Lect 5. Transformer: Encoder and Decoder

Young, Shen

August 11, 2024

1 Encoder

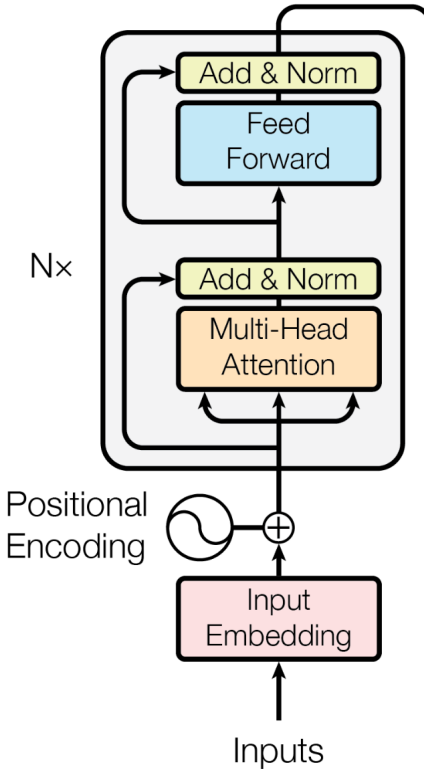

Fig. 1. The Structure of Encoder

The structure of an encoder is easy and clear in this figure. Here are detailed explanation towards the structure:

1. **Input embedding**: Embed each vector of the input sequence.
2. **Positional Encoding**: Add a vector to each of the input vector, signaling the position information of input.
3. **Multihead Attention**: Multihead self-attention, concatenate and transform the results in the end.
4. **Add and Norm**: Add the input vectors to the corresponding output vectors, then do **Layer Normalization**. (This is called **residual**)
5. **Feed Forward**: Each output vector then go through a Fully-Connected Network with **ONE hidden layer**.

6. **Add and Norm**: The output of FC then add with their corresponding inputs. Then do Layer Normalization.

Suppose the input sequence has the length l, then the output of one block will be l vectors. Then we take the output as the input of the block, repeat this process N times.

2 Decoder

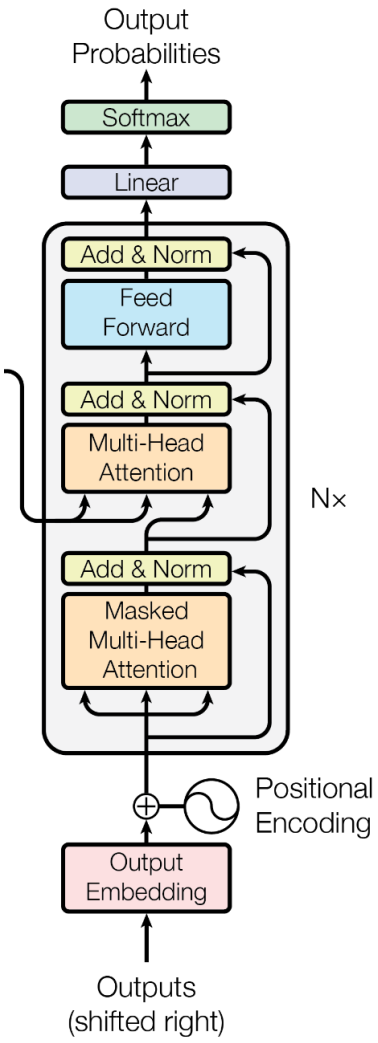

Fig. 2. The Structure of Decoder

There are 2 types of decoder, one being **Auto-Regression(AT)**, the other being **Non-auto-Regression(NAT)**. Here we mainly introduce AT.

We add 2 special vectors to all possible output, one stands for BEGIN and the other stands for END.

2.1 AT

During training, the Decoder directly takes the label sequence, add a BEGIN at the beginning, as input. Then do almost the same thing as Encoder. Here, **Masked Multi-Head Attention** means: each vector of the sequence ONLY calculate the attention to vectors in front of it, ignoring the attention after it. We need to mask, because although when training the decoder

has the whole output sequence, during testing, for AT, decoder only has outputs before current vector (for AT, output is generated one by one, the latter one takes the former output as input, with the first one taking special vector BEGIN as input).

In the Multi-Head Attention step, we focus one the 3 arrows pointing to it.

- The first 2 arrows takes from the output of Encoder. Calculating corresponding **V** and **K**.
- The last arrow takes the output of former layer, calculating **q**

With **q**, **K**, **V**, we then calculate the **Cross-Attention**. The output would be a vector. Then we do residual and normalization, the rest being the same as Encoder.

2.2 NAT

NAT is almost the same as AT, but it takes a sequence of BEGIN as input at a time, while for AT the latter input is the former output. For AT, since the output is generated one by one, it stops generating when the input is END. However, for NAT, the output sequence length is the same as the BEGINs it takes in. To control the output sequence length, there are several solutions.

- Train anther network that takes the Encoder output as input, predicting the length of output sequence length.
- Make sure the BEGINs Decoder takes in is longer than possible outputs, abandon all outputs after END.

2.3 Comparison

AT

More accurate, but time-consuming to train.

NAT

Parallelizable to train, but less accurate.