

Transformer

Attention is all you need

서울대학교 IDEA 연구실

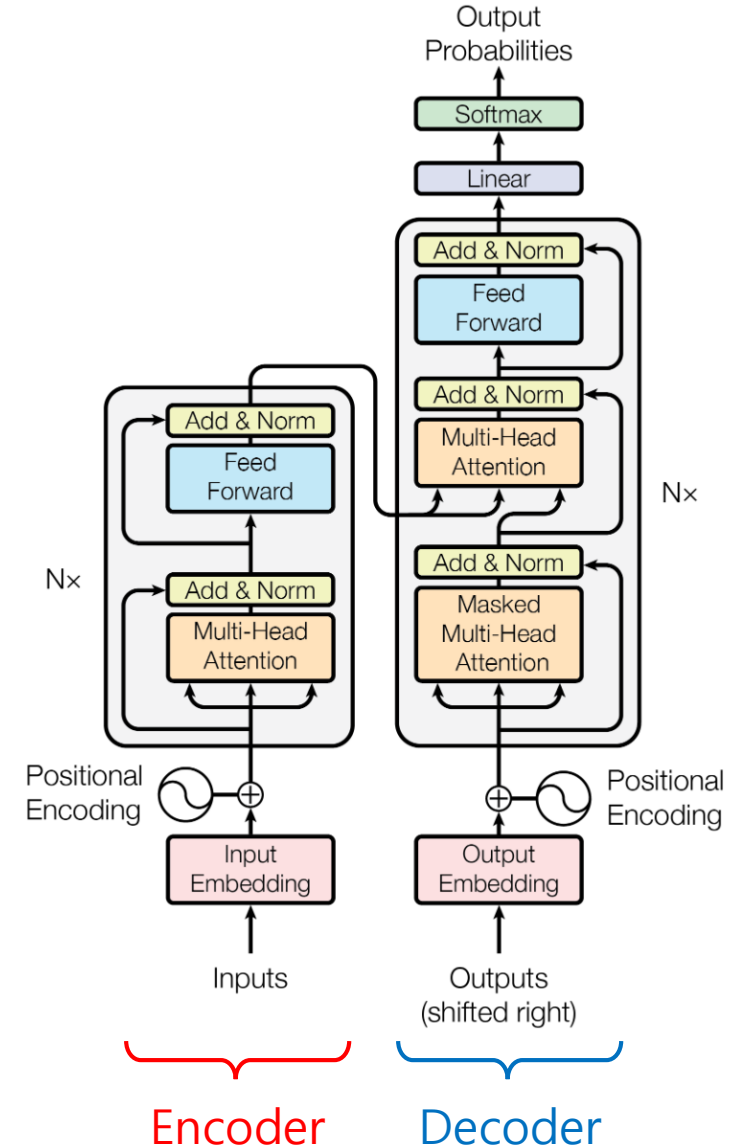
박사과정 박세현

OUTLINE

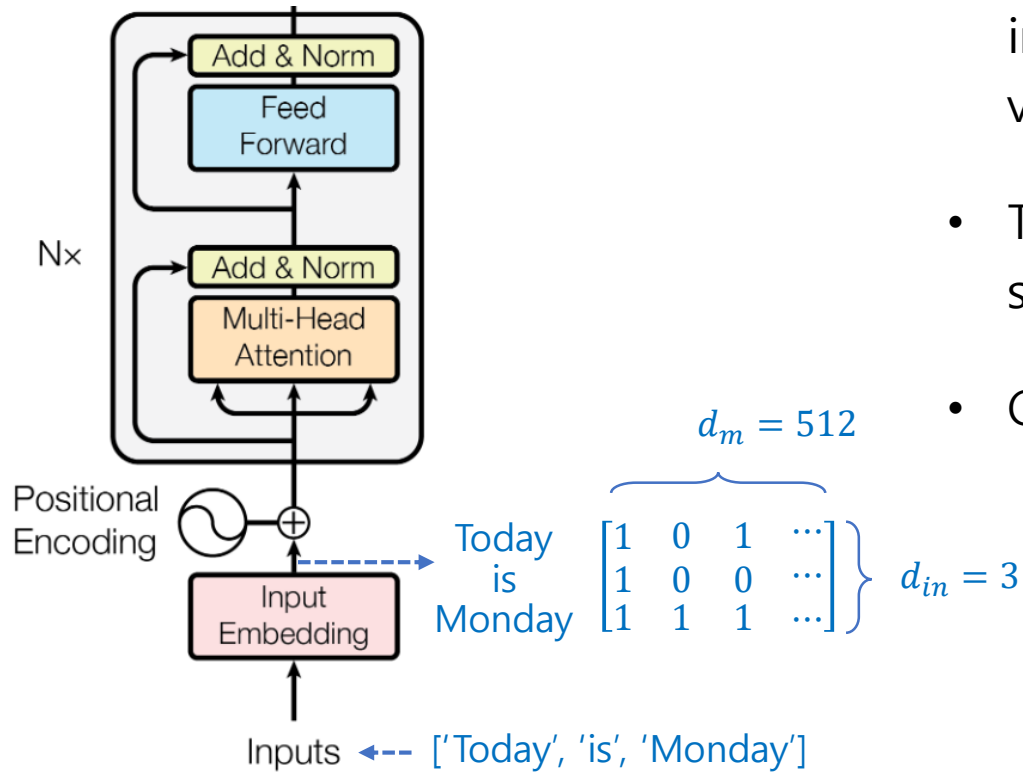
- 01** Transformer
- 02** Encoder
- 03** Decoder
- 04** Inference example

01. Transformer

- The Transformer algorithm was first proposed in a paper published in 2017 by researchers from Google Brain.
- To overcome the limitations of RNN used in traditional sequence-to-sequence models, the Transformer completely avoids RNN structures and instead utilizes Attention mechanisms.
(This is why the paper is titled "Attention is all you need.")
- The Transformer consists of an Encoder and a Decoder, each made up of multiple sub-layers.
- In this review, we will focus on the overall structure of the Transformer.

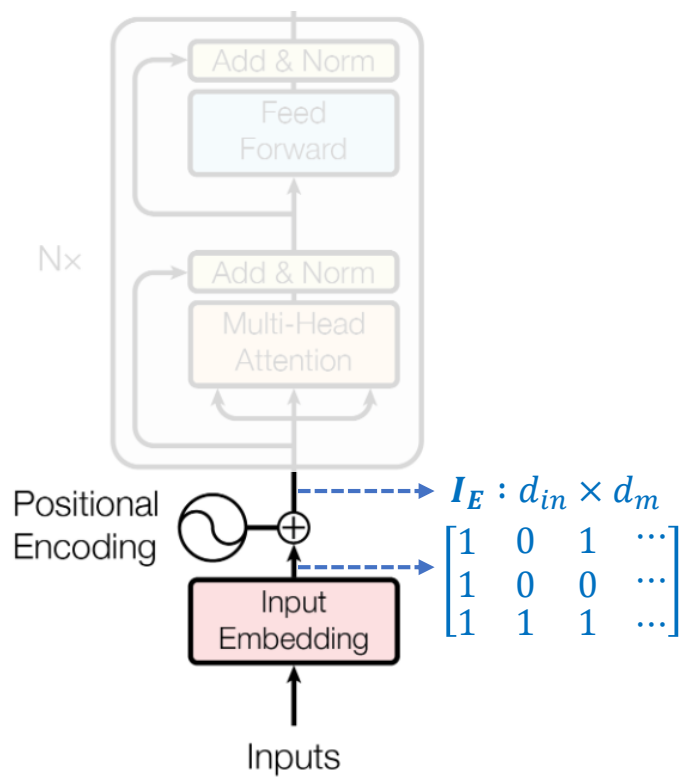


02. Encoder



- The encoder of the Transformer model serves to convert an input sequence (e.g., a sentence) into a high-dimensional vector.
- The Encoder part uses multiple identical Encoder blocks stacked together (the paper uses $N=6$).
- Components:
 - Positional Encoding
 - Multi-Head Attention
 - Feed Forward
 - Add & Norm

02. Encoder - Positional Encoding



- Positional Encoding adds positional information to input matrix that lack order information.
- In this work, they use sine and cosine functions of different frequencies:

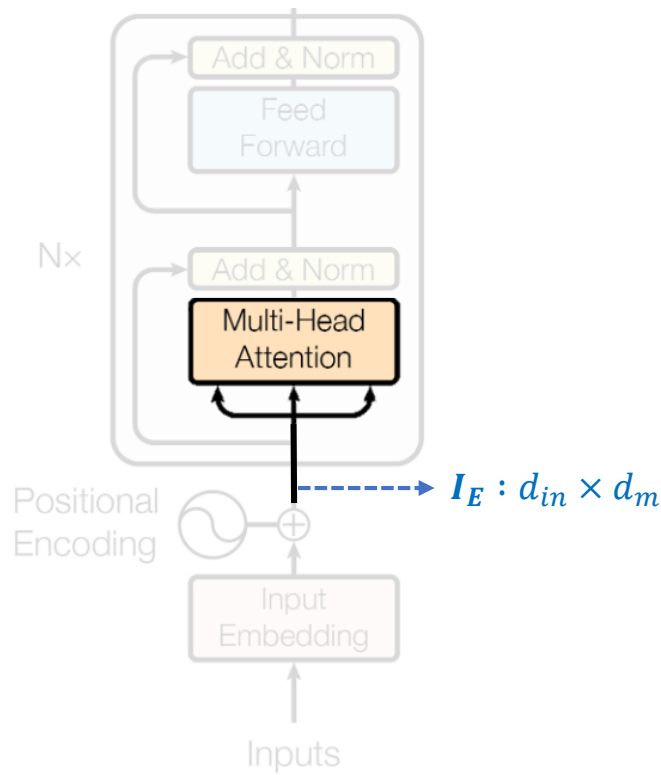
$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_m})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_m})$$

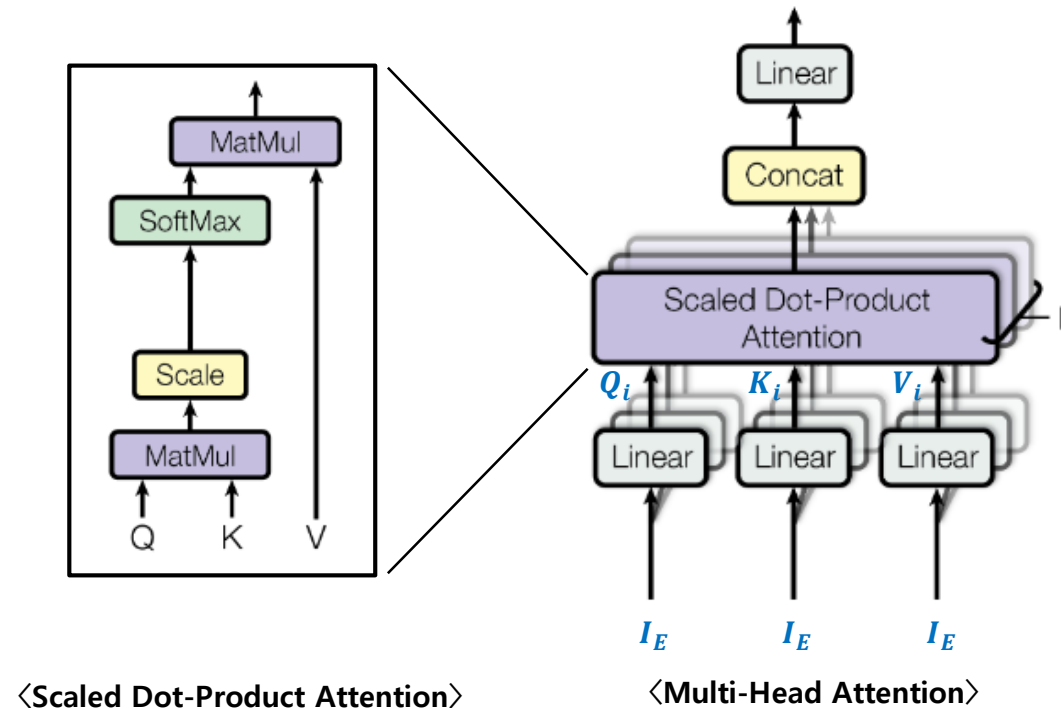
- Ex. Adding Positional Encoding to the sequence ['Today', 'is', 'Monday']

$$I_E = \begin{bmatrix} 1 & 0 & 1 & \dots \\ 1 & 0 & 0 & \dots \\ 1 & 1 & 1 & \dots \end{bmatrix} + \begin{bmatrix} \cos\left(\frac{1}{10^{4*(0/d_m)}}\right) & \sin\left(\frac{1}{10^{4*(2*1/d_m)}}\right) & \cos\left(\frac{1}{10^{4*(2*1/d_m)}}\right) & \dots \\ \cos\left(\frac{2}{10^{4*(0/d_m)}}\right) & \sin\left(\frac{2}{10^{4*(2*1/d_m)}}\right) & \cos\left(\frac{2}{10^{4*(2*1/d_m)}}\right) & \dots \\ \cos\left(\frac{3}{10^{4*(0/d_m)}}\right) & \sin\left(\frac{3}{10^{4*(2*1/d_m)}}\right) & \cos\left(\frac{3}{10^{4*(2*1/d_m)}}\right) & \dots \end{bmatrix}$$

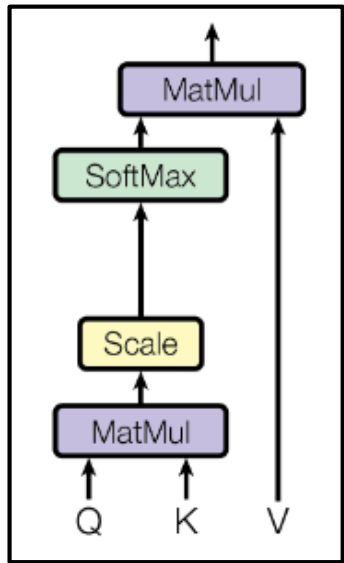
02. Encoder - Multi-head Attention (self-attention)



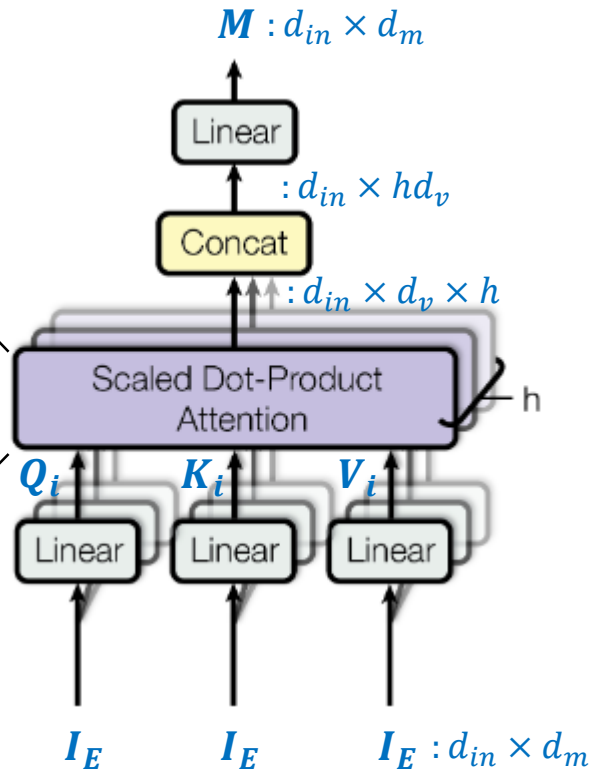
- Multi-Head Attention in the Encoder consists of h (the paper uses $h = 8$) self-Attention mechanisms.
- It calculates the relationships between words in the input sequence through these mechanisms.



02. Encoder - Multi-head Attention (self-attention)



<Scaled Dot-Product Attention>



<Multi-Head Attention>

- Query, Key, Value are the product of the embedded sentence matrix and each weight matrix.
- Scaled Dot-Product Attention is equivalent to the Dot-Product attention multiplied by a scaling factor of $\frac{1}{\sqrt{d_k}}$ (where $d_k = 64$).
- Multi-Head Attention can be expressed with the following formula.

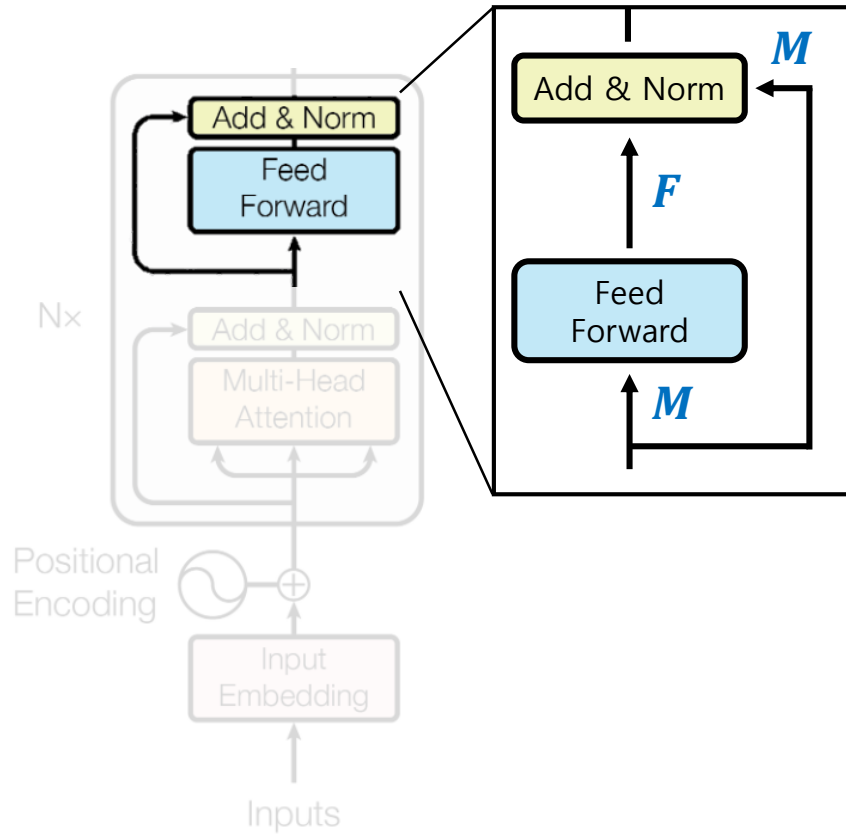
$$M = \text{MultiHead}(I_E, I_E, I_E) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^o$$

$$\text{where } \text{head}_i = \text{Attention}(I_E W_i^Q, I_E W_i^K, I_E W_i^V)$$

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

- Where the projection parameter matrix $W_i^Q, W_i^K \in \mathbb{R}^{d_m \times d_k}, W_i^V \in \mathbb{R}^{d_m \times d_v}, W^o \in \mathbb{R}^{hd_v \times d_m}$

02. Encoder - Feed Forward and Add & Norm layer



- **Feed Forward Neural Network**

Feed Forward

- The position-wise Feed-Forward Networks used in the Transformer are applied independently to each position.
- This consists of two linear transformation with a ReLU activation in between :

$$FNN(x) = W_2 \text{ReLU}(W_1 x + b_1) + b_2$$

(Input and output dim : d_m , inner-layer dim : $d_{ff} = 2048$)

- **Add & Norm layer**

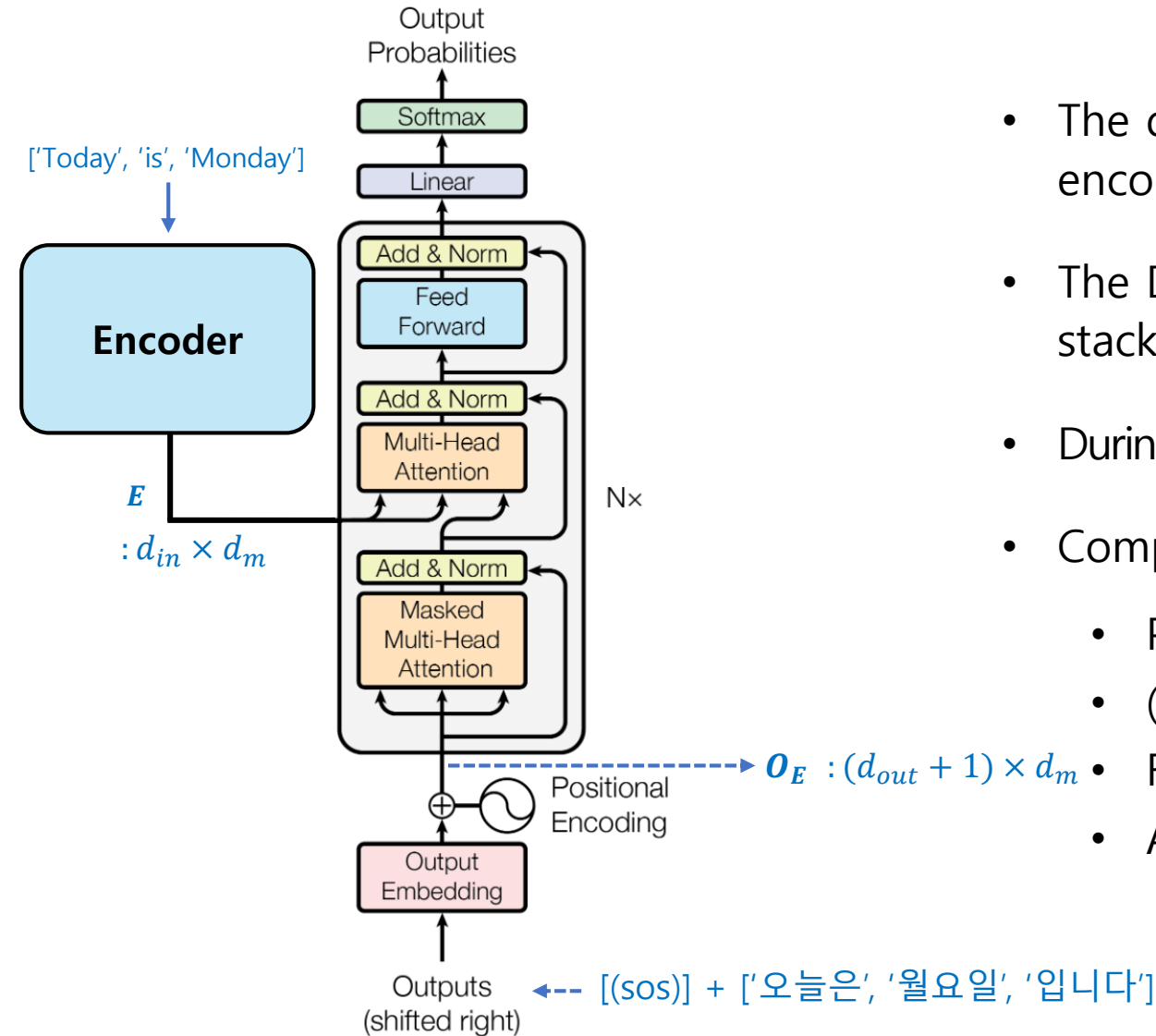
Add & Norm

- In the Transformer, each sub-layer's output is not used directly.
- Instead, the input and output of the sub-layer are added together and then layer normalization is applied.

$$Add\&Norm(M, S) = LayerNorm(M + S)$$

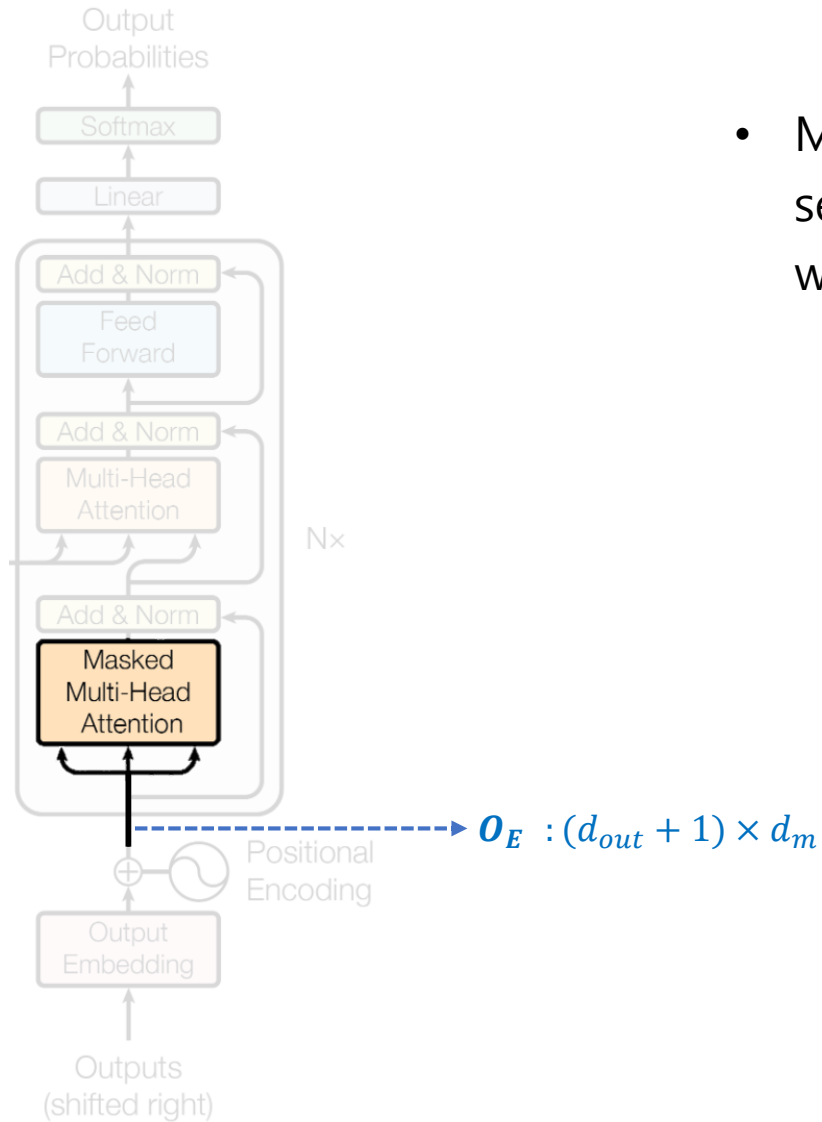
03. Decoder

Target : ['오늘은', '월요일', '입니다'] + [(eos)]

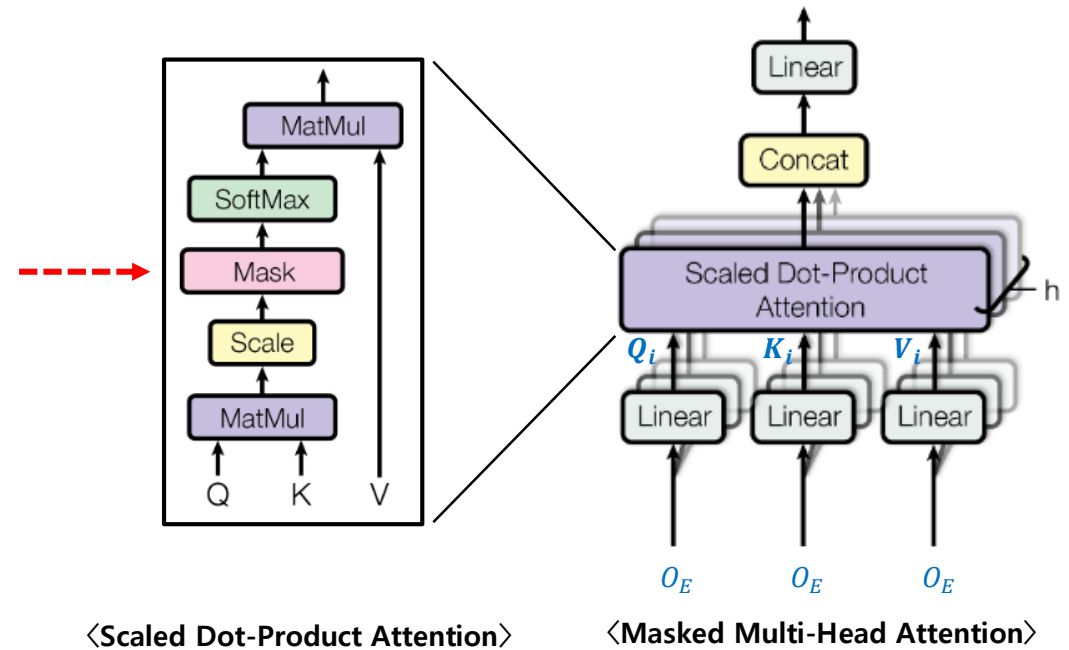


- The decoder in the Transformer receives the output from the encoder and generates the target sequence.
- The Decoder part also uses multiple identical Decoder blocks stacked together (the paper uses $N=6$).
- During training in the Decoder, the teacher forcing method is used.
- Components:
 - Positional Encoding
 - (Masked) Multi-Head Attention
 - Feed Forward
 - Add & Norm

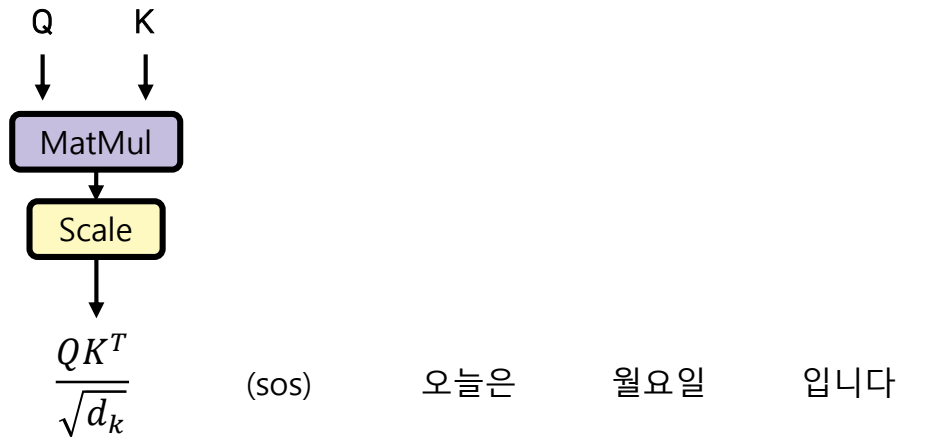
03. Decoder - Masked Multi-head Attention (self-attention)



- Masked Multi-Head Attention used in the Decoder employs a self-Attention mechanism similar to that in the Encoder, but with some differences.

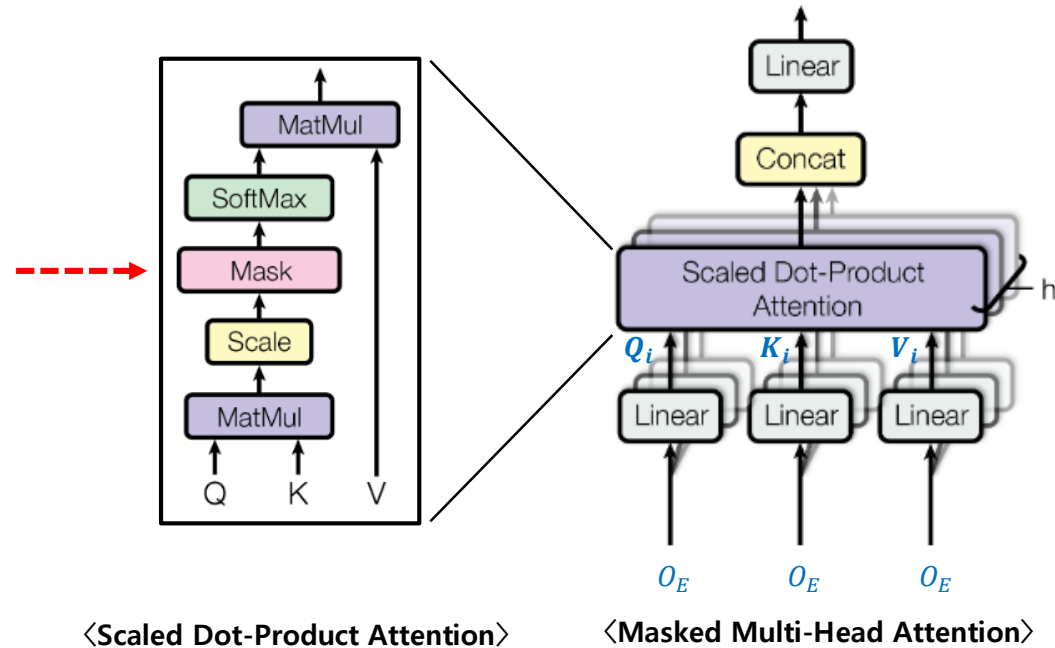


03. Decoder - Masked Multi-head Attention (self-attention)



	(sos)	오늘은	월요일	입니다
(sos)	10	2	1	1
오늘은	1	8	1	4
월요일	1	2	9	1
입니다	1	2	1	7

- Since the Decoder operates sequentially, it cannot use information about words that appear after the current word in the attention scores.
- Therefore, the attention scores for words that appear after the reference word are replaced with $-\infty$.

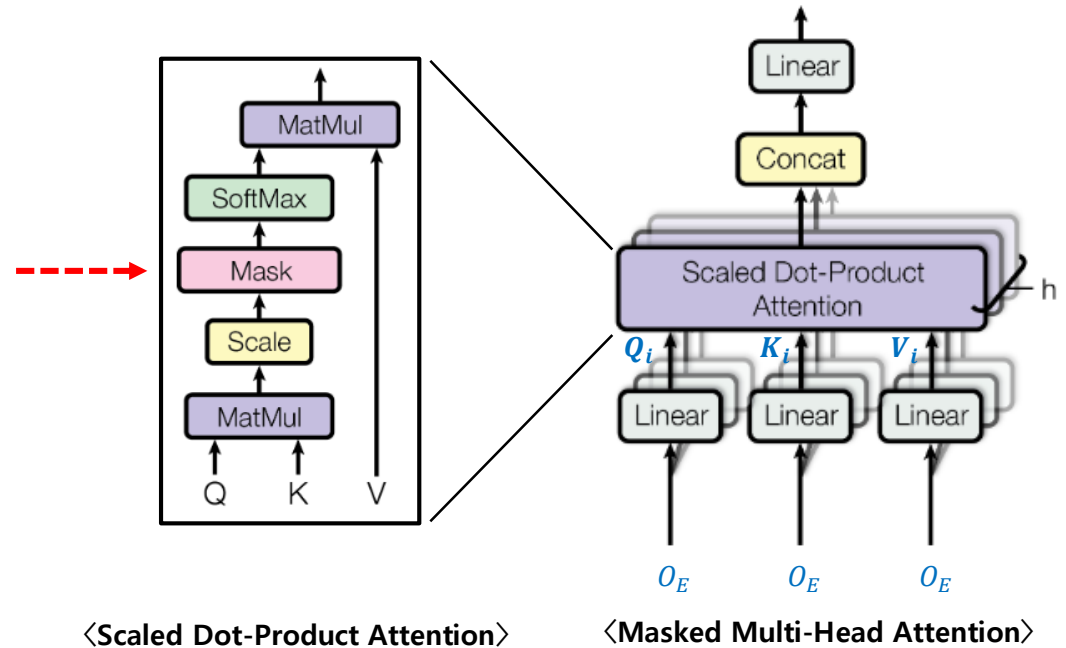


03. Decoder - Masked Multi-head Attention (self-attention)

- Since the Decoder operates sequentially, it cannot use information about words that appear after the current word in the attention scores.
- Therefore, the attention scores for words that appear after the reference word are replaced with $-\infty$.

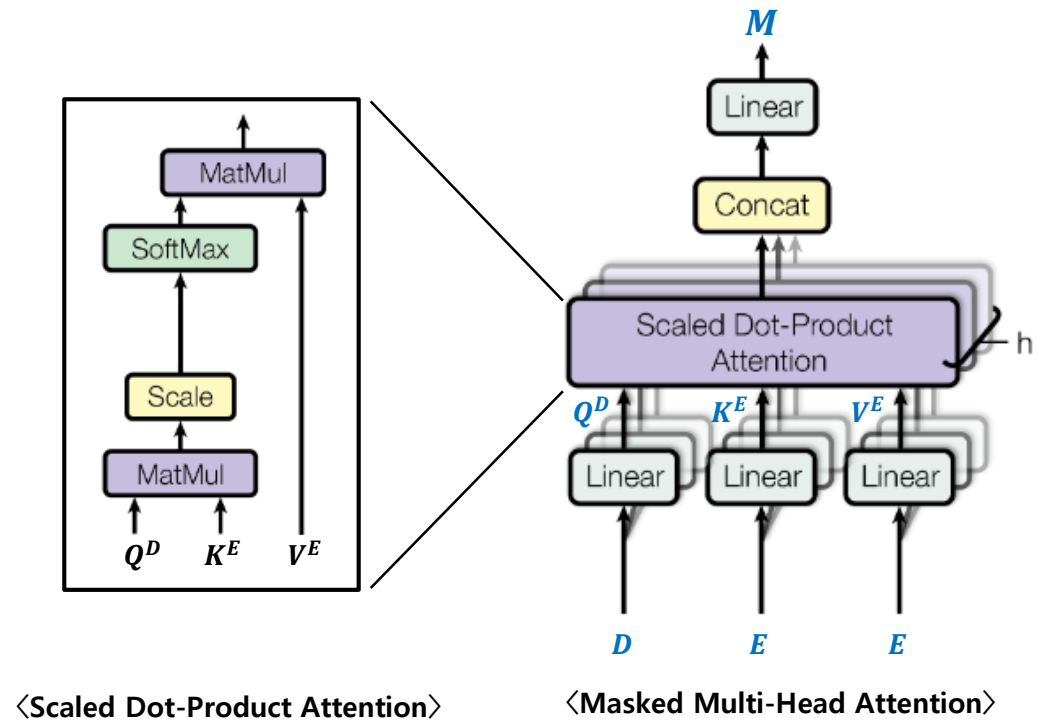
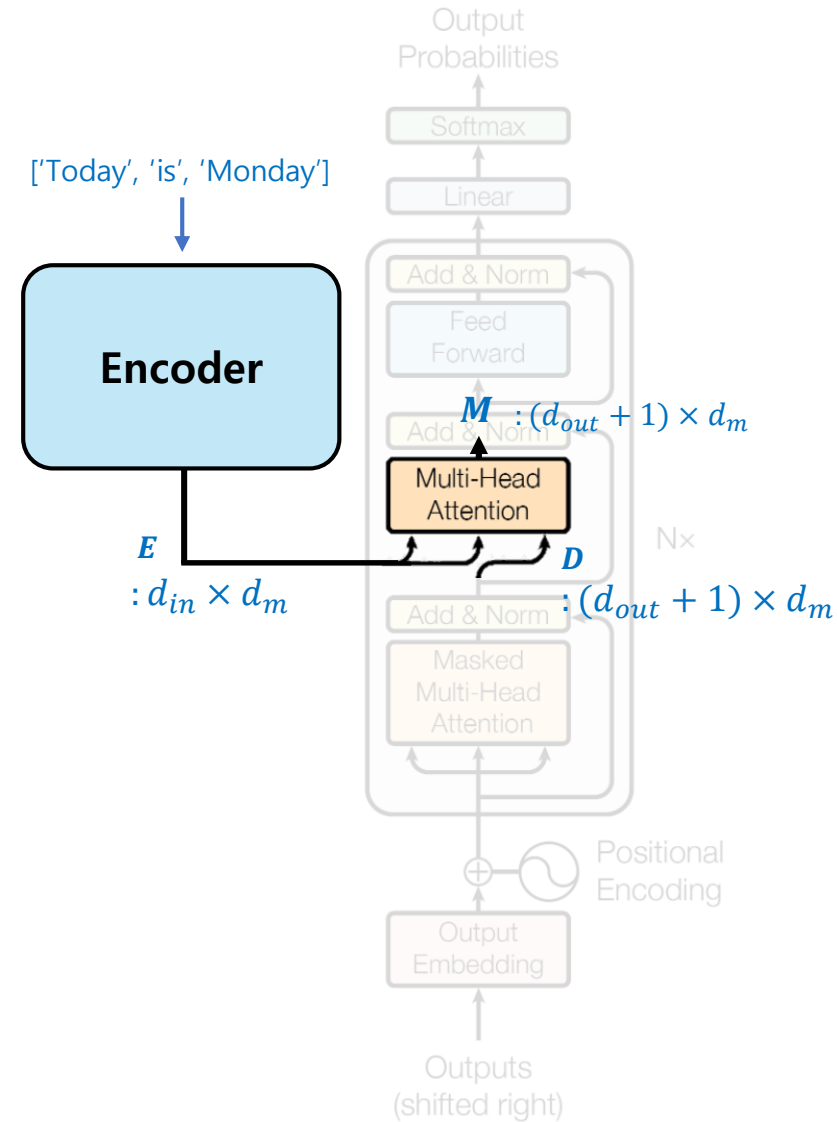
$Mask \left(\frac{QK^T}{\sqrt{d_k}} \right)$

	(sos)	오늘은	월요일	입니다
(sos)	10	$-\infty$	$-\infty$	$-\infty$
오늘은	1	8	$-\infty$	$-\infty$
월요일	1	2	9	$-\infty$
입니다	1	2	1	7



03. Decoder - Multi-head Attention(encoder-decoder attention)

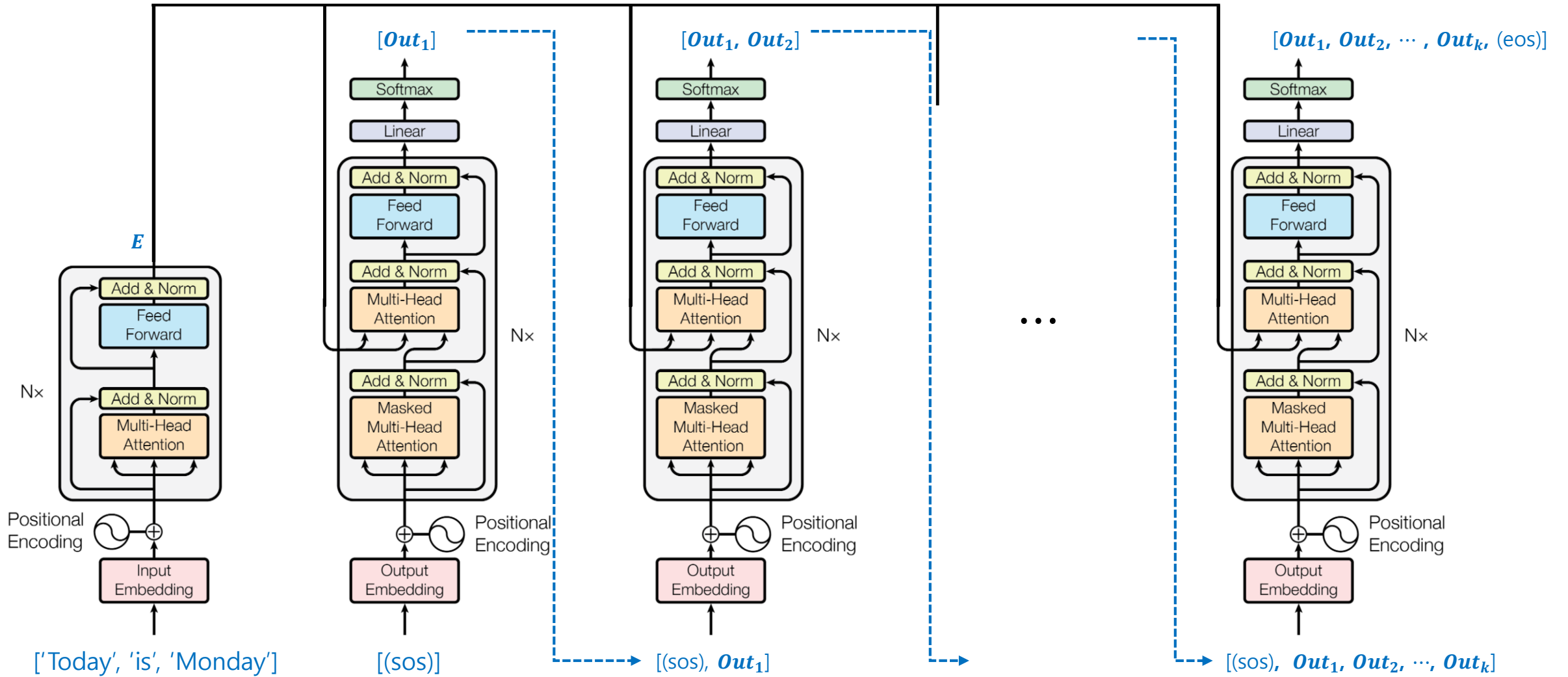
- In the Decoder block, the second Multi-Head Attention is the encoder-decoder attention.
- It uses the output from the encoder to compute the key and value matrices, and computes the query matrix using the output computed within the Decoder block.



<Scaled Dot-Product Attention>

<Masked Multi-Head Attention>

04. Inference example



End