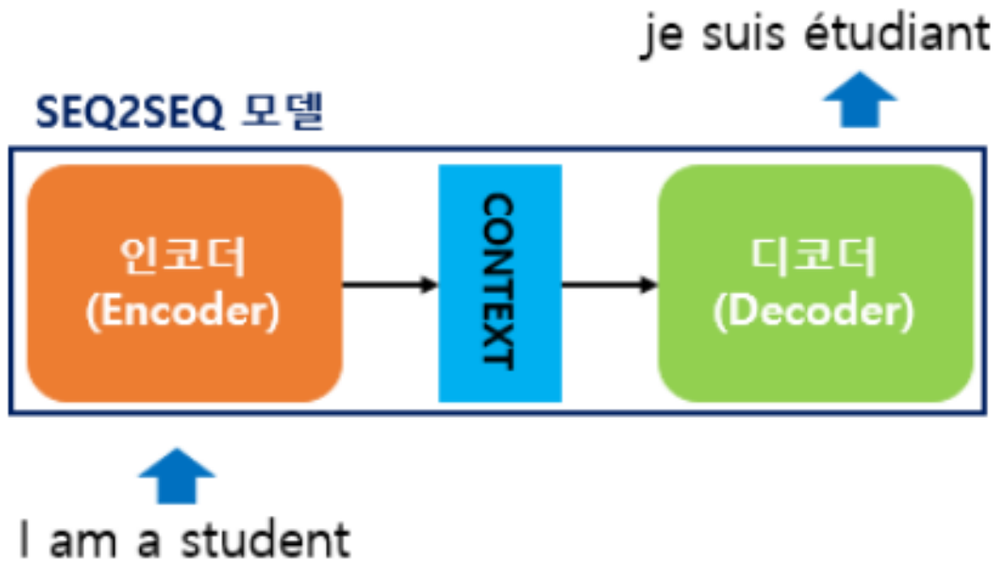
연세대학교
YONSEI UNIVERSITY

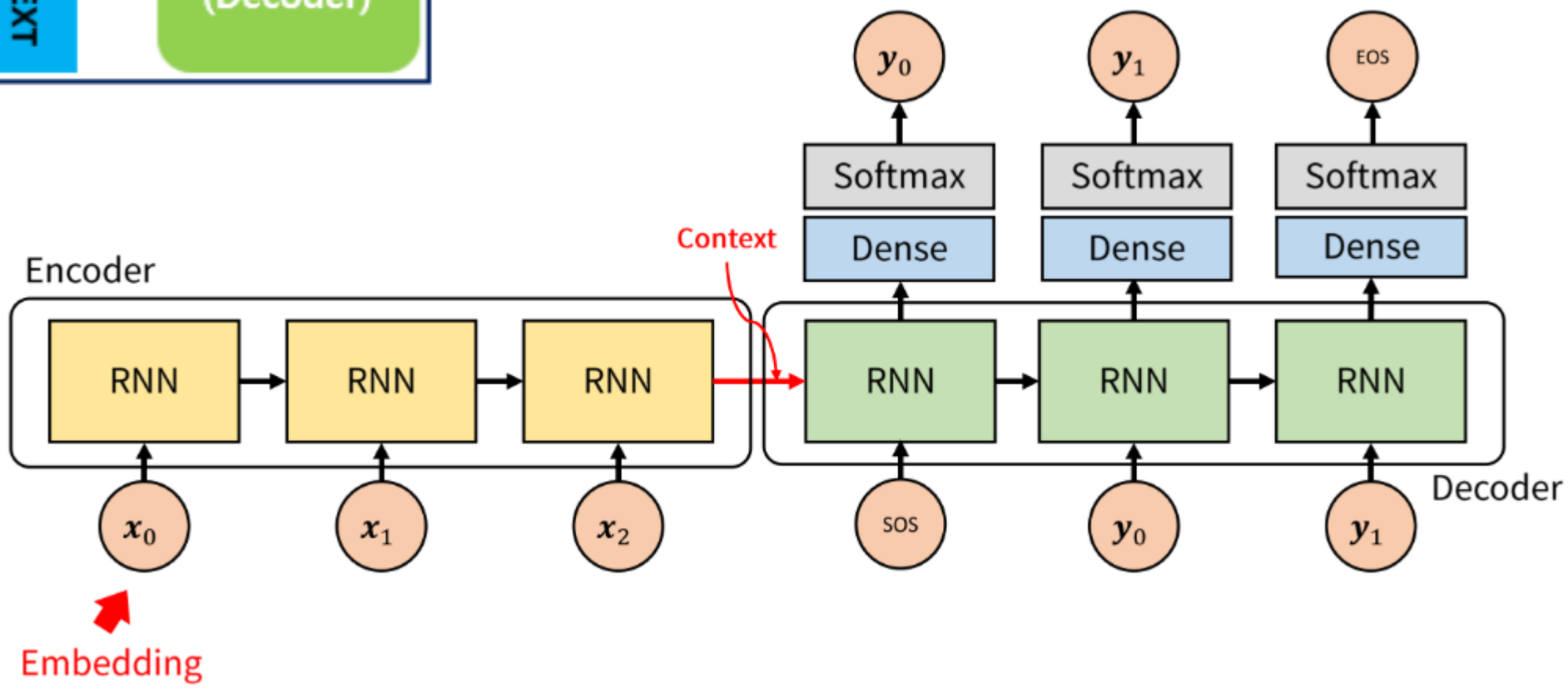
Transformer

TA. Bogyeong Suh

Encoder-Decoder using RNN

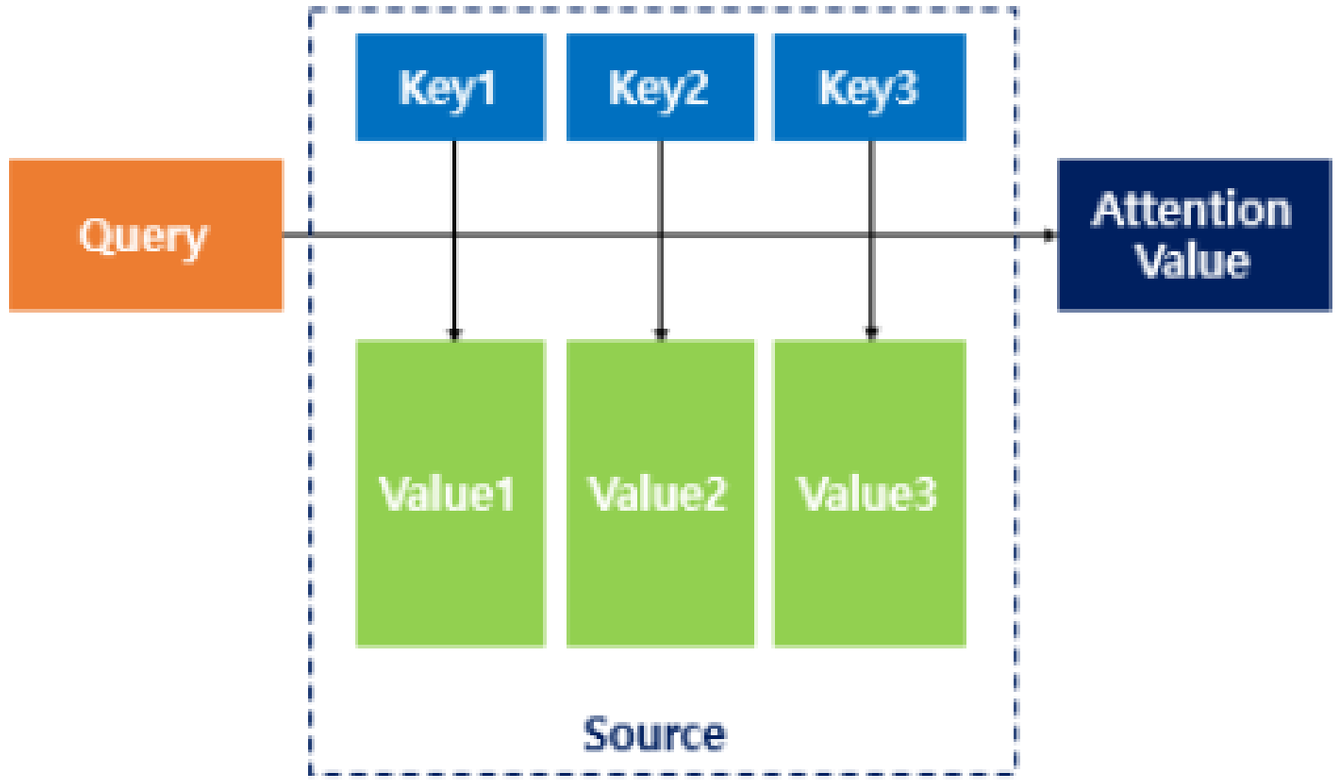
seq2seq





- With the input data consisting of sequences of words, the encoder makes a context vector and the decoder transforms this context vector into sequences of words
- e.g. Chatbot, Machine translation, text summarization, Speech to Text(STT)
- Limitation: information loss due to a fixed length of context vector, vanishing gradients

Attention mechanism



$$Attention(Q, K, V) = Attention\ Value$$

- At every time step the decoder predicts an output word, the entire input sentence at the encoder is being referred to once again with different amount of attention per each word
- It allows the model to focus on different parts of the input sequence when making predictions, by assigning different weights to each input element
- Attention function obtains the similarity with all ‘Keys’ for a given ‘Query’, and this similarity is reflected in each ‘Values’, mapped with the ‘Keys’. It returns the sum of all ‘Values’ as attention value

Types of attention

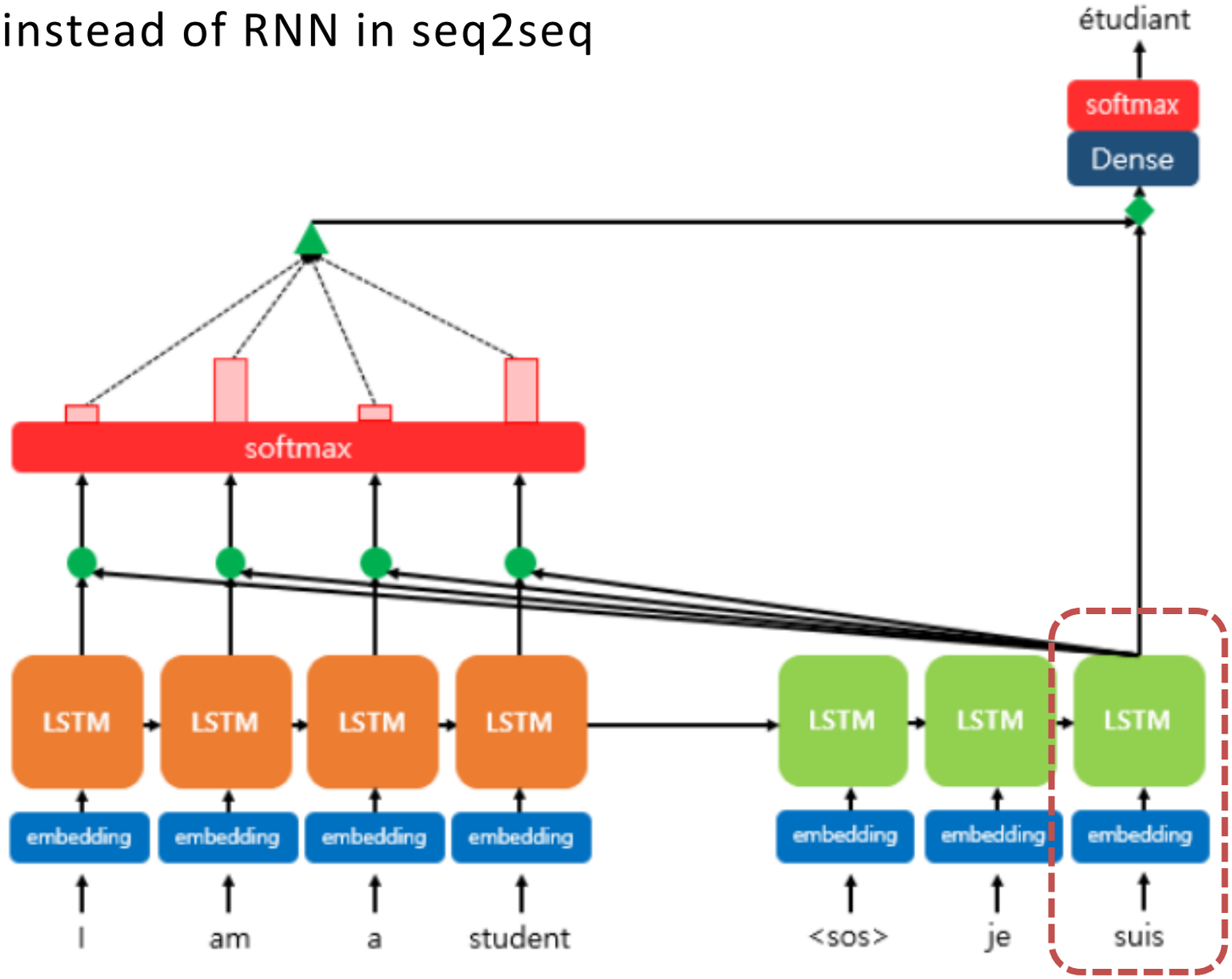
Attention mechanism	Score function	Defined by
dot	$score(s_t, h_i) = s_t^T h_i$	Luong et al. (2015)
$scaled\ dot$	$score(s_t, h_i) = \frac{s_t^T h_i}{\sqrt{n}}$	Vaswani et al. (2017)
$general$	$score(s_t, h_i) = s_t^T W_a h_i$	Luong et al. (2015)
$concat$	$score(s_t, h_i) = W_a^T \tanh(W_b[s_t; h_i])$ $score(s_t, h_i) = W_a^T \tanh(W_b s_t + W_c h_i)$	Bahdanau et al. (2015)
$location - base$	$\alpha_t = softmax(W_a s_t)$	Luong et al. (2015)

- There are different types of attention depending on how the attention score is calculated

Attention mechanism

Dot-Product Attention

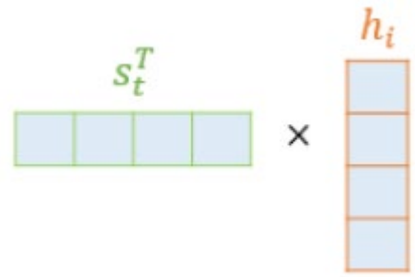
*We will use LSTM instead of RNN in seq2seq



- When the decoder tries to refer to the entire sentence of the encoder, **the degree of which each input word helps the decoder's prediction is quantified and measured**, and then sent to the decoder
- Softmax at encoder: outputs a numerical value of how helpful each of the words is in predicting the output word

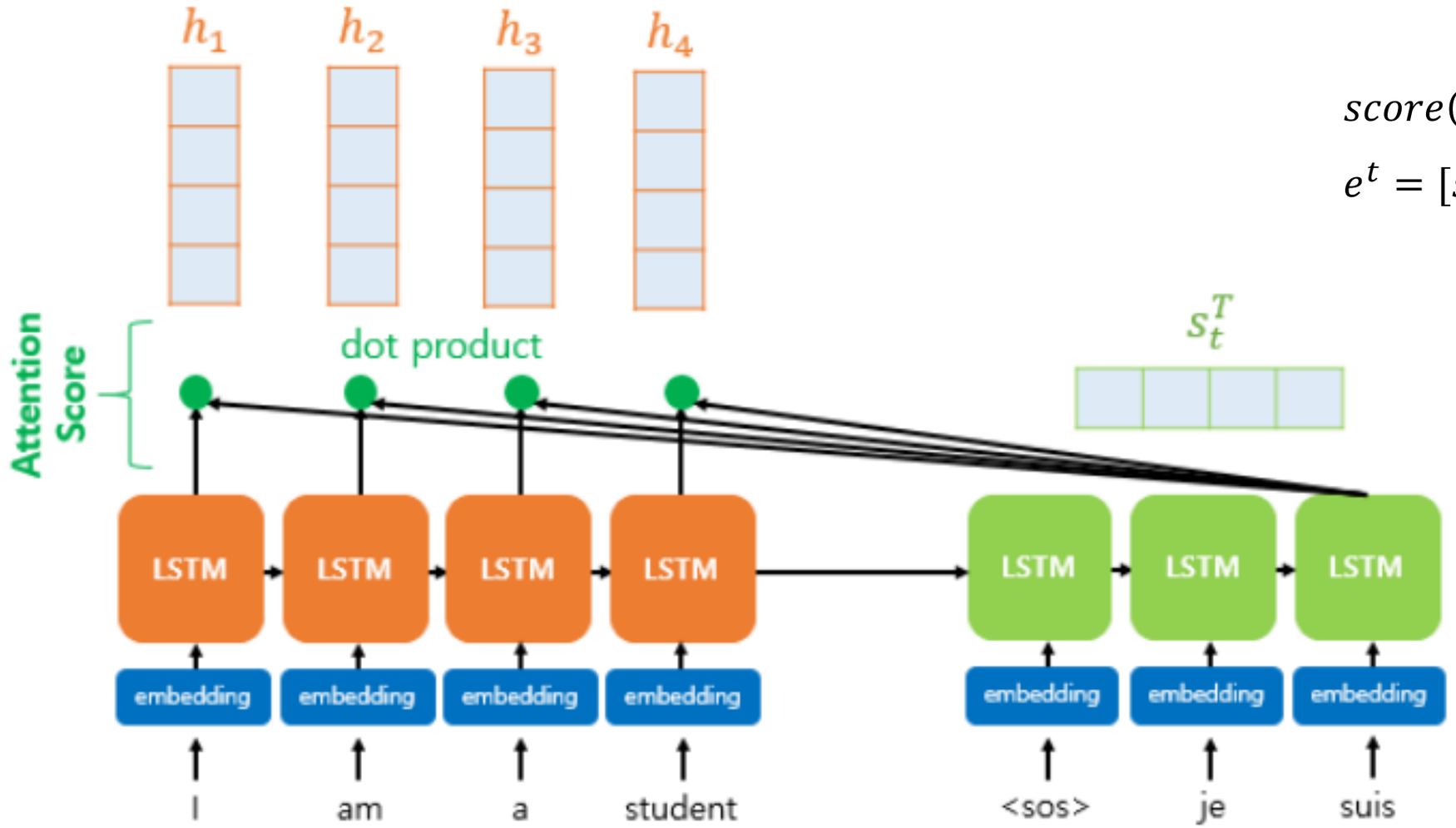
Dot-Product Attention

1. Obtain attention score a_t



$$score(s_t, h_i) = s_t^T h_i$$

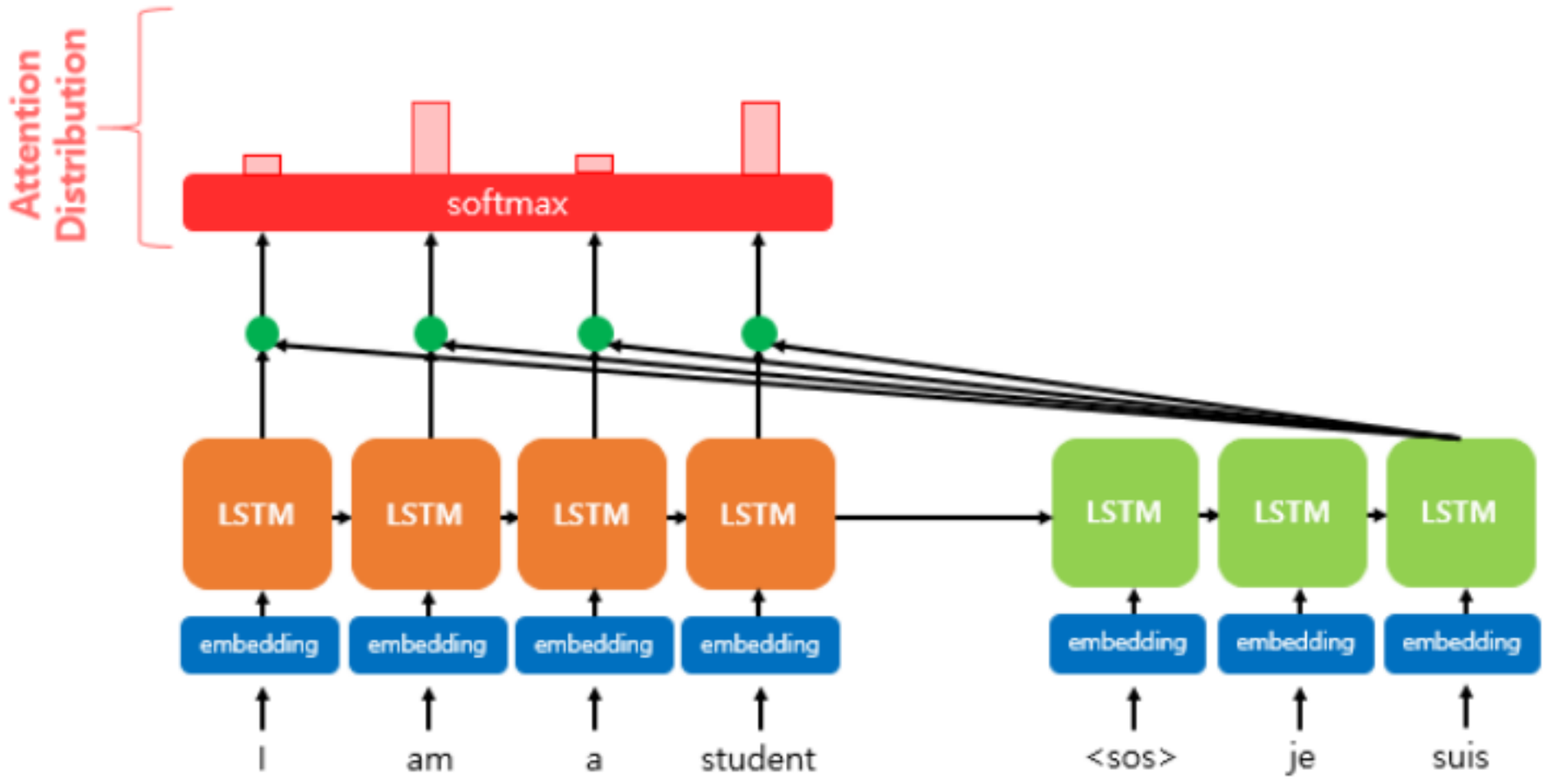
$$e^t = [s_t^T h_1, \dots, s_t^T h_N]$$



- Attention score determines how similar all hidden states of the encoder are to the current hidden state s_t of the decoder

Dot-Product Attention

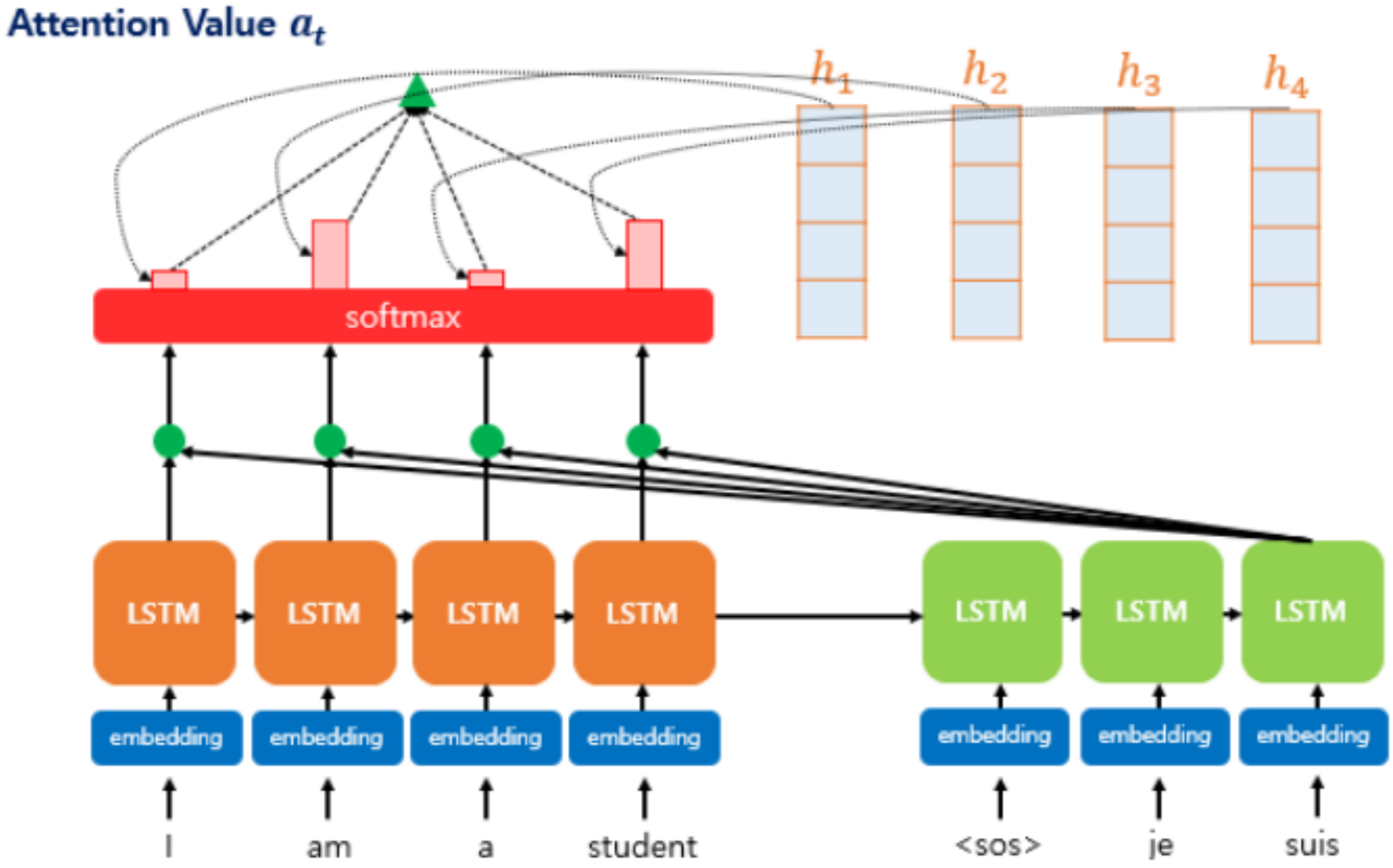
2. Obtain attention distribution via softmax function



- By applying the softmax function to e^t, obtain an attention distribution (probability distribution) where the sum of all values equals 1, and each value referring to attention weight
- Attention distribution $\alpha^t = softmax(e^t)$

Dot-Product Attention

3. Obtain attention value using weighted sum of each encoder's attention weight and hidden states

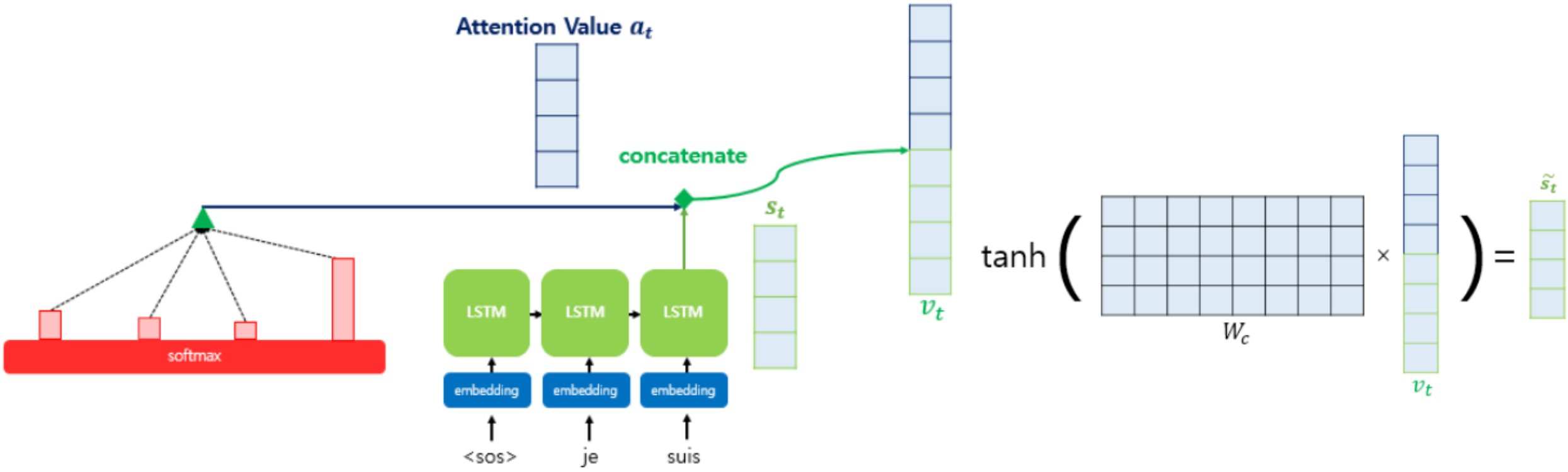


$$a_t = \sum_{i=1}^{N} \alpha_i^t h_i$$

- Attention value a_t is calculated by using weighted sum of each encoder's attention weight and hidden states
- Attention value also refers to context vector

Dot-Product Attention

4. Concatenate attention value and hidden state of decoder, and calculate $\widetilde{s_t}$ to use it as a input for output layer

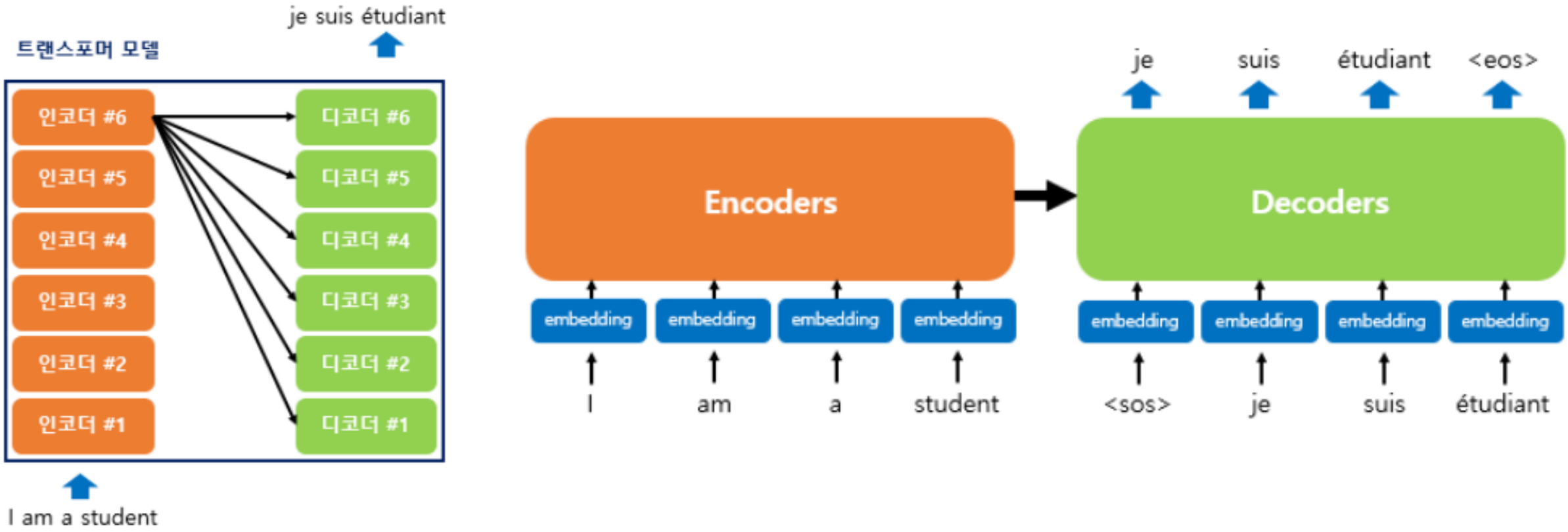


- Concatenate a_t and s_t to a single vector v_t
- With tanh activation function, obtain $\widetilde{s_t}$ and use it as an input for deriving $\hat{y}$

$$\widetilde{s_t} = \tanh(\mathbf{W}_c[a_t; s_t] + b_c)$$

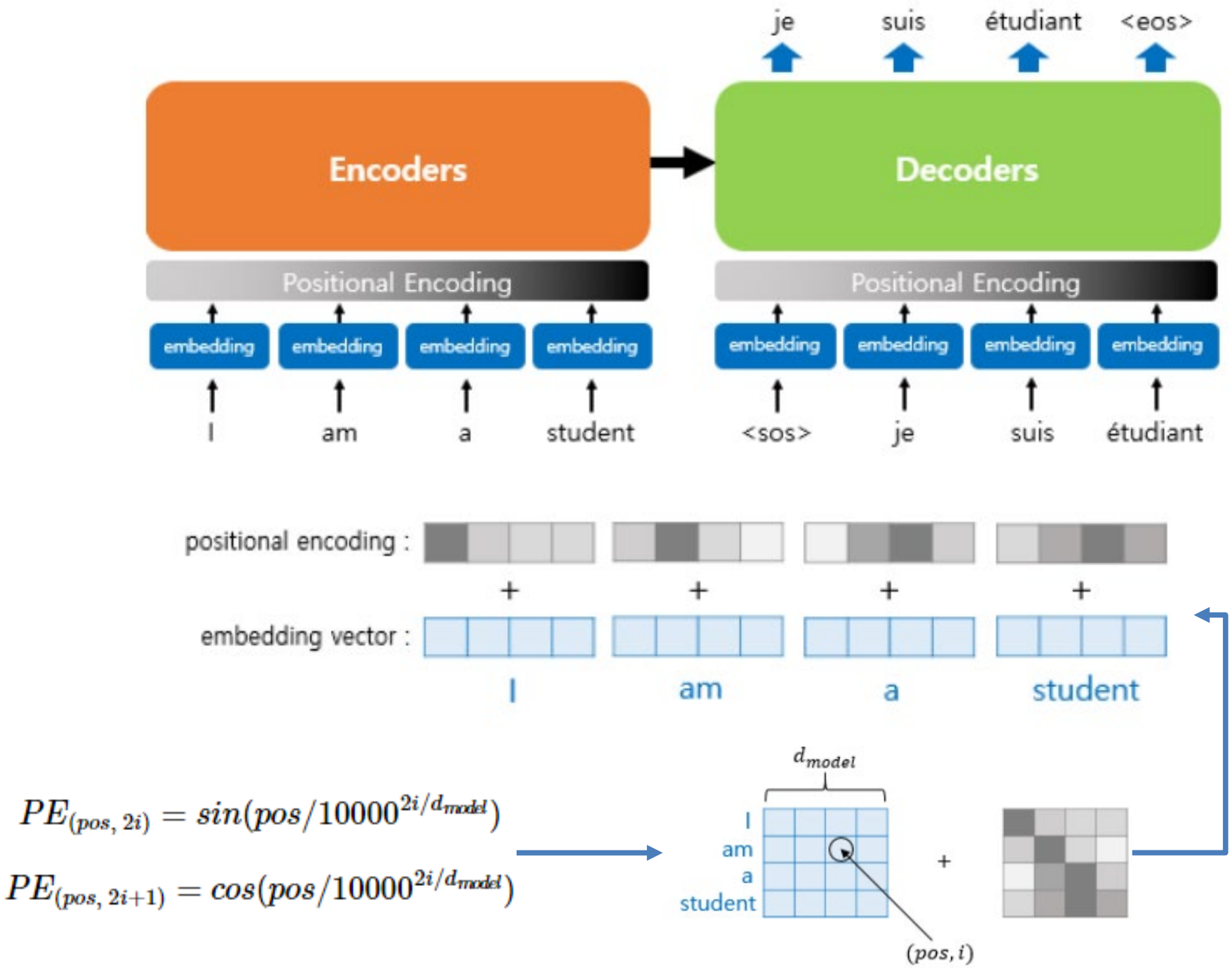
$$\hat{y_t} = \text{softmax}(W_y \widetilde{s_t} + b_y)$$

Transformer



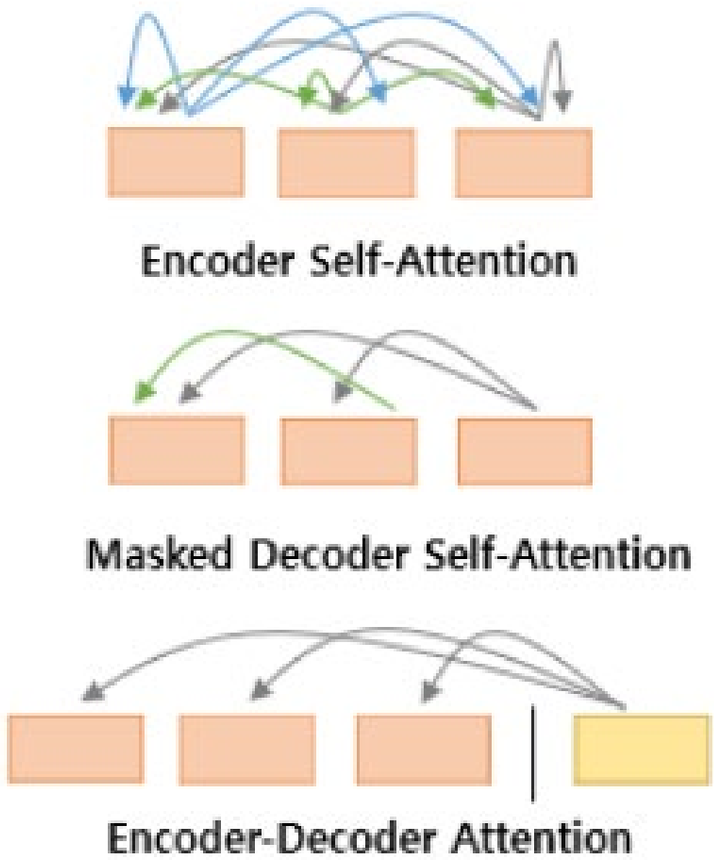
- In seq2seq, ‘soft attention’ is used where the attention weights are computed by aligning the input sequence according to the current state of the decoder
- However in the transformer, ‘multi-head attention’ is used to compute the attention scores between the query, key, and value matrices. In other words, the attention scores are used to compute the attended representation of the input sequence, which is then used as input to the feedforward layer.
- Attention scores of the self attention in the transformer are computed between the elements in the **same sequence**

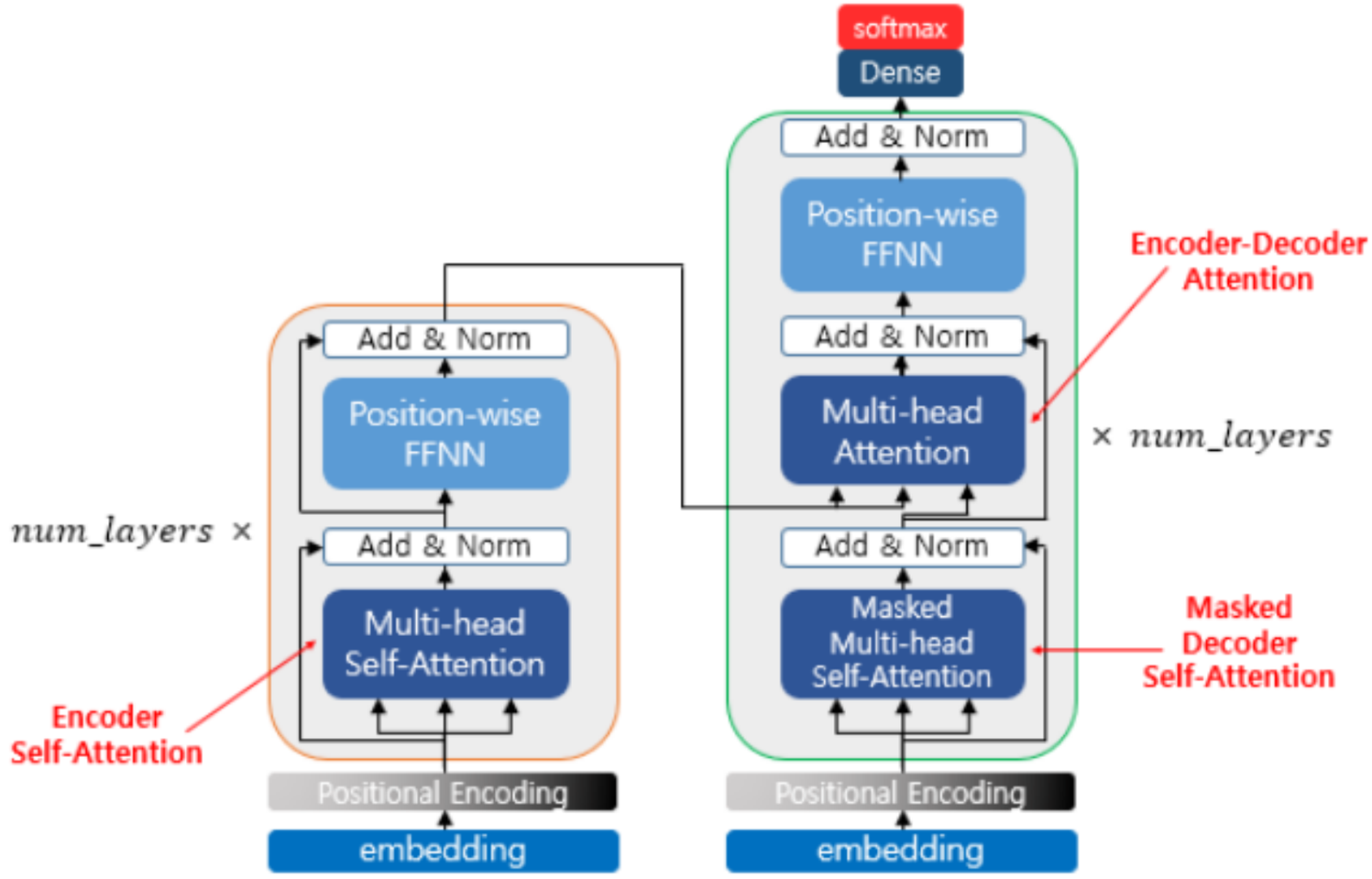
Positional Encoding



- Since the transformer receives word input in parallel rather than sequentially, the position information of the word is informed with positional encoding

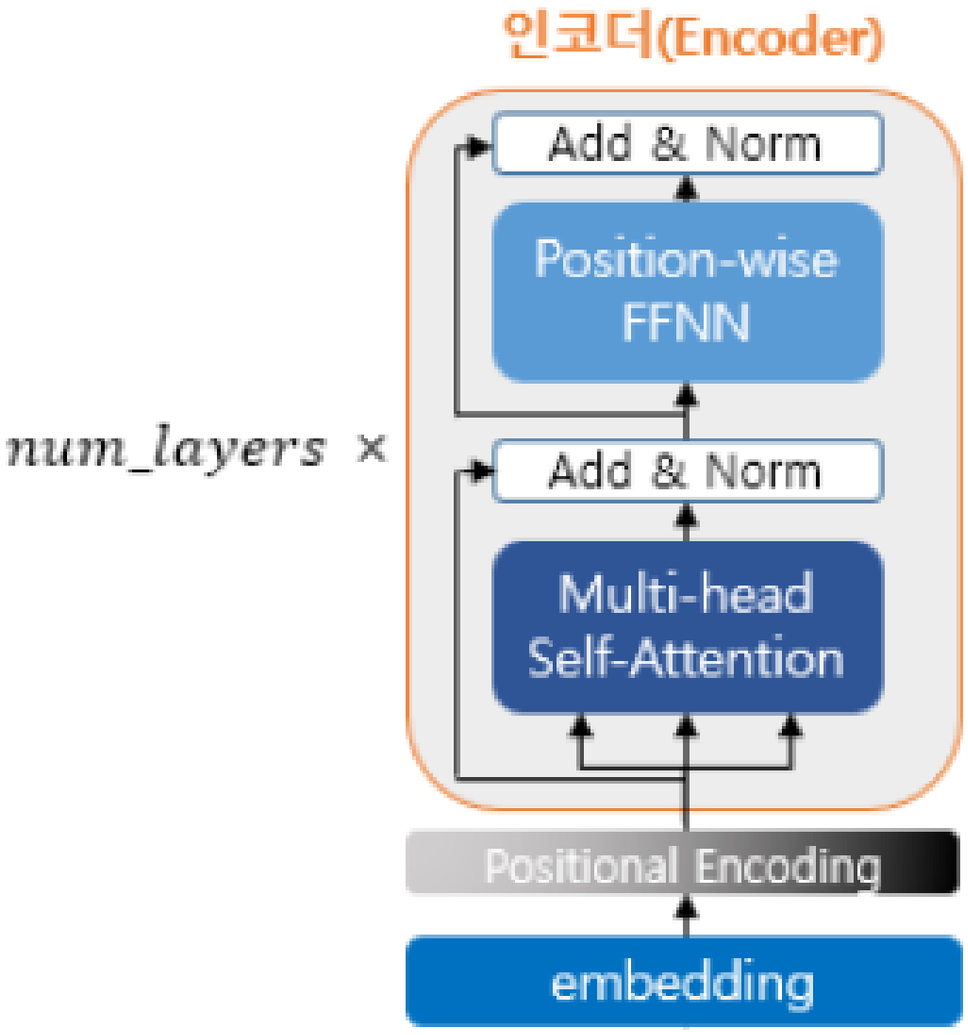
Attention in transformer

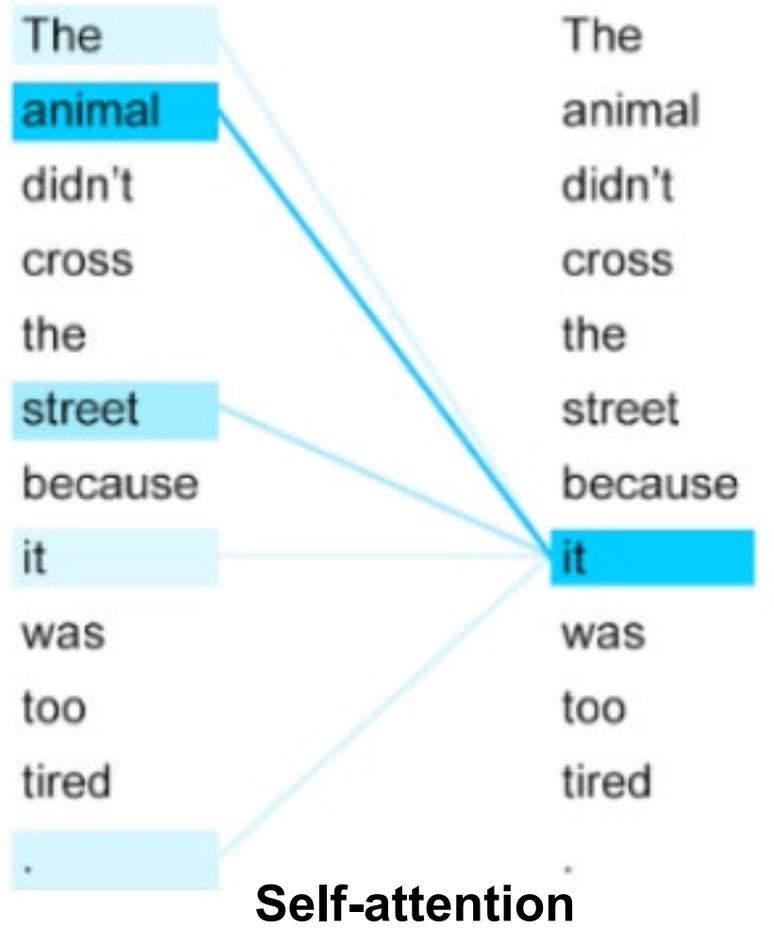




- The source of vectors (Q, K, V) are the same in self-attention. Thus, in the encoder self-attention and the masked decoder self-attention, the source of vectors (Q, K, V) are same
- In the Encoder-Decoder attention, Q is a vector from the decoder, where K, and V are from encoder

Encoder



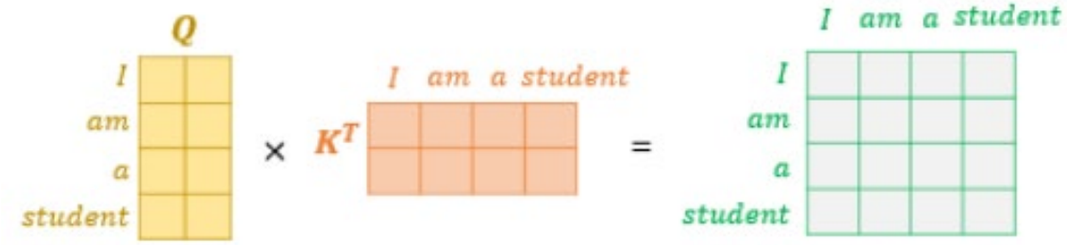


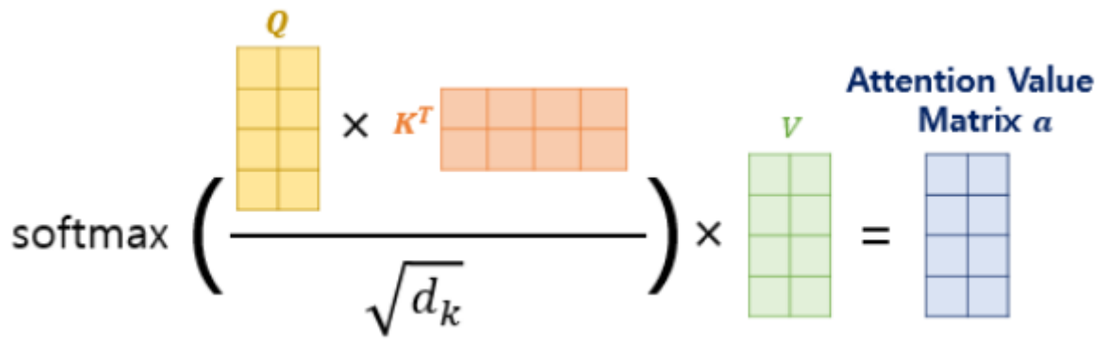
Self-attention

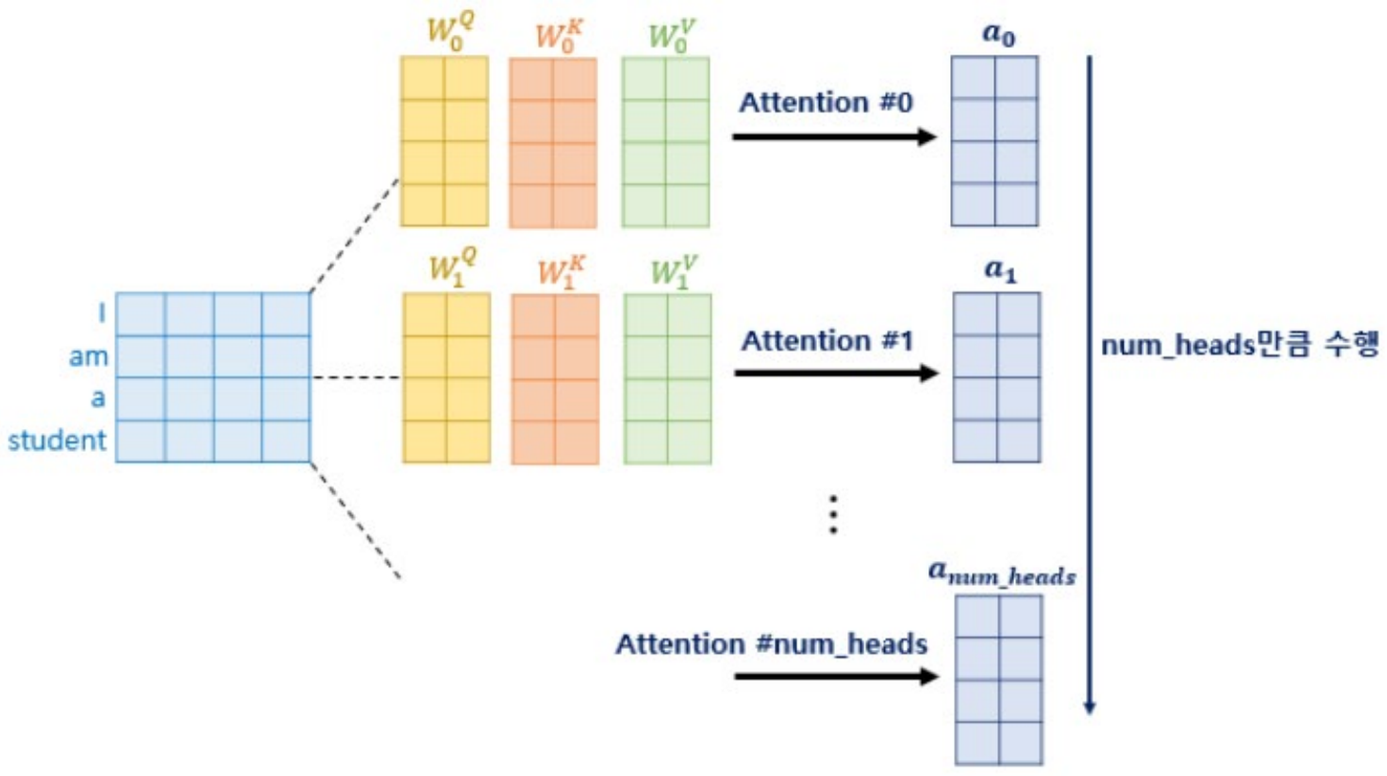
- Encoder consists of two sublayers: 1) **Multi-head Self-Attention**, 2) Position-wise FFNN
- In the self-attention, Q, K, and V refer to word vectors from ‘input sentence’, thus can be used for obtaining the similarity between words of input sentence
- Especially, scaled dot-product attention function is used

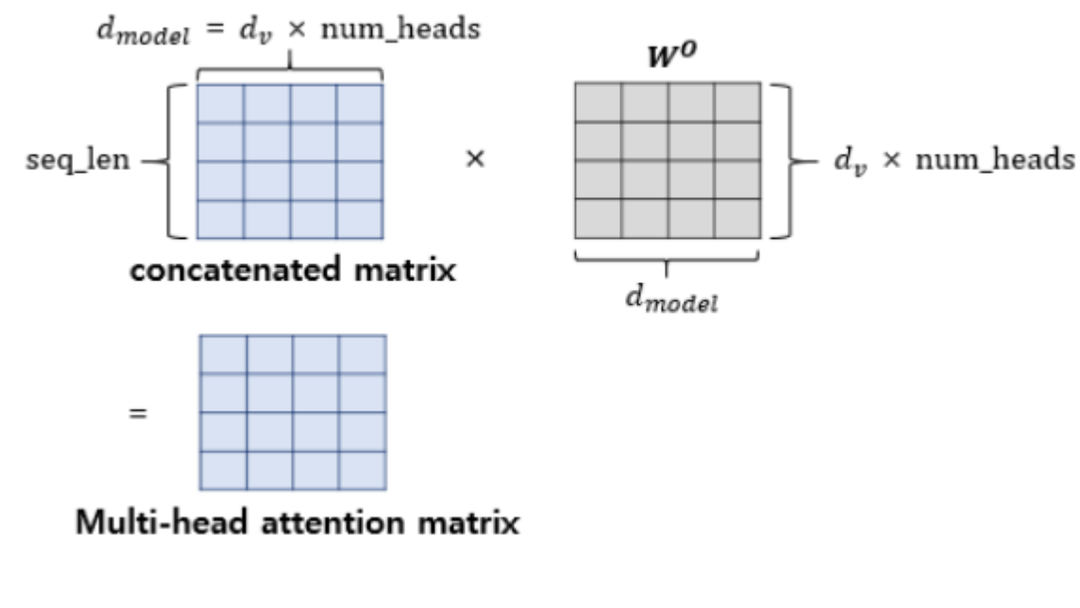
$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Encoder



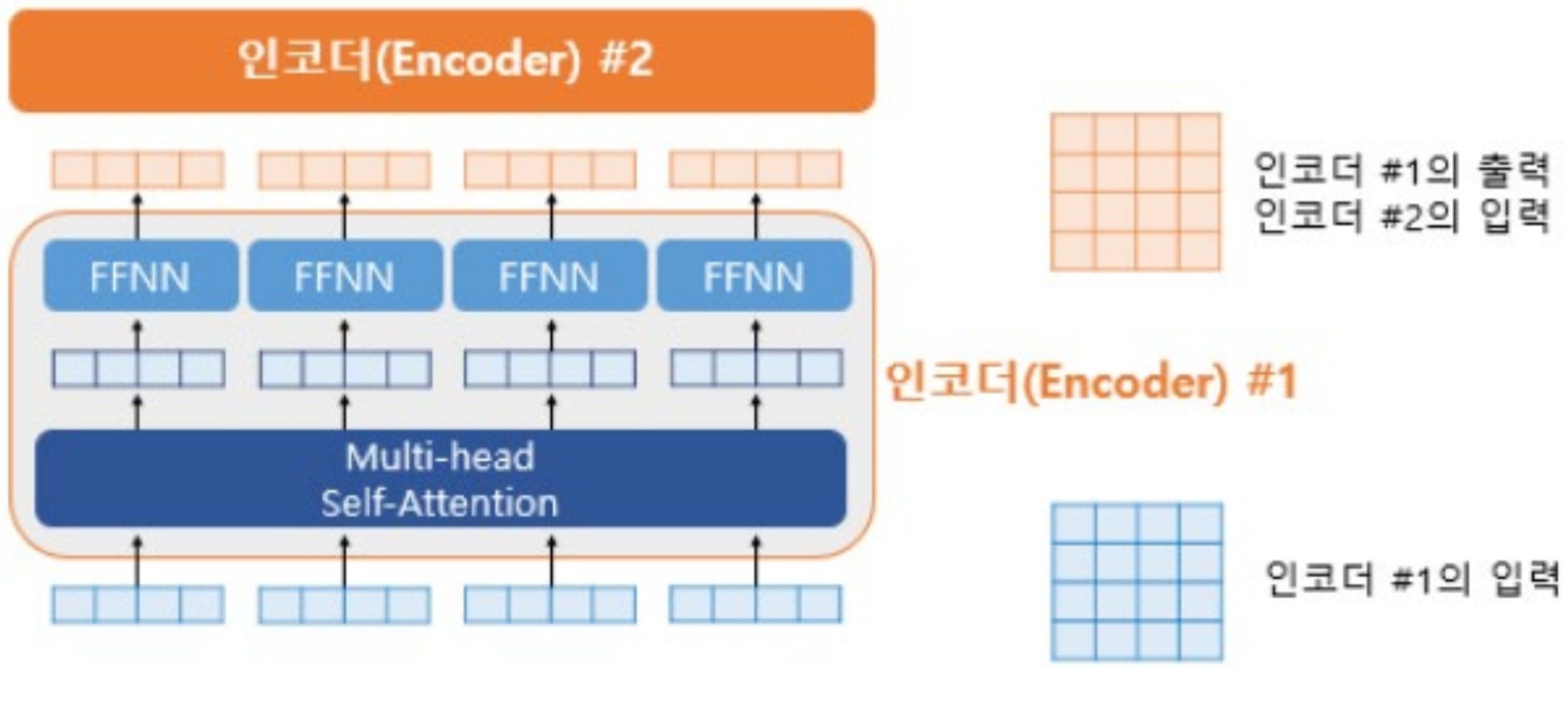






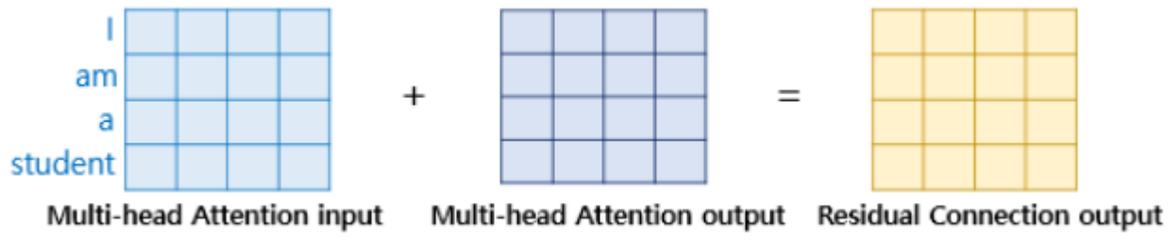
- Instead of using one attention, it is more effective to use multiple attentions in parallel since information from different perspectives can be collected
- Multi-head attention matrix is calculated by multiplying weight matrix W^O to concatenated attention matrix

Encoder



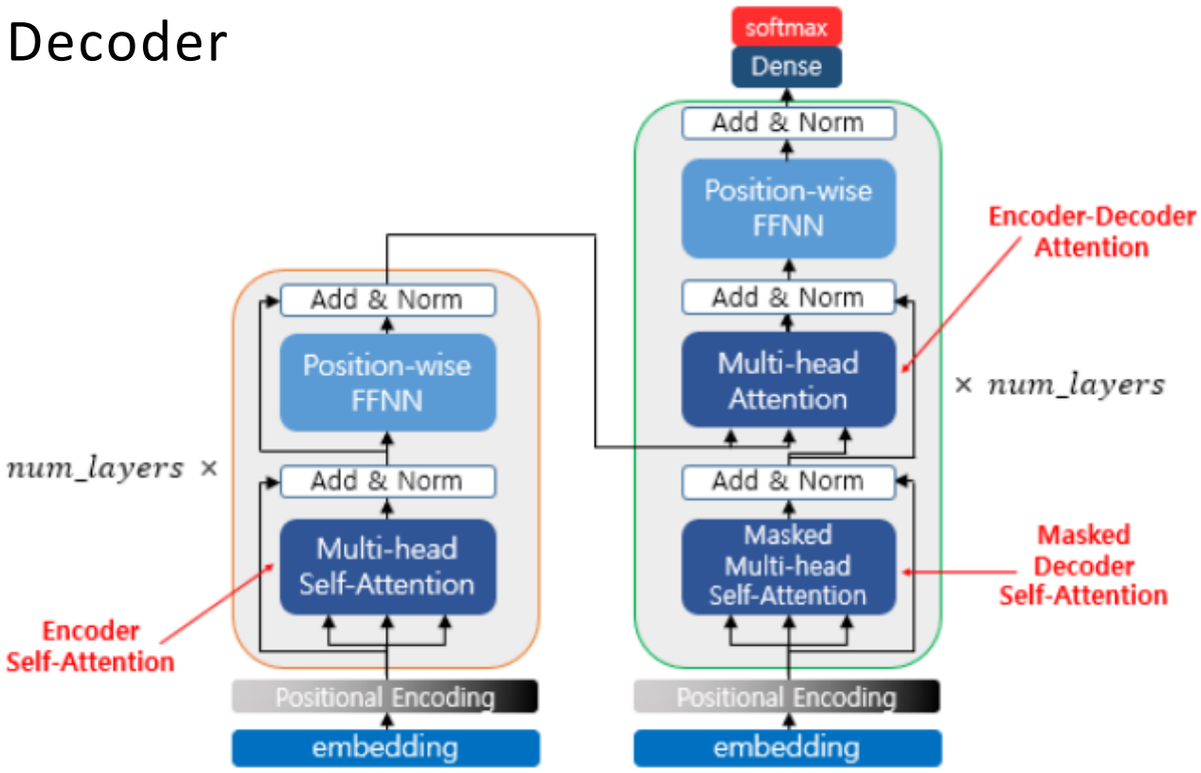
Position-wise FFNN

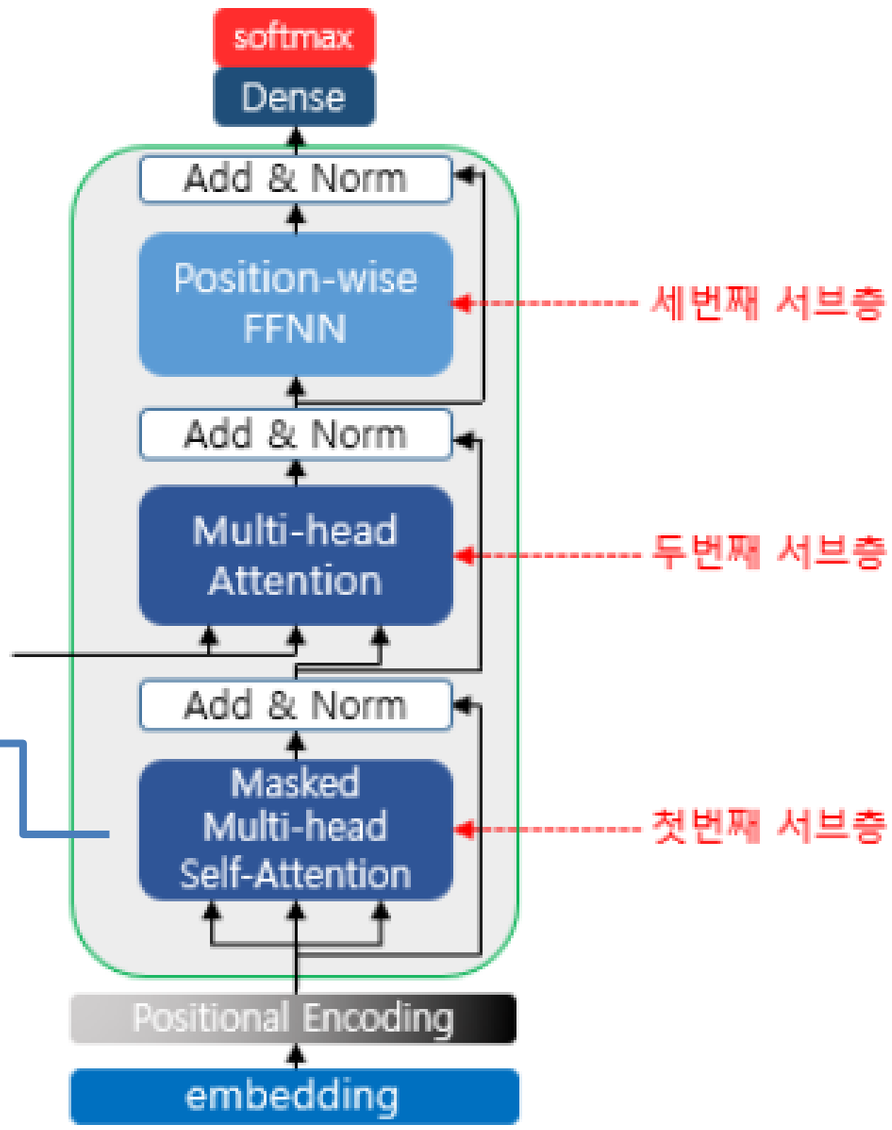
$$FFNN(x) = MAX(0, xW_1 + b_1)W_2 + b_2$$



- Residual connections are used to bypass the Multi-Head attention mechanism, which helps in improving the model's ability to capture long-range dependencies in the input
- Layer normalization is applied before and after each multi-head self-attention and FFNN, to maintain the scale and distribution of the activations during training

Decoder





- First sub-layer: masked multi-head self attention. In order to prevent the transformer referring to future words instead of past words, look-ahead mask is adopted in the first sub-layer
- Second sub-layer: encoder-decoder attention. (Key and value from encoder, and query from the decoder)

Application

1. Language Translation: NMT (Neural Machine Translation) used for Google Translate
2. Text Summarization: BART (Bidirectional and Auto-Regressive Transformer)
3. Question Answering: OpenAI's GPT-3 (Generative Pre-trained Transformer 3)
4. Sentiment Analysis: GLUE (General Language Understanding Evaluation)