Government Polytechnic, Pune-16

(An Autonomous Institute of Government of Maharashtra)

A

Seminar Report

On

"Transformer Model"

SUBMITTED BY:

Deshpande Sujay Hemantkumar
2106044

Under the Guidance of

Smt. S. S. Ingavale

DEPARTMENT OF COMPUTER ENGINEERING

(Academic Year: 2023-24)

Government Polytechnic, Pune-16

(An Autonomous Institute of Government of Maharashtra)

DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

This is to certify that **Mr. DESHPANDE SUJAY HEMANTKUMAR** with Enrollment Number **2106044** of Third Year Diploma in Computer Engineering has successfully completed the seminar titled "**Transformer Models**" as part of his diploma curriculum in academic year 2023-2024.

Seminar Guide	**H.O.D**	**Principal**
(Smt.S. S. Ingavale)	(Mrs. J. R. Hange)	(Dr. V. S. Bandal)

ACKNOWLEDGEMENT

I would like to sincerely acknowledge and express my heartfelt gratitude to all those who played a pivotal role in the successful completion of this report. First and foremost, I extend my deepest thanks to the Honorable Principal of the College, Dr. Vitthal Bandal, for his unwavering support throughout this journey. I am also immensely grateful to the Honorable Head of the Computer Department, Mrs. J.R. Hange, for her continuous encouragement and for providing us with the essential facilities necessary for the completion of this seminar. A special mention goes to Smt. S.S. Ingavale, whose guidance and mentorship were invaluable in steering this seminar toward success. Her constant encouragement and motivation sustained my efforts at every stage, and I am profoundly thankful for her unwavering support.

Furthermore, I would like to express my gratitude to my friends and family members who have been a pillar of strength throughout this endeavor. Their guidance and support in helping me choose the right path for this seminar were indispensable.

ABSTRACT

This seminar report is submitted in partial fulfillment of the requirements for the Diploma in Computer Engineering, following the guidelines of Government Polytechnic, Pune. The subject matter explored in this report revolves around the revolutionary Transformer Model, a groundbreaking development in the field of natural language processing and machine learning. The Transformer Model has garnered significant attention in recent years due to its transformative impact on various applications, including machine translation, text generation, and sentiment analysis. It represents a paradigm shift in the way we understand and process language, diverging from traditional sequence-based models.

This report aims to delve into the core concepts and mechanisms underpinning the Transformer Model's functioning. It will provide an in-depth exploration of the attention mechanisms, multi-head self-attention, and positional encoding, all of which are pivotal components of this model. Moreover, it will highlight the significance of pre-trained language models like BERT and GPT in the broader context of the Transformer architecture. Explore the forefront of natural language processing and machine learning with an in-depth analysis of the groundbreaking Transformer Model, a paradigm-shifting development that has revolutionized language understanding and reshaped the landscape of AI applications

INDEX

Chapter 1
Natural Language Processing

1.1 Natural Language Processing:

Natural Language Processing (NLP) is a branch of artificial intelligence (AI) that focuses on the interaction between computers and human language. It encompasses various tasks and techniques to enable machines to understand, interpret, and generate human language. With advancements in deep learning and the availability of large datasets, NLP has made significant strides in achieving human-level performance in tasks like text generation, question answering, and language translation, opening new possibilities for human-computer interaction and automated language processing.

In the realm of Natural Language Processing (NLP), several architectural approaches have emerged to tackle the challenges of understanding and processing human language. One prominent architecture is the Transformer model, which revolutionized NLP with its attention mechanism, allowing models to focus on different parts of the input text when making predictions. The Transformer architecture forms the foundation for various state-of-the-art models such as BERT (Bidirectional Encoder Representations from Transformers), GPT (Generative Pretrained Transformer), and T5 (Text-to-Text Transfer Transformer). These models have achieved remarkable results in tasks like language understanding, translation, and text generation, often through pretraining on massive text corpora and fine-tuning for specific tasks.

1.2 Different NLP Architectures:

Natural Language Processing (NLP) encompasses a wide range of architectures and models designed to process and understand human language. Here are some of the various NLP architecture types:

1) **Recurrent Neural Networks (RNNs):**
   Recurrent Neural Networks (RNNs) are a class of neural networks designed for processing sequential data. They have been widely used in natural language processing (NLP) and other sequential data tasks. Unlike feedforward neural networks, RNNs have loops that allow them to maintain a hidden state, which can capture information about

previous time steps in a sequence. This capability makes RNNs suitable for tasks like text generation, machine translation, sentiment analysis, and speech recognition.

2) **Long Short-Term Memory (LSTM):**
   LSTM, a specialized RNN architecture, was created to overcome traditional RNN limitations in capturing long dependencies in sequential data like language. It employs memory cells to retain information over extended sequences, making it great for tasks with long-term dependencies. Unlike standard RNNs, LSTMs use gating mechanisms to control information flow into and out of memory cells, learning and retaining relevant information while avoiding the vanishing gradient issue of earlier RNNs.
3) **Convolutional Neural Networks (CNNs):**
   Convolutional Neural Networks (CNNs) are deep learning models designed primarily for grid-like data like images and, to some extent, sequences such as text. They learn hierarchical patterns and features directly from data by using convolutional layers with learnable filters (kernels) to capture local patterns and build complex representations layer by layer. Max-pooling or average-pooling layers are used to reduce dimensions while retaining vital information. Thanks to their efficient capture of spatial hierarchies, CNNs excel in computer vision tasks like image classification, object detection, and image segmentation.

Like the above architectures, around one hundred and more transformer models are present in NLP.

Though these architectures were useful in several working assets, common disadvantages of traditional NLP models include their inability to effectively capture long-range dependencies in data, high sensitivity to the input data format, and limited scalability for handling large datasets and complex tasks. The need for the Transformer model arose due to its unique architecture, which employs self-attention mechanisms to capture contextual information across input sequences, making it highly effective at processing sequential data like text and achieving state-of-the-art results in various natural language processing tasks. This innovation addressed the limitations of previous models and revolutionized the field by enabling the efficient processing of long sequences while maintaining scalability and flexibility, leading to significant advancements in tasks such as machine translation and text generation.

Chapter 2
Transformer Models

2.1 Introduction to Transformer Model:

The Transformer model, named for its transformative architecture, represents a pivotal advancement in deep learning. Unlike traditional models that rely on sequential processing, the Transformer employs self-attention mechanisms to analyze and understand sequential data, which makes it highly efficient and parallelizable. This innovation has had a profound impact on various fields, particularly natural language processing. Transformer models are powerful tools for natural language processing (NLP) tasks, such as text generation, translation, summarization, and sentiment analysis.

The heart of the Transformer lies in its self-attention mechanism, which allows it to weigh the significance of different elements within a sequence, regardless of their position. This ability to capture long-range dependencies between words or tokens in a sentence has greatly improved its performance on tasks like language translation, text summarization, and question-answering.

It can accommodate sequences of varying lengths, making it versatile for processing text of different sizes. Its parallel processing capabilities also enable faster training times on modern hardware, contributing to its widespread adoption in the machine-learning community.

The influence of the Transformer extends beyond natural language processing. It has served as the foundational architecture for a range of state-of-the-art models such as BERT, GPT, and T5. Its adaptability and effectiveness have prompted its application in diverse domains, including computer vision, audio processing, and recommendation systems.

Born out of the need for more effective and context-aware language modeling, this marvel of modern technology has reshaped the landscape of machine learning and left an indelible mark on various domains, from natural language processing to computer vision. Its ability to capture intricate patterns and relationships within data, coupled with its unparalleled scalability, has made it the gold standard in cutting-edge AI research and applications. In this exploration of the Transformer model, we delve into its fascinating architecture, its impact on the world of AI, and its limitless potential to transform the way we interact with technology and information.

Chapter 3
Architecture of Transformer Models

The architecture of the Transformer model stands as “an architectural marvel in the realm of artificial intelligence, a symphony of innovation that has redefined the very essence of machine learning.” Its inception, marked by a revolutionary departure from traditional sequence-to-sequence models, has ignited a transformative journey into the heart of deep learning. This groundbreaking structure, with its intricate web of self-attention mechanisms and multi-layered neural networks, has not only shattered performance benchmarks but has also unlocked the gates to unprecedented possibilities in natural language processing, machine translation, and a myriad of other domains. In this intricate tapestry of interconnected layers and attention patterns, we uncover the blueprint of the Transformer—a testament to the relentless pursuit of excellence in the world of AI architecture.

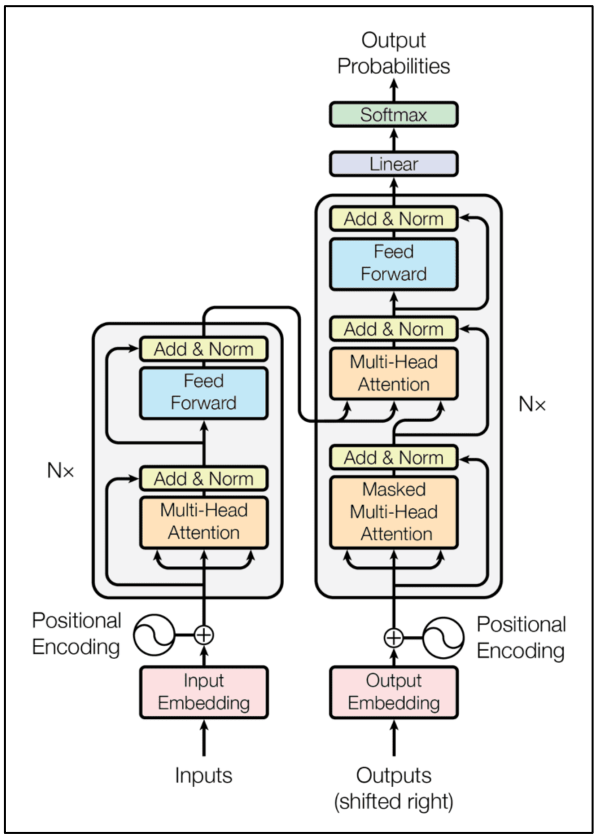


Fig1. Architecture of Transformer Model

At its essence, the Transformer model redefines AI architecture and revolutionizes language understanding.

3.1 Components of Transformer Model:

The Transformer Model consists of different parts for the working model where each part has significant importance according to the working. The components are as follows:

1) Encoders
2) Decoders
3) Self-Attention
4) Positional Encoding
5) SoftMax

Most competitive neural sequence transduction models have an encoder-decoder structure. Here, the encoder maps an input sequence of symbol representations $(x_1, \ldots, x_n)$ to a sequence of continuous representations $z=(z_1,\ldots,z_n)$. Given, the decoder then generates an output sequence $(y_1, \ldots, y_m)$ of symbols one element at a time. The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, shown in the left and right realms of Fig. 1, respectively.

3.2 Encoder and Decoder Stacks:

Encoder:

The encoder is composed of a stack of $N = 6$ identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-fully connected feed-forward network. We employ a residual connection around each of the two sub-layers, followed by layer normalization. That is, the output of each sub-layer is

$$LayerNorm(x + Sublayer(x))$$

where *Sublayer(x)* is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, and the embedding layers, produce outputs of dimension $dmodel = 512$.

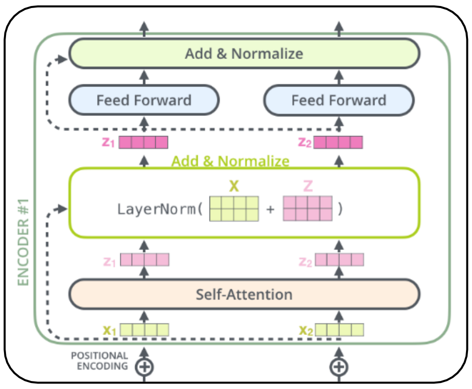


Fig.2 Encoder layer

Decoder:

The decoder is also composed of a stack of $N = 6$ identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Like the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with the fact that the output embeddings are offset by one position, ensures that the predictions for position *i* can depend only on the known outputs at positions less than *i*.

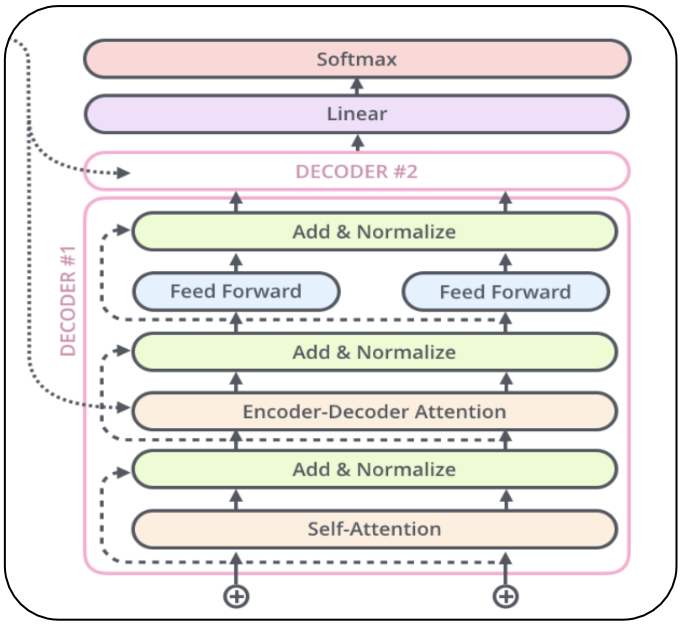


Fig.3 Decoder Layer

The decoder stack outputs a vector of floats. How do we turn that into a word? That is the job of the final Linear layer which is followed by a Softmax Layer.

The Linear layer is a simple fully connected neural network that projects the vector produced by the stack of decoders, into a much, much larger vector called a logits vector.

3.3 Self Attention:

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

3.3.1 Scaled Dot-Product Attention:

The input for the encoder consists of queries and keys of dimension d_k, and values of dimension d_v. We compute the dot products of the query with all keys, divide each $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values.

In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix *Q*. The keys and values are also packed together into matrices *K* and *V*. We compute the matrix of outputs as:

$$Attention(Q, K, V) = softmax\left(\frac{Qk^T}{\sqrt{d_k}}\right)V$$

While for small values of d_k the two mechanisms perform similarly, additive attention outperforms dot product attention without scaling for larger values of d_k. We suspect that for large values of d_k, the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients. To counteract this effect, we scale the dot products by $\frac{1}{\sqrt{d_k}}$.

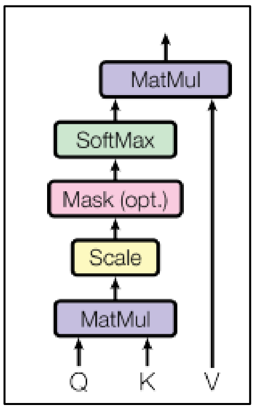
3.3.2 Multi-Head Attention:

Instead of performing a single attention function with d_{model}-dimensional keys, values, and queries, we found it beneficial to linearly project the queries, keys, and values h times with different, learned linear projections to d_k, d_k, and d_v dimensions, respectively. On each of these projected versions of queries, keys, and values we then perform the attention function in parallel, yielding dv-dimensional output values. These are concatenated and once again projected, resulting in the final values. Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.

$$\mathrm{MultiHead}(Q, K, L) = \mathrm{Concat}(\mathrm{head}_1, \dots, \mathrm{head}_\mathrm{h})W^0$$

$$Where, head_1 = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

That concludes the self-attention calculation. The resulting vector is one we can send along to the feed-forward neural network. In the actual implementation, however, this calculation is done in matrix form for faster processing. So, let us look at that now that we've seen the intuition of the calculation on the word level.



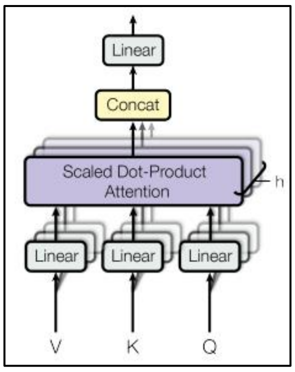


Fig.4.1 Scaled Dot-Product **Fig.4.2 Multi-Head Attention**

3.3.3 Applications of Attention in Transformer Model:

The Transformer uses multi-head attention in three different ways:

- In "Encoder-Decoder attention" layers, the queries come from the previous decoder layer, and the memory keys and values come from the output of the encoder. This allows every position in the decoder to attend to all positions in the input sequence.
- The encoder contains self-attention layers. In a self-attention layer all of the keys, values and queries come from the same place, in this case, the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder.
- Similarly, self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including that position. We need to prevent leftward information flow in the decoder to preserve the auto-regressive property. We implement this inside of scaled dot-product attention by masking out (setting to $-\infty$) all values in the input of the softmax which correspond to illegal connections. See Figure 2.

3.4 Position-wise Feed-Forward Networks:

In addition to attention sub-layers, each of the layers in our encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a *ReLU* activation in between.

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)\, W_2 + b_2$$

While the linear transformations are the same across different positions, they use different parameters from layer to layer. Another way of describing this is as two convolutions with kernel size 1. The dimensionality of input and output is $d_{model} = 512$, and the inner-layer has dimensionality $d_{ff} = 2048$.

3.5 Embeddings and Softmax:

Similarly, to other sequence transduction models, we use learned embeddings to convert the input tokens and output tokens to vectors of dimension d_{model}. We also use the usual learned linear transformation and softmax function to convert the decoder output to predicted next-token probabilities. In our model, we share the same weight matrix between the two embedding layers and the pre-softmax linear transformation. In the embedding layers, we multiply those weights by $\sqrt{d_{model}}$.

3.6 Positional Encoding:

Since our model contains no recurrence and no convolution, for the model to make use of the order of the sequence, we must inject some information about the relative or absolute position of the tokens in the sequence.

The model contains "positional encodings" to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encodings have the same dimension d_{model} as the embeddings so that the two can be summed. There are many choices of positional encodings, learned and fixed.

‘Positional encoding’ in the transformer model allows it to capture and differentiate the relative positions of words in a sequence, enabling it to process sequences with spatial and temporal information effectively, without relying solely on attention mechanisms.

In this section, we have delved into the transformative architecture of the Transformer model, showcasing its self-attention mechanisms that have reshaped natural language processing and machine learning. With unparalleled parallelization, scalability, and performance, the Transformer serves as the bedrock for state-of-the-art models. Its exceptional ability to handle long-range dependencies and adapt to diverse tasks is a profound achievement in deep learning.

Chapter 4
Training and Results of Transformer Model

Train the Transformer Model:

During training, an untrained model would go through the exact same forward pass. However, since the users are training it on a labeled training dataset, they can compare its output with the actual correct output. The following steps are involved in training a Transformer model:

1. *Data Preparation:*

   The first step is to prepare the data for training. You need to have a dataset that contains pairs of source and target sequences. For example, if you are training a machine translation model, the source sequence could be a sentence in English, and the target sequence could be the corresponding sentence in French. You need to preprocess the data by tokenizing the sequences and converting them into numerical representations. You can use libraries like NLTK or spaCy for tokenization.

2. *Model Architecture:*

   The next step is to define the architecture of the Transformer model. You need to decide the number of layers, the number of attention heads, the size of the hidden layers, and other hyperparameters. You can use a pre-trained Transformer model, such as BERT or GPT-3, as a starting point and fine-tune it for your specific task. Alternatively, you can build a Transformer model from scratch using frameworks like TensorFlow or PyTorch.

3. *Training with given datavalues:*

   The next step is to train the Transformer model on the prepared data. You need to define the loss function, the optimizer, and the learning rate schedule. The loss function is typically the cross-entropy loss, which measures the difference between the predicted and actual output sequences. The optimizer could be *Adam* or *SGD*, and the learning rate schedule could be a fixed schedule or a dynamic schedule like the Transformer's "warmup and decay" schedule.

   You also need to define the batch size and the number of epochs for training. It is recommended to use a large batch size and train for several epochs to achieve good results.

This information is just a brief overview of the training in the transformer mode. Even many open-source APIs are currently available in the market for training the dataset like Hugging Face Transformer API, OpenAI GPT API, Bert as a Service, etc. and these are best in use.

Results in Transformer Models:

The results after training a Transformer model can vary significantly depending on the specific task, dataset, model architecture, hyperparameters, and training process. However, here are some common types of results you might expect when training a Transformer model:

1. Text Generation: For tasks like language modeling or text generation, a trained Transformer model can produce coherent and contextually relevant text. The quality of the generated text often depends on the amount of training data and the model's architecture. It can be used to generate human-like text, write creative content, or even answer questions in natural language.
2. Machine Translation: In the context of machine translation, a trained Transformer model can accurately translate text from one language to another. The quality of translations improves with larger training datasets and better model architectures
3. Named Entity Recognition (NER): Transformer models can be trained for NER tasks to identify and classify entities (such as names of people, organizations, and locations) in text. The result is a model that can extract and categorize entities from unstructured text.
4. Sentiment Analysis: After training on labeled data, a Transformer model can classify text sentiments (e.g., positive, negative, neutral). This is useful for sentiment analysis of social media content, reviews, and customer feedback.
5. Summarization: Transformer-based models can be trained for text summarization, where they generate concise and coherent summaries of longer texts, making them useful for automated content summarization.

The quality and performance of the trained model will depend on the amount and quality of training data, the model architecture (e.g., BERT, GPT, RoBERTa), and the fine-tuning process. Evaluation metrics specific to the task, such as accuracy, F1 score, BLEU score, or ROUGE score, are used to assess the model's performance.

Chapter 5
Different Transformer Models

We can see Transformer models outperforming all the so-called 'state-of-the-art' models. Transformers have also been proven to improve long-term dependencies and can be used to leverage the technique called Transfer learning, which is very useful when we have less amount of data.

GPT:

GPT stands for Generative Pre-trained Transformer. GPT is essentially the decoder stack in the Transformer. The GPT was developed by OpenAI in 2018 The decoder stack in Transformers was responsible for predicting the next word of the output sequence. Given the previous words, the decoder is trained to predict the current word in the sequence. The previous stage input along with the current predicted word forms the input for predicting the next word. This type of model is called the Autoregressive model. So, we can understand the GPT model as a *next-word prediction* model, which can be useful in sentence completion, natural language generation, and many more.

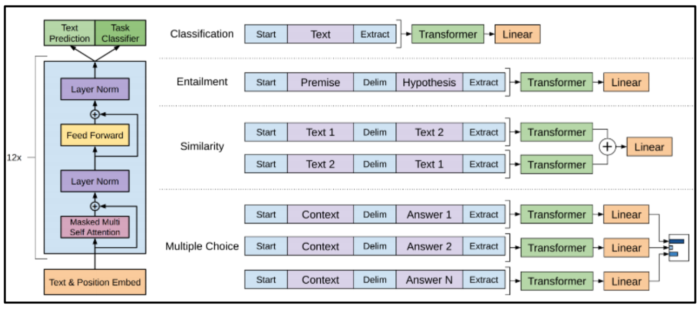


Fig 5. Architecture of GPT

GPT models power generative AI applications such as:

- ChatGPT: A model that interacts conversationally. ChatGPT can answer follow-up questions, admit its mistakes, challenge incorrect premises, and reject inappropriate requests.
- iCIMS Copilot: A GPT-powered copilot for recruiters.

Currently, many GPTs are developed around the world like ChatGPT, Davinci, Grammerly, etc. which are moving to the deep in the transformer models.

BERT:

BERT stands for Bidirectional Encoder Representations from Transformers. A BERT model is trained using the masked language model (MLM) and next sentence prediction (NSP) simultaneously. Google researchers proposed BERT in 2018. It is based on the Transformers deep learning model, which connects every output element to every input element. The weightings between them are dynamically calculated based on their connection.

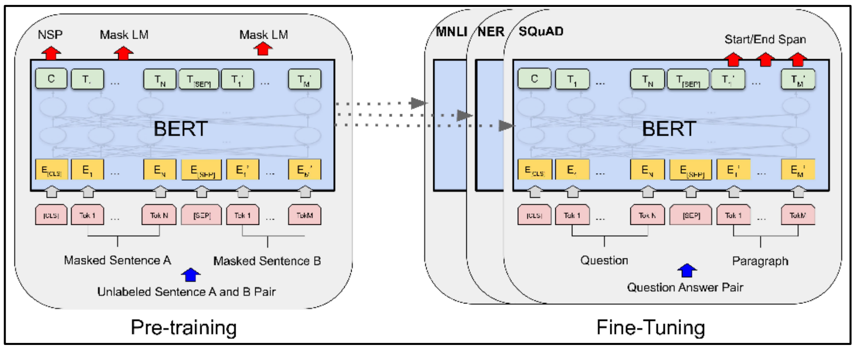


Fig 6. BERT Architecture

In the market, many BERTs are present which gives significant results in the Transformer Model.

T5:

T5, Text-To-Text Transfer Transformer, is an encoder-decoder model pre-trained on a multi-task mixture of unsupervised and supervised tasks for which each task is converted into a text-to-text format. T5 works well on a variety of tasks out-of-the-box by prepending a different prefix to the input corresponding to each task.

The T5 transformer model works by using the same standard encoder-decoder structure as standard transformer models. It consists of 12-pair blocks of encoder-decoder. Each block contains self-attention, a feed-forward network, and optional encoder-decoder attention.

To understand that, we need to first understand two unique features of the T5 model:

1. Input/Output Representation: Text-to-Text Framework
2. Training dataset: C4 dataset

Similarly, around twenty or more efficient instances of the Transformer model are available which are prior built upon their structural variation. alBARTa, RoBART, BART, etc. uses similar architecture for developing transformer model.

Conclusion:

In conclusion, the Transformer model represents a monumental leap in the field of deep learning and natural language processing. Its innovative self-attention mechanism and parallelization capabilities have revolutionized the way we approach a wide range of tasks, from machine translation to text generation and beyond. As we have explored its architecture and applications throughout this seminar report, it is evident that the Transformer's impact is profound and far-reaching. Its adaptability, scalability, and exceptional performance make it a foundational tool for researchers and practitioners in the ever-evolving landscape of artificial intelligence. With ongoing advancements and continued research, the Transformer model promises to remain at the forefront of AI innovation, pushing the boundaries of what is possible in language understanding and generation. As we embark on this transformative journey, understanding the Transformer's intricacies and harnessing its full potential is not just a pursuit of knowledge but a commitment to shaping the future of AI-powered applications.

References:

1. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). “Attention is All You Need.”
2. Jay Alammar (2020) “The Illustrated Transformer”
3. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Sutskever, I. (2020). “Language Models are Few-Shot Learners.”
4. Ronan Collobert (2015) “Natural Language Processing”.
5. Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova (2016) “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”.
6. Hugging Face Transformers Documentation: https://huggingface.co/transformers/