**CS410 Text Information System Tech Review**

# A Brief Review of the Transformer and

Hyeonjae Cho
hc53@illinois.edu

These days, all cutting-edge models are based on BERT or GPT. Both two models use transformer architecture as the basis. Therefore, it is quite important to understand the transformer thoroughly. Before I start to briefly go over the transformer, let's review the evolution history of machine translation techniques which are based on deep learning. In 1986, RNN first came out and LSTM, Seq2Seq followed. From 1986 to 2017, RNN was mainly used as a base model for NLP tasks. In 2015, the notion of attention was first suggested but the NLP model was still combined with RNN. In 2017, the Transformer model removed RNN or CNN and is solely based on attention architecture. That's the reason why the paper named "Attention is all you need". After the paper had been published, almost every NLP model is based on Transformer architecture.

So why did we come out with attention? The Seq2Seq model, which was the leading model before attention architecture, consists of an encoder and decoder. Here, the encoder creates a fixed-length-sized context vector containing the input sequence's information. This context vector is pushed to the first hidden state of the decoder. A decoder is based on RNNLM(RNN Language Model), so it predicts the next word based on the input. The problem occurs at this point. As the input sequence gets longer and longer, the worse model performs. As the context vector goes through decoder layers, the gradient vanishes, which is the intrinsic flaw of RNN. This is also called a long-term dependency problem. Therefore, attention alleviates the problem by providing weight vectors that contain relevance between words within the input sequence. Attention is used in every decoder step, but the weight keeps changing depending on the given step. As you see in the picture on the left, the previous hidden state in the decoder and all input sequence terms are used and produce weight after applying softmax to make a value range from 0 to 1.

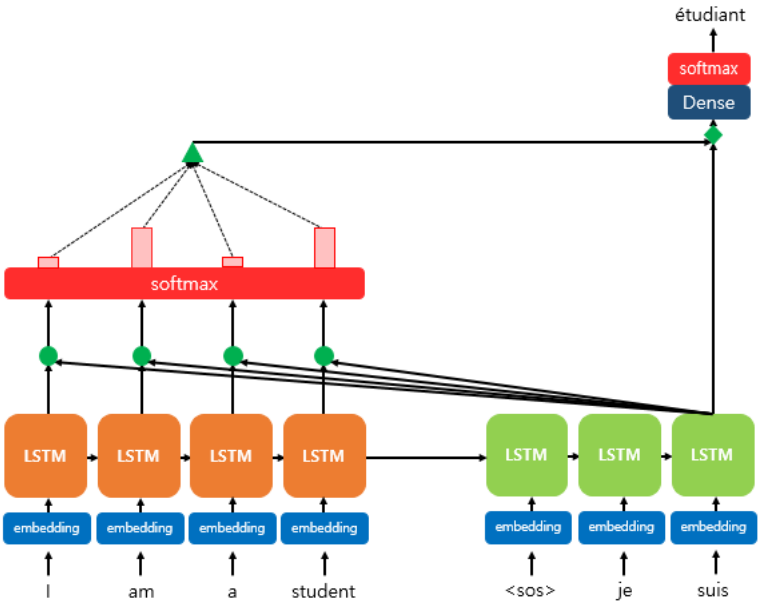


**figure 1 : https://velog.io/@sjinu/개념정리- Attention-Mechanism**

In the encoder part of the Transformer, there are four main components. 1) embedding matrix, 2) positional encoding, 3) multi-head attention, 4) add+norm. In the beginning, the input sequence is

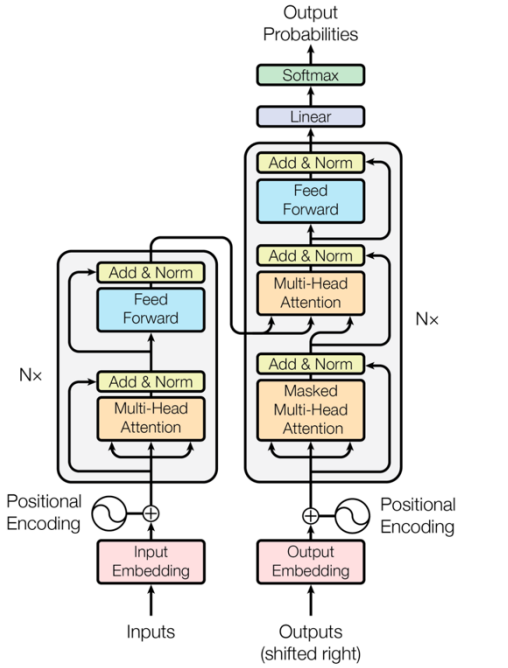


**figure 2 : Attention is all you need**

represented as an embedding matrix and it is combined with positional encoding in an element-wise way. Then encoding has self-attention architecture inside, which computes the attention score representing the relevance degree of tokens within the input sequence. This helps the model train the information of the input sequence. Next, in add + norm part, the residual learning technique is used to enhance the training speed of the model. The combined result of positional encoding and embedding matrix is given as input again here by jumping off the multi-head attention part. Finally, the residual input and the output of multi-head attention are combined and then normalized. This becomes the output of the encoder and is used as an additional input in each layer of the decoder.

The decoder of the transformer has two attention architectures. First is self-attention and it works in the same way as that of the encoder. The self-attention part in the decoder assists the model to train the information of the output sequence it produces. The other attention part requires the output of the encoder and learns which output sequence is most related one to the input sequence. This part is also called encoder-decoder attention. In this way, all the layers in the decoder get the output of the encoder as input.

To summarize, the transformer also uses the encoder-decoder architecture. However, it does not use RNN and rather exploits encoders and decoders for multiple times. This makes the input sequence fed into the transformer model all at once unlike the previous models. Normally, we use the same number of layers in both the encoder and decoder for the transformer architecture.