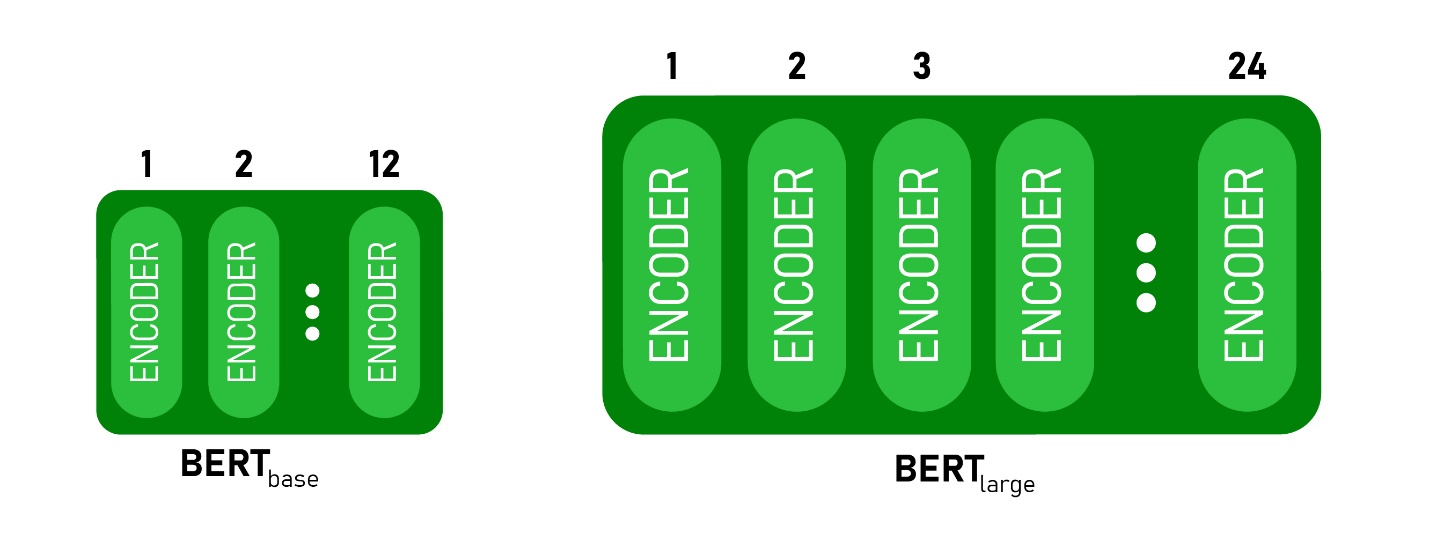
BOW: Bag of word

TF-IDF=term frequency — inverse document frequency

FC= fully connected

Model 1 : Text Classification using Bert

Bert: is a Natural Language Processing Model proposed by researchers at Google Research in 2018. When it was proposed it achieve state-of-the-art accuracy on many NLP and NLU tasks

Bert Base vs Bert Large 

And we used the base version that contain 12 encoders to make out classification.



So, the components of this model are   
**1. Tokenizer**

Tokenization is breaking the raw text into small chunks. Tokenization breaks the raw text into words, sentences called tokens. These tokens help in understanding the context or developing the model for the NLP. The tokenization helps in interpreting the meaning of the text by analyzing the sequence of the words.

For example, the text “It is raining” can be tokenized into ‘It’, ‘is’, ‘raining’

There are different methods and libraries available to perform tokenization. NLTK, Gensim, Keras are some of the libraries that can be used to accomplish the task.

Tokenization can be done to either separate words or sentences. If the text is split into words using some separation technique it is called word tokenization and same separation done for sentences is called sentence tokenization.

Stop words are those words in the text which does not add any meaning to the sentence and their removal will not affect the processing of text for the defined purpose. They are removed from the vocabulary to reduce noise and to reduce the dimension of the feature set.

One of the biggest challenges in the tokenization is the getting the boundary of the words. In English the boundary of the word is usually defined by a space and punctuation marks define the boundary of the sentences, but it is not same in all the languages. In languages such as Chinese, Korean, Japanese symbols represent the words and it is difficult to get the boundary of the words.

Even in English there are lot of symbols used as £, $, € followed by numerical to represent money and there are lot of scientific symbols such as µ, α etc. which create challenges in tokenization.

There are also lot of short forms used in English such as I’m (I am), didn’t (did not) etc. which needs to resolved or else these cause a lot of problems in next steps of NLP.

**2. encoding**

1. Machine doesn’t understand characters, words or sentences.

2. Machines can only process numbers.

3. Text data must be encoded as numbers for input or output for any machine.

So, we cannot pass raw text into machines as input until and unless we convert them into numbers, hence we need to perform text encoding.

Text encoding is a process to convert meaningful text into number / vector representation so as to preserve the context and relationship between words and sentences, such that a machine can understand the pattern associated in any text and can make out the context of sentences.

There are a lot of methods to convert Text into numerical vectors, they are:

- Index-Based Encoding

- Bag of Words (BOW)

- TF-IDF Encoding

- Word2Vector Encoding

- BERT Encoding

**3.SequenceClassifier (Bert base)**

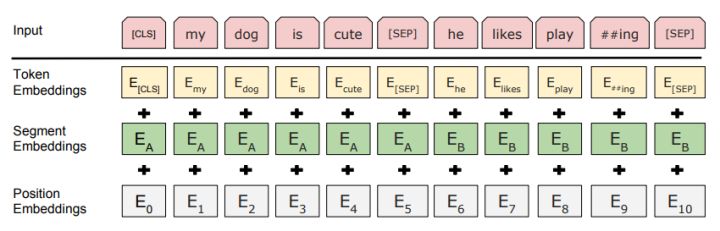
1.architecture: BERT Base: 12 layers (transformer blocks), 12 attention heads, and 110 million parameters

2. text processing

The developers behind BERT have added a specific set of rules to represent the input text for the model. Many of these are creative design choices that make the model even better.

For starters, 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



3.pre-training tasks:

 Masked Language Modeling

Need for Bi-directionality

BERT is designed as a *deeply bidirectional*model. The network effectively captures information from both the right and left context of a token from the first layer itself and all the way through to the last layer.

Traditionally, we had language models either trained to predict the next word in a sentence (right-to-left context used in GPT) or language models that were trained on a left-to-right context. This made our models susceptible to errors due to loss in information.

 Next Sentence Prediction

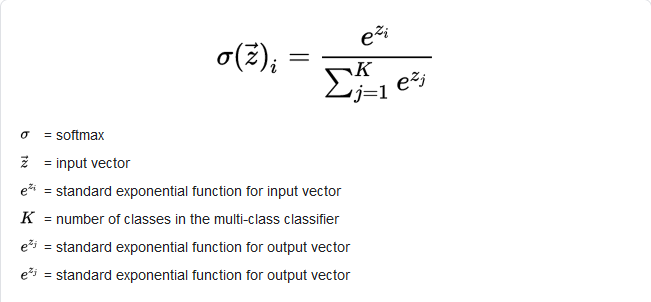
**Masked Language Models (MLMs) learn to understand the relationship between words. Additionally, BERT is also trained on the task of Next Sentence Prediction for tasks that require an understanding of the relationship between sentences.**

The task is simple. Given two sentences – A and B, is B the actual next sentence that comes after A in the corpus, or just a random sentence?

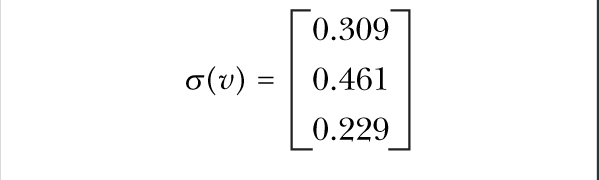
Since it is a binary classification task, the data can be easily generated from any corpus by splitting it into sentence pairs

**4.classification layer (SoftMax)**

The softmax function is often used as the activation function in the output layer of neural networks when the network’s goal is to learn classification problems. The softmax will squash the output results between 0 and 1, and the sum of all outputs will always add up to 1. That way, the results of an output layer with a softmax function can be considered as probabilities.



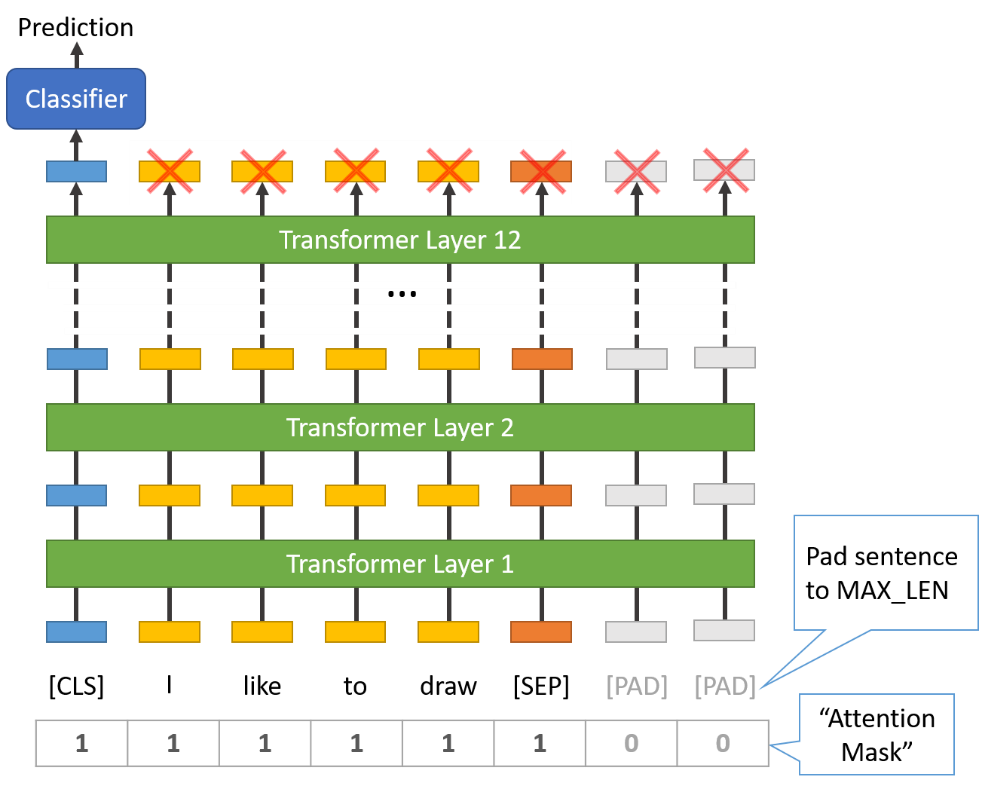
The output will be like that

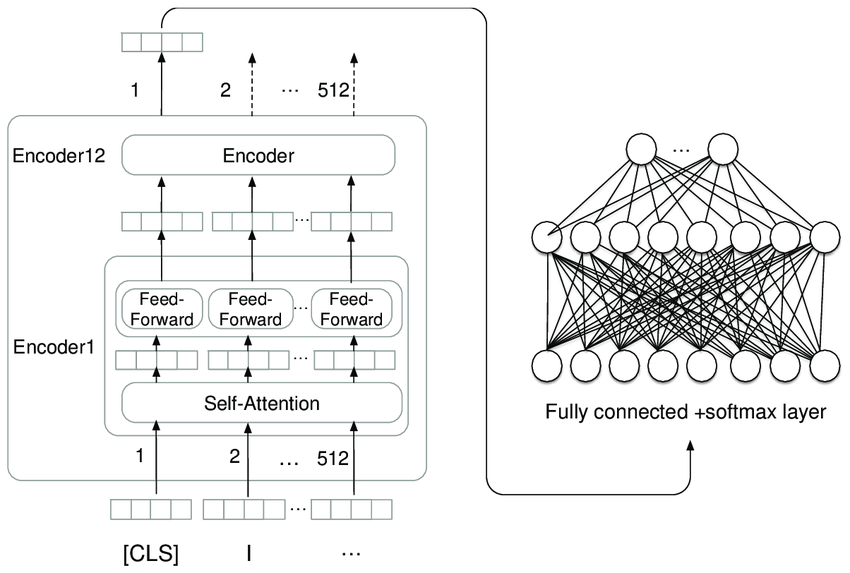


5.assign the label with higher probability to the sentence

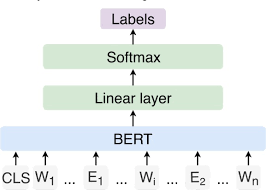
Here we specify max length for words then we pad the rest if lower than max\_len with zeros

And truncate if more then max\_len



This the architecture of bert base when we use it as sequence classifier (bert+FC+SoftMax)

High level architecture for the text classification model



We use Adam as our optimizer

**Adam** is a replacement **optimization** algorithm for stochastic gradient descent for training deep learning models. **Adam** combines the best properties of the AdaGrad and RMSProp algorithms to provide an **optimization** algorithm that can handle sparse gradients on noisy problems

The parameters for adam optimizer are

* β1 — This is used for decaying the running average of the gradient
* β2 — This is used for decaying the running average of the square of gradient
* α — Step size parameter
* ε- It is to prevent Division from zero error.

Our values for adam hyperparameters

* weight\_decay(β1)=0.02
* learning\_rate (α)= 25e-6
* adam\_epsilon (ɛ)= 3e-8

batch size = 25

The algorithm for adam optimizer

