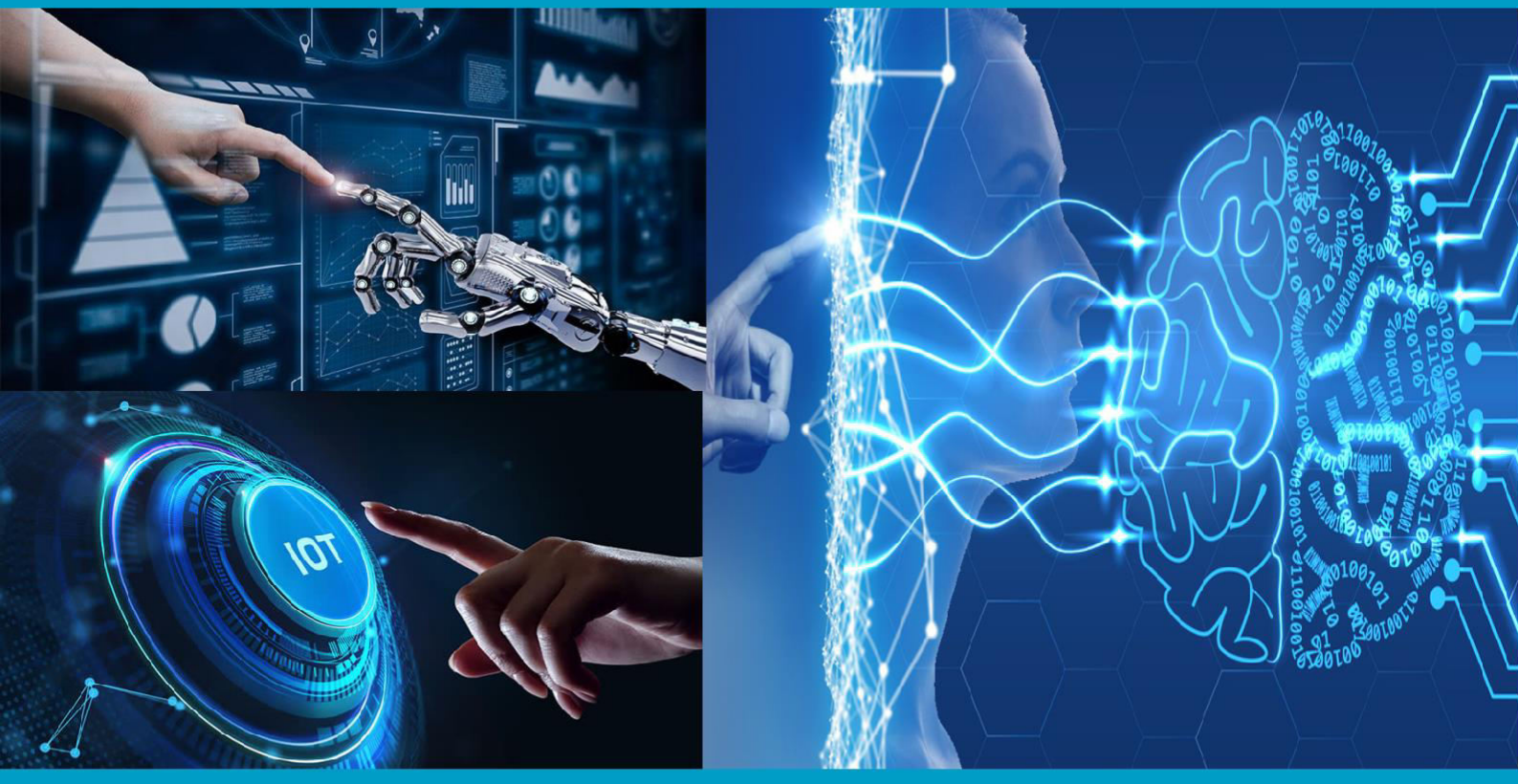
IJMRSETM

e-ISSN: 2395 - 7639



INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH

IN SCIENCE, ENGINEERING, TECHNOLOGY AND MANAGEMENT

Volume 12, Issue 1, January 2025

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.802

International Journal of Multidisciplinary Research in Science, Engineering, Technology & Management (IJMRSETM)

| ISSN: 2395-7639 | www.ijmrsetm.com | Impact Factor: 7.802 | A Monthly Double-Blind Peer Reviewed Journal |

| Volume 12, Issue 1, January 2025 |

Pre-Training and Fine-Tuning Bidirectional Encoder Representations with Transformers

Rohan Rao, Swati Chopade

P.G. Student, Department of Master of Computer Applications, Veermata Jijabai Technological Institute, Matunga, Mumbai, India

Professor and Head of Department, Department of Master of Computer Applications, Veermata Jijabai Technological Institute, Matunga, Mumbai, India

ABSTRACT: This research paper describes a language model called Bidirectional Encoder Representations using Transformers (BERT), used to perform several natural language tasks. With several language models for several natural language tasks already in existence, one might wonder what is so special about BERT. Developed by Google in 2018, BERT widely promoted the concept of transfer learning, in which a pre-trained model is built over large amounts of data, and the same model is fine-tuned or adjusted to be used for several other natural language tasks. This paper mentions the short-comings of existing language models, and describes how BERT is able to overcome these short-comings with its bidirectional training approach and context-based modelling. This paper describes the architecture of BERT, which is loosely based on the transformer architecture, especially paying close attention to the encoder representation, which forms the majority of the BERT architecture. We also delve into the inner workings of the BERT model, describing the two unsupervised tasks, Masked Language Modelling (MLM) and Next Sentence Prediction (NSP) on which BERT is trained. The objective of this paper is to verify if we can use a pre-trained BERT model, and fine-tune its parameters to develop a high-quality model that provides accurate results, with lesser training time and effort. In this paper, we fine-tune a pre-trained BERT model over a text classification task, and evaluate its performance against the General Language Understanding Evaluation (GLUE) benchmark, on which several other language models are competing.

KEYWORDS: Word embeddings, Context-based modelling, Bidirectional, Pre-training, Fine-tuning, Text Classification, Transformer, Encoder, Self-attention, Hidden size, Masked Language Modelling, Next Sentence Prediction, GLUE Benchmark, CoLA dataset, Matthew's Correlation Coefficient

I. INTRODUCTION

The core solution to most natural language problems these days is Language Models. Language models are trained on large amounts of text in an unsupervised manner, and given a sentence or a sequence of text, they have the ability to predict the next word (or token) of the sequence. Language models are able to do so by means of a probability distribution that allows the model to predict the likelihood of the next word in the sentence, given the previous words of the sentence.

Language models input a sentence (or sequence of words) in the form of tokens, which are the words of the sentence. Each of these tokens are converted into feature vector representations called word embeddings. These word embeddings can then be compared for similarity retrieval tasks, or can be fed to an output layer that performs some natural language task on them.

Most language models these days are context-free models i.e.- they produce the same word embeddings for words that are spelt the same way, but have different contexts. Context-free models do not take into consideration the context of words. This was an inherent flaw of language models that had to be rectified. Fig. 1 shows the word embedding created by context-free models.

| ISSN: 2395-7639 | www.ijmrsetm.com | Impact Factor: 7.802 | A Monthly Double-Blind Peer Reviewed Journal |

| Volume 12, Issue 1, January 2025 |

Fig. 1 Context-free models produce the same word embedding for the word 'fair' although the word has a different meaning in both the sentences

BERT, on the other hand, is a context-based model i.e.- it produces different word embeddings for words that are spelt the same way, but have different contextual meanings. Fig. 2 shows the word embeddings created by context-based models.

Fig. 2 Context-based models produce different word embeddings for the word 'fair' on the basis of their context in the sentences

Existing language models are capable of performing a single natural language task at a time. If we wanted to perform 2 natural tasks, for example Text Classification and Question Answering, we would have to perform both the tasks on separate language models designed for their respective purpose. BERT paves the way for transfer learning mechanisms, in which, we pre-train a single model, which acts a base for several other models. We can re-use the same pre-trained model for several natural tasks, all we have to do is to fine-tune the parameters of the pre-trained model on the basis of the natural language task to be performed. For example, the output of a pre-trained model can be fed to a classifier that performs classification, while the same output can be provided to a question-answering model. This ensures quicker development, while also saves memory since we are re-using the same model and parameters for several natural language tasks.

Existing language models are auto-regressive (unidirectional) in nature i.e.- they are able to read a piece of text either from left to right or from right to left, but not both simultaneously. Due to this learning approach, the model restricts itself to understanding the sentence from one perspective only. BERT, on the other hand, is an auto-encoding (bidirectional) model i.e.- it reads a text from both left to right, as well as from right to left simultaneously. This approach allows the model to better understand the context of the sentence.

These features have led to the popularity of BERT among other language models. BERT is built using the transformer architecture, and it is imperative to briefly understand the working of a transformer before we delve into the architecture of BERT. We now discuss certain concepts about transformers that will enhance our understanding of the BERT model.

| ISSN: 2395-7639 | www.ijmrsetm.com | Impact Factor: 7.802 | A Monthly Double-Blind Peer Reviewed Journal |

| Volume 12, Issue 1, January 2025 |

II. LITERATURE REVIEW

Since BERT is built on top of the Transformer architecture, it is imperative to first understand what is a transformer and how it works internally. A transformer is a sequence-to-sequence neural network that transforms a given sequence from one representation to another. It is mostly used for machine translation, in which a sentence is translated from one language to another.

Sequence-to-sequence models make use of an encode and a decoder. The encoder maps an input sentence into a higher-dimensional (n-dimensional vector). The decoder maps this vector into an output sequence, which can be in another language [1].

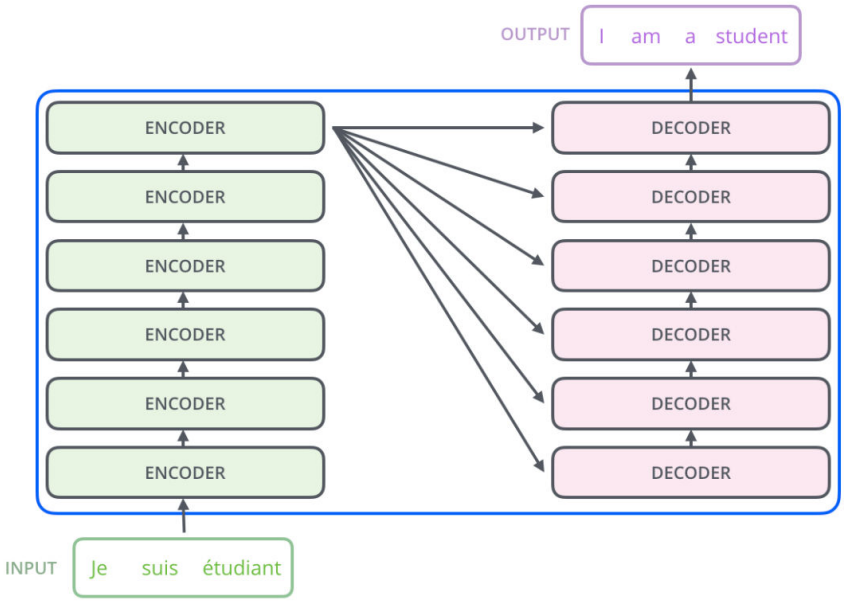


Fig. 3 Transformer Architecture

As shown in Fig. 3, a transformer generally consists of 6 encoder layers, which are connected to 6 decoder layers. The input sentence is provided to the first encoder, which processes the input, and consequently feeds it to the next encoder layer and so on. Once the sentence is converted into its vector representation, it is fed to the decoders, which map the vector into the translated output sentence [2].

III. ARCHITECTURE OF BERT

BERT builds on top of the transformer architecture by eliminating all the decoder layers from it, and simply retaining all the encoder layers, stacked one above each other. There are several variants of the BERT model, but two of the most popular and widely-used variants are BERT Base and BERT Large. Fig. 4 shows the internal configurations, such as number of transformers, size of hidden layers, number of attention heads, number of parameters etc. for the BERT Base and BERT Large models.

	Transformer Layers	Hidden Size	Attention Heads	Parameters	Processing	Length of Training
BERTbase	12	768	12	110M	4 TPUs	4 days
BERTlarge	24	1024	16	340M	16 TPUs	4 days

Fig. 4 Configurations for the BERT Base and BERT Large variants

| ISSN: 2395-7639 | www.ijmrsetm.com | Impact Factor: 7.802 | A Monthly Double-Blind Peer Reviewed Journal |

| Volume 12, Issue 1, January 2025 |

BERT Base consists of 12 encoder layers, while BERT Large consists of 24 encoder layers. As shown in Fig. 5, each encoder consists of 2 sub-layers – a Self-Attention layer (or Attention Head), and a Feed-forward neural network. An input sentence is first passed to the Self-attention layer of the encoder, which understands the context of each word in the sentence by comparing it with every other word in the sentence. The output of the Self-attention layer is fed to a feed-forward neural network, which consists of several hidden layers [2].

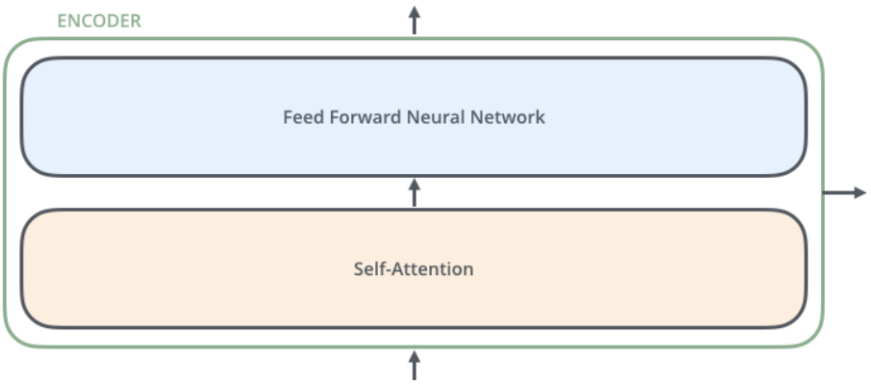


Fig. 5 Components of an Encoder

In the BERT Base configuration, each encoder contains 12 attention heads, and the feed-forward neural network contains 768 hidden layers, while in the BERT Large configuration, each encoder contains 16 attention heads, and the feed-forward neural network contains 1024 hidden layers [3].

IV. WORKING OF BERT

A. INPUT

We cannot directly input sentences in their raw form into the BERT model. We need to pre-process these sentences, by transforming each word of the sentence into a word embedding. A word embedding is a feature vector representation of that word.

The BERT vocabulary contains a total of 30,000 tokens, each having a size of 768 features. In order to transform a word into its word embedding, we have to perform 3 types of embeddings – Token Embedding, Segment Embedding, and Positional Embedding [4]. Fig. 6 shows how a sentence is broken down into token embeddings, segment embeddings, and positional embeddings.

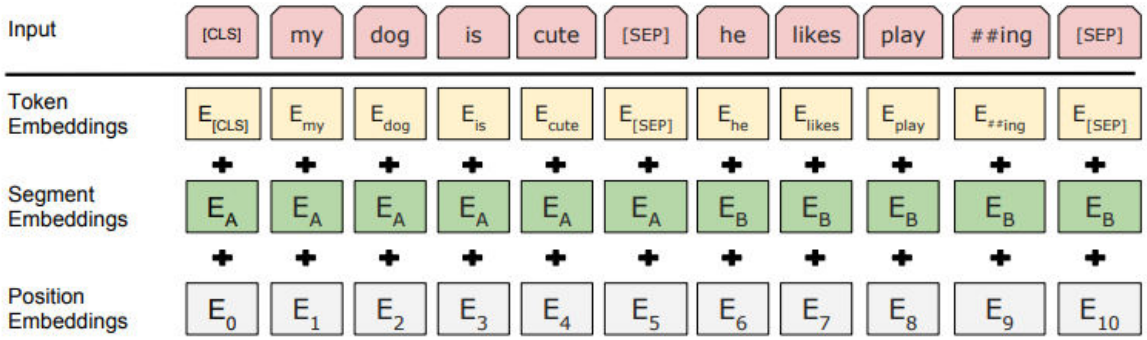


Fig. 6 An input sentence broken down into token embeddings, segment embeddings, and position embeddings

In token embedding, we assign a token to each word of the sentence. In addition to this, we have to add 2 special tokens – [CLS] and [SEP] to the sentence. The [CLS] token, which stands for Classification token, is appended to the start of the input sentence. If there are multiple input sentences, the [CLS] token is appended only at the start of the first sentence. The [SEP] token, which stands for Separator token, is appended to the end of every input sentence. For

| ISSN: 2395-7639 | www.ijmrsetm.com | Impact Factor: 7.802 | A Monthly Double-Blind Peer Reviewed Journal |

| Volume 12, Issue 1, January 2025 |

example, for a given input sentence “My dog is cute”, the assigned tokens will be E(CLS), E(my), E(dog), E(is), E(cute), E(SEP).

In segment embedding, we assign a token to each input sentence provided i.e.- all words within a given sentence are assigned the same token. For example, if there are two sentences, sentence A and sentence B, all the words in sentence A are assigned a token E(A), while all the words in sentence B are assigned the token E(B).

In positional embedding, we assign a numerical token to each word of the input sentences. These numerical tokens are like indexes, which represent the position of each word in the input sentences. For example, we assign tokens of the form E(0), E(1), E(2), and so on.

Finally, we add the token embedding, segment embedding, and positional embedding of each word to form its word embedding. The word embeddings for all the words of the input sentence form the input to the BERT model.

B. PRE-TRAINING

BERT is pre-trained on large amounts of existing literature used to train language models. It is primarily trained on the BooksCorpus (800 Million words) and Wikipedia (2500 Million words). Since BERT is a bidirectional model, we do not train it on any traditional left-to-right or right-to-left models. Instead, we train it on two unsupervised tasks, called Masked Language Modelling (MLM) and Next Sentence Prediction (NSP) [4].

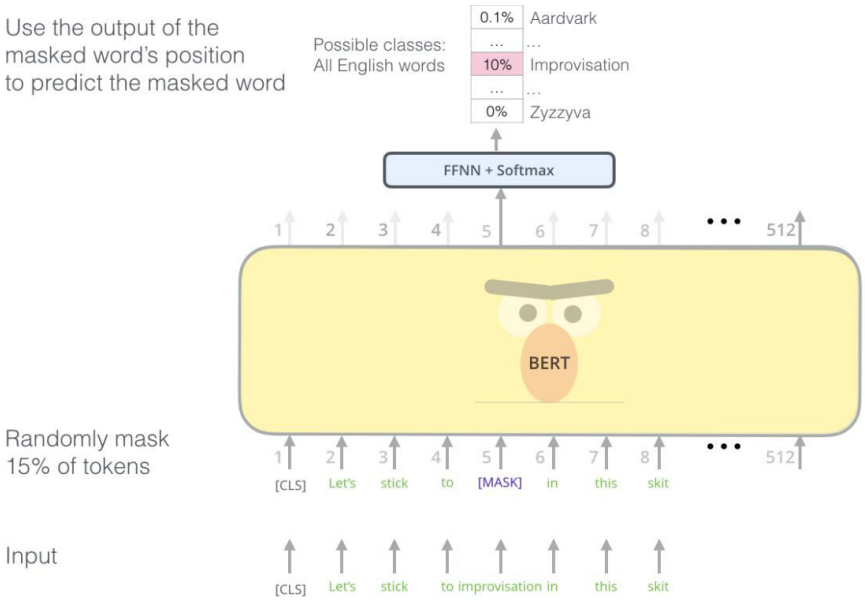


Fig. 7 Pre-training the model on Masked Language Modelling (MLM)

In Masked Language Modelling, we randomly mask a certain percentage (generally 15%) of the input tokens. The randomly selected tokens are replaced by the [MASK] token. As shown in Fig. 7, the model then forms a probability distribution to predict the words that were masked. In order to improve the performance of the fine-tuning stage, we use certain permutations while masking the tokens. Around 80% of the time, we replace selected tokens with the [MASK] token, around 10% of the time, we replace the selected tokens with other tokens, and around 10% of the times, we leave the selected tokens as it is with no replacement [4].

| ISSN: 2395-7639 | www.ijmrsetm.com | Impact Factor: 7.802 | A Monthly Double-Blind Peer Reviewed Journal |

| Volume 12, Issue 1, January 2025 |

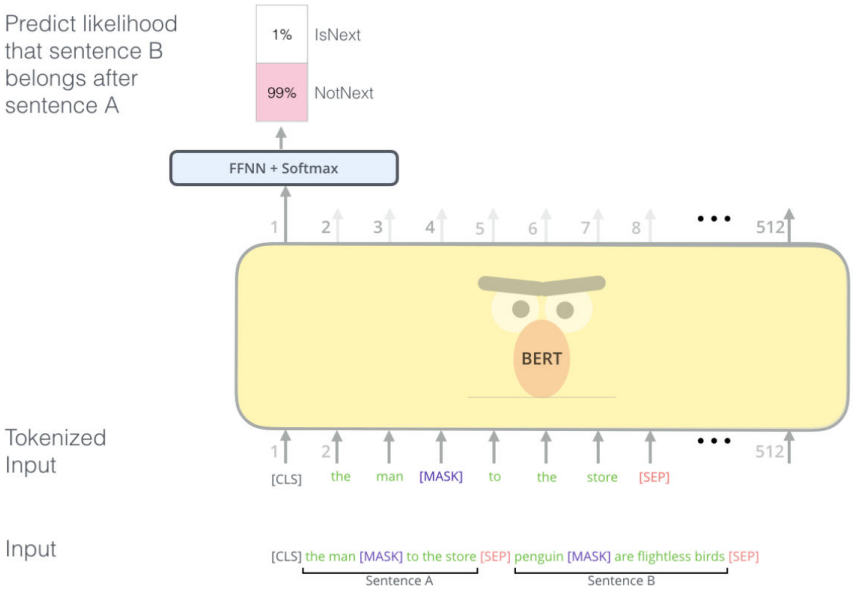


Fig. 8 Pre-training the model on Next Sentence Prediction (NSP)

In Next Sentence Prediction, the model tries to understand the relationship between two sentences. As shown in Fig. 8, given two sentences, sentence A and sentence B, the model tries to predict the likelihood of sentence B being a logical successor to sentence A. While pre-training the model, in 50% of the training samples, sentence B is the logical successor to sentence A (labelled as IsNext), while in 50% of the training samples, sentence B is not the logical successor to sentence A (labelled as NotNext) [4].

C. FINE-TUNING

The pre-trained BERT model can now be used to perform almost any Natural Language Processing task. All we have to do is to provide the task-specific inputs and outputs to the model, while also fine-tune the parameters of the model as per the task to be performed. For example, the sentence pairs (sentence A and sentence B) used in the pre-training phase can be used for question answering, hypothesis testing, text classification, sequence tagging etc.

Based on the NLP task to be performed, we feed the output token vectors of the pre-trained BERT model to an output layer, which performs the required task. For example, for token-specific tasks like question answering or sequence tagging, we feed all the token vectors to the output layer, while for sequence-specific tasks like classification, we feed the [CLS] token vector to the output layer [4].

D. OUTPUT

As shown in Fig. 9, the word embeddings for all the input tokens are sent to the first encoder in the BERT model. These embeddings pass through the self-attention phase and feed-forward neural network phase of the first encoder, which are then passed to the second encoder, and so on till the last encoder in the BERT model. The BERT model finally outputs a feature vector of size X, where X is the hidden size of the model i.e.- the number of hidden layers present within the feed-forward neural network of the encoders. For BERT Base, the hidden size of the model is 768. Therefore, the BERT Base model outputs feature vectors of size 768 for all the input tokens.

These feature vectors can then be fed to an output layer, whose parameters can be fine-tuned based on the requirement, and can be used to perform any natural language task [5].

| ISSN: 2395-7639 | www.ijmrsetm.com | Impact Factor: 7.802 | A Monthly Double-Blind Peer Reviewed Journal |

| Volume 12, Issue 1, January 2025 |

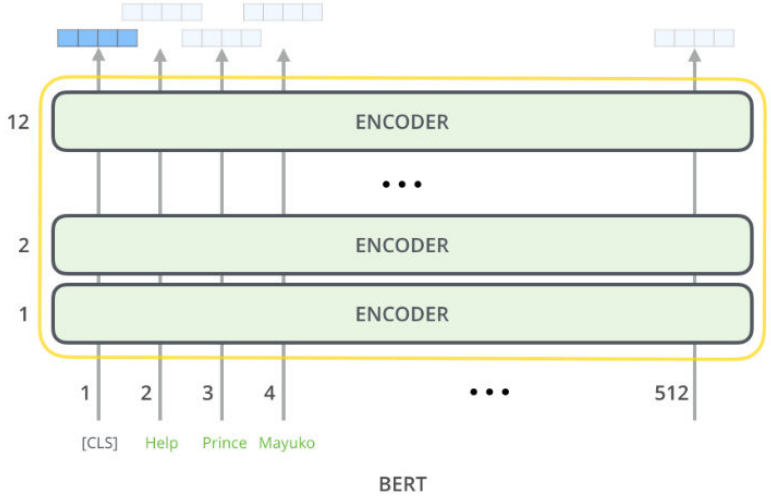


Fig. 9 Input processed through the 12 encoder layers to output feature vectors for each token

V. IMPLEMENTATION

In this research paper, we are going to fine-tune the pre-trained BERT Base model on a text-classification task. Given a sentence, our model will try to classify a sentence as grammatically correct (1) or grammatically incorrect (0). We will evaluate our model's performance on the GLUE benchmark against a single-sentence classification task using the Corpus of Language Acceptability (CoLA) dataset. GLUE is an evaluation benchmark that measures the performance of language models across various natural language tasks. It consists of nine different natural language tasks, including sentiment analysis, text classification etc.

A. DATASET

We will be using the CoLA dataset, which is used for single-sentence classification tasks. CoLA is one of the tests included in the GLUE benchmark, on which the several language models like BERT are competing. The dataset contains a total of 10,657 sentences, of which, 9,594 sentences are reserved for the training set, while 1,063 sentences are reserved for the test set [6].

The dataset provides data in 2 forms – raw form and tokenized form. However, we will utilize the raw data for our model training, since BERT follows its own process for tokenization.

clc95	0	*	In which way is Sandy very anxious to see if the students will be able to solve the homework problem?
c-05	1		The book was written by John.
c-05	0	*	Books were sent to each other by the students.
swb04	1		She voted for herself.
swb04	1		I saw that gas can explode.

Fig. 10 Sample records from the CoLA dataset

As shown in Fig. 9, the CoLA dataset contains 4 columns, the first column displays the source of the record, the second column displays the acceptability judgement label for the record (1 – acceptable, 0 – not acceptable), the third column displays the acceptability judgement label for the record as specified by the author, while the fourth column displays the sentence.

| ISSN: 2395-7639 | www.ijmrsetm.com | Impact Factor: 7.802 | A Monthly Double-Blind Peer Reviewed Journal |

| Volume 12, Issue 1, January 2025 |

B. PRE-PROCESSING

As discussed above, we cannot directly input sentences to the BERT model. We need to tokenize our sentences to form word embeddings, which are then input to the model. For this purpose, we use the BertTokenizer class from the transformers library, which creates a list of all the tokens (words) in the input sentence, maps each token to its token id in the BERT vocabulary, and appends special tokens [CLS] (classifier) to the beginning of the sentence, and [SEP] (separator) to the end of the sentence.

We then pad (truncate) all sentences to the same length using the [PAD] token. Finally, we create attention masks to differentiate between normal tokens and padded tokens. Normal tokens are masked to a value (1), while padded tokens are masked to a value (0). At last, we input the token ids and their corresponding attention masks to the BERT model.

C. METHODOLOGY

In our implementation, we use the Hugging Face library "transformers", which provides PyTorch interfaces for several pre-trained models like BERT, GPT, GPT-2 etc. From the transformers library, we will use the BertForSequenceClassification class to fine-tune our pre-trained model. We follow a train-test split of 90% and 10% respectively.

We will train our model over 4 epochs, in which, each epoch will consist of a training phase and a validation phase. In each epoch, we will train our samples in batches of size 32 each.

After our model is trained, we will test the model's performance using the Matthew's Correlation Coefficient (MCC) metric. The MCC score ranges between -1 (worst score) to +1 (best score). There are 2 reasons for using MCC as our test metric instead of simply measuring the accuracy of the model's predictions – the NLP community generally uses this metric while testing the CoLA dataset, and apart from this, it is observed that our test set contains a majority of positive sentences (i.e.- grammatically correct sentences). Hence, our test set is imbalanced. MCC accounts for this imbalance and gives us an accurate metric that represents the true performance of our model.

The full implementation for the fine-tuned model can be accessed from the given link - https://colab.research.google.com/drive/1x0y2e9FI2gqyNTTm6BRuJPt8YYNiToV-?usp=sharing

VI. RESULTS

The training results of our fine-tuned model are shown in Fig. 10 as follows –

epoch	Training Loss	Valid. Loss	Valid. Accur.	Training Time	Validation Time
1	0.383018	0.476657	0.797840	0:01:15	0:00:03
2	0.266396	0.474082	0.820988	0:01:18	0:00:03
3	0.163721	0.525671	0.831404	0:01:21	0:00:03
4	0.121782	0.591360	0.818287	0:01:21	0:00:03

Fig. 11 Training statistics for our fine-tuned BERT model

As shown in Fig. 11, it is observed that with each epoch, our training loss decreases, which means our model is being trained well.

| ISSN: 2395-7639 | www.ijmrsetm.com | Impact Factor: 7.802 | A Monthly Double-Blind Peer Reviewed Journal |

| Volume 12, Issue 1, January 2025 |

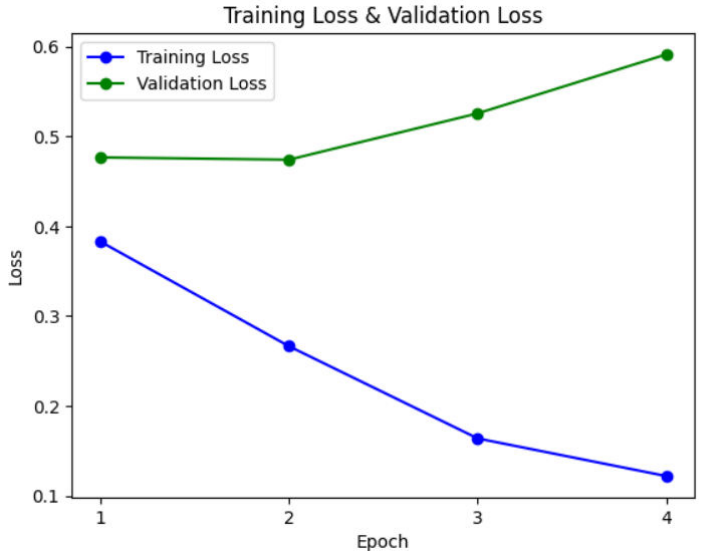


Fig. 12 Learning curve for training loss

On evaluating our model's performance on the test set, we observe the MCC score (ranging between -1 to +1) for each batch in Fig. 12 as follows –

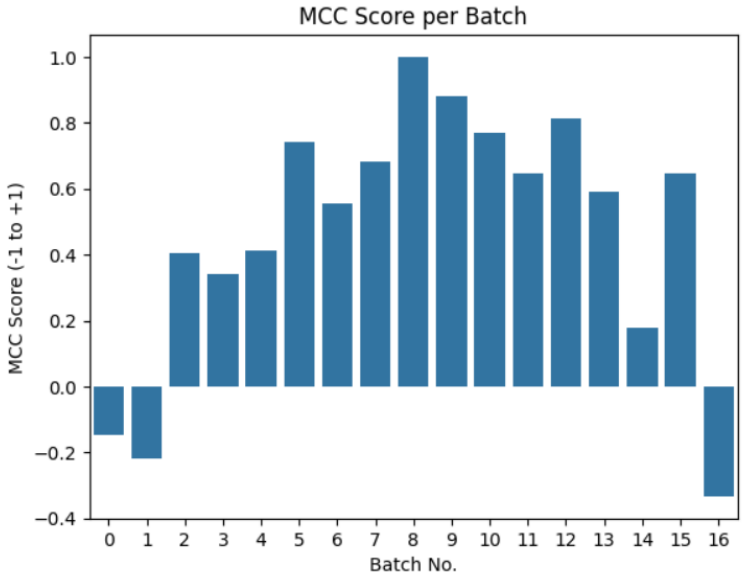


Fig. 13 MCC Score across batches of test set

The final (aggregate) MCC score for our test set is 0.545.

VII. CONCLUSION

Fig. 13 represents the GLUE benchmark scores, and provides a comparative analysis of the performance of the BERT Base model against several other language models, tested against nine different datasets, including the CoLA dataset.

| ISSN: 2395-7639 | www.ijmrsetm.com | Impact Factor: 7.802 | A Monthly Double-Blind Peer Reviewed Journal |

| Volume 12, Issue 1, January 2025 |

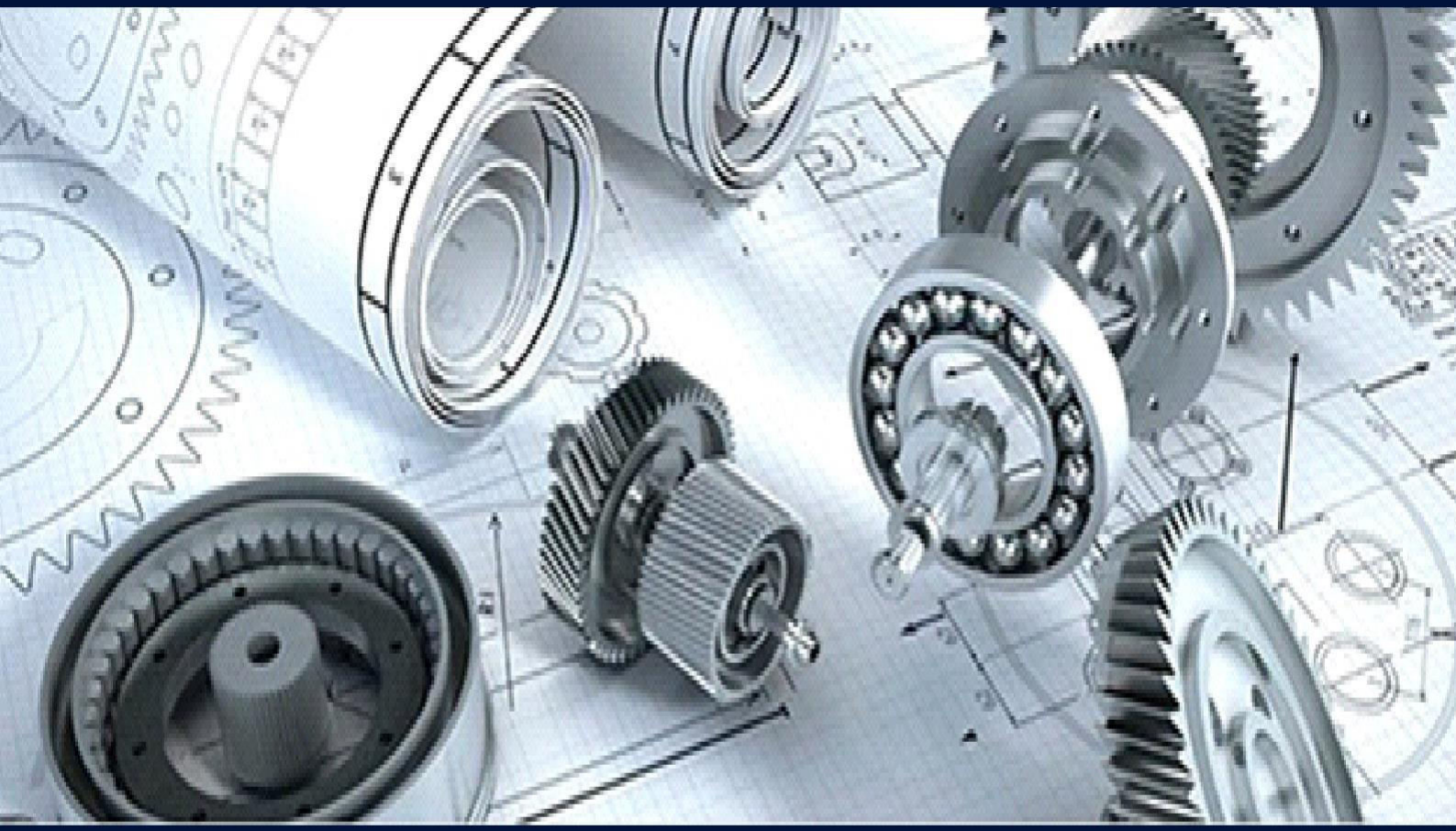
System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$BERT_{BASE}$	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**

Fig. 14 GLUE Benchmark scores for 9 various natural language tasks

On evaluating our model's performance against the GLUE benchmark, we observe that the GLUE score for the BERT Base model, tested on the CoLA dataset, is 52.1, while the MCC score of our model is 0.545 (or 54.5%). This proves that by fine-tuning a pre-trained BERT model, we can quickly and efficiently create high-quality models with minimal training time and effort, regardless of the natural language task to be performed.

REFERENCES

[1] Maxime, "What is a Transformer?," Inside Machine learning, 2019.
[2] J. Alammar, "The Illustrated Transformer [Blog post]," jalammar.github.io, 2018.
[3] B. Muller, "BERT 101 - State Of The Art NLP Model Explained [Blog Post]," huggingface.co, 2022.
[4] J. Devlin, M. Chang, K. Lee, K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in arXiv preprint arXiv:1810.04805, 2018.
[5] J. Alammar, "The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)," jalammar.github.io, 2018.
[6] A. Warstadt, A. Singh, S. Bowman, "Neural Network Acceptability Judgments," in Transactions of the Association for Computational Linguistics 7, 2019.



INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH

IN SCIENCE, ENGINEERING, TECHNOLOGY AND MANAGEMENT

+91 99405 72462 +91 63819 07438 ijmrsetm@gmail.com