BERT MODEL

BERT- Bi-Directional Encoder Representations from Transformers

In this Article I am going through Overview of BERT Model Explanation and Sample Code snippets.

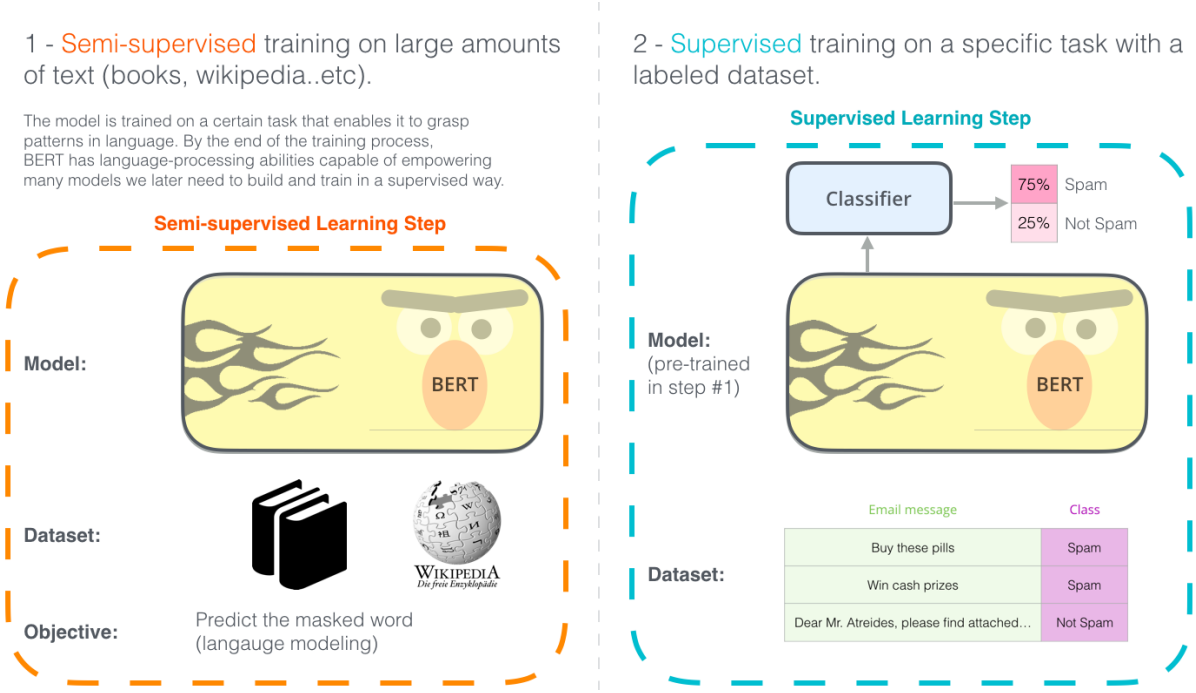
What is BERT?

From internet there multiple of Answers for this Questions but I understand in easy that,

BERT stands for Bidirectional Encoder Representations from Transformers. It is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks.

This is the greatest approach from NLP Tasks became a 2-Step process:

1. Train a language model on a large un labelled text corpus (unsupervised or semi-supervised)
2. Fine-tune this large model to specific NLP tasks to utilize the large repository of knowledge this model has gained (supervised).



Source

I going to explain these below concepts.

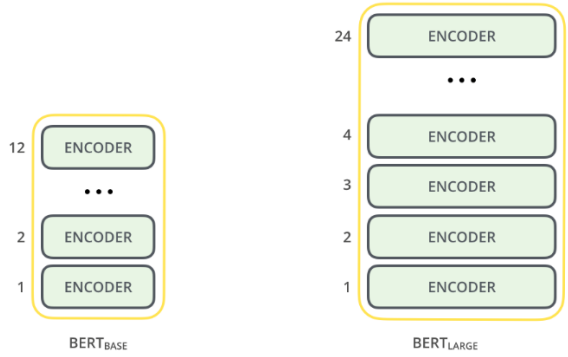
1. BERT Architecture.
2. Introduction on Transformers.
3. BERT-Text Processing.
4. Pre-Trained Tasks.

Architecture of BERT:

Bert Architecture Builds on top of Transformer Architecture.

We have two kind of Architectures

1. BERT - Base:
    a. 12-Layers (Transformers Blocks), 12 Attention heads.
2. BERT – Large
    a. 24-Layers (Transformers Blocks), 16-Attention heads.



As We discussed BERT Model builds based on Transformers right, Okie now lets Illustrates what *TRANSFORMERS* first

There are different models which are running based on *Transformers* only on that some are ULMFiT, Elmo, OpenAI GPT, and BERT etc.,

Why Transformers are Very Powerful?

If we have one paragraph of data, from the data capturing relationships and sequence of words in sentences is vital for machine to Understand a natural language.

"Here the Transformer concept player Major Roles"

There are (Seq 2 Seq) Sequence-TO-Sequence models, RNN (Seq2Seq) in NLP are used to convert Seq of Type A to Seq Type B.

These SeqtoSeq kind of Models are used in

1. Machine Translations
2. Text Summarization
3. Speech Reorganization.

4. Q and A Models

There is further enhancement happened to improve the performance of Seq2Seq models by adding "Attention Mechanism"

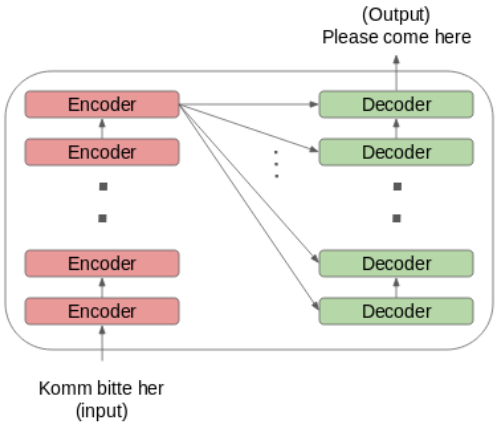
TRANSFORMER:

The Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence aligned RNNs or convolution.

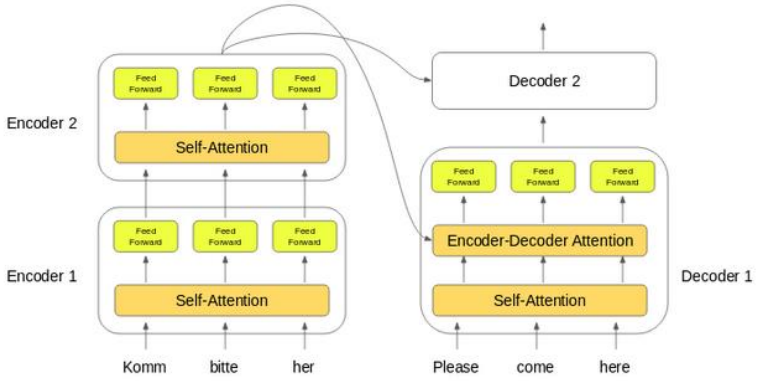
Transduction means Conversion of Input seq into Output Seq

The Idea behind the transformers is to handle the dependencies b/w input and output with "Attention" and Recurrence Completely.

The encoder and decoder blocks are multiple identical encoders and decoders stacked on top of each other. Both the encoder stack and the decoder stack have the same number of units.



- The word embeddings of the input sequence are passed to the first encoder
- These are then transformed and propagated to the next encoder
- The output from the last encoder in the encoder-stack is passed to all the decoders in the decoder-stack



In addition to the self-attention and feed-forward layers, the decoders also have one more layer of Encoder-Decoder Attention layer.

As per Research papers:

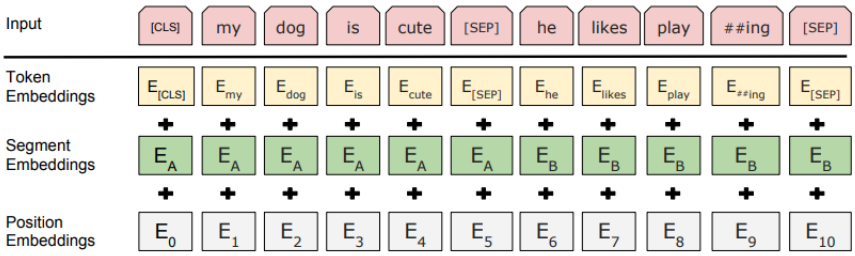
"Self-attention, sometimes called intra-attention, is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence."

Text Processing:

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



BERT is pre-trained on two NLP tasks:

1. Masked Language Modeling.
2. Next Sentence Prediction.

Masked Language Modeling.

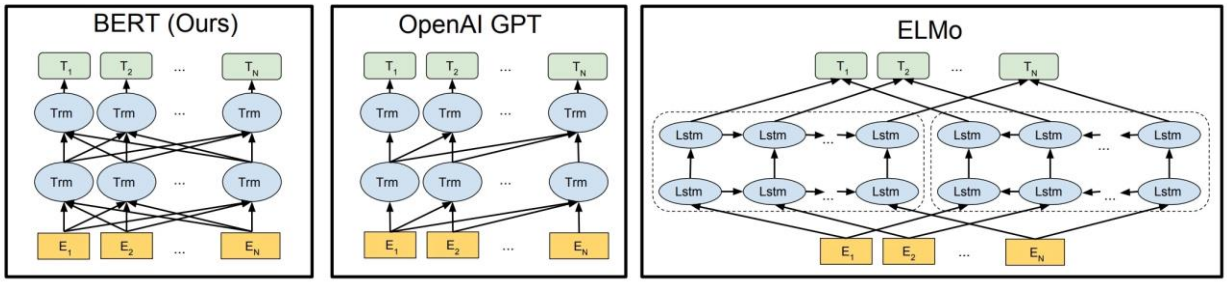
BERT is designed as a deeply bidirectional model. The network effectively captures information from both the right and left context of a token from the first layer itself and all the way through to the last layer

BERT vs ELMO vs OpenAI GPT

ELMo tried to deal with this problem by training two LSTM language models on left-to-right and right-to-left contexts and shallowly concatenating them. Even though it greatly improved upon existing techniques, it wasn't enough. **--ELMo**

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. **– Open AI GPT**

It is reasonable to believe that a deep bidirectional model is strictly more powerful than either a left-to-right model or the shallow concatenation of a left-to-right and a right-to-left model. **– BERT**



BERT is bi-directional, GPT is unidirectional (information flows only from left-to-right), and ELMO is shallowly bidirectional.

"I love to read data science blogs on Analytics Vidhya". We want to train a bi-directional language model. Instead of trying to predict the next word in the sequence, we can build a model to predict a missing word from within the sequence itself.

Let's replace "Analytics" with "[MASK]". This is a token to denote that the token is missing. We'll then train the model in such a way that it should be able to predict "Analytics" as the missing token: "I love to read data science blogs on [MASK] Vidhya."

This is the crux of a Masked Language Model. The authors of BERT also include some caveats to further improve this technique:

To prevent the model from focusing too much on a particular position or tokens that are masked, the researchers randomly masked 15% of the words

The masked words were not always replaced by the masked tokens [MASK] because the [MASK] token would never appear during fine-tuning

So, the researchers used the below technique:

80% of the time the words were replaced with the masked token [MASK]

10% of the time the words were replaced with random words

10% of the time the words were left unchanged

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.

Since it is a binary classification task, the data can be easily generated from any corpus by splitting it into sentence pairs. Just like MLMs, the authors have added some caveats here too. Let's take this with an example:

Consider that we have a text dataset of 100,000 sentences. So, there will be 50,000 training examples or pairs of sentences as the training data.

- ✓ For 50% of the pairs, the second sentence would actually be the next sentence to the first sentence
- ✓ For the remaining 50% of the pairs, the second sentence would be a random sentence from the corpus
- ✓ The labels for the first case would be 'IsNext' and 'NotNext' for the second case.

And this is how BERT is able to become a true task-agnostic model. It combines both the Masked Language Model (MLM) and the Next Sentence Prediction (NSP) pre-training tasks.

I have gone through lot articles and which illustrates on BERT from those I love to read from Analytics Vidya, and Jay Alammar Blogs.

Reference Articles:

http://jalammar.github.io/illustrated-bert/

https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/