Tech Review on BERT in Apache SparkNLP Context

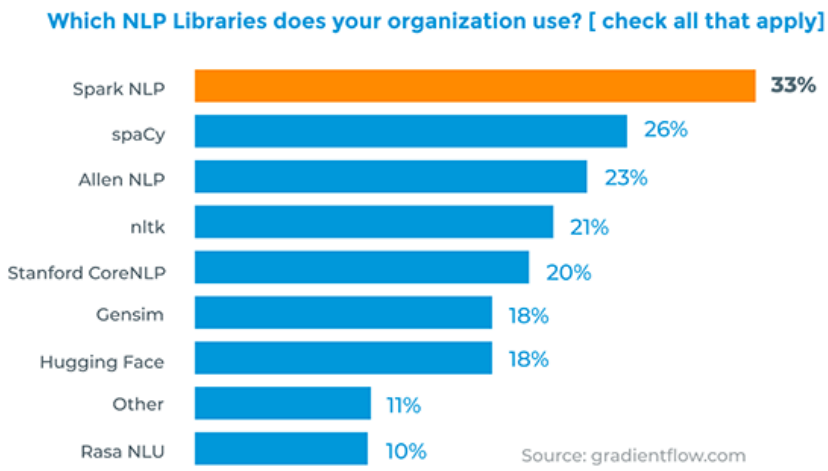
Rahul Sharma | UIUC CS 410 (Text Information Systems)

Abstract

BERT stands for Bidrectional Encoder Representation from Transformers and deep bidirectional in nature. It's new technique for pre-training. Since Google has released it in 2019, there're multiple versions of BERT implemented in the open-source community. BERT architecture is referenced from the many existing NLP models, i.e. GPT, Elmo.

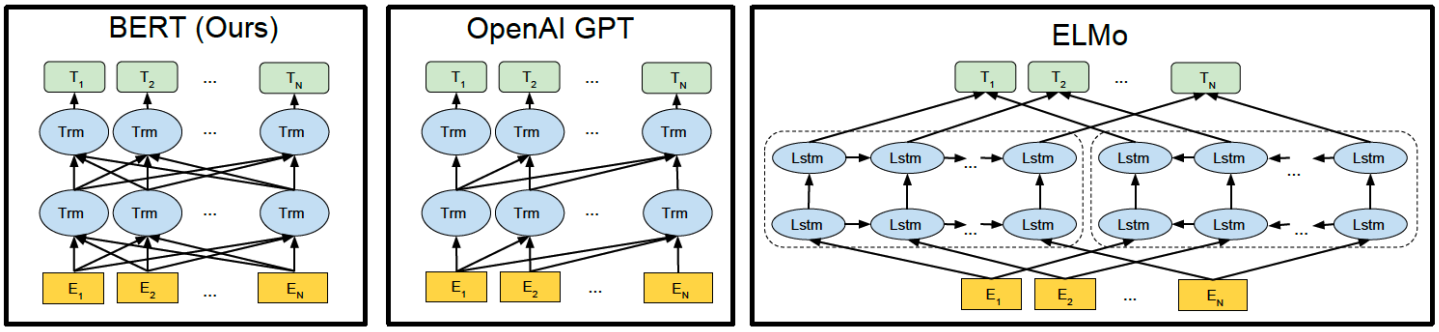
BERT achieves the state-of-art nlp results and performance on NLP tasks i.e. Questional Answering, Langauage inferences, NER, Next sentence prediction and analyze sentences holistically.

In this paper we will be reviewing the BERT architecture and explore BERT in SparkNLP context. SparkNLP allows to perform NLP tasks at the large scale using distributed processing.



BERT Architecture

The standard NLP Language Models extracts context sensitive features from left to right (LTR) or Right to Left (RTL) where token can only attend to LTR or RTL tokens in self-attention layers of transformer and are not deeply bidirectional.



- BERT is deeply bidirectional and overcome with the limitations of unidirectional LMs using Mask Language Model (MLM).
- The existing LM ELMo follow feature based approach whereas BERT is fine-tuning based approach like GPT
- BERT uses a bidirectional Transformer.
- BERT representations are jointly conditioned on Left and Right contexts in all the layers.

Left to right context:

Learning

Learning **Text**

Learning Text **Information**

Learning Text Information **Systems**

Right to Left context:

Systems

Information Systems

Text Information Systems

Learning Text Information Systems

BERT joint on Left and Right contexts:

Learning Text Information Systems

Learning **Text** Information Systems

Learning Text **Information** Systems

Learning Text Information **Systems**

In BERT's Input/Output representation, the first token of every sequence is always a special classification token ([CLS]). Sentence pairs are packed together into a single sequence. It differentiate the sentences in two ways. First, separate them with a special token ([SEP]). Second, It add a learned embedding to every token indicating whether it belongs to sentence A or sentence B.

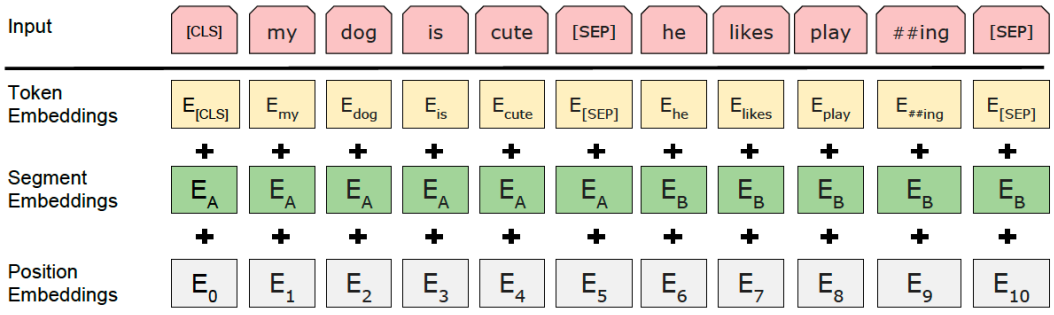


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

BERT- SparkNLP by Johnsnowlabs

Johnsnowlab has implemented NLP models trained several embeddings/pipelines. Here are some of the reasons to use SparkNLP.

- Polyglot: Python, Scala, Java

- Natively Scalable: Backend on Apache Spark project
- Pretrainined NLP pipelines: It covers wide range of NLP pipelines and embedding

Below are SparkNLP sentence embedding model that support 109 languages:

BERT LaBSE, BioBERT Clinical, BioBERT Discharge, ,BioBERT PMC,BioBERT Pubmed,BioBERT Pubmed Large,BioBERT Pubmed PMC,BioBERT Clinical,BioBERT Discharge,BioBERT PMC,BioBERT Pubmed,BioBERT Pubmed Large,BioBERT Pubmed PMC

Usecase exploration BERT

- Named Entity recognition (NER) I am using SparkNLP BERT word embeddings which Leverages Google's Bert via Pyspark. First step is declare the pipeline. NLP Pipelines provide a uniform set of functions built on top of Spark Dataset that is further used to define and tune the model.

  `assembler->sentence->token->bert->loaded_ner_model->converter`

Example:

```
import os
# Install java
! apt-get update -qq
! apt-get install -y openjdk-8-jdk-headless -qq > /dev/null

os.environ["JAVA_HOME"] = "/Library/Java/JavaVirtualMachines/jdk1.8.0_151.jdk/Contents/Home"
os.environ["PATH"] = os.environ["JAVA_HOME"] + "/bin:" + os.environ["PATH"]
! java -version

# Install pyspark
! pip install --ignore-installed pyspark==2.4.4

# Install Spark NLP
! pip install --ignore-installed spark-nlp

from pyspark.sql import SparkSession
from pyspark.ml import Pipeline
from pyspark.sql.functions import *
import sparknlp
from sparknlp.annotator import *
from sparknlp.common import *
from sparknlp.base import *
spark=sparknlp.start()

text = [["Peter Parker is a nice guy and lives in New York."],["Learning Text Information System"]]
spark_df = spark.createDataFrame(text).toDF("text")
doc_assembled_df=assembler.transform(spark_df)
#doc_assembled_df.head(1)
train_df = spark.read.csv("train.csv")

assembler = DocumentAssembler().setInputCol("text").setOutputCol("document")

sentence = SentenceDetector().setInputCols(['document']).setOutputCol('sentence')
doc_sentenced_df=sentence.transform(doc_assembled_df)
#doc_sentenced_df.head(1)

token = Tokenizer().setInputCols(['sentence']).setOutputCol('token')
converter = NerConverter().setInputCols(["document", "token", "ner"]).setOutputCol("ner_span")

bert = BertEmbeddings.pretrained('bert_base_cased', 'en').setInputCols(["sentence",'token']).setOutputCol("bert").

bert_base_cased download started this may take some time.
Approximate size to download 389.1 MB
[OK!]

loaded_ner_model = NerDLModel.pretrained('ner_dl_bert').setInputCols(["sentence", "token", "bert"]).setOutputCol("
```

```
    ner_dl_bert download started this may take some time.
    Approximate size to download 15.4 MB
    [OK!]

    ner_prediction_pipeline = Pipeline(
    stages = [document
    ,sentence
    ,token
    ,bert
    ,loaded_ner_model
    ,converter])

    model=ner_prediction_pipeline.fit(train_df)
    predicted= model.transform(spark_df)
    predicted.show()
```

- Word Embedding:

  Pipeline- `df->sentence->token->bert`

```
word_embeddings = BertEmbeddings.pretrained('bert_base_cased', 'en').setInputCols(["sentence", "token"]).setOutputCol(


    bert_base_cased download started this may take some time.
    Approximate size to download 389.1 MB
    [OK!]


pipeline = Pipeline().setStages(
  [
    assembler,
    sentence,
    token,
    word_embeddings
  ]
)

result=pipeline.fit(spark_df).transform(spark_df)
result.select('embeddings').show()
```

References

- https://arxiv.org/pdf/1810.04805.pdf
- http://jalammar.github.io/illustrated-bert/
- https://www.quora.com/What-is-a-masked-language-model-and-how-is-it-related-to-BERT
- https://nlp.stanford.edu/seminar/details/jdevlin.pdf
- https://arxiv.org/abs/1907.11692
- https://johnsnowlabs.github.io/spark-nlp-workshop/databricks/index.html
- https://colab.research.google.com/github/JohnSnowLabs/spark-nlp-workshop/blob/master/tutorials/blogposts/3.NER_with_BERT.ipynb?authuser=1#scrollTo=8IR30lxz_foj