

Attention Is All You Need

2021. 08. 16

GDSC Sookmyung – DeepSleep 논문 리뷰 스터디

발표자 | 컴퓨터과학전공 19 남수연

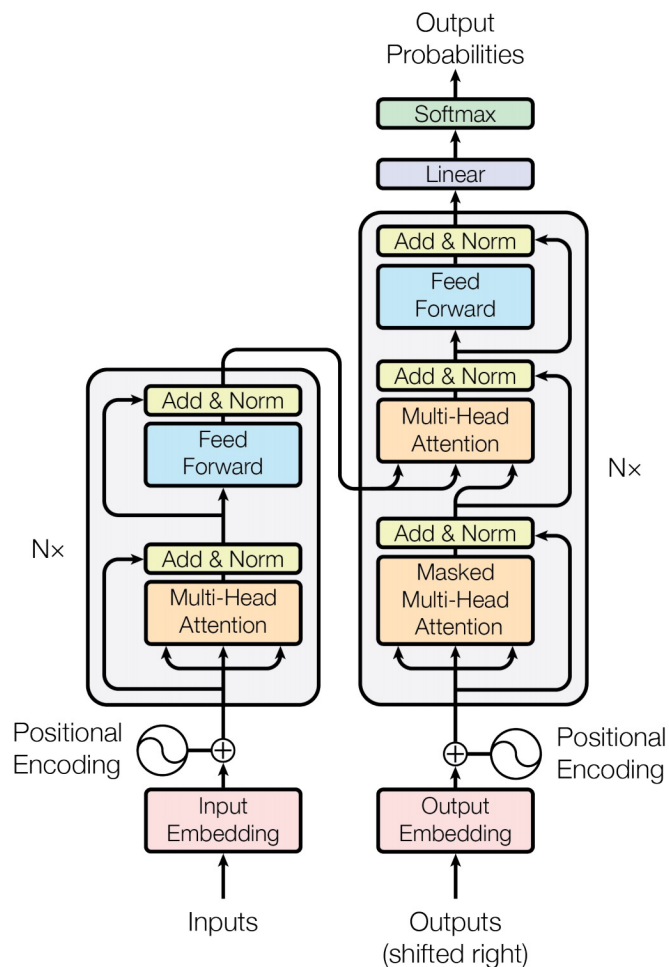
Introduction

- RNN(순환 신경망)은 시퀀스 모델링 / 언어 모델링, 기계 번역같은 변환 문제에 유용하게 사용됨
- 순환 모델의 **순차적**인 특징으로 인해 학습 데이터의 병렬 처리가 불가능, 긴 시퀀스를 처리하는 데 치명적 (long-term dependency)
- **Attention** 메커니즘은 시퀀스 모델링과 변환 모델에서 필수적인 부분이 되었으며, input과 output 시퀀스 사이의 거리를 고려하지 않고 의존성을 모델링할 수 있음
하지만, Attention은 보통 RNN과 함께 사용됨(2017년 기준) 🖐 RNN의 문제점을 Attention도...

이 논문에서는 “**Transformer**”를 제안
recurrence를 사용하지 않고, Attention에 전적으로 의존하여 input과 output 사이의 전역 의존성을 catch
병렬 처리가 가능하며, 성능 아주 굿

Model Architecture

참고: [YouTube] [\[딥러닝 기계 번역\] Transformer: Attention Is All You Need \(꼼꼼한 딥러닝 논문 리뷰와 코드 실습\)](#)



* Positional Encoding

- Transformer는 RNN, CNN을 사용하지 않으므로, 단어의 순서 정보를 담고 있는 Positional Encoding을 이용한다.

* Encoder

- 첫번째 sub-layer: Inputs에 대한 **Self Attention**
Self Attention: 입력 시퀀스의 각각의 단어가 서로에게 얼마만큼의 영향(연관성)을 주는가?
- **Residual Learning(잔여 학습):** 특정 레이어를 건너뛰어 입력, gradient vanishing 문제 완화
- 레이어를 중첩하여 Attention과 Normalization을 반복
- Encoder의 출력은 Decoder의 레이어 안 2번째 Attention의 입력으로 들어감

* Decoder

- 첫번째 sub-layer: Outputs에 대한 **Self Attention**
 - 두번째 sub-layer: Encoder의 정보에 대한 Attention (**Encoder-Decoder Attention**)
Encoder의 출력을 받아 입력 시퀀스 중 어떤 단어에 가장 주목해야 하는지 계산
- Residual Learning** 이용

Model Architecture

참고: [YouTube] [트랜스포머 \(어텐션 이즈 올 유 니드\)](#)

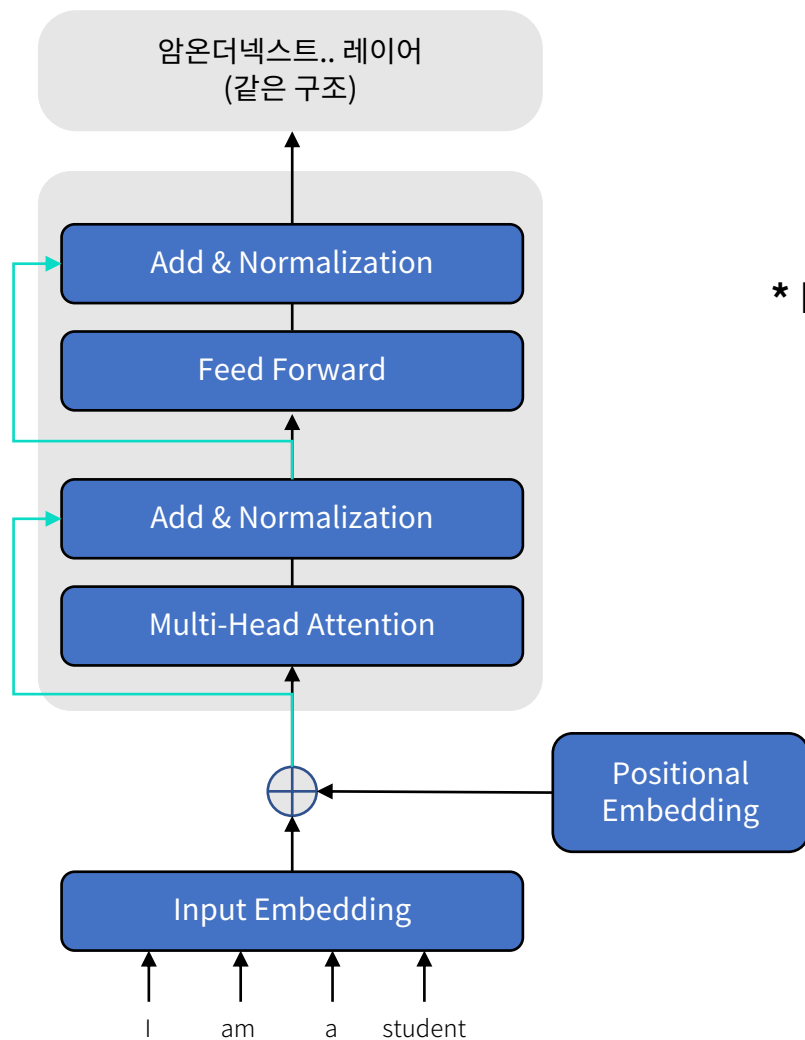
$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$

장점 1. 인코딩 값이 무조건 $[-1, 1]$ 사이의 값을 갖는다.

장점 2. 학습 시 사용했던 데이터들보다 더 긴 문장이 들어온 경우에도 무리없이 처리 가능하다.

Model Architecture

참고: [YouTube] [\[딥러닝 기계 번역\] Transformer: Attention Is All You Need \(꼼꼼한 딥러닝 논문 리뷰와 코드 실습\)](#)

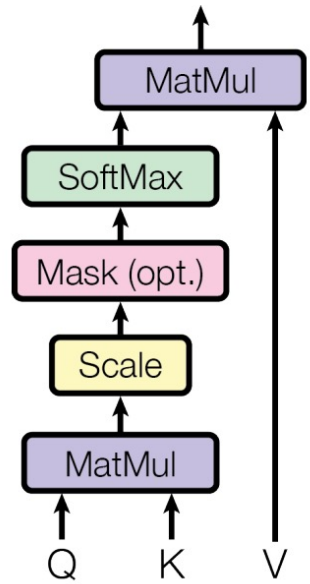


* Encoder

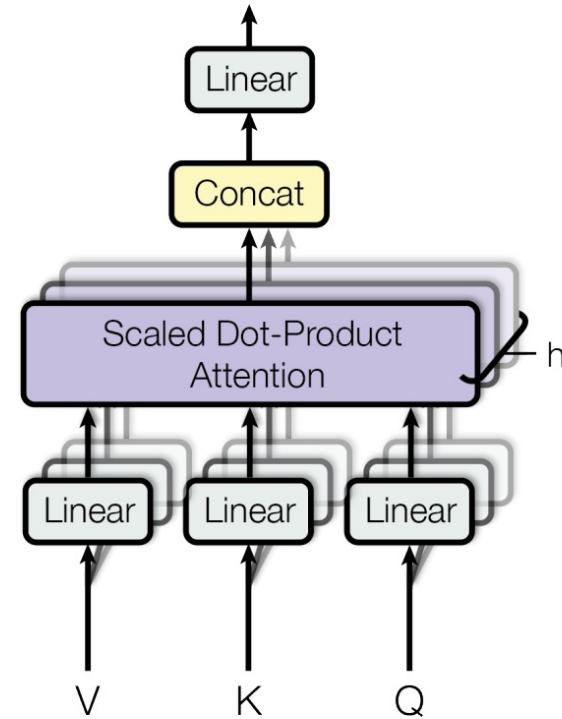
- 첫번째 sub-layer: Inputs에 대한 **Self Attention**
Self Attention: 입력 시퀀스의 각각의 단어가 서로에게 얼마만큼의 영향(연관성)을 주는가?
- **Residual Learning(잔여 학습):** 특정 레이어를 건너뛰어 입력, gradient vanishing 문제 완화
- 레이어를 중첩하여 Attention과 Normalization을 반복(각 레이어는 서로 다른 파라미터를 가짐)
- Encoder의 출력은 Decoder의 레이어 안 2번째 Attention의 입력으로 들어감

Model Architecture

Scaled Dot-Product Attention

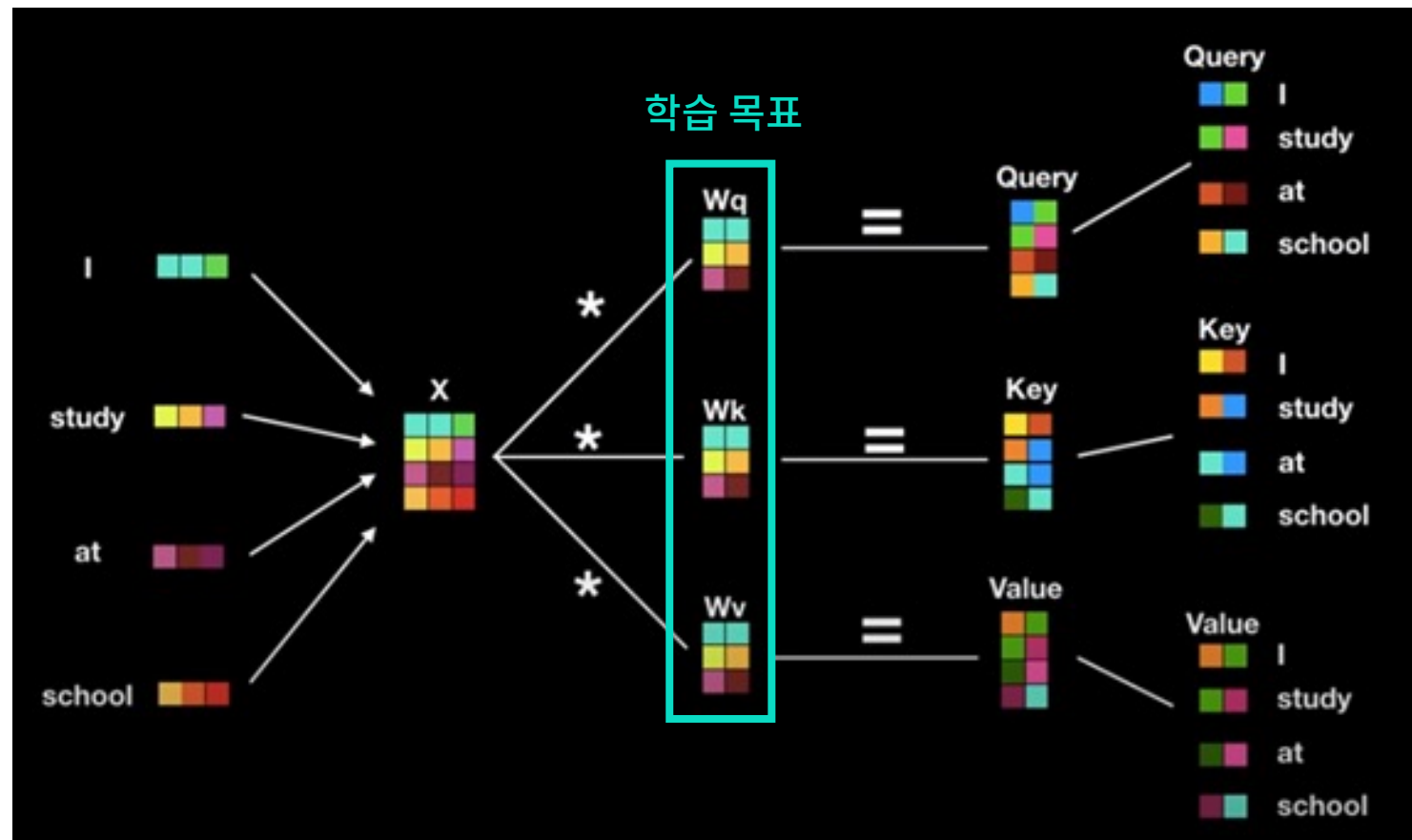


Multi-Head Attention



Model Architecture – Scaled Dot-Product Attention

참고: [Tistory] [Transformer \(Attention is All You Need\) 논문 리뷰](#)











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


- $Q(\text{query})$: 영향을 받는 단어
- $K(\text{key})$: 영향을 주는 단어
- $V(\text{value})$: 영향에 대한 가중치

QK 행렬곱 값이 너무 크면
gradient가 소실될 가능성이 있으므로
스케일링이 필요함

Model Architecture – Scaled Dot-Product Attention

참고: [Tistory] [Transformer \(Attention is All You Need\) 논문 리뷰](#)

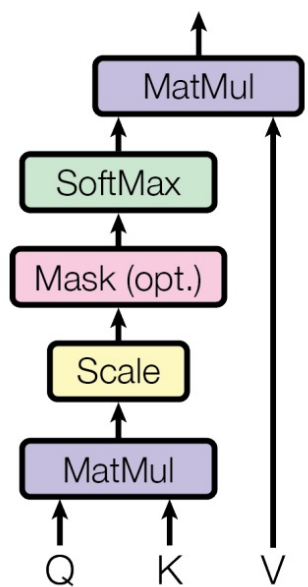
	Query * Key ^T	Score	Softmax
I	I * I  *  = 130	130	0.92
	I * study  *  = 50	50	0.05
	I * at  *  = 20	20	0.02
	I * school  *  = 10	10	0.01

	Query * Key ^T	Score	Softmax	Value	Softmax * Value	\sum Softmax * Value (Attention layer output)
I	I * I  *  = 130	130	0.92	I 		
	I * study  *  = 50	50	0.05	study 		
	I * at  *  = 20	20	0.02	at 		
	I * school  *  = 10	0.01	0.01	school 		

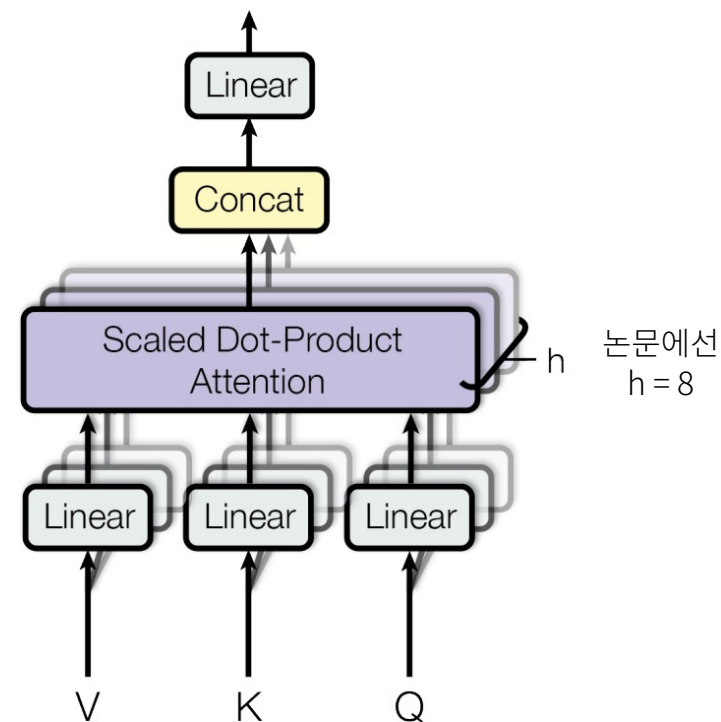
모든 단어에 대해서
동일한 작업 수행

Model Architecture – Multi-head Attention

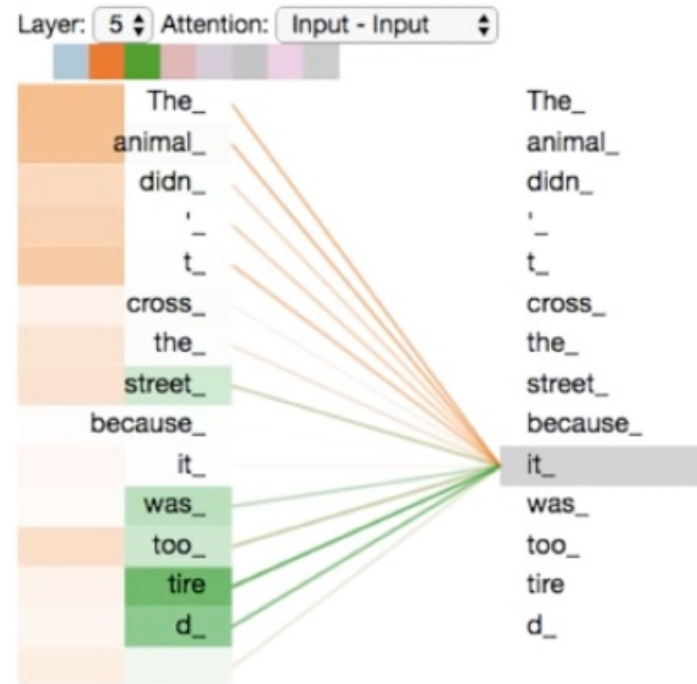
Scaled Dot-Product Attention



Multi-Head Attention



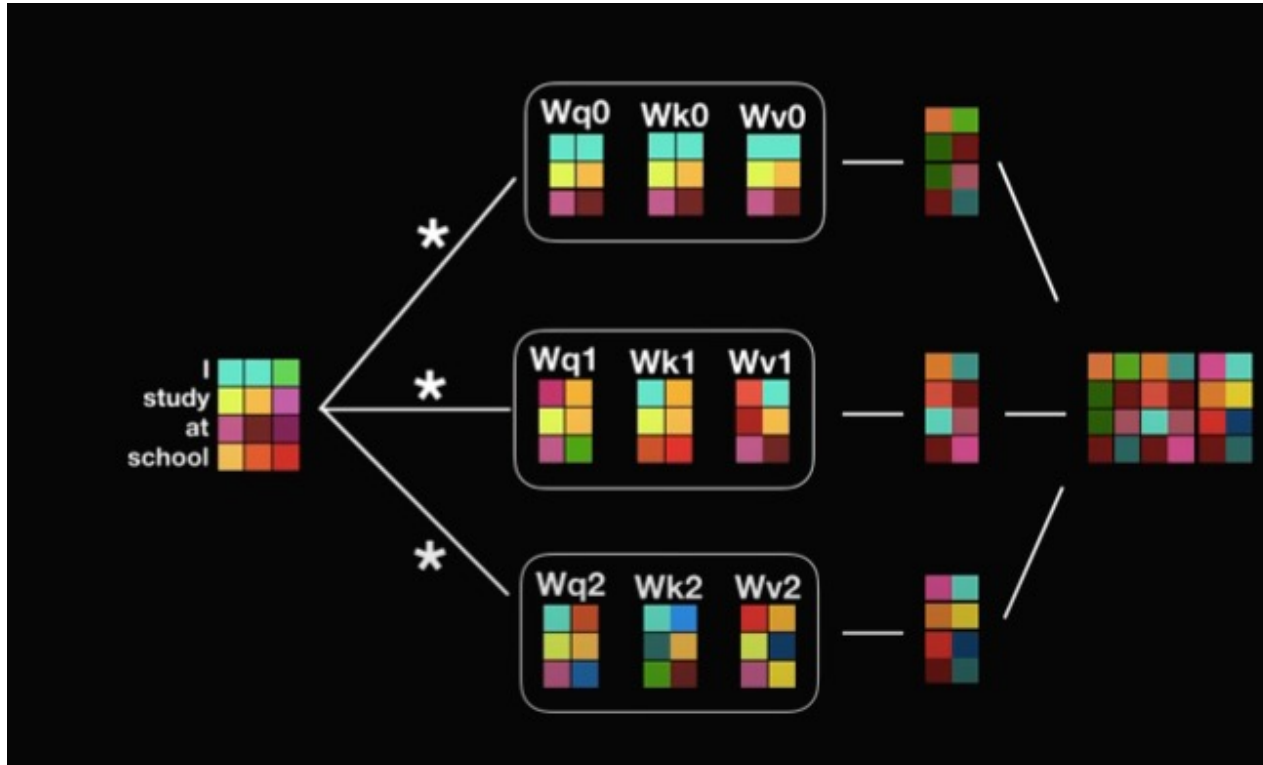
Model Architecture – Multi-head Attention



As we encode the word "it", one attention head is focusing most on "the animal", while another is focusing on "tired" -- in a sense, the model's representation of the word "it" bakes in some of the representation of both "animal" and "tired".

Model Architecture – Multi-head Attention

참고: [Tistory] [Transformer \(Attention is All You Need\) 논문 리뷰](#)



Self Attention을 8개의 병렬 구조로 확장

Model Architecture – Multi-head Attention

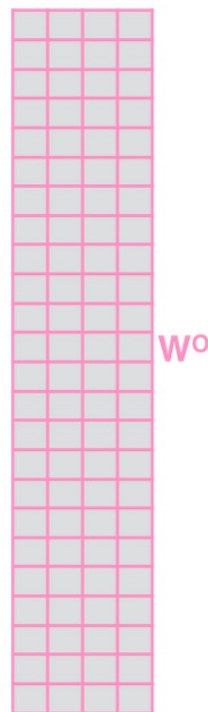
참고: [Tistory] [Transformer \(Attention is All You Need\) 논문 리뷰](#)

1) Concatenate all the attention heads

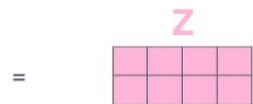


2) Multiply with a weight matrix W^O that was trained jointly with the model

X



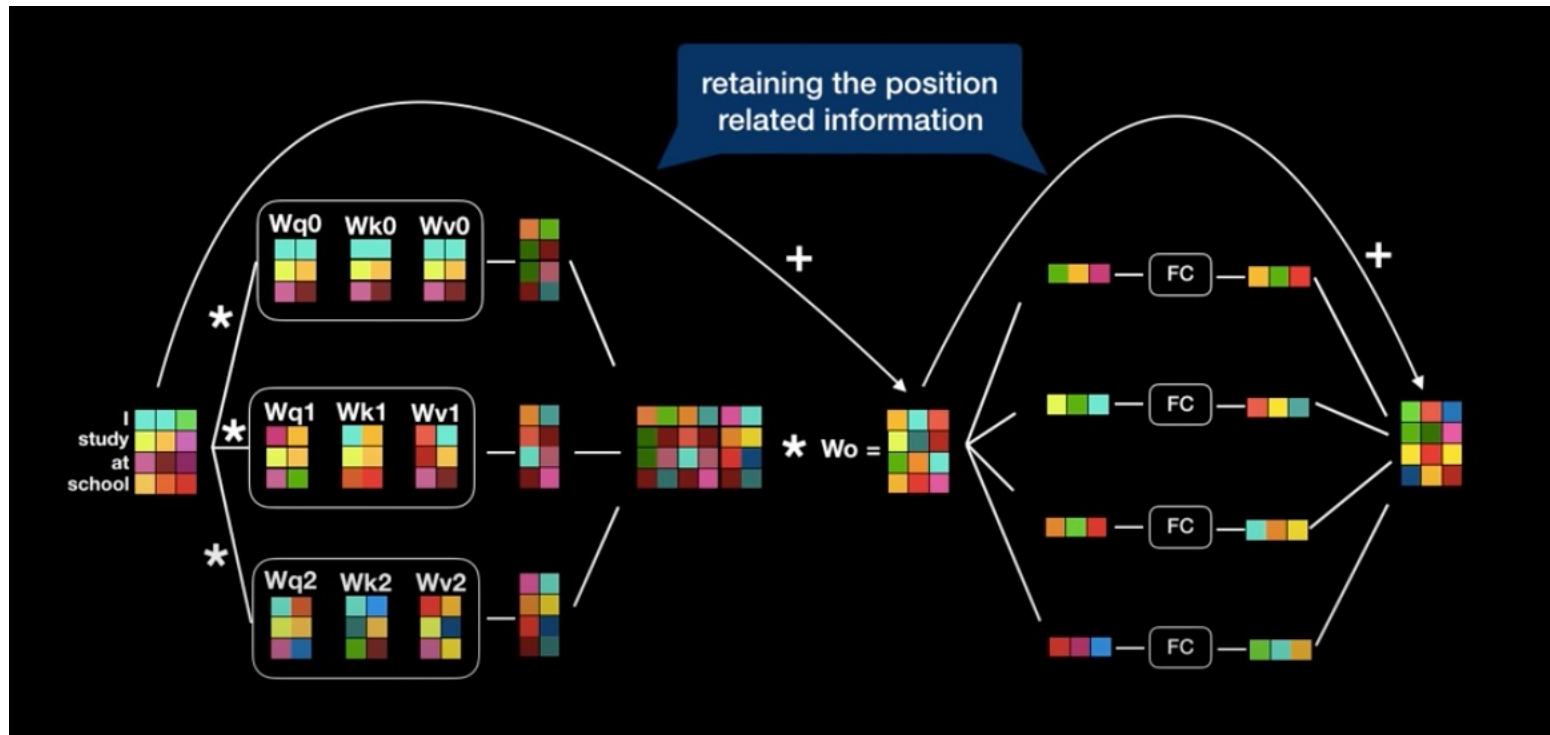
3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN



8개의 matrix를 1개로 합침

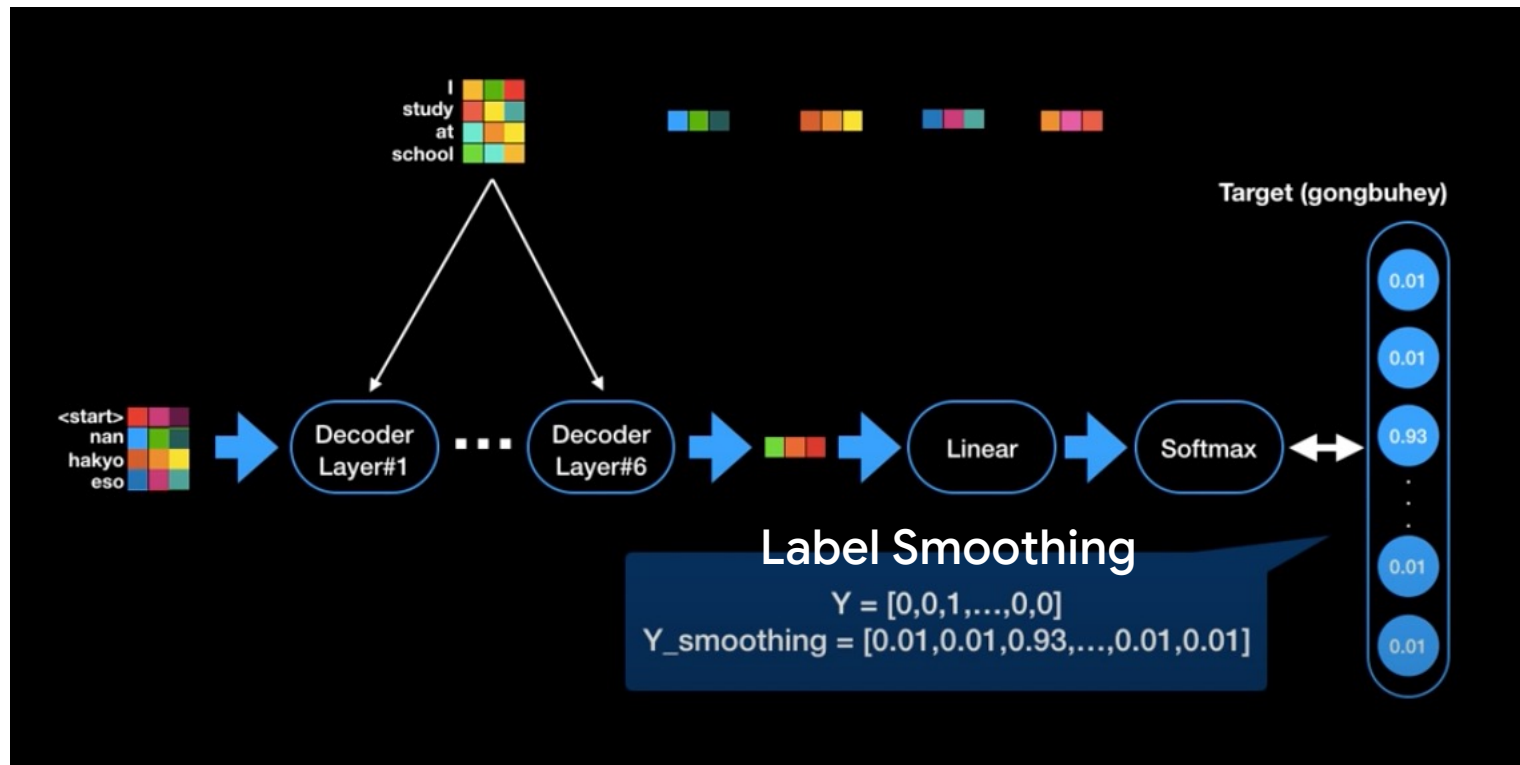
Model Architecture – Multi-head Attention

참고: [Tistory] [Transformer \(Attention is All You Need\) 논문 리뷰](#)



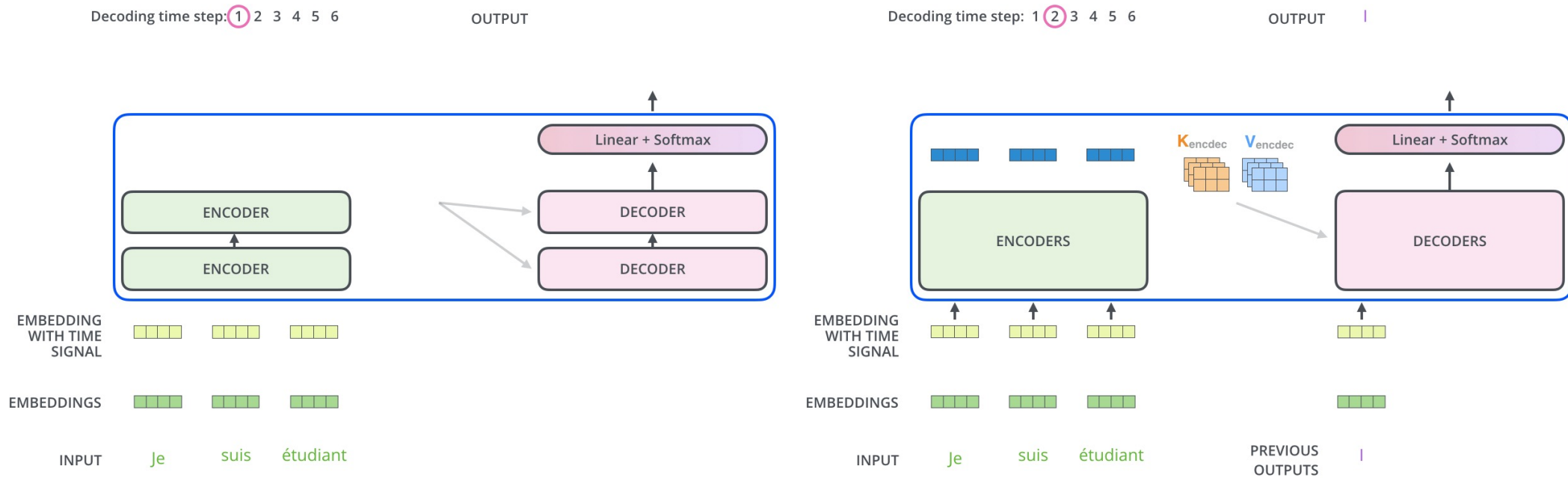
Model Architecture – Masked Multi-head Attention

추가 자료: [Blog] [Label Smoothing 이해하기](#)



Model Architecture – Masked Multi-head Attention

참고: [Tistory] [Transformer \(Attention is All You Need\) 논문 리뷰](#)



Model Architecture – Feed Forward NN

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Fully Connected Feed-Forward Layer

2개의 linear layer + ReLU

- 입출력 차원: 512
- 중간 차원: 2048

Why Self-Attention?

1. 각각의 레이어마다 계산 속도가 줄어든다.
2. 순환을 없앴으로써 병렬 처리가 가능하다.
3. Long-range dependency 처리가 용이하다.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

n = 단어의 개수, d = 표현 차원, k = 컨볼루션의 커널 크기, r = 제한된 이웃의 수(restricted)