Transformers - Handout Paper

Siva Indraneel Dhulipala
3046805
Siva.I.Dhulipala@stud.leuphana.de

1 Introduction

Transformer Models were introduced to essentially compensate for the flaws of Recurrent Neural Network models which could not handle extremely long input sequences and faced the vanishing/exploding gradients problem due to the nature of Backpropagation Through Time (BPTT).

In the original "Attention Is All You Need" paper [3], the authors mention how the Transformers Model can not only handle longer sequences, but it can do so with high efficiency thanks to parallelization which enables the input to be fed in at once (a.k.a one time-step).

2 Architecture

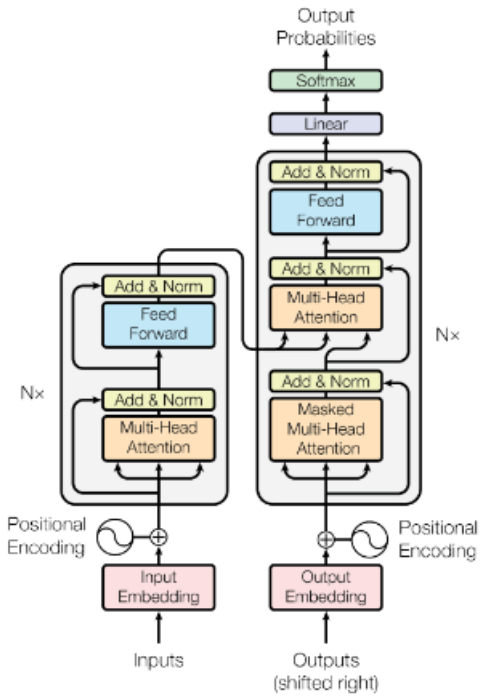

Figure 1: Transformer Architecture [3]

We shall briefly discuss the purpose of each important component in the architecture diagram.

2.1 Encoder

Input

The input sequence is split into tokens and converted into Embeddings. These embeddings could be pre-trained (GloVe etc.) or could be trained with the model. The latter is what is done in LLMs.

Positional Encodings are then added to each input token to note the relative position of each token with respect to the other tokens. This is because the tokens are passed in parallel.

Multi-Head Attention

Query, Key, and Value (Q, K, V) vectors/matrices are prepared based on the input sequence. The terms potentially refer to their database counterparts and mean "What we are looking for", "What are the keywords we have", "What do we get after the search" respectively.

These vectors/matrices are then split into h blocks along the feature dimension each of which is then multiplied with parameter matrices W_Q, W_K, W_V to finally create h "heads". These heads are then fed into the following equation.

$$MultiHead(Q,K,V) = Concat(head_1, ..., head_h)W^O$$
$$\text{where } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$
$$\text{and } Attention(Q,K,V) = softmax(QK^T/\sqrt{d_k})V$$

By doing multi-head attention, more context can be included in the attention mechanism as individual heads are considered for the entire process [Figure 2]. The end result is an attention matrix. Note that since the Q, K, V are from the same input, this is called Self-Attention.

Add and Norm

After the attention matrix is calculated the original input matrix is added back to it via a residual/skip connection to avoid vanishing gradients. Following this, Layer Normalization is done across the feature dimension for each word/token to avoid issues in backpropagation that come with the cost function being skewed.

Feed Forward Network

The purpose of this layer is supposed to be to promote more interaction between the heads calculated in the Multi-Head Attention block. The Network includes a Linear Layer with ReLU activation and Dropout and another Linear Layer that brings back the input to its original dimension.

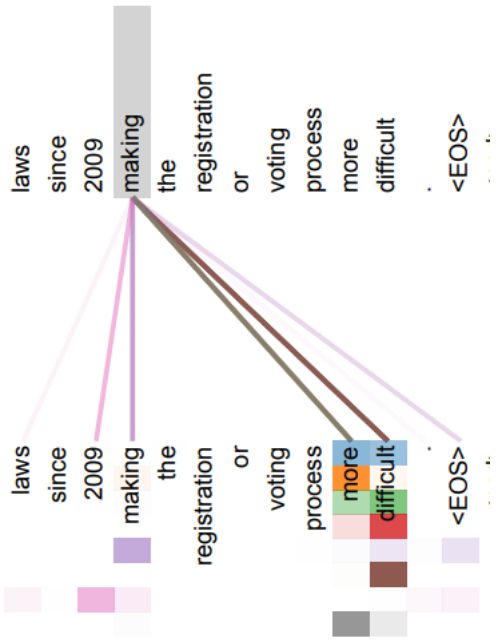

Figure 2: Attention Visualization [3]

Output
The output of the Encoder block can be thought of as a representation of the input in the latent space. Note that the Encoder block is repeated N times to achieve good contextual representation vector.

2.2 Decoder

Input
The Decoder part of the Transformer is always "autoregressive" in nature. This means that it generates the next output token based on the previous output token. To achieve this, the inputs are fed after being shifted right during training. Say the output needs to be "Bist du fertig", the input would be <SOS> Bist du fertig" where <SOS> is the start of sequence token. During inference, the input would simply take the tokens generated so far as in the input to generate the next token. Of course, the Positional Encoding are also added as in the case of the Encoder

Masked Multi-Head Attention
Since the training in the decoder focuses on its autoregressive properties, we feed the entire expected output sequence at once. This could mean that in the attention mechanism, a token could pay attention to another token in the future which is not desired. Thus we add a mask to the input Q, K, V matrices to avoid this very scenario. The mask matrix itself would be an upper triangular matrix filled with negative infinities as that would translate to zeroes in the softmax, nulling the effect of future tokens for each given token in the attention matrix.

Multi-Head Attention - Cross Attention
After the add and norm over the attention matrix, which is similar to the one discussed in the Encoder, the resulting matrix is fed as the Value matrix into the multi-head attention block. The Query, Key matrices in this case come from the Decoder and the attention mechanism is executed. Since the Q, K, V matrices are not from the same input sequence, the attention mechanism performed here is referred to as Cross-Attention. The intuitive end goal of this is to utilize the context vectors from the Encoder to find what we need in the final output sequence.

Beyond the multi-head attention done here, we can see that the components are similar to those seen in the Encoder. Their functionality as well is exactly the same. The Decoder Layer is also stacked *N* times.

Final Output
The output of the Decoder stack cannot be directly used to find out the next token in the sequence. In order to achieve this, the Linear layer and the subsequent Softmax layer map the output to a matrix that shows, for each word/token in the target language vocab space, what is its probability of being the next token.

From here, either the token with the maximum probability is picked directly, or a "Beam Search" can be done where the top few possible tokens are picked and then further steps are taken.

3 Transformer Variations

Finally, we shall briefly look at 2 popular models whose architecture is based on the original Transformer architecture and see what they achieved overall.

3.1 GPT

The Generative Pre-Trained Transformer model brought in the idea of Pre-Training where the model itself is trained in an unsupervised manner with the help of a large number of unlabelled examples. Its architecture replicates mainly the Decoder stack of the original Transformer [2].

3.2 BERT

The Bidirectional Encoder Representation from Transformers took a step ahead from GPT and considered the importance of bidirectional training since context drawn from a single direction may not be as effective as its counterpart [1]. As its name states, BERT's architecture mainly draws from the Encoder stack of the original Transformer and also focuses on the Pre-Training introduced in GPT. In fact, all LLMs in the current day focus on Pre-Training and Fine-Tuning for the required downward stream tasks.

References

[1] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. cite arxiv:1810.04805Comment: 13 pages. 2018. URL: http://arxiv.org/abs/1810.04805.

[2] Alec Radford et al. "Improving language understanding by generative pre-training". In: (2018).

[3] Ashish Vaswani et al. "Attention is all you need". In: *Advances in Neural Information Processing Systems*. 2017, pp. 5998–6008.