1 Attention is all you need

1.1 Introduction

Before the introduction of the transformer, the dominant sequence transduction models were based on the complex recurrent or convolutional neural networks that include an encoder and a decoder. But this paper introduced a new network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train.

Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, allowing the modeling of dependencies without regard to their distance in the input or output sequences. In all but a few cases, however, such attention mechanisms are used in conjunction with a recurrent network.

1.2 Transformer Model Architecture

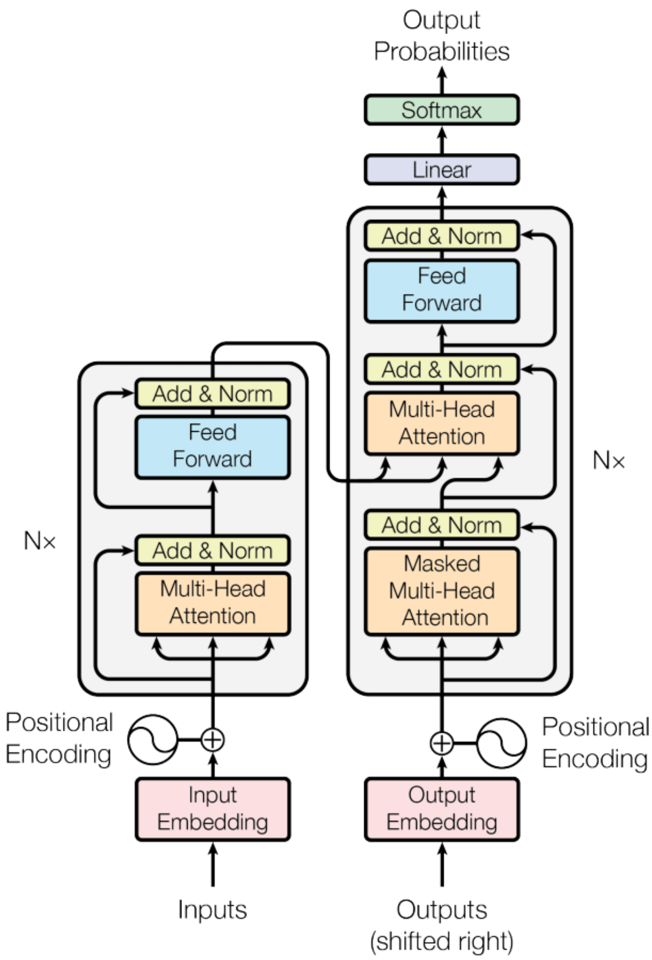

Figure 1: The Transformer - model architecture

1.2.1 Encoder and Decoder Stacks

The encoder is composed of a stack of N = 6 identical layers. Each layer has two sublayers: multi-head self-attention mechanism, and the simple position wise fully connected feed-forward network. Residual connection around each of two sub layers, followed by a layer normalization is employed in the encoder.

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. Similar to the encoder, residual connections around each of the sub layers, followed by layer normalization is employed. Self attention sub-layer in the decoder stack is modified to prevent the positions from attending to subsequent positions.

1.2.2 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.

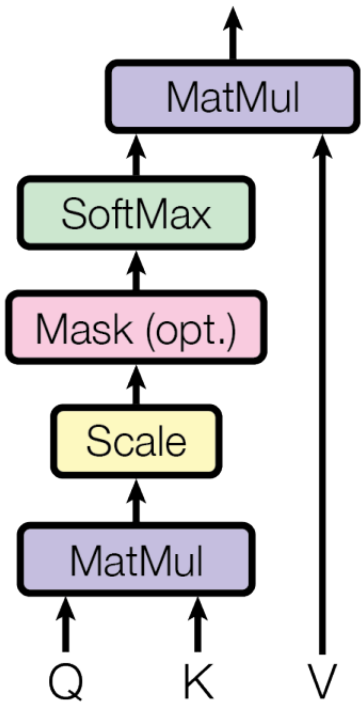

Figure 2: Scaled Dot-Product Attention

In scaled dot-product attention, the input 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 by d_k and apply a softmax function to obtain the weights on the values.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

In 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 d_v-dimensional output values. These are concatenated and once again projected, resulting in the final values, as depicted in figure 3.

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.

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_2)W^o \tag{2}$$
$$\text{where } head_i = \text{Attention}(QW_i^Q,\ KW_i^K,\ VW_i^V)$$

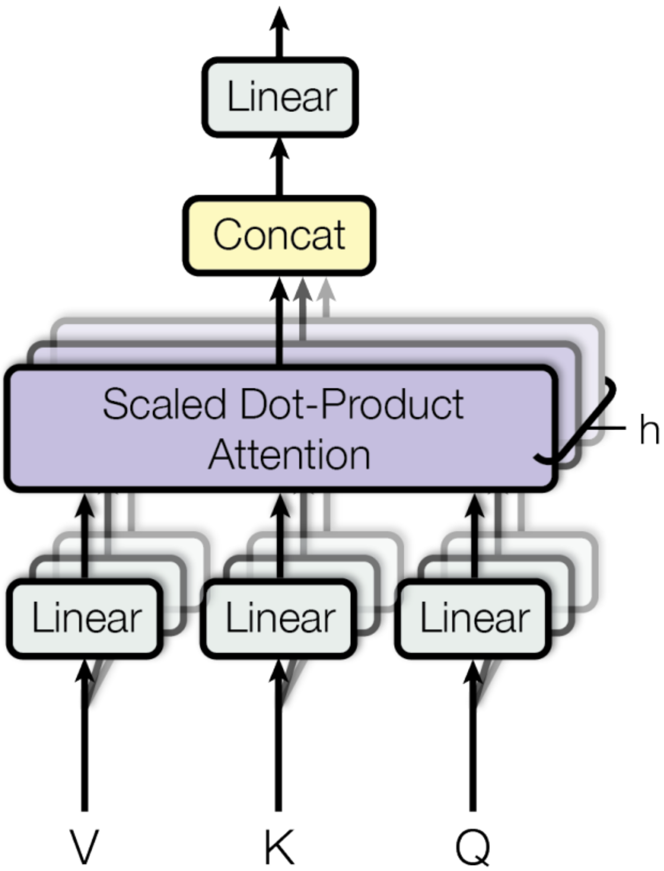

Figure 3: Multi-Head Attention consists of attention layer running in paralle

1.2.3 Position wise Feed Forward Neural Network

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.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{3}$$

1.2.4 Embedding and Softmax

Similarly to other sequence transduction models, Learned embeddings are used to convert the input tokens and output tokens of dimension d_{model}. Usual learned linear transformation and softmax function are used to convert the decoder output to predict next-token probabilities.

1.2.5 Positional Encoding

Since the model contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, some information about the relative or absolute position of the tokens in the sequence must be injected. To this end, "positional encodings" are added to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encodings have the same dimension model as the embeddings, so that the two can be summed. Sine and cosine functions of different frequencies are used for positional encoding as follow

$$PE_{(pos,2i)} = sin(\frac{pos}{10000^{2i/d_{model}}}) \tag{4}$$

$$PE_{(pos,2i+1)} = cos(\frac{pos}{10000^{2i/d_{model}}}) \tag{5}$$

where *pos* is the position and i is the dimension. That is, each dimension of the positinal encoding corresponds to a sinusoid.

1.3 Conclusion and Results

In the machine translation task, the transformer model outperformed all previous state of the art models which have used the RNNs, GRUs and LSTMs. To evaluate if the Transformer can generalize to other tasks, experiments on English constituency parsing were performed and despite the lack of task-specific tuning, the transformer model performed surprisingly well, yielding better results than all previously reported models with the exception of the Recurrent Neural Network Grammar.

In this work, the Transformer, the first sequence transduction model based entirely on attention, replacing the recurrent layers most commonly used in encoder-decoder architectures with multi-headed self-attention is presented.

Till today, various transformer models have been developed which have performed better on all kinds of natural language processing tasks like language modeling, text classifications, questions answering, machine translation, sentence similarity, summarization, etc.