Transformer

Seungcheol Oh

I. Encoder-Decoder Transformer

One main draw back of conventional RNN architectures such as LSTM and GRU is that it suffers from capturing long term dependencies between the naturally sequential data. Further, it does not have a functionality to relate local time stamp information to the current time stamp data. With the multi-headed attention mechanism, Transformer architecture is able to capture the relationship between different parts of the input sequence, including long-range dependencies and local timestamp information. In Fig. 1, the transformer architecture is illustrated. In this section, we will elaborate on how each block is contributing to the whole transformer architecture.

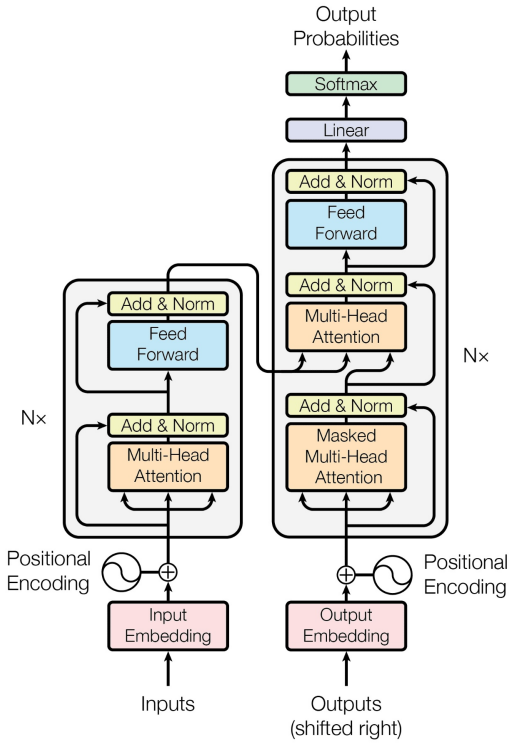

Fig. 1. Transformer Architecture

A. Input Embedding & Positional Encoding

As mentioned in sec. **??**, each token of the input gets mapped to an embedding vector which has dimension of d. These embedding vectors get stacked as row vectors in a matrix and we call this matrix an input embedding matrix. When LSTM is used for machine translation, it loops through the tokens one by one; therefore, it has a sense of ordering. However, for transformers, a chunk of tokens becomes an input. This loses the ordering between the tokens. For this reason, positional encoding is added.

B. Self Attention

As explained, each token of the input is represented by an embedding vector. The objective of self-attention then, is to update these embedding vectors such that it conveys the information about each token's relationship between all the other tokens in the sequence. There are three trainable parameters for the self-attention: query (Q), key (K) and value (V). The self-attention is described by these parameters as

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

where Q, K and V are query key and value matrices, respectively. The procedure to find these matrices is as follow: take the input embedded matrix which is stacked row embedded vectors of each token added with positional encoding, make three

copies of the matrix, matrix multiply it with W^q, W^k and W^v to find Q, K and V, respectively. What each row of Q and K represents is the query and key of the tokens of the input. As shown in (1), dot product QK^T is computed, which provides the metric of how strongly a particular query is related with a particular key. It then gets scaled with $\sqrt{d_k}$ for numerical stability. Finally softmax, described as

$$\alpha(\vec{z}_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \text{ for i} = 1, ..., \text{K}, \tag{2}$$

is applied to each row of QK^T matrix. This keeps each row to sum to 1, as well giving score of how one token is related to another.

For example, consider an input sentence "YOUR CAT IS A LOVELY CAT", which has sequence length 6, and $d_k = 512$ (embedding vector dimension). First, find Q and K, then compute the score matrix. An illustration of the score matrix is shown by Fig. 2. Take a closer look at first row of the matrix. Each element of the first row gives the "score" of how related

	YOUR	CAT	IS	A	LOVELY	CAT	Σ
YOUR	0.268	0.119	0.134	0.148	0.179	0.152	1
CAT	0.124	0.278	0.201	0.128	0.154	0.115	1
IS	0.147	0.132	0.262	0.097	0.218	0.145	1
A	0.210	0.128	0.206	0.212	0.119	0.125	1
LOVELY	0.146	0.158	0.152	0.143	0.227	0.174	1
CAT	0.195	0.114	0.203	0.103	0.157	0.229	1

Fig. 2. QK Score Matrix

each word, "YOUR CAT IS A LOVELY CAT" is to YOUR. All the other row convey the same information but for other input words in the sequence. Now, by (1), we see that the score matrix is matrix multiplied by V to finish generating the attention matrix.

After we find the representation, we divide by number of dimension for numerical stability. Then, softmax is operated on this matrix. This will represent highest number in each column to have highest probability. Then, we multiply this matrix via V, which is the value matrix. Main job of V is to enhance the representation of correlated vectors in the word embedding space this step allows the model to focus on capturing meaningful contextual information from the input sequence, ultimately improving its ability to generate accurate outputs.

C. Multi-Headed Attention

II. Decoder-Only Transformer