

Content

- **Introduction to Business Case: Data Scientist at Medical Research Publication company**
- **Introduction to BERT: Why is it SOTA?**
 - Differences between RNN and BERT
 - BERT: Embedding vs Pretrained model
- **How Bert is Pretrained**
 - Masked Language Modelling
 - Next Sentence Prediction
 - MLM vs NSP
- **Bidirectionality of BERT**
- **Short recap of Transformers**
 - Attention mechanisms
 - Positional encoding
- **Tokenization in BERT**
 - Word Piece Tokenizer
- **Types and Applications of BERT**
 - Types of BERT
 - Application Classes in huggingface for BERT

✓ Business Case

- You are a Data scientist working in a Medical Research publication company.
- As 100s of articles are published every single day, the organizers wants to identify which diseases are being spoke in the article.
- Manually reading the articles and identifying the disease is a laborious process.
- As a data scientist you are asked, if you can train a machine Learning model to automatically identify the diseases mentioned in the corpus with the limited training data available

To solve this problem we will be training a NER model using BERT.

- We can use NER models available in Spacy or train RNN models, etc

- Since BERT has been the State Of The Art in the almost all NLP task in the recent years, we will use it to train with our corpus

✓ But why is BERT the SOTA and how was it able to achieve?

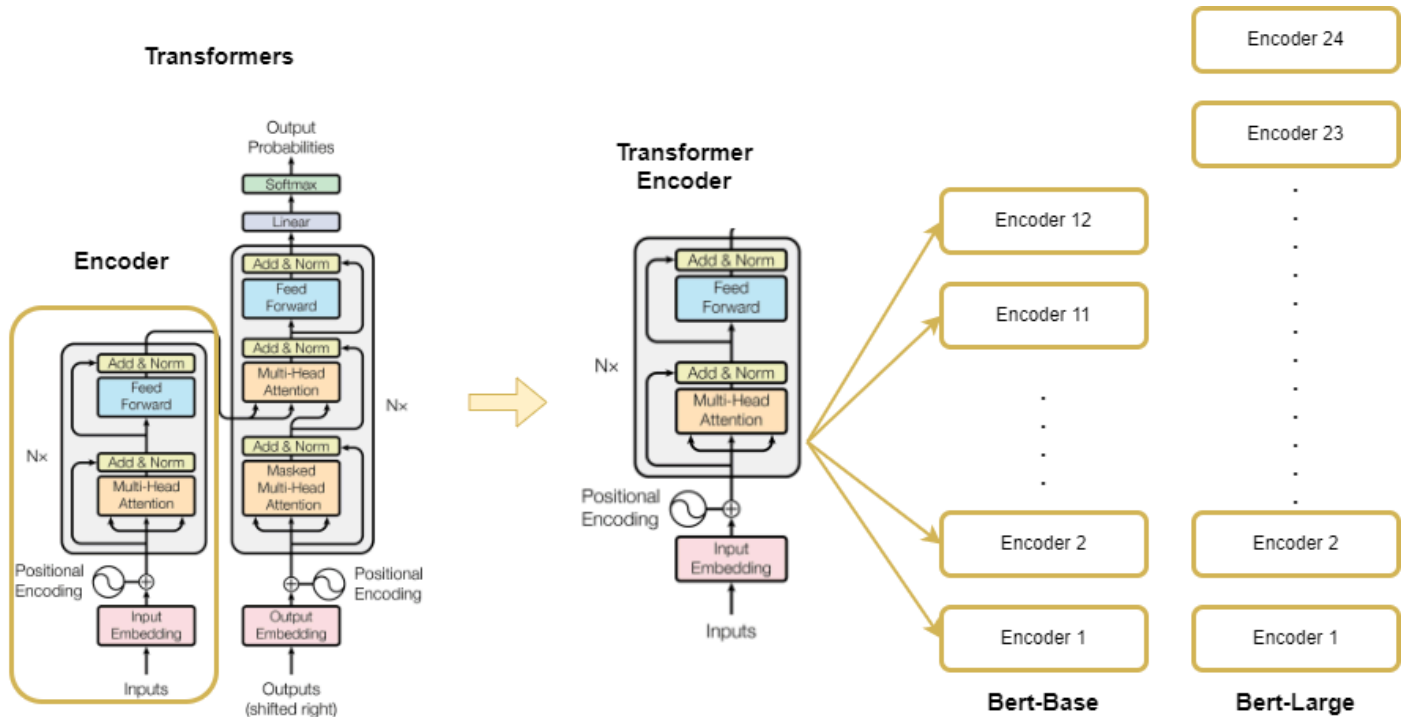
- BERT has been outperforming in almost all benchmarking NLP task like Question Answering, Next Sentence Prediction tasks etc
- Because of the way it was pretrained with rich corpus of data it was able to achieve SOTA results.
- We will see the details in the subsequent sections

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT _{BASE}	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	92.7	94.9	60.5	86.5	89.3	70.1	82.1

Table 1: GLUE Test results, scored by the evaluation server (<https://gluebenchmark.com/leaderboard>). The number below each task denotes the number of training examples. The “Average” column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.⁸ BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

✓ Let us begin with what is BERT?

- BERT stands for BiDirectional Encoder Representation from Transformers released by Google Research team in 2018
- BERT uses only the Encoder portion of the transformer's architecture which is the reason it's called **Encoder Representation from transformers**.



How is BERT different from other architectures like RNN(LSTM/GRU)?

RNN	BERT
It is built with recurrent blocks	It is built with transformer blocks
Each word in a sequence are passed one after the other	The entire sentence is passed at once
If N is the no.of words in a sequence, RNNs has to compute N sequential steps	It performs only one step for the entire sequence of length N
Similarly for the gradients to propagate from last word to the first word it takes N steps	In Bert, it is just one step process
Because of sequential processing there is no parallelization	It can be parallelization since it is not passed sequentially
They suffer from Vanishing gradients though LSTM/GRU can handle to an extent	Bert does not suffer from Vanishing gradient problem and can learn long term dependencies very well
No self attention	Has self attention

Is BERT a word embedding model or a pretrained model?

Though we might have learnt it before. Let's understand the difference:

Embedding model:

- In Embedding models you only get the embeddings for each word or sentence of specified dimensions
- With the embeddings, you will have to fit a ML model like Logistic Regression, Random Forest, XGboost etc or even NN using the embeddings as features.
- Example: Word2Vec, Glove, ELmo etc

Pretrained model:

- If you are familiar with transfer learning/pretrained models in computer vision like AlexNet, VGGNet, the same concept is used here where the architecture of the pre-trained models with weights are fixed.
- You can just add the output layer with different number of neurons based on your task.

So now is BERT a word embedding model or a pretrained model?

- Actually it is both.
 - * Just like other word embedding models like Word2Vec, ELmo, Glove etc you can extract the word embedding for each word/sentences.
 - * This can be achieved by accessing the embedding layer before it is passed to the Encoder portion of the architecture.
 - * You can also use the BERT architecture with random weights or pretrained weights to train a model just like pretrained models in computer vision by transfer learning.

Can we use Random weights instead of Pretrained weights?

- In >90% of the cases we use the pretrained weights since these weights are trained weights are trained on general purpose corpus, it's easy to optimize.
- If we start with random weights, It can even take days/weeks to get good results compared to starting from pre-trained weights

But is there any usecase we can start with Random weights?

- Yes it can be useful in some cases where you are trying to pre-train a BERT model from scratch specific to a domain. Example:

BIO-BERT: It is pretrained on the corpus specific to Medical domain.

LawBERT: It is pretrained on the corpus specific to Legal domain.

Once pretrained these weights can be used for task specific to that domain instead of starting from the native BERT embeddings.

✓ How was BERT pre-trained?

- Bert was trained on two tasks
 1. Predicting the masked words
 2. Next Sentence Prediction

What was the data that it was pretrained on?

- It was Pretrained on Wikipedia(2500 million words) and Book Corpus(800 million words)

1. Predicting the masked words

- Masking 15% of the words and predict them
 - * By Randomly masking 15% of the words in the corpus the, model was trained to predict the masked words.

Example:

Actual sentence	Masked train sentence	Label
Today morning, I went for a tooth removal to my dentist	Today morning, I went for a < MASK > removal to my dentist	tooth
Did you go to school today	Did you < MASK > to school today	go

✓ 2. Next Sentence Prediction

- Predicting if the next sentence is continuation of the previous sentence or not

Example:

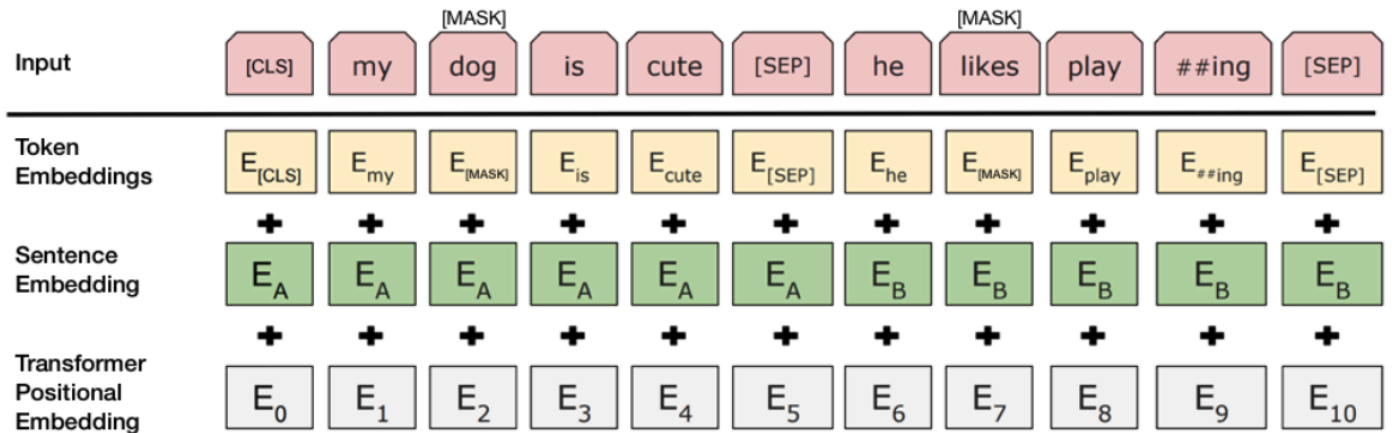
Sentence A	Sentence B	IsNext?
The Indian cricket team won the match against West Indies.	Obama served 2 terms as the president of USA.	No
Nadal won the 2022 Australian open.	With this victory he now has 21 GrandSlam titles to his name.	Yes

But why do we need a NSP and why not just MLM alone?

- In masked language modeling (MLM), the model is learning the contextual information well. But for someother downstream tasks like QnA, text generation etc, MLM was not capturing the relations relationship between sentences.
- By pretraining with the MLM and NSP together, they were able to achieve good performance on QnA and Natural Language Inference (NLI) tasks

Training data and label generation

- The data was generated in such a way that 50% of the time the Sentence B is a continuation of Sentence A and labeled as True for < IsNext >
- In the rest 50%, Sentence B is randomly sampled which is not a continuation of Sentence A and labeled as False for < IsNext >
- The combined length of the two chosen sentences was <= 512 tokens.
- Once each sequence was built, 15% of its tokens were masked for the MLM task



✓ How is the loss calculated?

- As we saw BERT is pretrained on two tasks **MLM** and **NSP** together, the losses from both the tasks are summed together to get the final loss.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

But the training data is not labelled, What type of learning is it?

- This method of training is called semi-supervised learning method.
 - * Semi-supervised training sits between supervised and unsupervised training.
 - * In this method we don't have labels like supervised learning but we try to create labels from the data available and not manually labelling them.

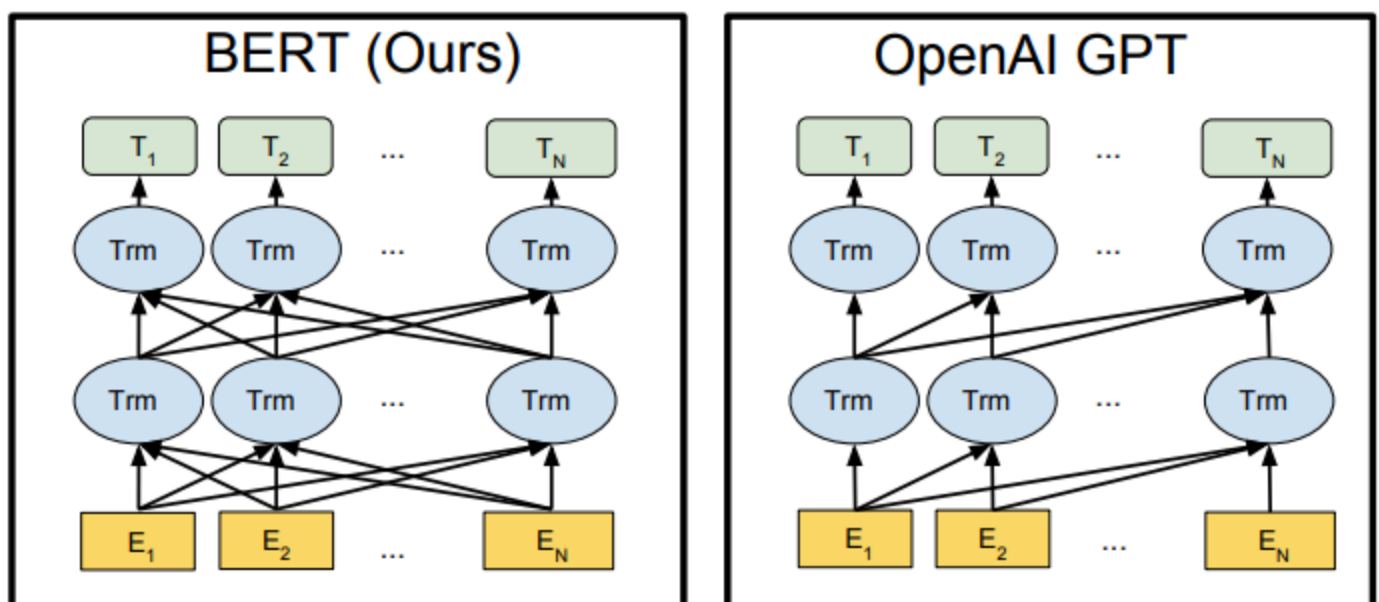
✓ Which of the 2 is it easier to predict?

- Today morning, I went for a ____
- Today morning, I went for a ____ removal to my dentist

In the example above it is easier to predict in the second instance compared to 1st, since we can see what are the words coming after the ____.

What is Bidirectional about Bert?

- Has access to the words to the **left** and **right** of the masked word.
 - * When training the model to predict the masked word it has access to the words to the left and the right side of the masked word.
 - * This method helps in understanding the context of the language better unlike having access only to the words from left of the masked words.
 - * For this reason **BERT** is called a **Masked Language Model**, whereas other architectures like **GPT** are called **Language model** since they have access to words only from the left to make the prediction of the next word.




✓ How are the words tokenized in Bert?

- Word Piece Tokenizer method
 - Bert uses a special tokenization mechanism called the WordPiece Tokenizer where compound words are split into root word and subwords separated with ##.
 - The word-based tokenizer suffers from very large vocabulary, large number of OOV tokens which is addressed by Word Piece Tokenizer
- Let's look at a example

```
from transformers import BertTokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased', do_lower_case=True)

words = ['parachute', 'paraglide', 'paragliding', 'scubadiving', 'scubadiver', 'scubadive']

for word in words:
    tok_word = tokenizer.tokenize(word)
    print(f'{word:<12} tokenized to {tok_word}')
```



parachute	tokenized to ['parachute']
paraglide	tokenized to ['para', '##gli', '##de']
paragliding	tokenized to ['para', '##gli', '##ding']
scubadiving	tokenized to ['scuba', '##di', '##ving']
scubadiver	tokenized to ['scuba', '##di', '##ver']
scubadive	tokenized to ['scuba', '##di', '##ve']