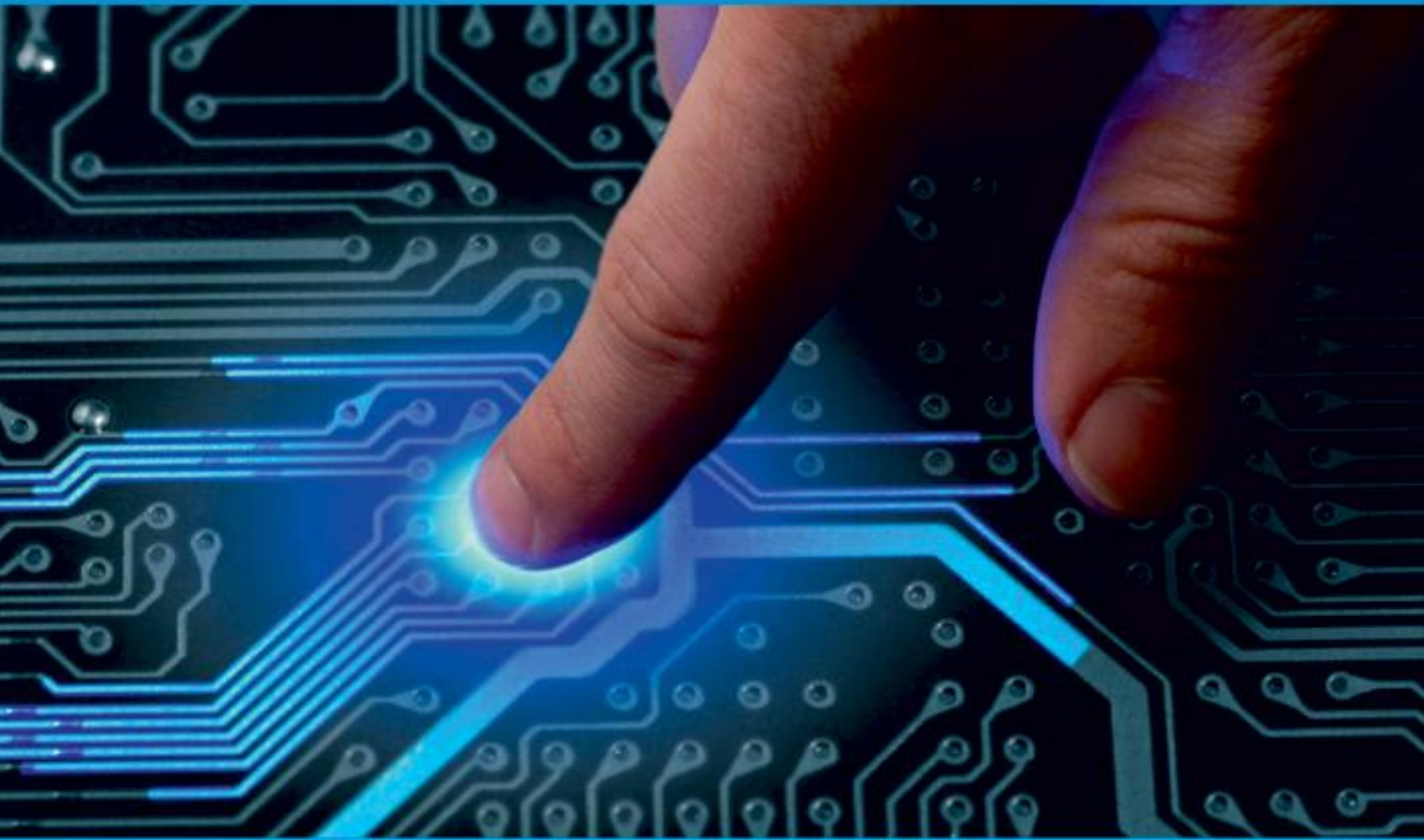
IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 9, Issue 6, June 2021

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.542

Google BERT advanced NLP model

Aakash Singh, Prof. Mrs. Pratibha Adkar

PG student, Department of MCA, Modern College of Engineering Pune, India

Assistant Professor, Department of MCA, Modern College of Engineering, Pune, India

ABSTRACT: BERT is an adjective of Bidirectional Encoder Representatives from Transformers), a network-based approach to processing natural languages prior to training. Training a machine to understand human language writing, speaking, comprehension, and people is a must. And these days the understanding of natural language is widely used in organizations like Google so the concept of NLP is in the picture. Used to help search engines such as Google better understand the content of search terms in search queries.BERT is the work of Google search engines, found by Google in 2018. This means that anyone can use BERT to train their language processing system which has 11 major NLP language functions such as guessing sentences, answering questions, sentence breaks, etc. This paper contains an introduction to BERT, its Architecture, how it differs from traditional NLP models, major functions used by Google using BERT, Fine-tune BERT for its data sets

KEYWORDS*:* BERT, NLP, Fine-tuning, Architecture, Advantages, Applications.

I.INTRODUCTION

Natural Language Processing(NLP) is the field of study that basically focuses on interactions between computers and human language.It is the subset of artificial intelligence. Developers can organize knowledge to perform tasks such as text summarization, translation, sentiment analysis, named entity recognition, speech recognition. NLP can rely on machine learning algorithms rather than hand set large rules since NLP algorithms are based on machine learning algorithms.

BERT is an open machine learning platform for Natural Language Processing. Based on transformers, an in-depth learning model in which every output neuron is connected to its input neuron, and the metals between them are calculated dynamically based on their interactions. Google has launched the BERT open source in 2018. Trained with a vast array of linguistic details, the draft yielded impressive results in major works of understanding natural languages, such as emotional analysis, paragraph labeling, sentence classification, and word meaning.

II. TRADITIONAL APPROACH TO NLP

The traditional way to solve NLP is a consistent flow of many steps, and it is a mathematical method. If we look at the traditional NLP learning model, we can see a set of unique tasks that take place, such as cleaning data with unwanted data, to get good presentations of text data numbers. data representation was a time-consuming step to get better performance on a given NLP task.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542 |

|| Volume 9, Issue 6, June 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0906086 |

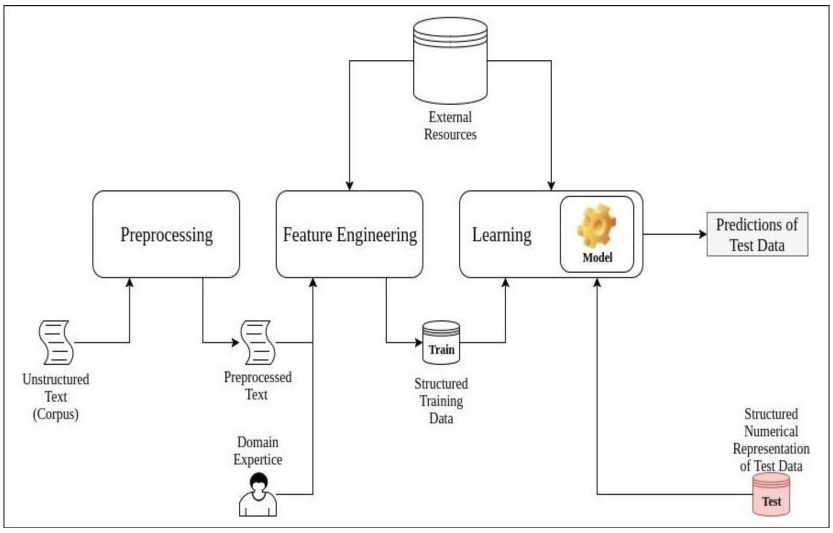


Fig.1: The general approach of traditional NLP

Preprocessing:

The text has operations, such as stemming or lemmatization, removing punctuation, in order to reduce the memory, thus reducing the vocabulary, thus requirement. In short, reduce the size of text while keeping important information.

Tokenization:

It is another preprocessing step that needs to be performed. Tokenization is the process of dividing a corpus into small entities (for example, the tokenization of the sentence “I am eating an apple” would be I, am, eating, an, apple).

Feature engineering :

It is used to convert raw text data into value formats so that the model is trained in that data, for example, to convert text into a wallet or to use n-gram representation. For example a 2-gram representation of the sentence "I eat an apple" would be [(I, am), (am, eating), (eating, an), (an apple)].

III.THE PROBLEMS WITH USING THE TRADITIONAL APPROACH

The progression measures used in traditional NLP can result in the loss of useful information embedded in the text (for example, punctuation and sensitive data) to make learning possible by reducing wording.
Feature engineering needs to be done manually. To build a reliable system, good features need to be set. This process can be tedious as spaces for various features need to be explored more thoroughly.

IV.THE NEEDS OF BERT

A key part of the BERT framework is Transformers and the advantage of Transformer models is that they are not sequential, which means they do not require the input sequence to be processed sequentially. This allows Transformers to be compared and measured more easily than traditional NLP models. Transformer models have so far shown better performance and speed than other, more traditional models. The basic design of the BERT model is based on the use of training from two Transformer model models in language modeling. In contrast to direct models, which read text input sequentially (from left to right or right-left), the Transformer encoder reads it in two directions, meaning word order simultaneously. The results shown by the BERT model show that a bilingual-trained language model may have a deeper sense of language and flow status than self-directed language types. The BERT model variety now hits all types of records throughout the NLP work range, such as text classification, document submission, emotional analysis, answer questions, sentence matching, etc.

V.BERT ARCHITECTURE

The BERT construction model is a multi-layer bidirectional Transformer encoder. Because the use of Transformers has become ubiquitous recently and our implementation is very similar to the original.
In this case, the number of layers (i.e., Transformer blocks) is denoted as L, the hidden size as H, and the number of self-attention heads as A. In all cases the feed-forward/filter size is 4H, i.e., 3072 for the H = 768 and 4096 for the H = 1024.
BERT has the following major component in its architecture:

The Transformers :
The transformer is an architecture for transforming one sequence into another one with the help of two parts (Encoder and Decoder), but it differs from the sequence-to-sequence models because it does not imply any Recurrent Networks (GRU, LSTM, etc.). Recurrent Networks were, until now, one of the best ways to capture the timely dependencies in sequences. However, the transformers researchers proved that architecture with only attention-mechanisms without any RNN (Recurrent Neural Networks) can improve on the results in translation tasks and other tasks.

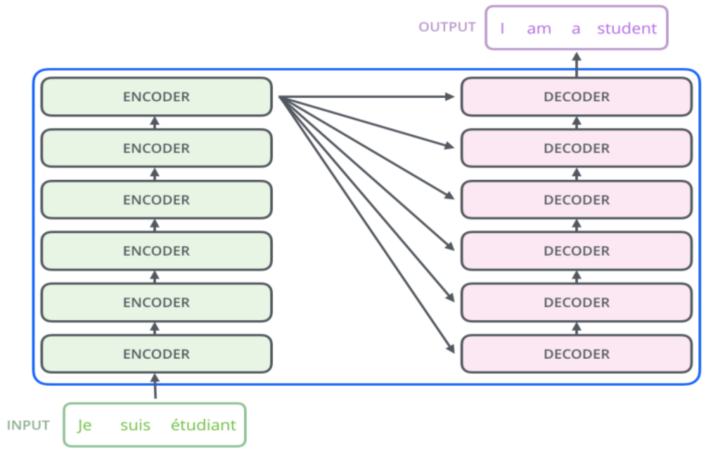


Fig.2: Encoder-decoder in a transformers

The encoder's inputs first flow through a self-attention layer, it is a layer that helps the encoder look at other words in the input sentence as it encodes a word. The exact same feed-forward network is independently applied to each position. The decoder has both those layers, but between them is an attention layer that helps the decoder focus on relevant parts of the input sentence.

Input Representation:
BERT requires a specific input format to train one's own dataset as it was trained on. Input representation is able to represent both a single text sentence or a pair of text sentences in one token sequence.

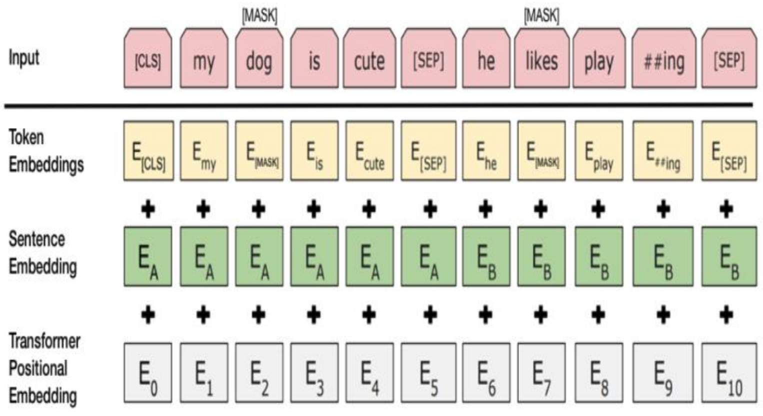

Input
[MASK]
[MASK]
[CLS] my dog is cute [SEP] he likes play ##ing [SEP]
Token Embeddings
Sentence Embedding
Transformer Positional Embedding

Fig.3: Input representation

Token Embeddings

The first token of every sequence is always the special classification embedding ([CLS]). The final hidden state (i.e., the output of Transformer) corresponding to this token is used as the aggregate sequence representation for classification tasks. For non-classification tasks, this vector is ignored. Sentence pairs are packed together into a single sequence. Which differentiates the sentences in two ways. First, we separate them with a special token ([SEP]). Second, add a learned sentence A embedding to every token of the first sentence and a sentence B embedding to every token of the second sentence.

Sentence Embedding

For single-sentence inputs only used the sentence Aembeddings, B embeddings to the next sentence and so on.

Positional Embedding

The position of each word with a unique id is assigned so the word individually can be identified.

Pre-training Tasks:

Unlike traditional models, BERT doesn’t use traditional left-to-right or right-to-left language models to pre-train BERT. Instead, pre-train BERT using two novel unsupervised prediction tasks.

Task #1: Masked LM

It is reasonable to believe that a deep bidirectional model is strictly more powerful than either a left-to-right and right-to-left model. Unfortunately, standard conditional language models can only be trained left-to-right or right-to-left, since bidirectional conditioning would allow each word to indirectly “see itself” in a multi-layered context. In order to train a deep bidirectional representation, BERT takes a straightforward approach of masking some percentage of the input tokens at random and then predicting only those masked tokens. This procedure is known as a “masked LM” (MLM). In this case, the final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary, as in a standard LM. In all of our experiments, we mask 15% of all WordPiece tokens in each sequence at random.

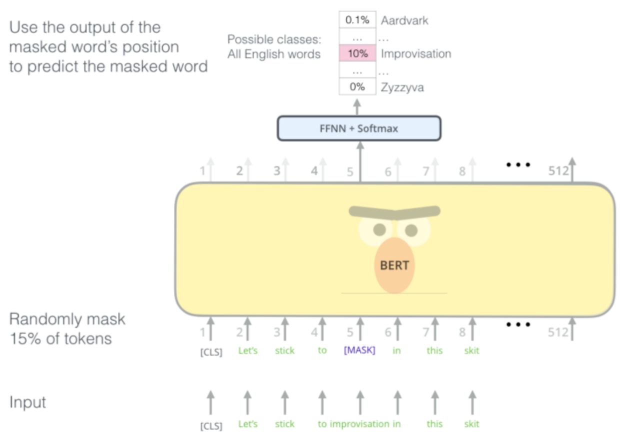

Use the output of the masked word's position to predict the masked word
Possible classes: All English words
0.1% Aardvark
...
10% Improvisation
...
0% Zyzzyva
FFNN + Softmax
1 2 3 4 5 6 7 8 ... 512
BERT
Randomly mask 15% of tokens
[CLS] Let's stick to [MASK] in this skit
Input
[CLS] Let's stick to improvisation in this skit

Fig.4 Masked Language Model

Rather than always replacing the chosen words with [MASK], the data generator will do the following:

- 80% of the time: Replace the word with the [MASK] token, e.g., I love my dog -- I love my [MASK].
- 10% of the time: Replace the word with a random word, e.g, I love my dog -- I love my cat.
- 10% of the time: Keep the word unchanged, e.g., I love my dog --I love my dog.

The purpose of this is to bias the representation towards the actual observed word.
The Transformer encoder doesn't know which words it will be asked to predict or which have been replaced by random words, so it is forced
to keep a distributional contextual representation of every input token.
The second downside of using an MLM is that only 15% of tokens are predicted in each batch, which suggests that more pre-training steps may be required for the model to converge.

Task #2: Next Sentence Prediction
Many important downstream tasks such as Question Answering (QA) are based on understanding the relationship between two text sentences, which is not directly captured by language modeling. In order to train a
model that understands sentence relationships, when choosing the sentences A and B for each pretraining example, 50% of the time B is the actual next sentence that follows A, and 50% of the time it is a random sentence from the corpus. For example:
Input = [CLS] the man went to [MASK] store [SEP] he bought a gallon [MASK] milk [SEP]
Label = IsNext
Input = [CLS] the man [MASK] to the store [SEP] penguin [MASK] are flight ##less birds [SEP]
Label = NotNext
BERT training chooses the NotNext sentences completely at random, and the final pre-trained model achieves 97%-98% accuracy at this task.

VI.FINE-TUNING PROCEDURE

Fine-tuning, in general, means making small adjustments to a process to achieve the desired output or performance. Fine-tuning deep learning involves using weights of a previous deep learning algorithm for programming another similar deep learning process.
Assume our Task is of sequence classification, we would use this for our purpose. This expects our sentences to be started and terminated with ‘cls’ and ‘sep’ respectively. We will firstly reformat our reviews or text column in this format. This is done because Bert uses ‘sep’ in the next sentence prediction task. Then, we would use BertTokenizer to tokenize text data in the Bert format. For a given token or word, the tokenizer will keep the word as such if the token is found in Bert’s vocabulary else it will find the small subword which will be in Bert’s vocabulary.

We will now convert tokens to word ids as per Bert's vocabulary. We also need to pad the sequences to make them of fixed length. For a given tokenized text, we also need to distinguish whether it is a part of the token or padding. This is done using attention masks.
Now, we will split our train data into 80:20. We will train our model on 80 percent of data and test that on 20 percent of data. Rest is simply following a routine Pytorch method of making a data loader and training the model. All of it has been done and explained extensively in the repository. By using a pre-trained model, we get an accuracy of nearly 90.7 on our test data. For the last step of this tutorial, we used the fine-tuned language model and kept all other controllable parameters the same such as learning rate, epochs and so on. The observed accuracy is 90.9 on test data in this case. This is a negligible improvement inaccuracy. We can see that with the language model, the score improved by 0.15 percent. See, that our data is reviews written in the English language, this is pretty similar to the task on which Bert was trained. We can see a significant result if our data is relatively different from the data on which Bert was pre-trained.

Fine-Tuning for Question Answer:
BERT model is fine-tuned to perform this task in the following way:

1. Context and the question are preprocessed and passed as inputs.
2. Take the state of the last hidden layer and feed it into the start token classifier. The start token classifier only has a single set of weights which it applies to every word. After taking the dot product between the output embeddings and the start weights, apply the softmax activation to produce a probability distribution over all of the words. Whichever word has the highest probability of being the start token is the one that was picked.
3. Repeat this process for the end token — we have a separate weight vector for this.

For example, we have the reference text as below to feed our trained model,
Reference text = "ViratKohli is an Indian cricketer who currently captains the India national team. A right-handed top-order batsman, Kohli is regarded as one of the best batsmen".

Question = "Who is ViratKohli?"

Our model will calculate separate start word scores and end scores as per the question and together both word scores combined and returns the answer to the asked question.

Answer= "an Indian cricketer"

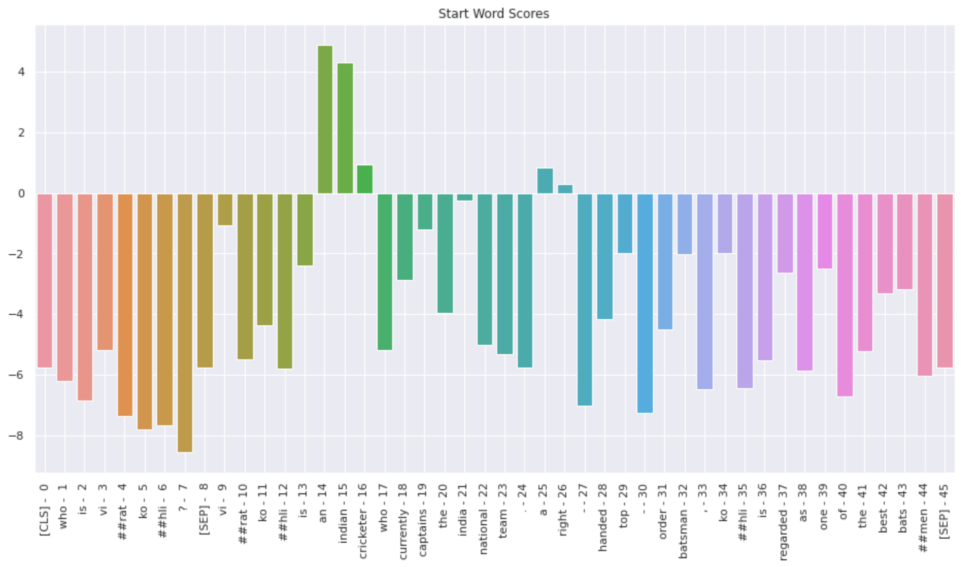


Fig.5: Start word scores to the reference text.('an',' indian' words has high score)

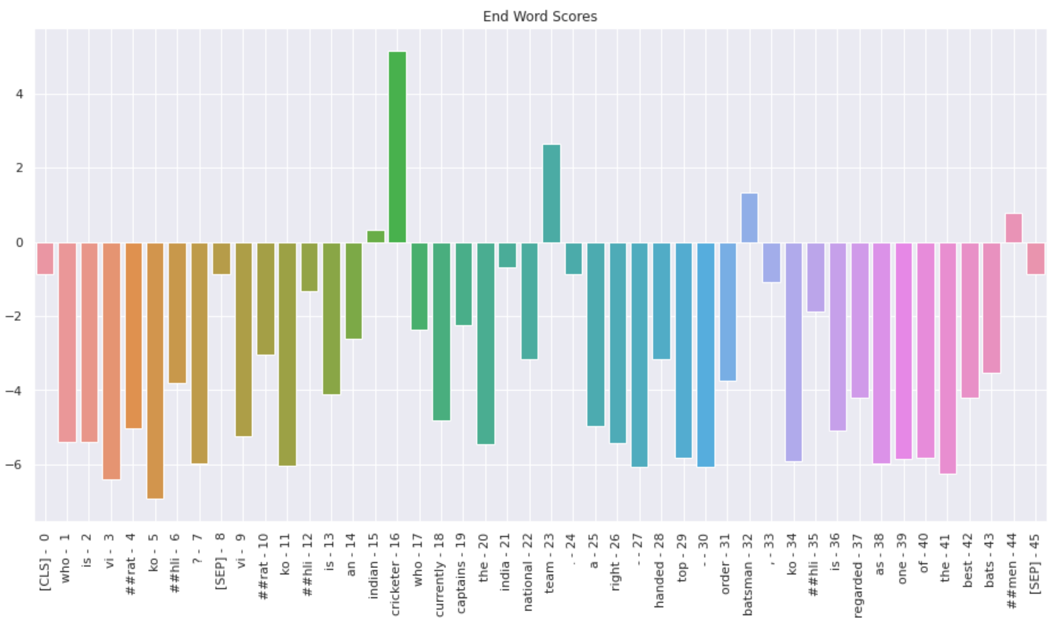


Fig.6: Endword scores to the reference text. ('cricketer' word has high score)

VII.ADVANTAGES AND DISADVANTAGES OF BERT

Advantages:

Outperformed Accuracy:

Transformers being a major component of BERT's architecture is the main reason behind BERT's outperformed accuracy over other modern models or of course traditional models. Transformers have the capability to look over text from left-to-right, right-to-left which makes fine-tuned models to predict accurately.

Speed:

GPU is required to run any deep learning model, deep learning calculates the input weights very fast, to train any traditional NLP model may take too long but it takes less time to train the BERT model.

Fine-tuned with any dataset:

Based on one's need one can use BERT with their own dataset for the major tasks like Question-Answer, Sentiment classification, next sentence prediction, etc.

Helps Google better discern the context of words in search queries:
It helps Google understand natural language better, particularly in conversational search. In the year preceding its implementation, BERT has caused a storm of activity in production search.

Disadvantages
Not easy to implement:
Fine-tuning BERT isn't an easy task, to train the own dataset requires the same input format which BERT was trained on, and parameters to modify is the crucial task.
High computational resources are needed to train or fine-tuned:To get efficient accuracy, more data is required, and to train that much data a highly GPU-enabled system is needed.

VIII.APPLICATIONS

1. Google: Google uses to discern its search queries.
2. SEO: BERT analyzes search queries, not web pages. On-page SEO becomes more important in terms of using words in precise ways.
3. Chatbots: Questions - Answering system helps to find information more efficiently in many cases, and goes beyond the usual search, answering questions directly instead of searching for content similar to the query.

4. Language models: A sentiment classification model can be used by big companies like Facebook, Twitter to classify good or bad tweets.

IX. CONCLUSION

BERT is a groundbreaking technology, not only for NLP in general but also for Google Search. However, it’s important to understand that BERT doesn’t change the way in which websites get ranked but simply improves Google’s understanding of natural language. For Google, this is an important step in reaching its ultimate goal: understanding exactly what users want in each and every situation.

REFERENCES

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” Association for Computational Linguistics, (2018) , Vol.1, No.3128.
[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” Association for Computational Linguistics, (2019), Vol.1, No.4171.
[3] Guida, G.; Mauri, G. (July 1986). "Evaluation of natural language processing systems: Issues and approaches". *Proceedings of the IEEE*. 74 (7): 1026–1035.
[4]Goldberg, Yoav (2016). "A Primer on Neural Network Models for Natural Language Processing".*Journal of Artificial Intelligence Research*. 57: 345–420.
[5] Clark, Kevin; Khandelwal, Urvashi; Levy, Omer; Manning, Christopher D. (August 2019). "What Does BERT Look at? An Analysis of BERT's Attention".*Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Florence, Italy: Association for Computational Linguistics: 276–286.
[6] https://en.wikipedia.org/wiki/Natural_language_processing
[7] https://www.lexalytics.com/technology
[8]https://medium.com/analytics-vidhya/fine-tuning-bert-language-model-to-get-better-results-ontext-classification-3dac5e3c348e
[9]https://subscription.packtpub.com/book/application_development/9781788478311/1/ch01lvl1s ec12/the-traditional-approach-to-natural-language-processing
[10]https://searchenterpriseai.techtarget.com/definition/BERT-language-model