TRANSFORMERS

V Sriram

March 2025

1 Introduction

Transformer is a deep learning architecture which relies entirely on the attention mechanism and feed-forward neural networks to model sequential datasets, for instance, language translation.Its ability to immensely parallelize training of large amounts of data and ability to get scaled without degradation made them the state-of-the-art models in the domain of natural language processing.

Transformers eliminated two main disadvantages of the RNNS:

- RNNs depended on the previous output for the next output.Hence, training takes a long time.
- RNNs lose context when fed large amounts of data.

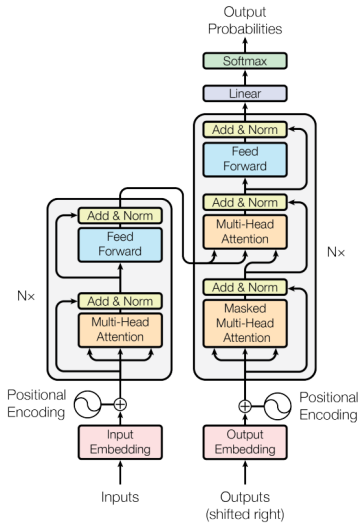
2 Model Architecture

2.1 Encoder And Decoders

On a very high level, the encoders are a stack of 6 identical layers stacked on top of each other. The output of one of the encoder layer is fed into the next layer. Each Encoder layer consists of 2 sub-layers. The first is the multi-head attention layer, the other is the feed-forward neural network layer. After each layer there are also residual connections and layer normalization for numeric stability. Decoder layers also the follow the same structure except for the fact that each of the layer consists of three sub-layers namely the masked-mutihead attention, Encoder-decoder attention layer and then the feed-forward neural network layers.It also contains residual connection and layer norm after each sub-layer.

2.2 Attention

Attention in transformers are implemented in the form of Query, key and value. First, the Query and key are calculated for each embedding with a learnable Query and Key matrix and these values are then taken dot product and a soft max function is taken to find a relative correlation within the embeddings.Then



the values are calculated by another learnable value matrix which are then taken weighted sum (with weights as the result of the dot product between the queries and the keys) which are then added to the embedding to create a new embedding.

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

where the d_k represents the dimension of the embedding. Sometimes to prevent the model from getting information about the future inputs, some of the query-key matrix values are made $-\infty$.This is called masking.

2.3 Multi-Head Attention

In order to get a more nuanced understanding of the data, the weight matrices of the query, key and the value are divided into equal number of columns, which are then computed separately, thus capturing more complex and multi-way attentions in the data. All of this process can be done simultaneously and thus can be parallelized.

2.4 Feed-Forward Neural Networks

The feedforward neural networks consists of normal deep neural network with ReLU as activation function.

2.5 Embeddings

Embeddings are derived from the word dictionary using a learnable matrix. Also learnable matrix and softmax function is used to calculate the next word output.

2.6 Positional Encoding

Since transformers are parallelized, there is a very high chance of losing the sequential relationships between the inputs, and thus need to be provided separately in some other form. For this we use positional encoding. For each position p, the encoding is a vector of same dimension as the embedding and will be added to it. For each value j in the embedding the encoding is given by: If $j = 2i$

$$PE = sin(\frac{p}{10000^{\frac{2i}{d_{model}}}})$$

else if $j = 2i + 1$ it is given by:

$$PE = cos(\frac{p}{10000^{\frac{2i}{d_{model}}}})$$

This encoding is linear in the sense that there exists a linear transformation from i to $i + k$ for any i.

3 Training

Training is done through cross entropy as loss for language translation and uses backpropagation to update learnable parameters. The learning rate is changed through a form regularization given by:

$$lrate = d^{-0.5} \cdot min(step_num^{-0.5}, step_num \cdot warmup_steps^{-0.5})$$

Residual connections are given to avoid vanishing gradient problem and to make the model remember the flow of information. Label smoothing is done to avoid overfitting and to improve accuracy.

4 Final Flow

The inputs are given in the encoder and decoder. During the training the decoder gets one step shifter input with the first be a $< START >$ token. The encoder output is added to the decoder.The final output is just probabilities of the next most probable value for machine translation. During inference the model is given inputs to the encoder and only given a $< START >$ signal to the decoder. The output of the decoder is again fed back as input to the decoder for the next step. It continues till the decoder predicts an end of sentence $< EOS >$ token.