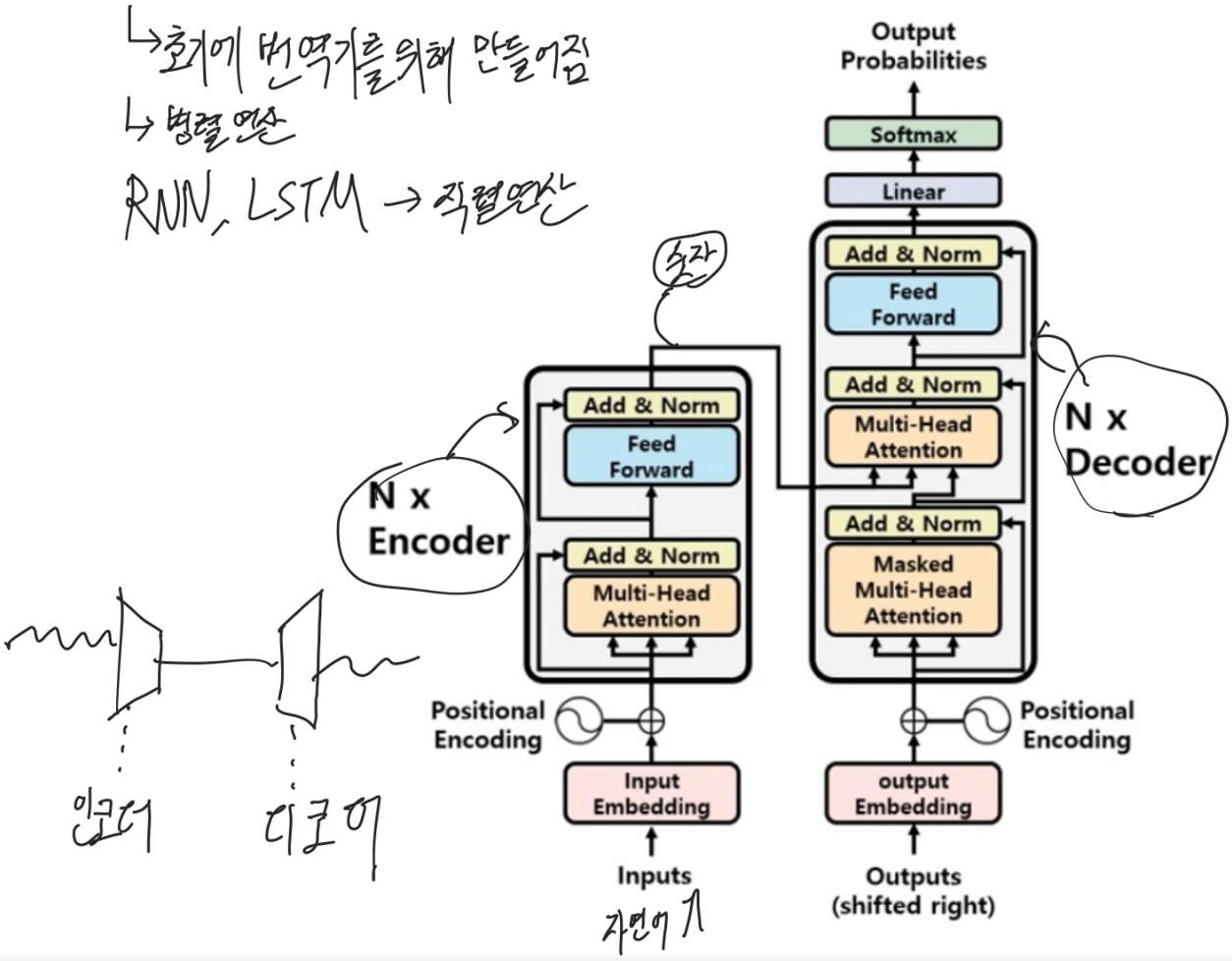
Transformer

↳ 최초에 번역기를 위해 만들어짐

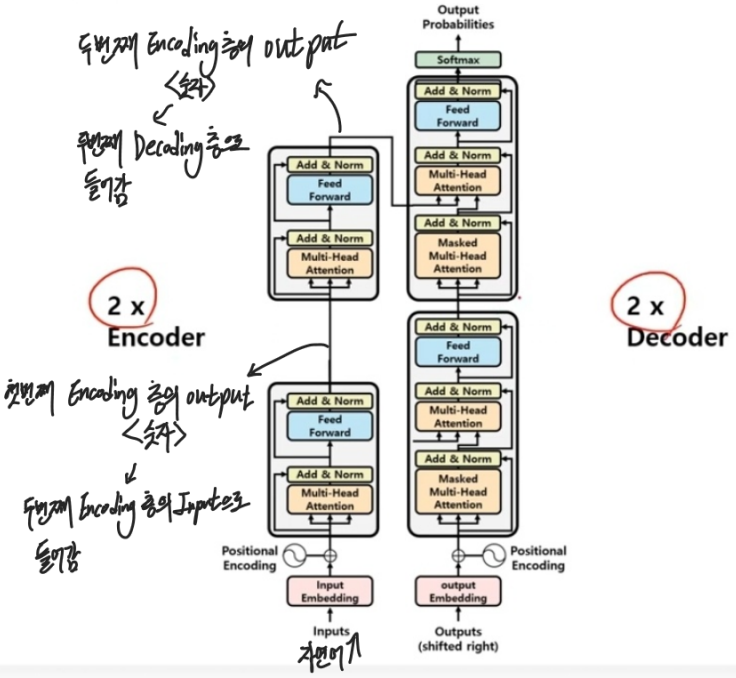
↳ 병렬연산

RNN, LSTM → 직렬연산



위에 있는 그림은 N=1일때 transformer 모델그림이고 아래의 그림은 N=2일때 transformer 모델의 그림이다.

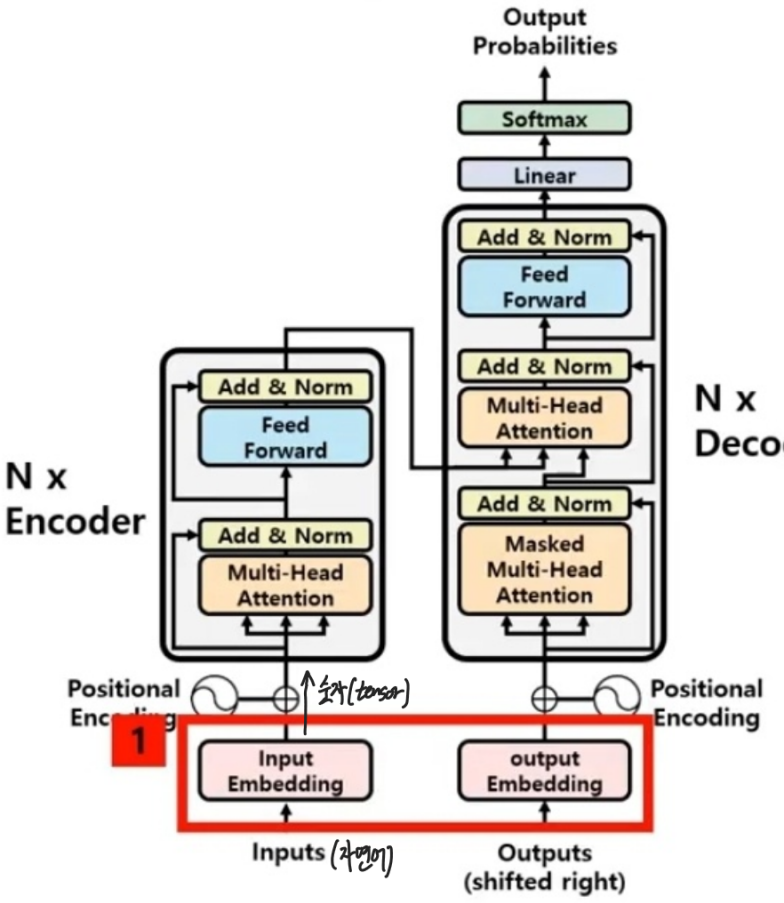
Transformer



⇓

즉, 마지막 끝에있는 Encoder, Decoder끼리만 연결된다.

Embedding



I am a good student

자연어

one-hot

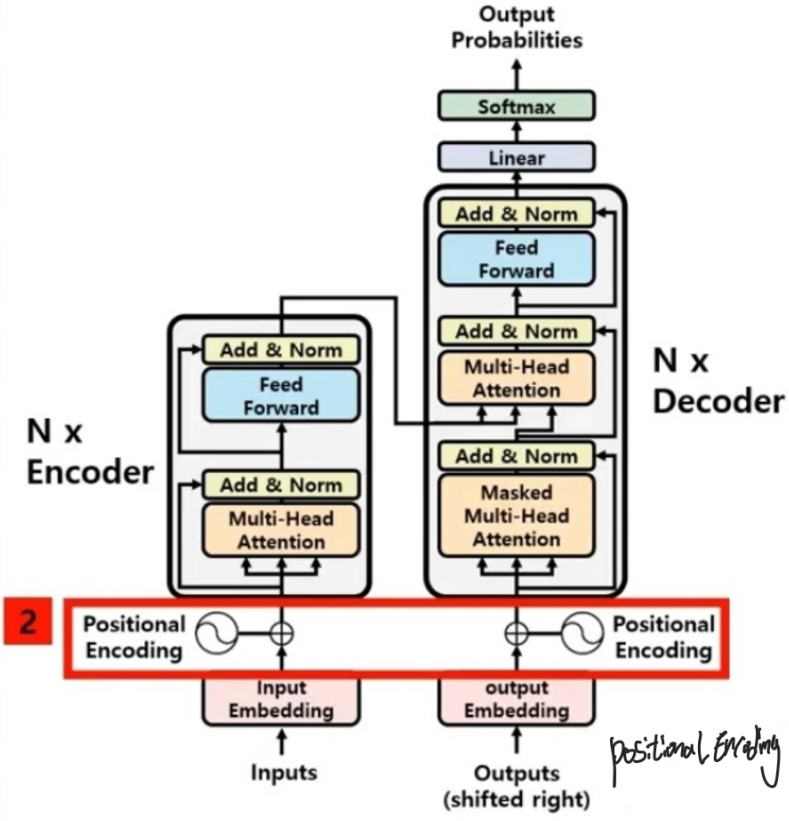
TFIDF

Embedding

tensor

7, 3, 1, 6, 8

Positional Encoding → 자연어에선 어순이 중요하기 때문에...



Tom ate the meat. ≠ The meat ate Tom.

Postional encoding vector

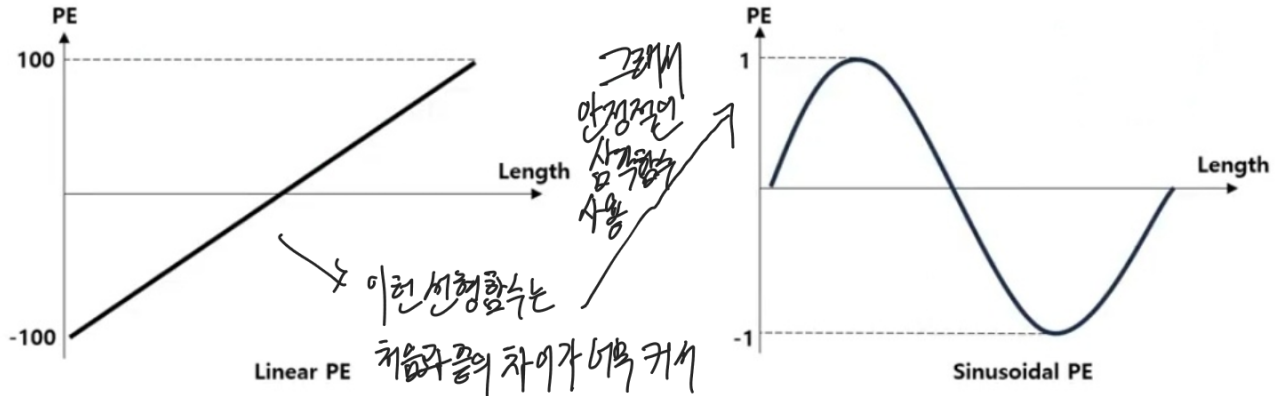
+

embedding vector

positional Encoding = Embeding Vector에 position 정보를 더한것

어떤 방식으로 position 정보를 줄것인가?

Positional Encoding



그래서 안정적인 삼각함수 사용

이런 선형함수는 처음과끝의 차이가 너무 커서 이런값들은 놀허버릴수 있다.

positional 인코딩값을 삼각함수사용

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

→ 왜 안정적이어야 하는가? ⇒ 정규화를 하기위해

- 범위가 -1~1로 안정적
- 주기함수로 글자수에 상관없음

Positional Encoding

임베딩 벡터의 위치(아래그림의 i값 참고)

Position of word | Location of embedding vector | Size of embedding vector

문장에서의 단어위치

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

→ 짝수인 sin

홀수인 cos사용

같은 i임

i 0 1 2 3 → i가 위치에따라 0, 1, 2, 3… 값을 가짐

1	1	1	1	pos = 0
	4			
1	1	1	0	pos = 1
1	1	0	0	pos = 2
sin		sin		
	cos		cos	

d_model = 4

임베딩 벡터의 크기

→ 왜사용? → 정규화를 위해

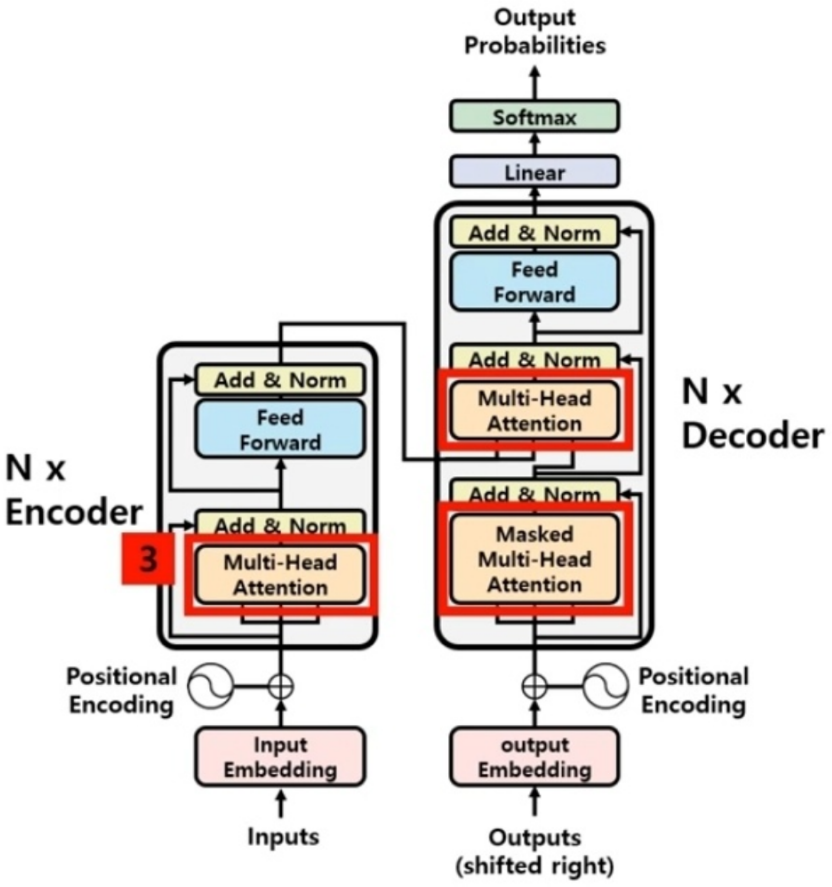
홀수인 cos, 짝수인 sin사용하는 이유 I am student

→ 값의 수를 늘리기 위해서

→ 더 긴 문장도 겹치지 않기 위해서

→ sin이나 cos만 단독으로 사용했을때 문장의 길이가 2π(약 7)를 넘어가면 위치 값이 겹치지만 혼용해서 사용하면 이를 방지할 수 있다.

Multi-Head Attention



Attention

↓

Multi-head Attention을 알기위해선 Attention을 알아야한다.

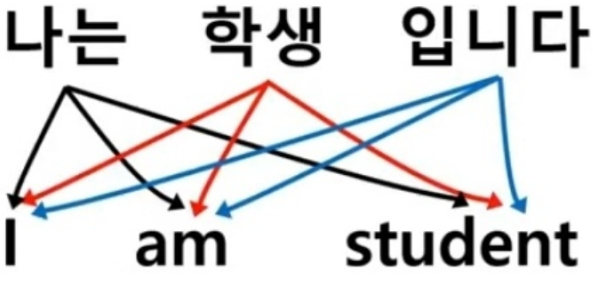
Attention

The animal didn't cross the street because it was too tired .

The animal didn't cross the street because it was too tired

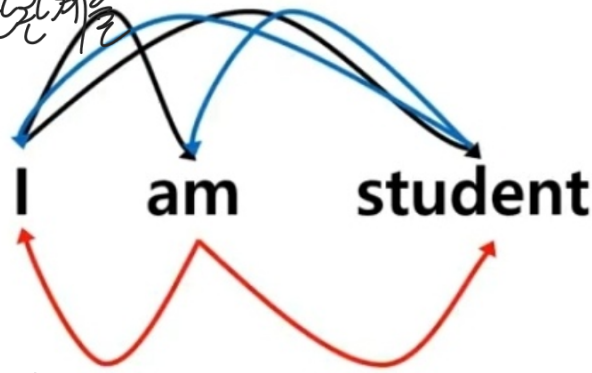
→ Attention의 등장

글의 문맥이해가 필요



<Attention>

문장과문장의 단어들끼리 상관관계를 추출



<Self-Attention>

자기 자신에게 Attention을 취해서 각 단어들 간의 상관관계 추론

Attention

고객아이디	고객이름	나이	등급	직업	적립금
apple	김현준	20	gold	학생	1000
banana	정소화	25	vip	간호사	2500
carrot	원유선	28	gold	교사	4500
orange	정지영	22	silver	학생	0

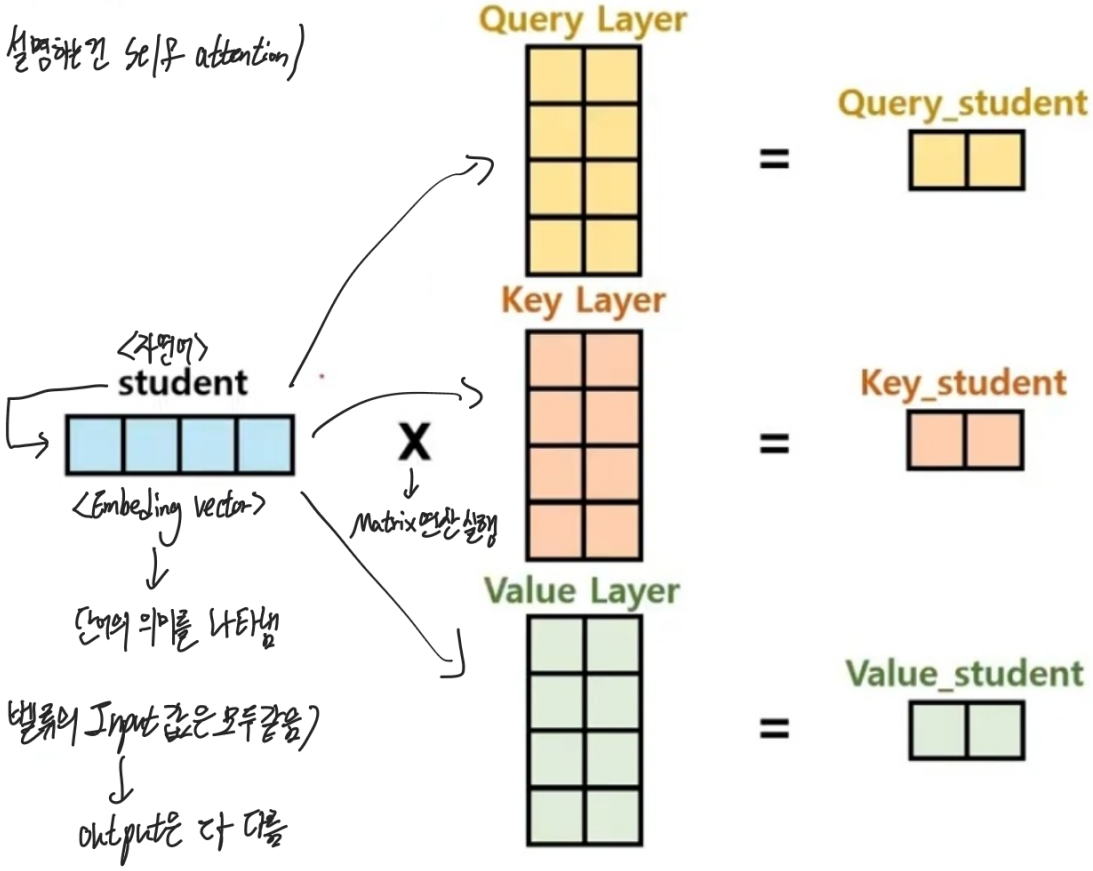
Query

Key

Value

Attention

(이 그림에서 설명해준 Self-attention)



단어의 의미를 나타냄

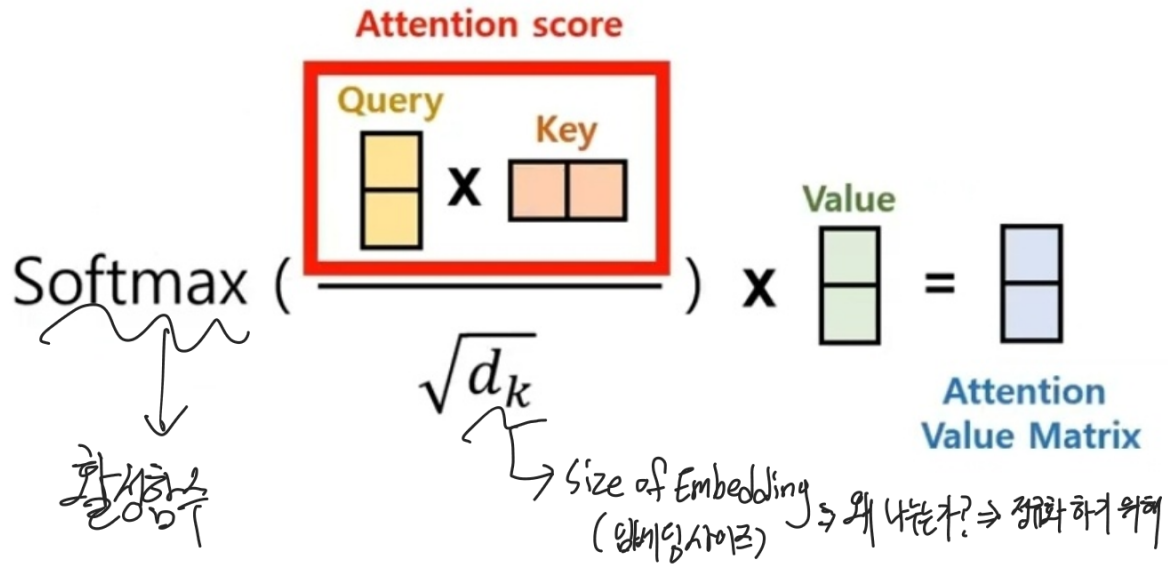
(쿼리, 키, 밸류의 Input값은 모두같음)

↓

output은 다 다름

Attention

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



Attention

$$\frac{QK^T}{\sqrt{d_k}} = \frac{3}{\sqrt{4}} = 1.5$$

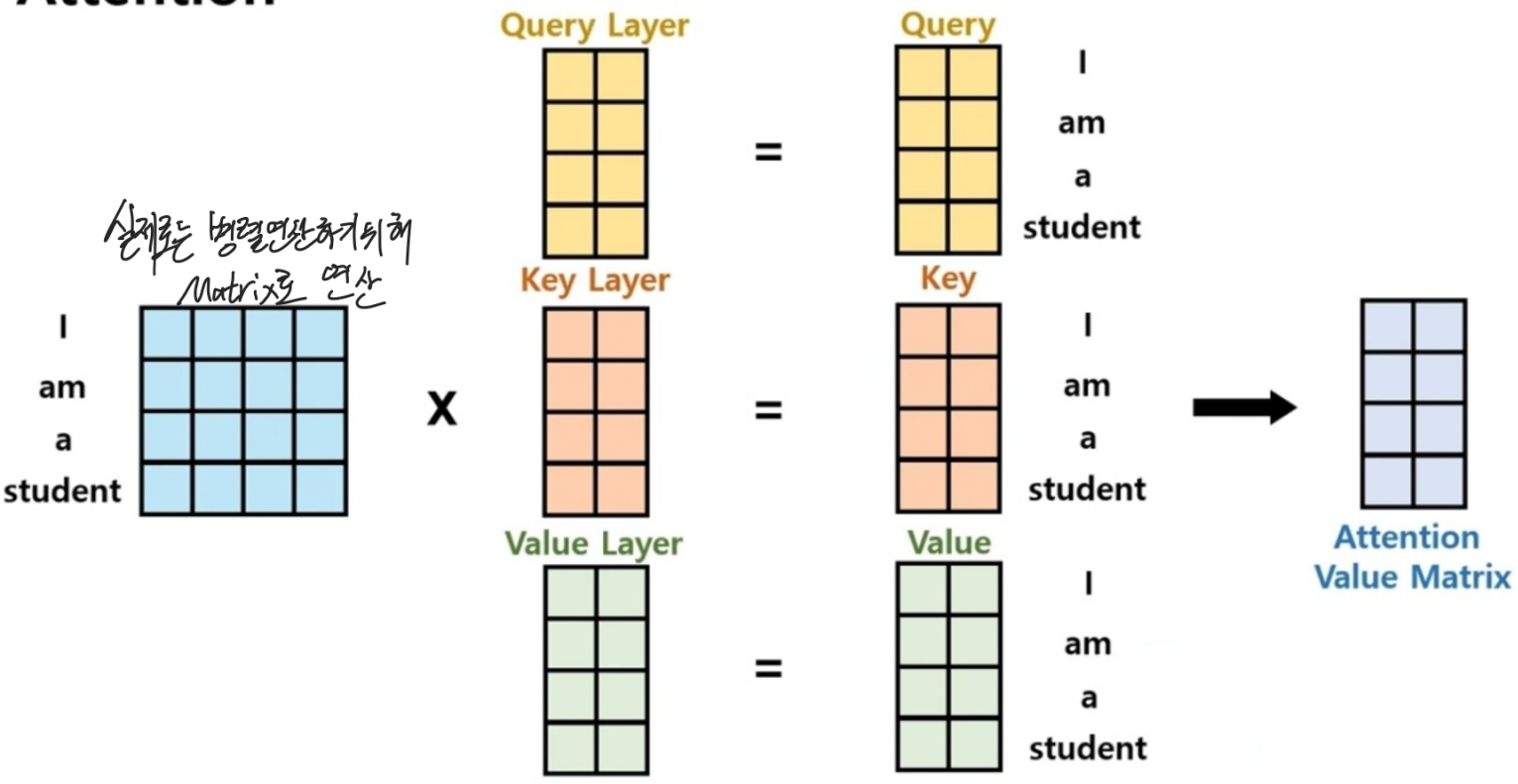
Self-Attention Map

	I	am	a	student
I	1.0	0.5	0.3	1.0
am	0.5	1.0	0.4	0.2
a	0.3	0.4	1.0	0.1
student	1.0	0.2	0.1	1.0

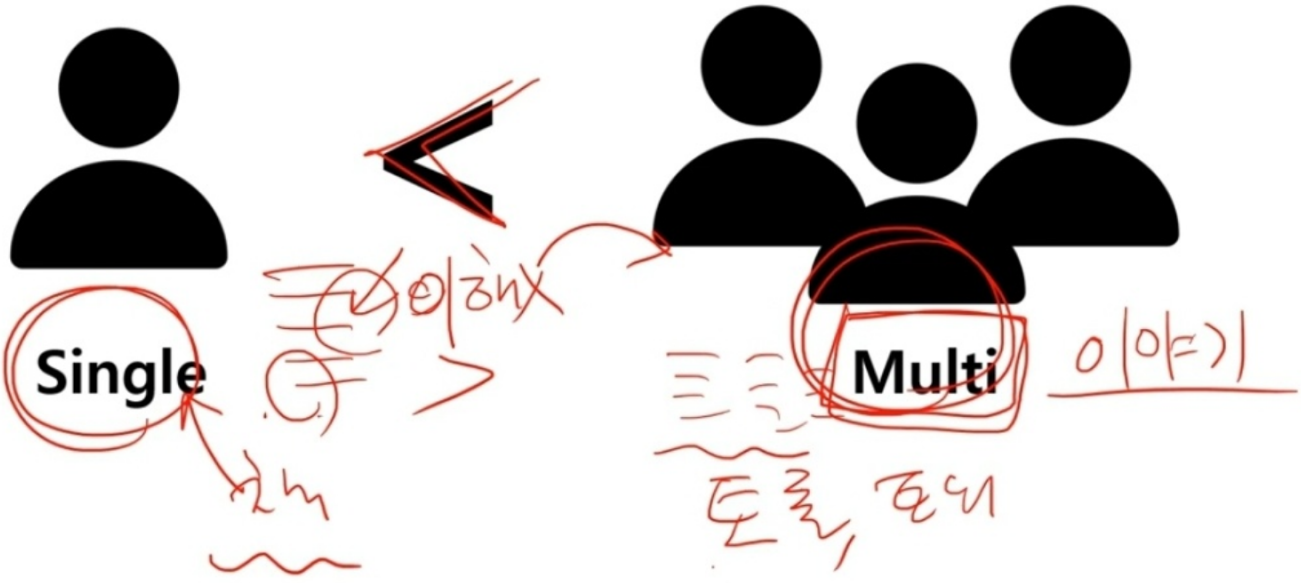
I = student라는 정보를 알려주는 유의미한 정보

당연히 그게 자신하고 연관이 있으니 무의미한 정보

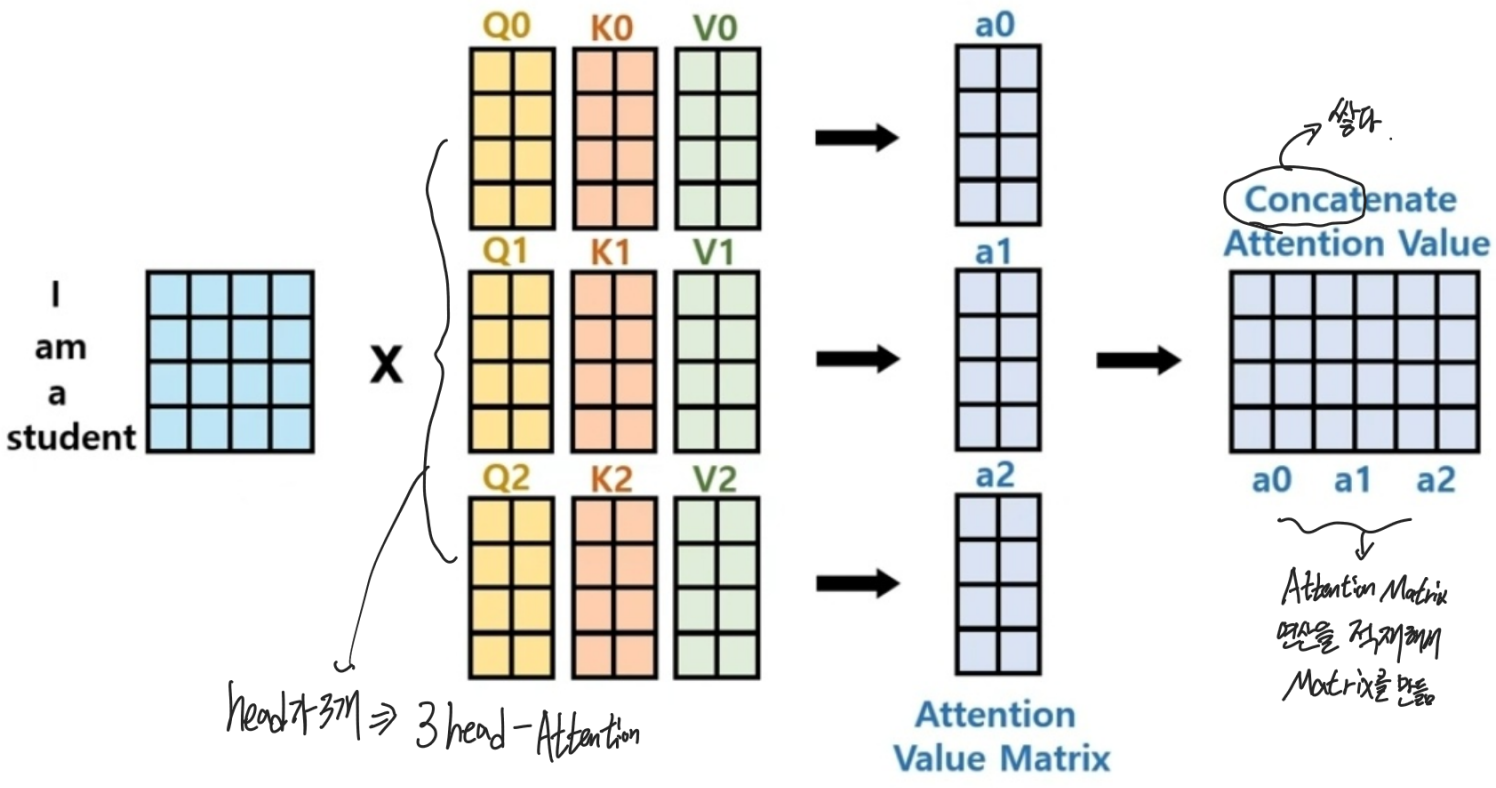
Attention



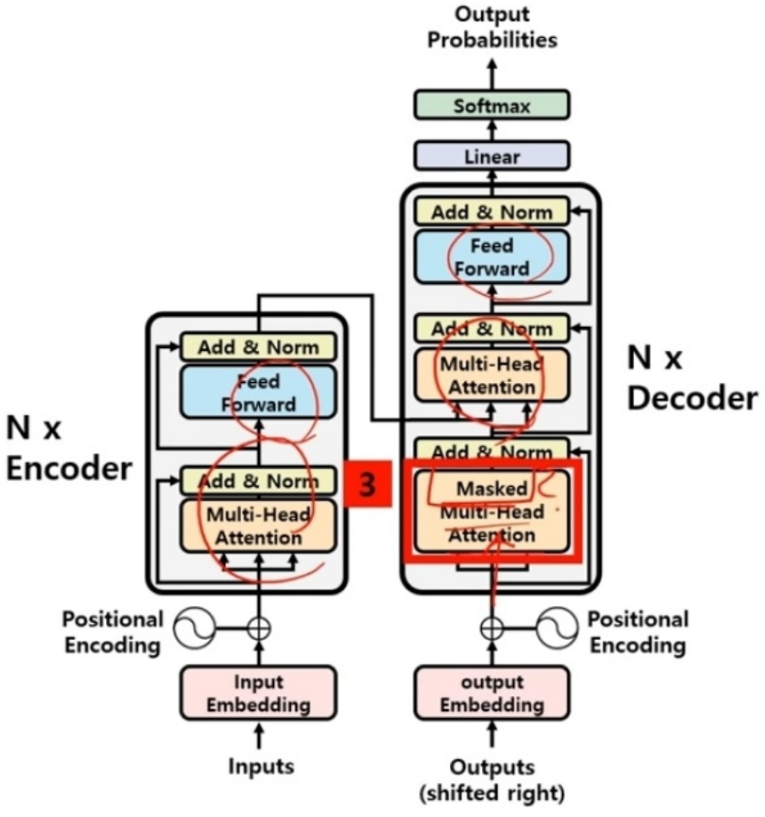
Multi-head Attention

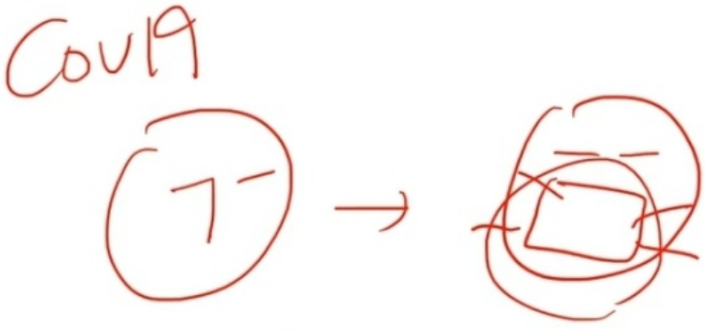


Multi-head Attention



Masked Multi-Head Attention

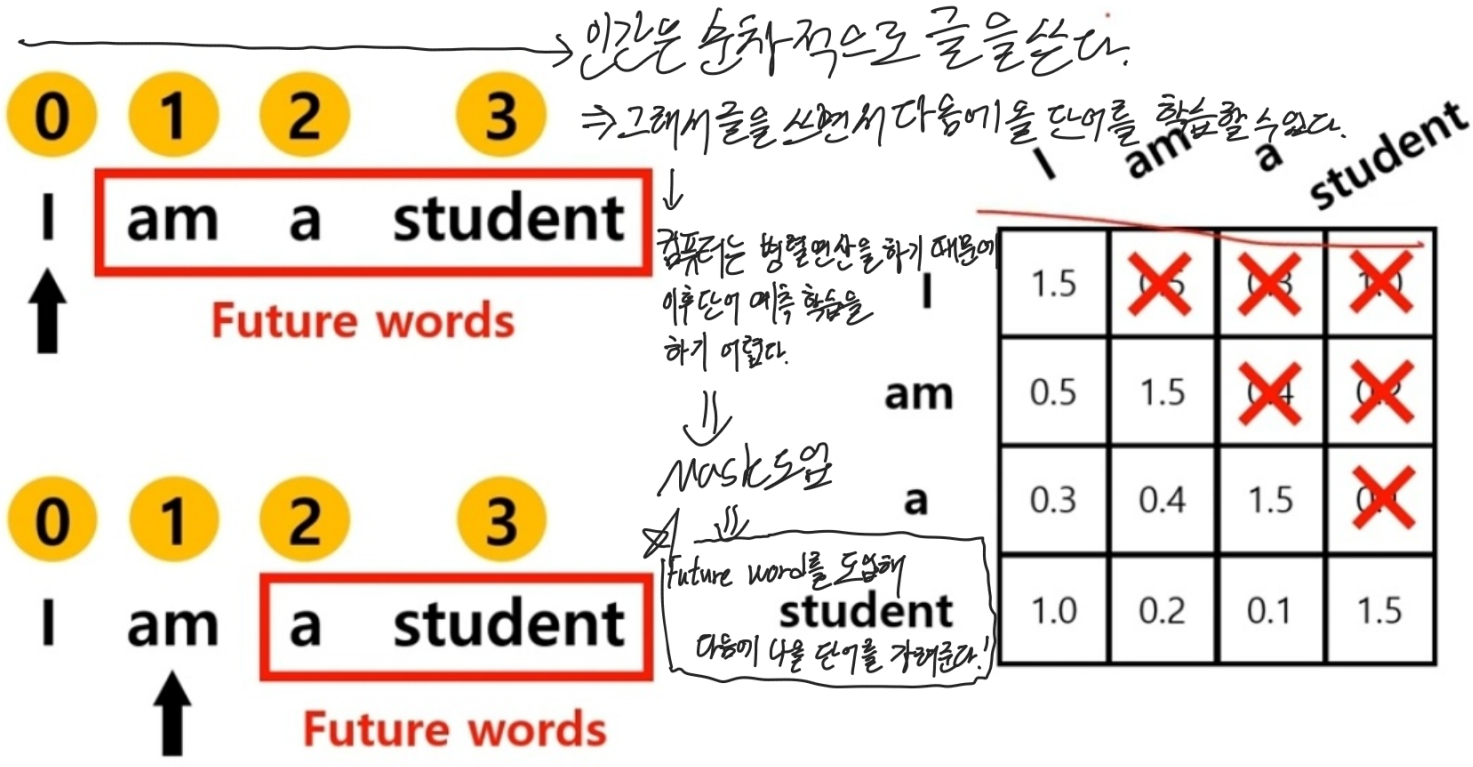




Masked ???

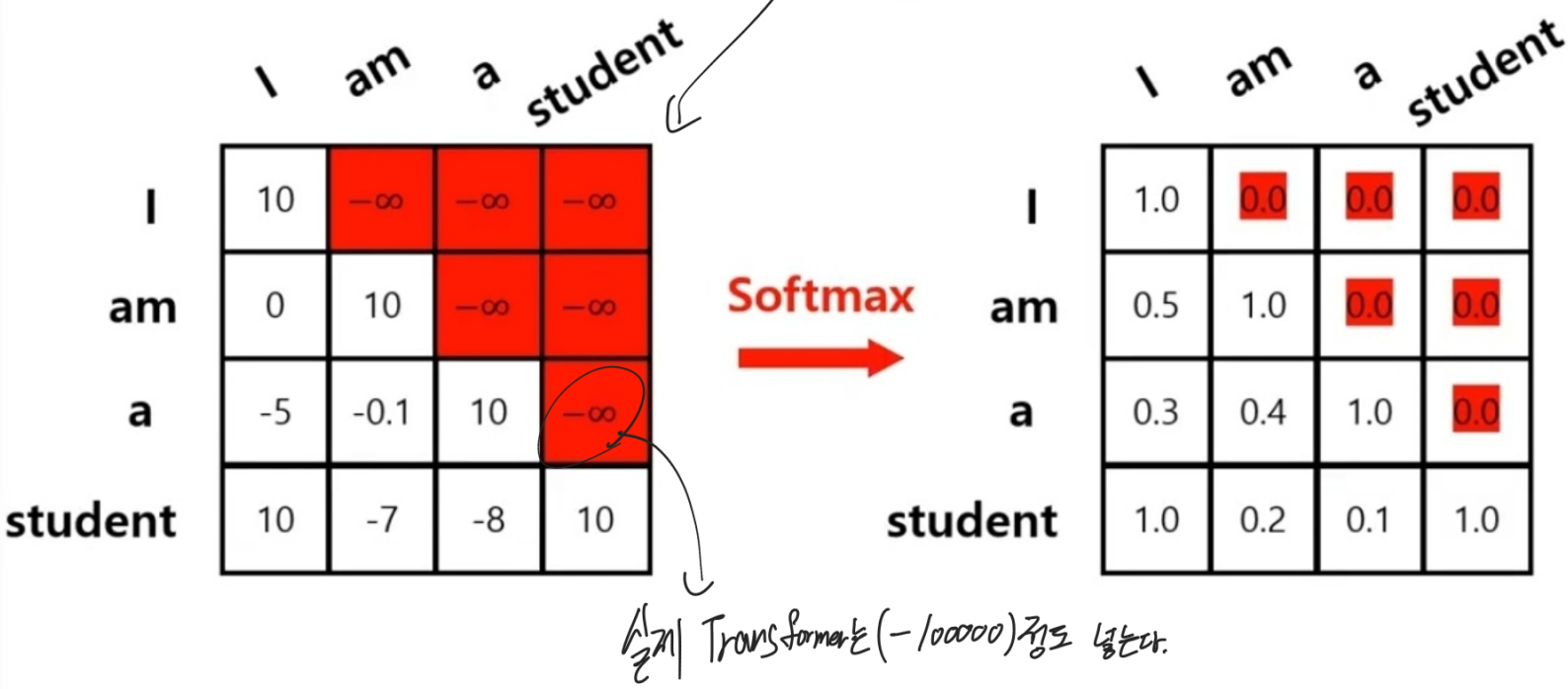
입력 → 가리고

Masked Multi-Head Attention

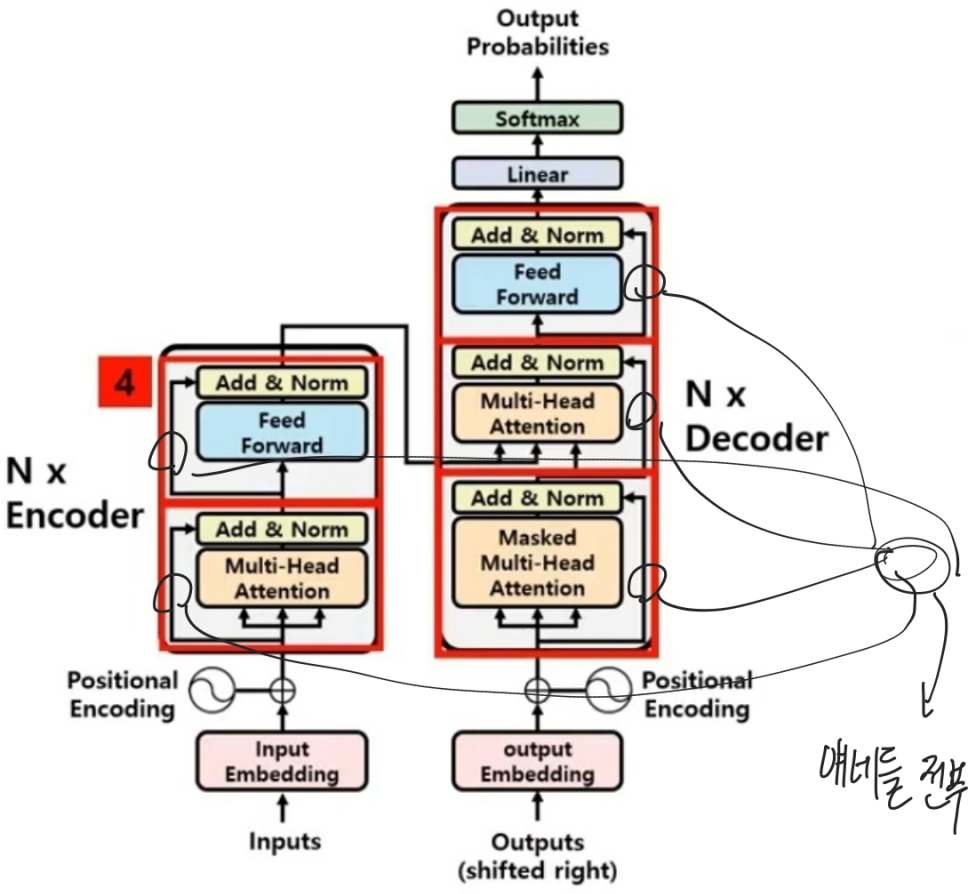


Masked Multi-Head Attention

↳ 어떤 방식으로 Future word를 가려줄까?



Residual connection

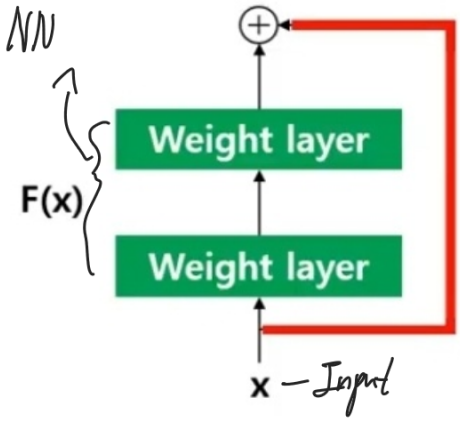


결과값에 Input값을 그대로 더하는 Residual connection

output Input

F(x) + x

NN



<Residual Connection>

에너를 전부

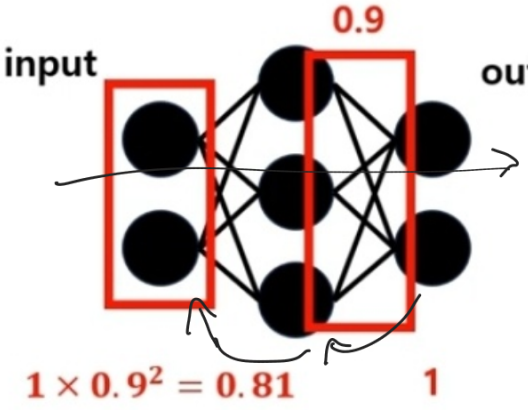
Vanishing gradient ⇒ Residual connection을 사용하는 이유

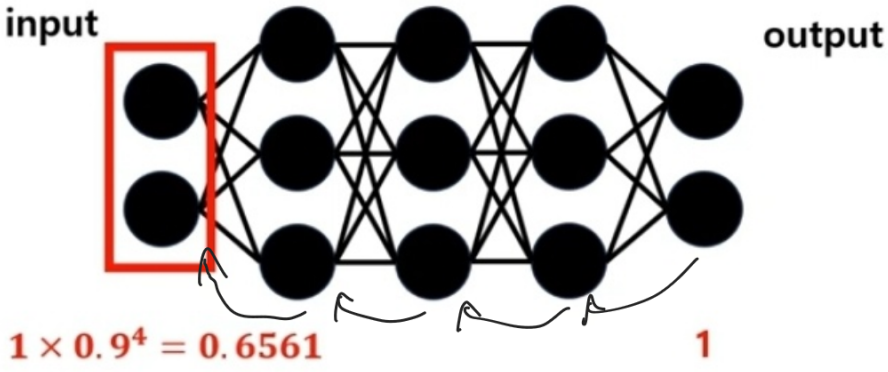
사라지다.

model depth가 증가함에 따라

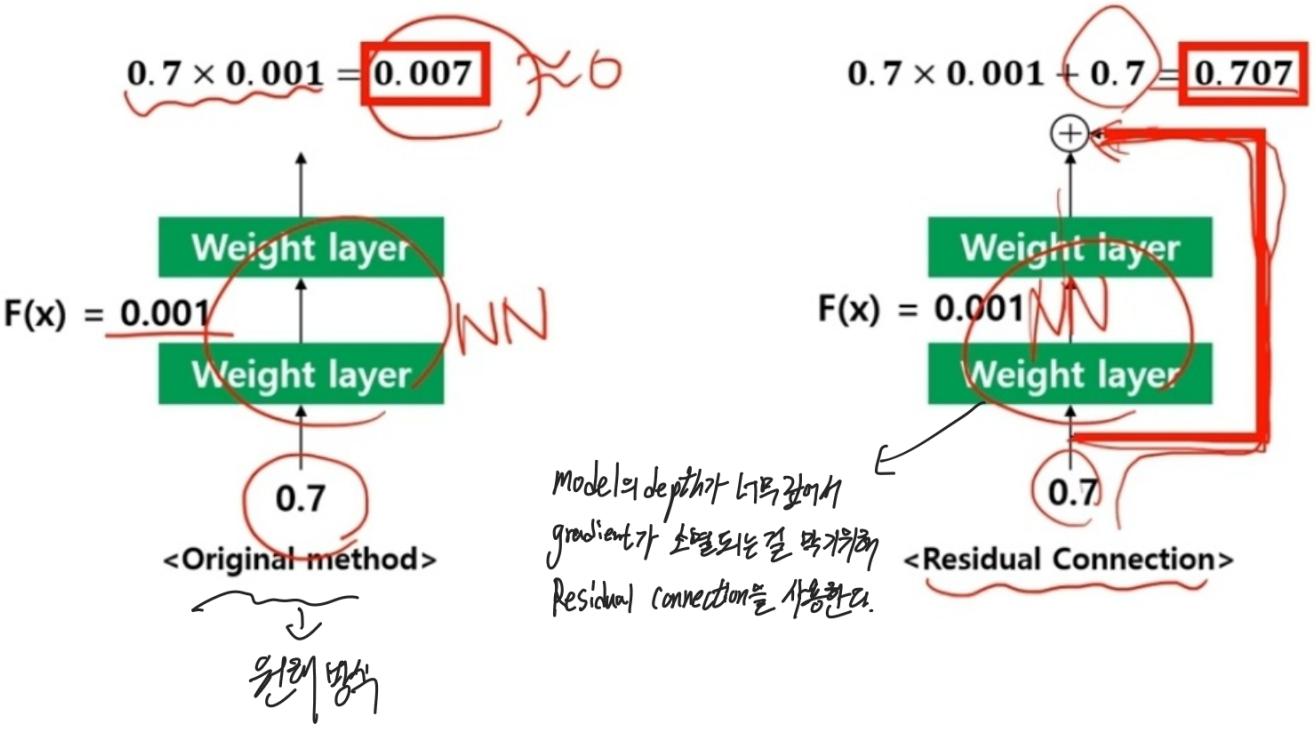
model의 Input gate가 0에 수렴하는 현상

back propagation





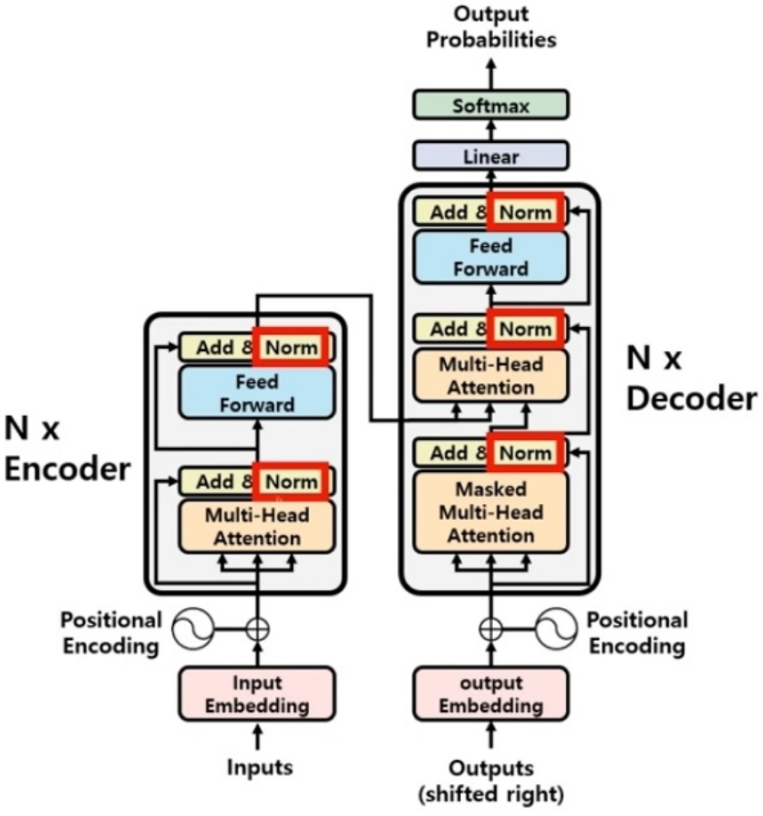
Residual connection



model의 depth가 너무 깊어서 gradient가 소멸되는 걸 막기위해 Residual connection을 사용한다.

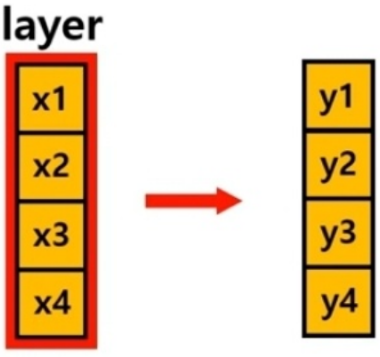
Layer normalization

AI에서 가장 중요한것 ⇒ 정규화 ⇒ 왜? ⇒ Data들의 상대적인 차이를 인식해서 학습하기위해



Layer normalization

layer



$$y = \frac{x - E[x]}{\sqrt{V[x] + \epsilon}} * \gamma + \beta$$

분산 — 0에 수렴하는 값

<Layer normalization>