Transformer

"Attention is all you need", Google, 2017, NeurIPS

조건희

목차

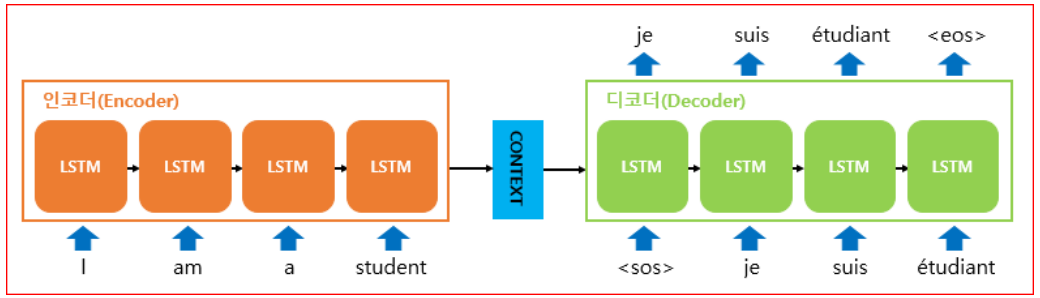
Transformer 개요

Transformer 개요

2017년 구글에서 발표한 “Attention is all you need”

신경망 기계번역(Neural Machine Translation, NMT) 문제

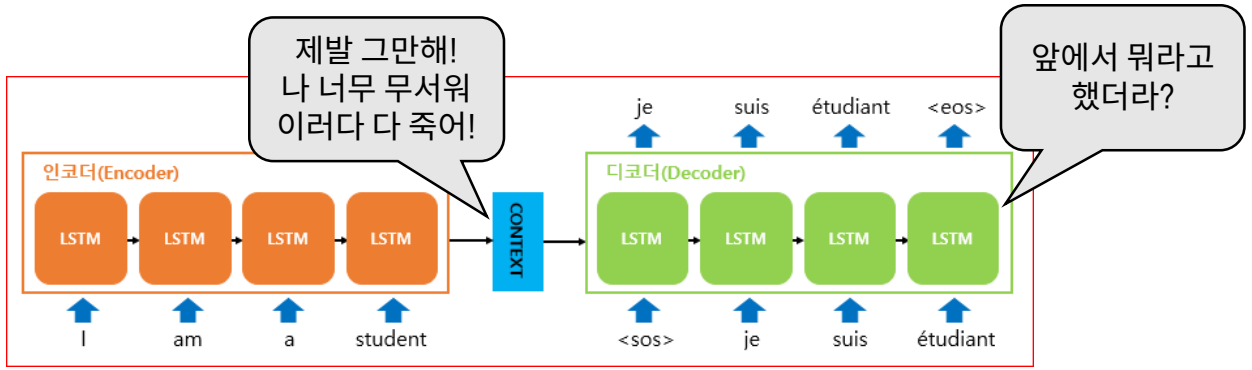
NMT 문제를 다룬 기존의 대표적인 연구



Seq2Seq

Transformer 개요

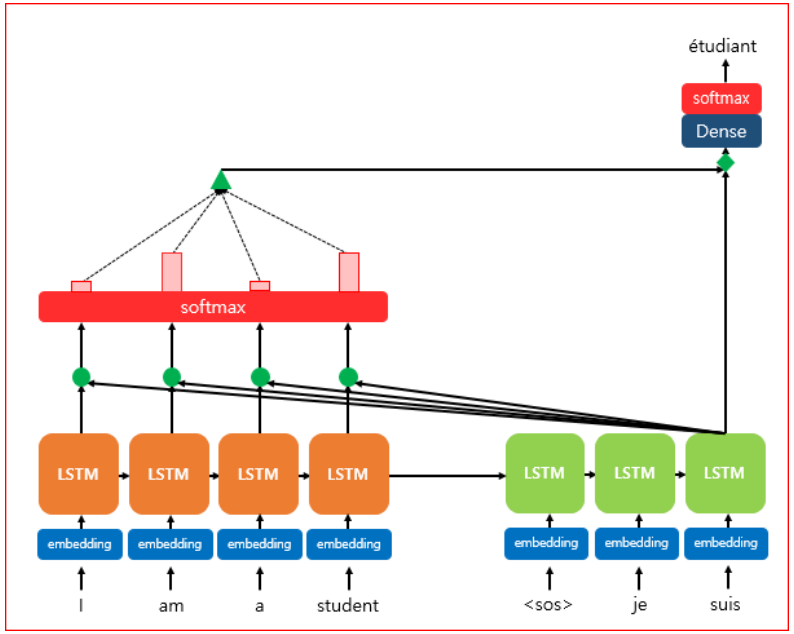
Seq2Seq 의 단점



1) 고정된 크기의 컨텍스트 벡터에 모든 정보를 압축하려다보니 정보 손실

2) RNN 기반 ➔ 그래디언트 소실(Gradient vanishing)

Transformer 개요



이런 seq2seq의 문제를 해결하기 위하여 등장한 기법

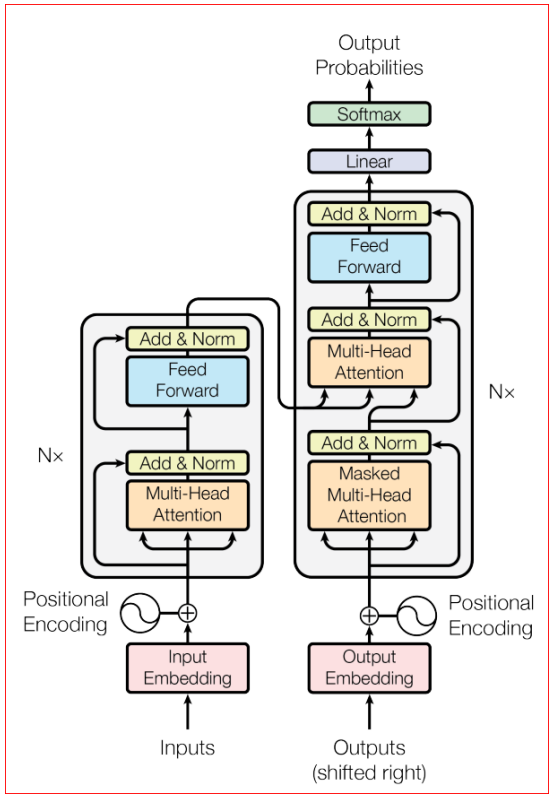
➔ 어텐션 메커니즘(Attention Mechanism)!

Transformer 개요

근데...

RNN 기반을 고집하지 않고 그냥 어텐션 메커니즘만으로 신경망을 설계해볼 순 없을까?

Transformer 개요



➔ “Attention is all you need!”

RNN LSTM 버리고, 어텐션만 사용해서 기계 번역해보자!

Transformer 개요

오늘 이 개념만큼은 이해하자

1. 어텐션 메커니즘
2. 1번만이라도 알고가자

Attention mechanism

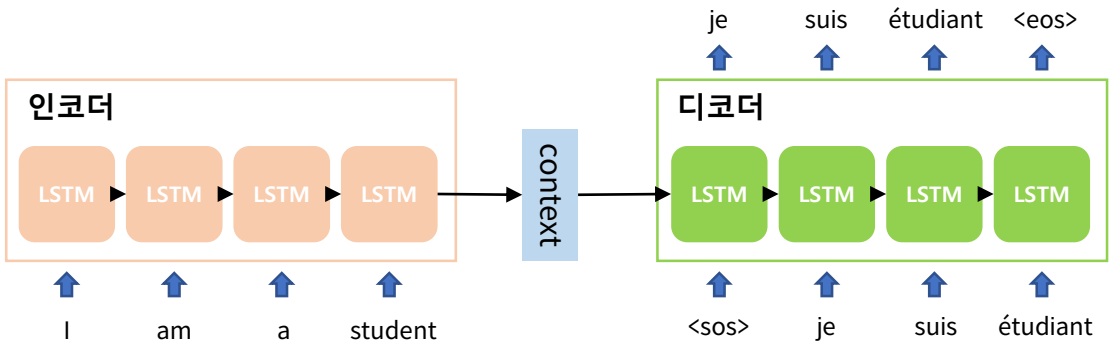
Attention Mechanism

어텐션?

디코더에서 출력 단어를 예측하는 매 시점(time step)마다,

인코더의 전체 문장을 한번 더 확인시켜주자!

단, **해당 시점에 예측해야 할 단어와 연관있는 입력 단어 부분**을 좀더 집중(attention)해서 알려주자!



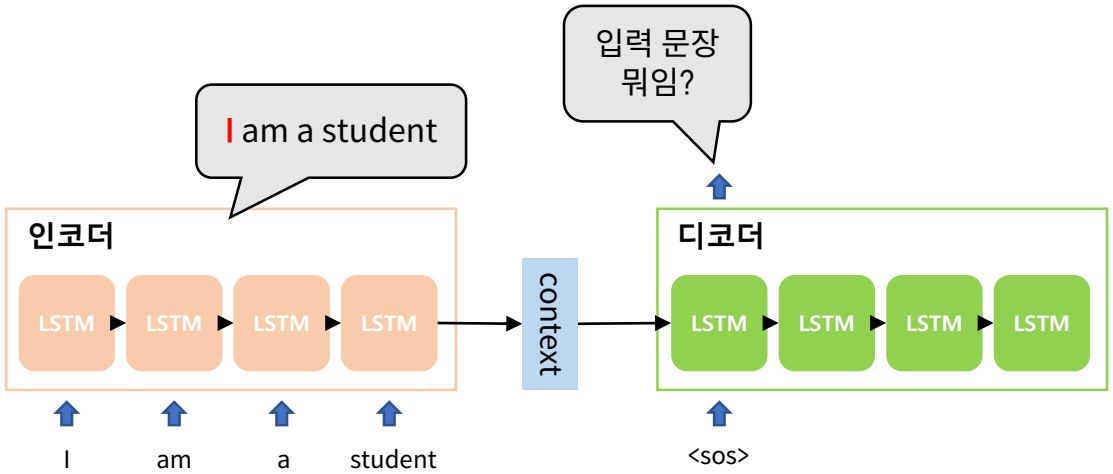
Attention Mechanism

어텐션?

디코더에서 출력 단어를 예측하는 매 시점(time step)마다,

인코더의 전체 문장을 한번 더 확인시켜주자!

단, **해당 시점에 예측해야 할 단어와 연관있는 입력 단어 부분**을 좀더 집중(attention)해서 알려주자!



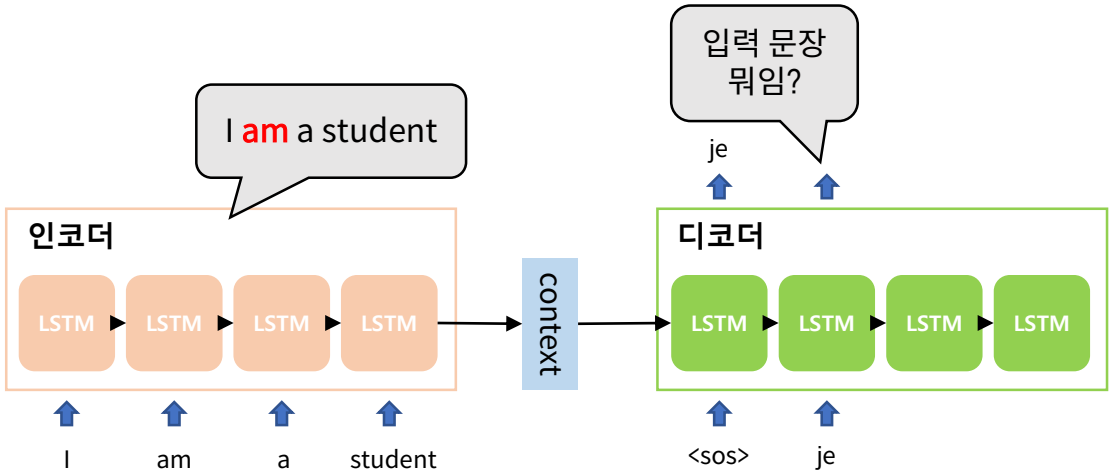
Attention Mechanism

어텐션?

디코더에서 출력 단어를 예측하는 매 시점(time step)마다,

인코더의 전체 문장을 한번 더 확인시켜주자!

단, **해당 시점에 예측해야 할 단어와 연관있는 입력 단어 부분**을 좀더 집중(attention)해서 알려주자!



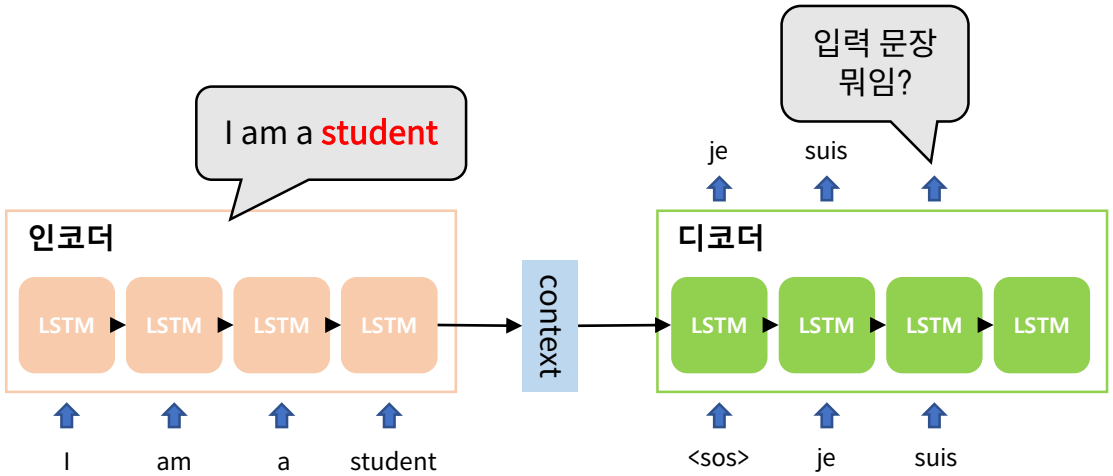
Attention Mechanism

어텐션?

디코더에서 출력 단어를 예측하는 매 시점(time step)마다,

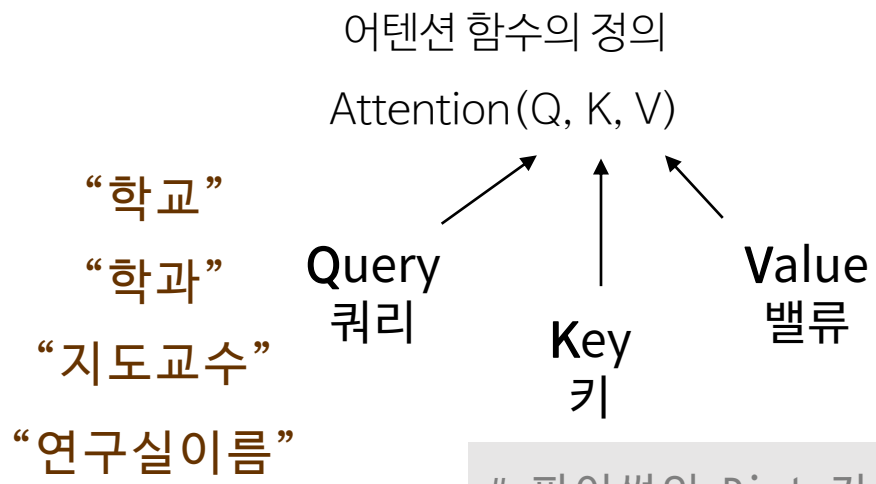
인코더의 전체 문장을 한번 더 확인시켜주자!

단, **해당 시점에 예측해야 할 단어와 연관있는 입력 단어 부분**을 좀더 집중(attention)해서 알려주자!



Attention Mechanism

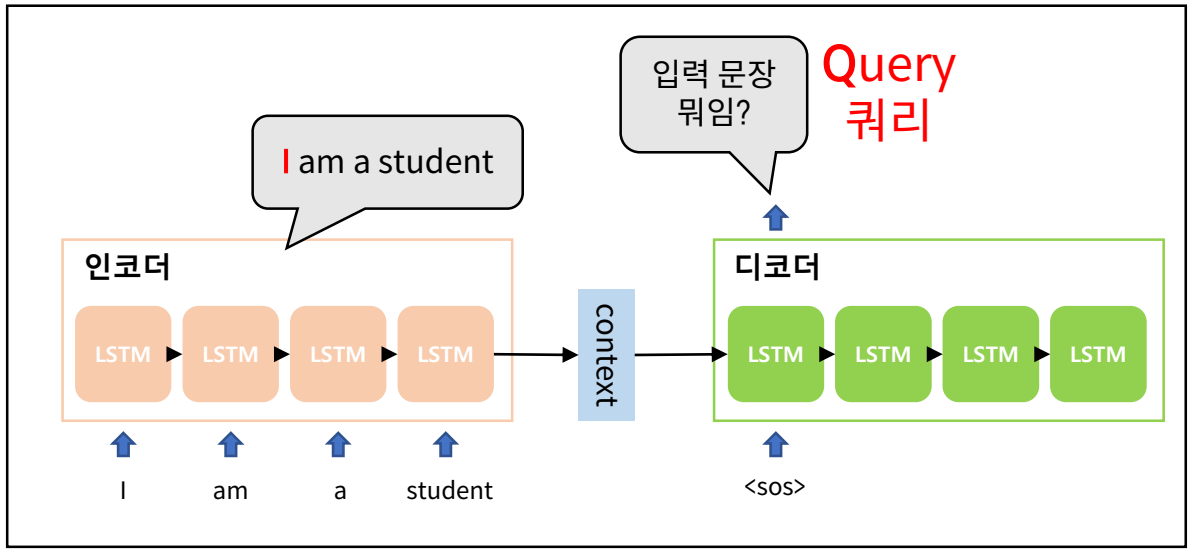
그래서 어텐션을 어떻게 함?

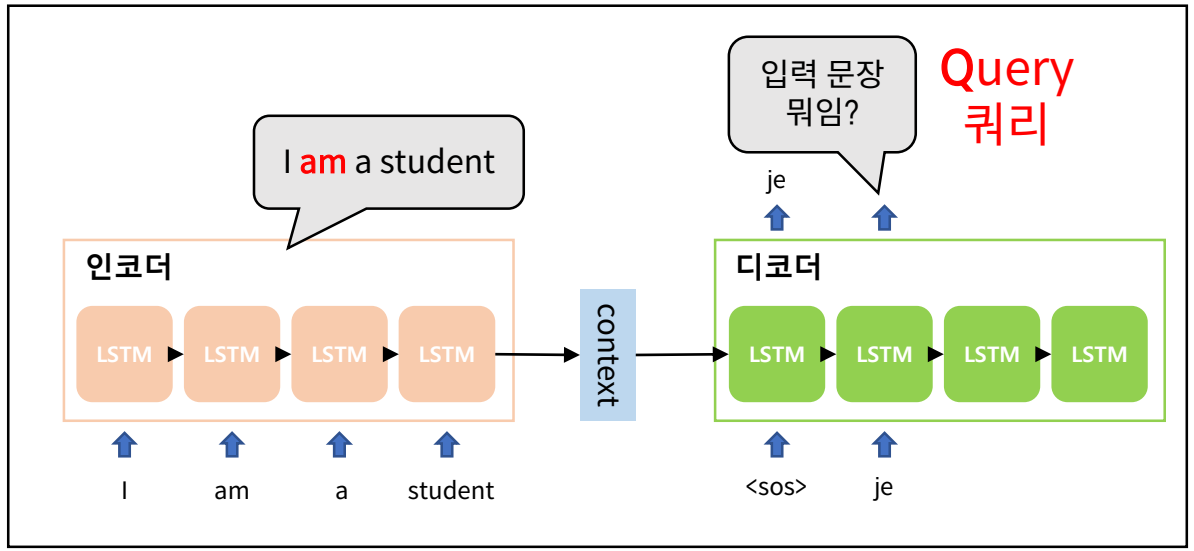


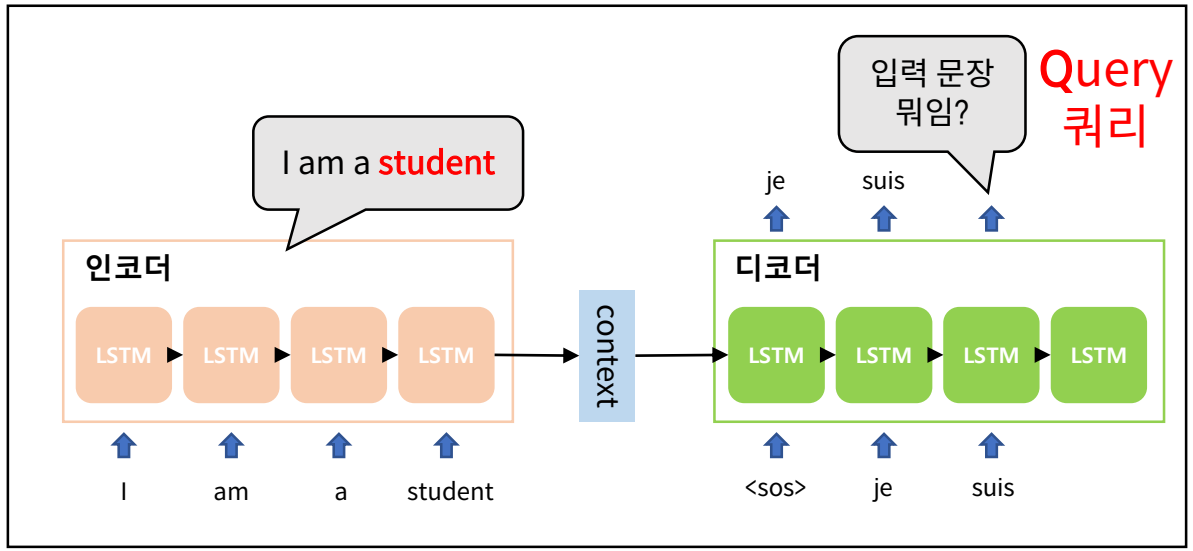

```
# 파이썬의 Dict 자료형
# 키(Key) : 값(Value) 형식

dict = {“학교” : “한양대”, “지도교수” : “최용석”}
```

Attention Mechanism







Attention Mechanism

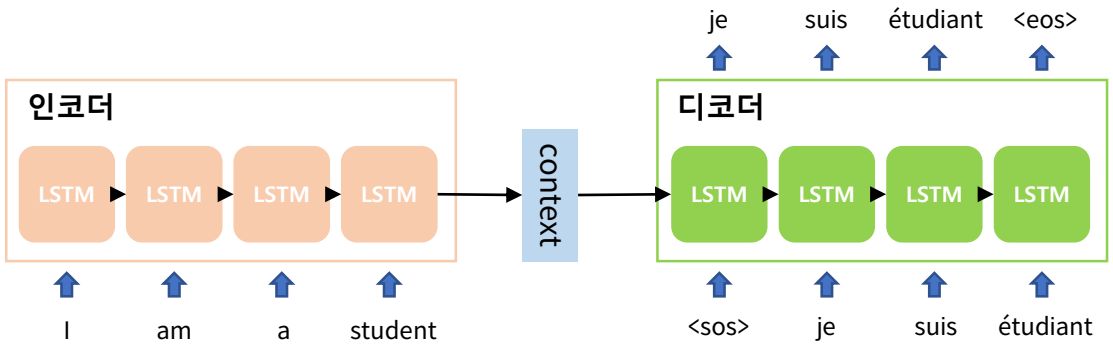
Attention(Q, K, V) = Attention value

주어진 쿼리(Q)에 대해서 모든 키(K)와의 유사도를 구하고,
밸류(V)에 각각의 유사도를 반영하여 모두 더해주는 함수

그 함수의 결과를 “Attention value” 라고 부르자

Attention Mechanism

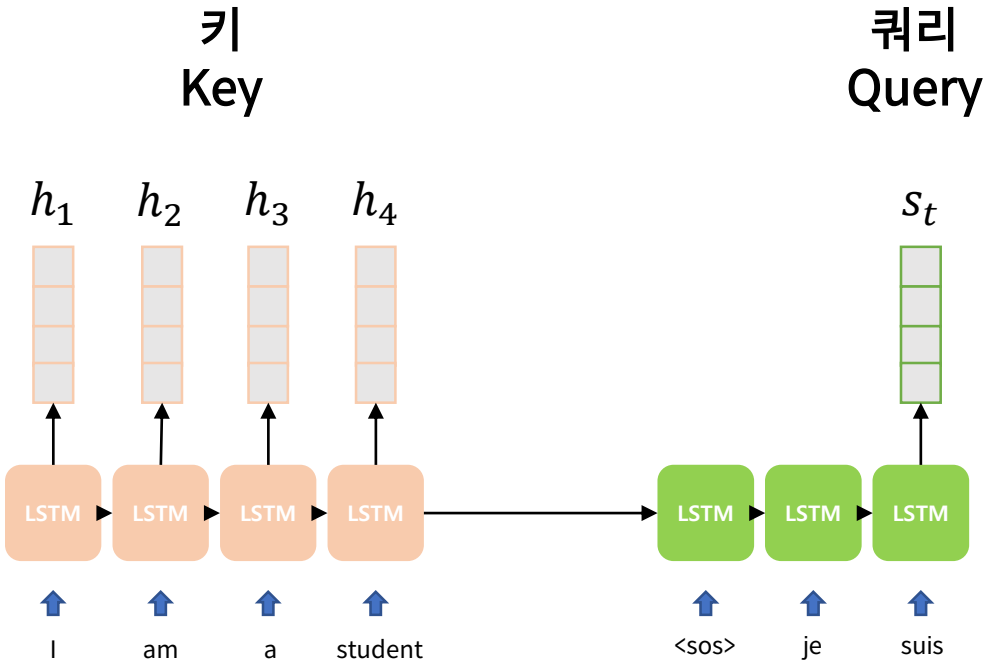
seq2seq + 어텐션 메커니즘



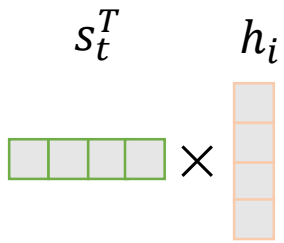

```
Q : t 시점의 디코더 셀의 hidden state
K : 모든 시점의 인코더 셀의 hidden state
V : 모든 시점의 인코더 셀의 hidden state
```

Attention Mechanism

Q : t 시점의 디코더 셀의 hidden state
K : 모든 시점의 인코더 셀의 hidden state
V : 모든 시점의 인코더 셀의 hidden state



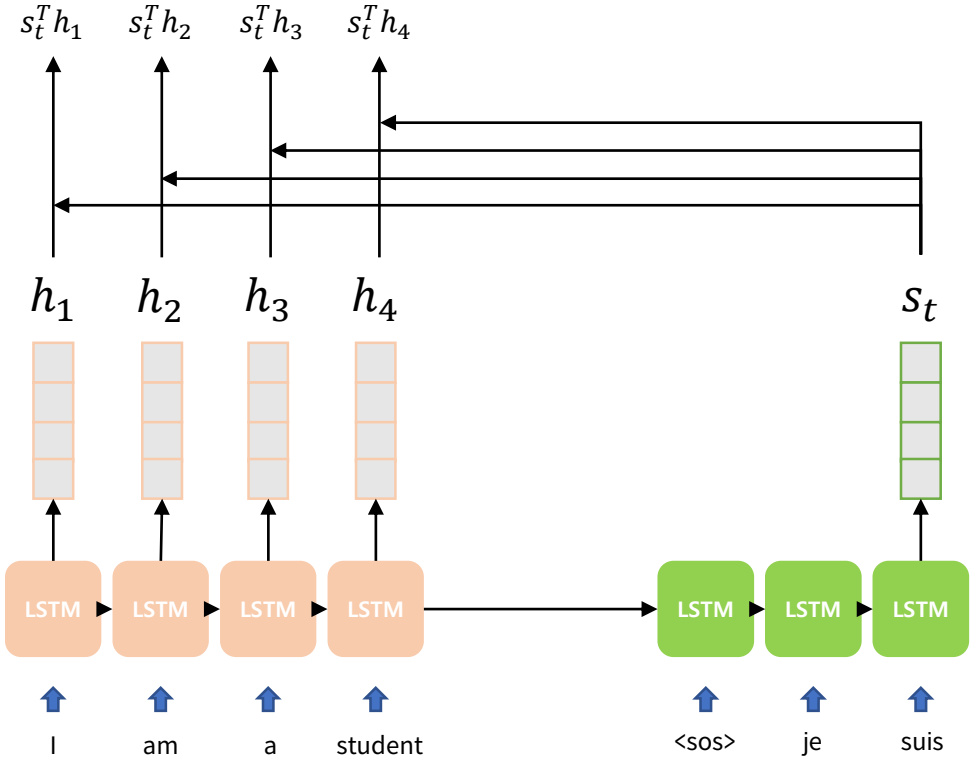
$$\mathbf{score}(\boldsymbol{s_t}, \boldsymbol{h_i}) = s_t \cdot h_i = s_t^T h_i$$



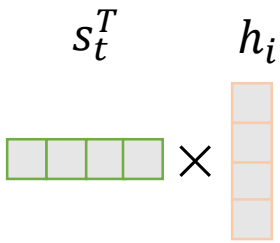
seq2seq + 어텐션 메커니즘

Attention Mechanism

Q : t 시점의 디코더 셀의 hidden state
K : 모든 시점의 인코더 셀의 hidden state
V : 모든 시점의 인코더 셀의 hidden state



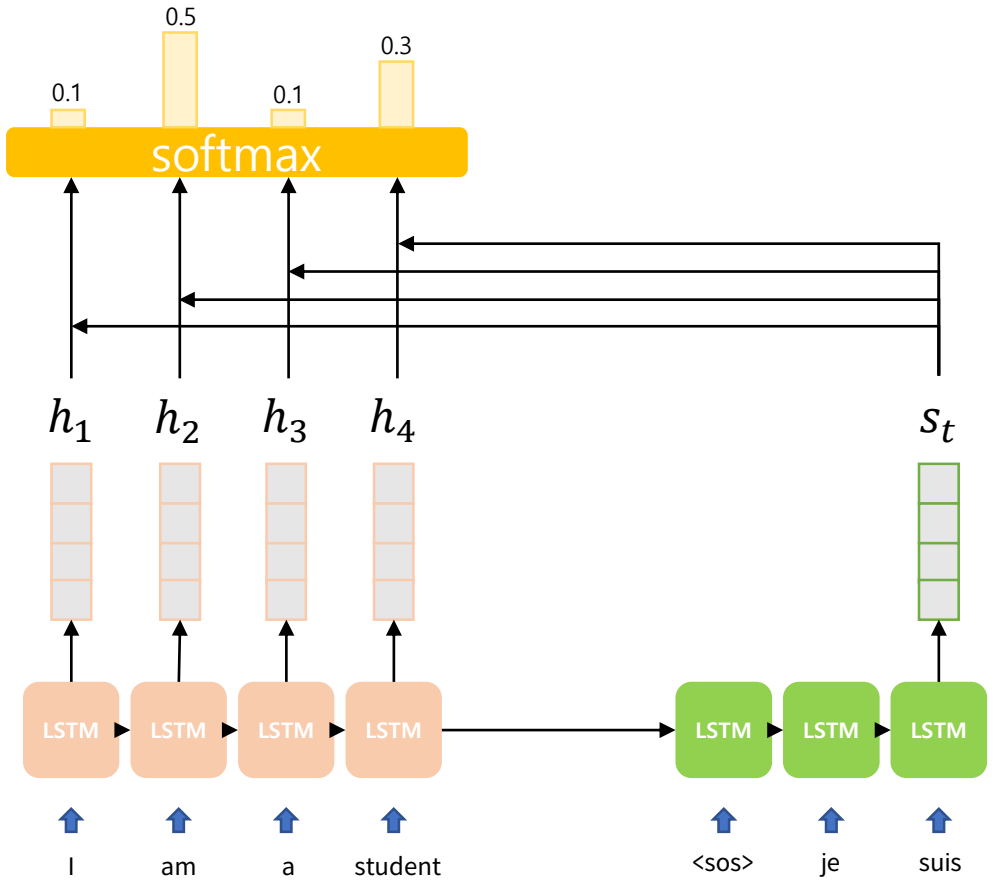
$$\mathbf{score}(\boldsymbol{s_t}, \boldsymbol{h_i}) = s_t \cdot h_i = s_t^T h_i$$



seq2seq + 어텐션 메커니즘

Attention Mechanism

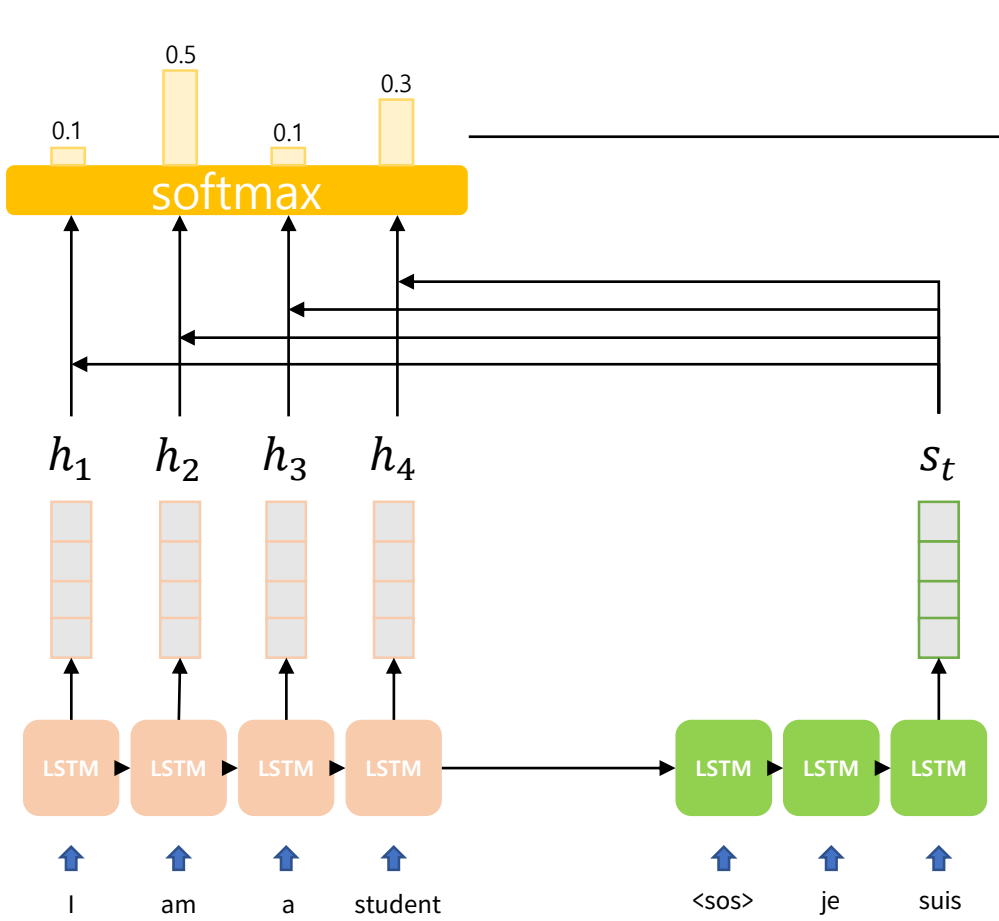
Q : t 시점의 디코더 셀의 hidden state
K : 모든 시점의 인코더 셀의 hidden state
V : 모든 시점의 인코더 셀의 hidden state

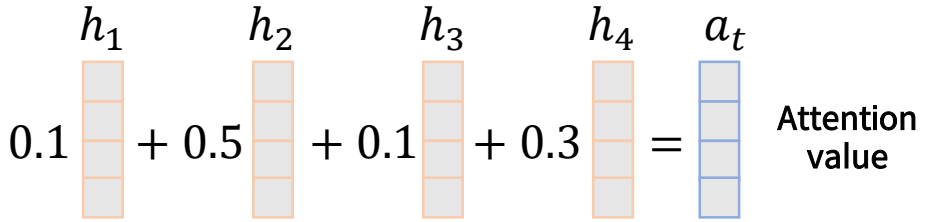


seq2seq + 어텐션 메커니즘

Attention Mechanism

Q : t 시점의 디코더 셀의 hidden state
K : 모든 시점의 인코더 셀의 hidden state
V : 모든 시점의 인코더 셀의 hidden state



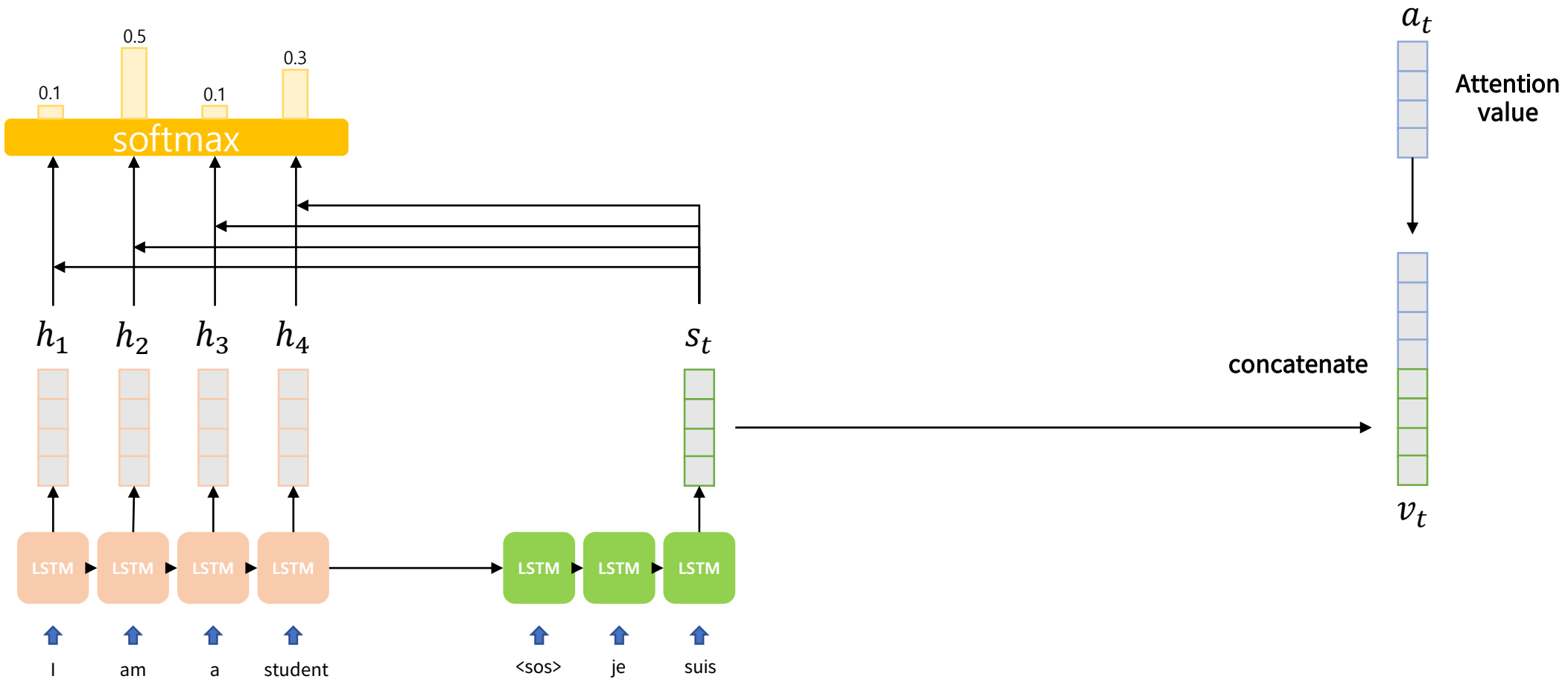


Value
밸류

seq2seq + 어텐션 메커니즘

Attention Mechanism

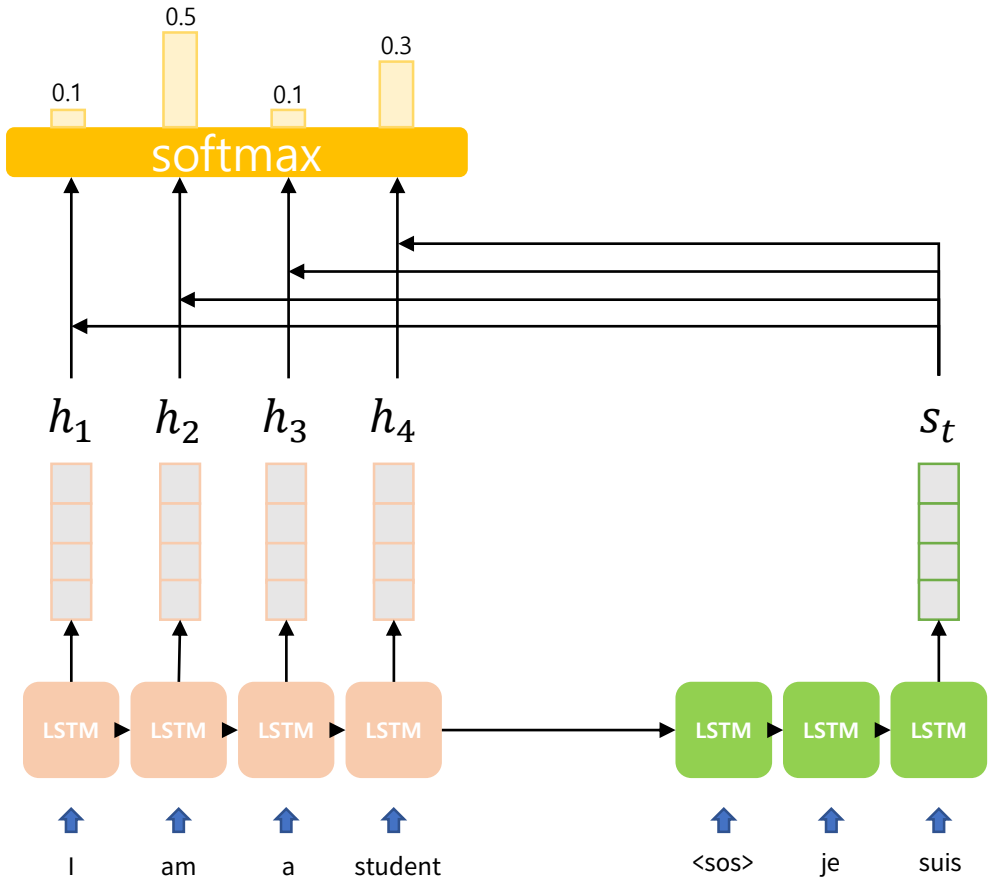
Q : t 시점의 디코더 셀의 hidden state
K : 모든 시점의 인코더 셀의 hidden state
V : 모든 시점의 인코더 셀의 hidden state



seq2seq + 어텐션 메커니즘

Attention Mechanism

Q : t 시점의 디코더 셀의 hidden state
K : 모든 시점의 인코더 셀의 hidden state
V : 모든 시점의 인코더 셀의 hidden state



$$\tilde{s_t} = \tanh(W_c[a_t; s_t] + b_c)$$

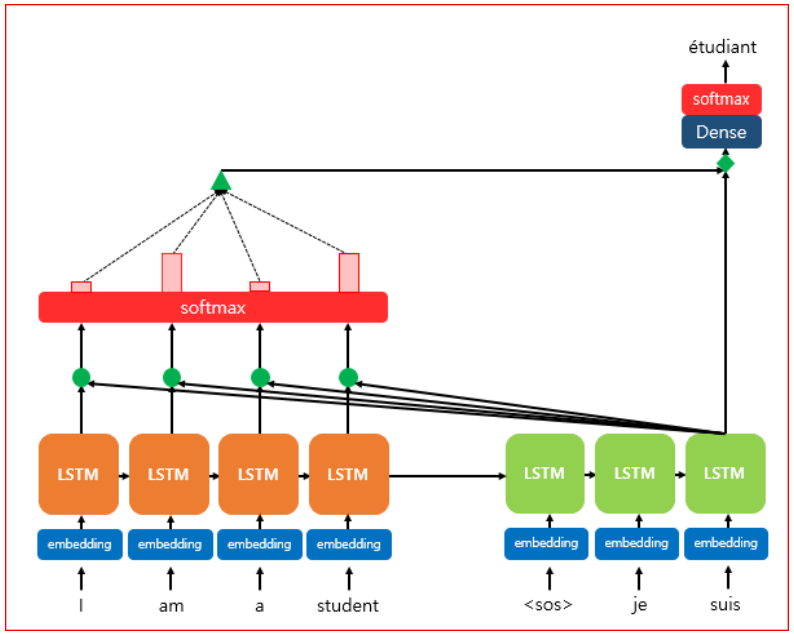
$$\tanh\left(\begin{bmatrix} & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \end{bmatrix} \times \begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \end{bmatrix} + \begin{bmatrix} \\ \\ \\ \end{bmatrix} \right) = \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

W_c v_t b_c $\tilde{s_t}$

seq2seq + 어텐션 메커니즘

Attention Mechanism

seq2seq + 어텐션 메커니즘



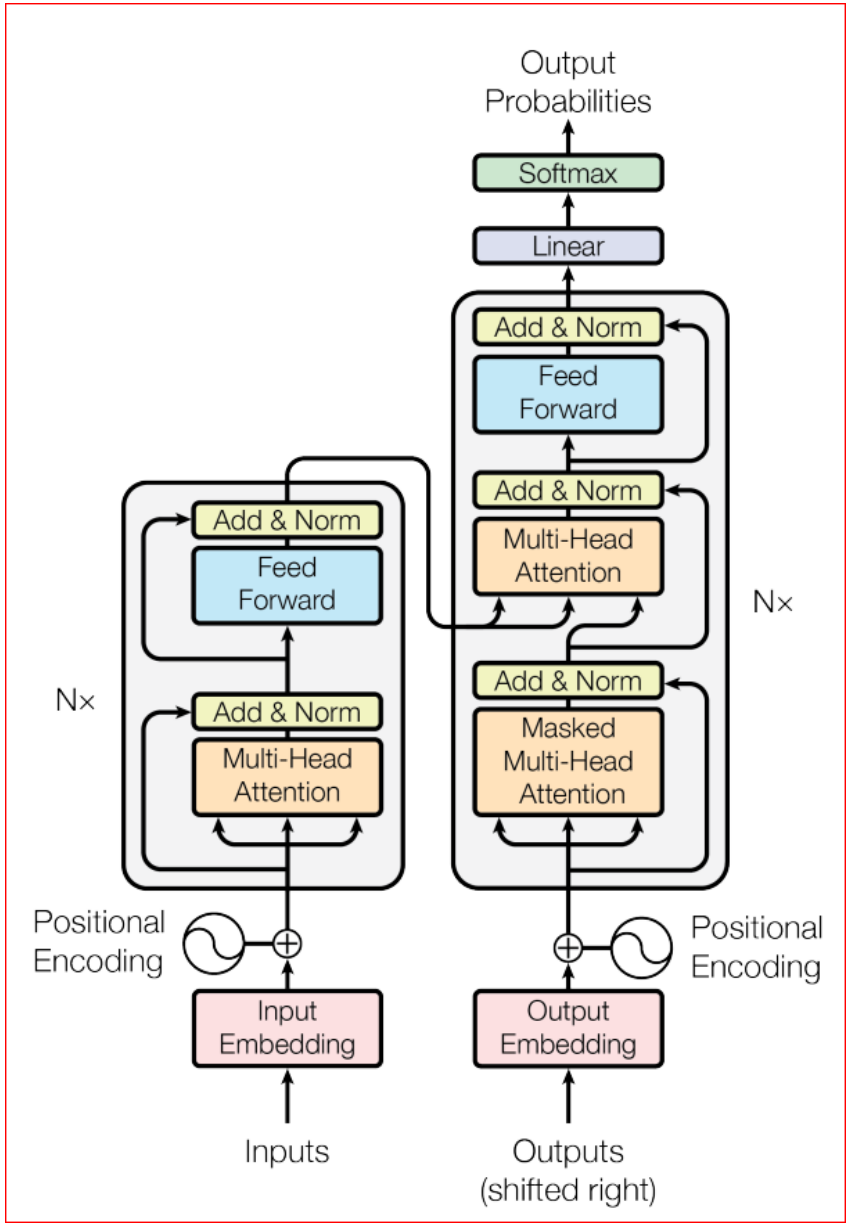
Attention Mechanism

다양한 어텐션

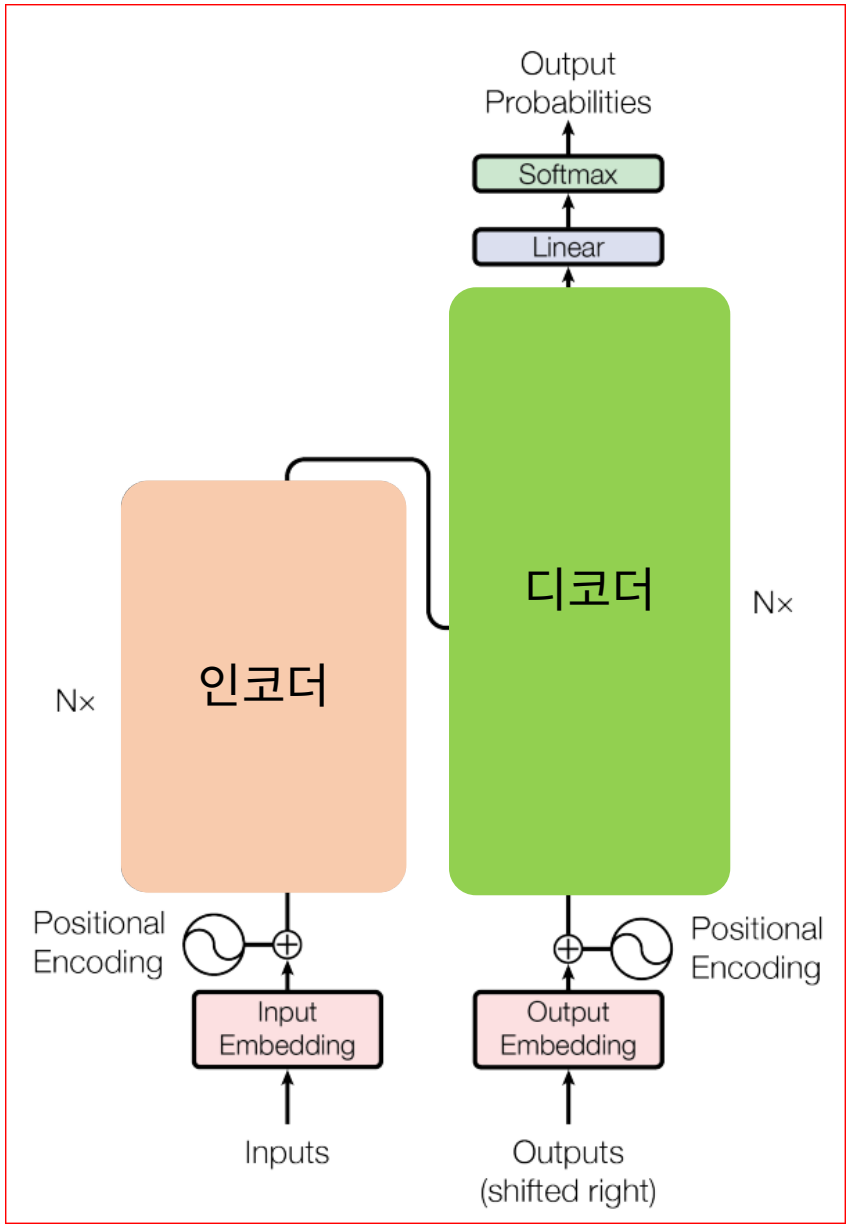
이름	스코어 함수
dot product attn.	$score(s_t, h_i) = s_i^T h_i$
scaled dot product attn.	$score(s_t, h_i) = \frac{s_i^T h_i}{\sqrt{n}}$
Bahdanau attn.	$score(s_t, h_i) = W_a^T \tanh(W_b[s_t; h_i])$

Transformer 모델 디테일

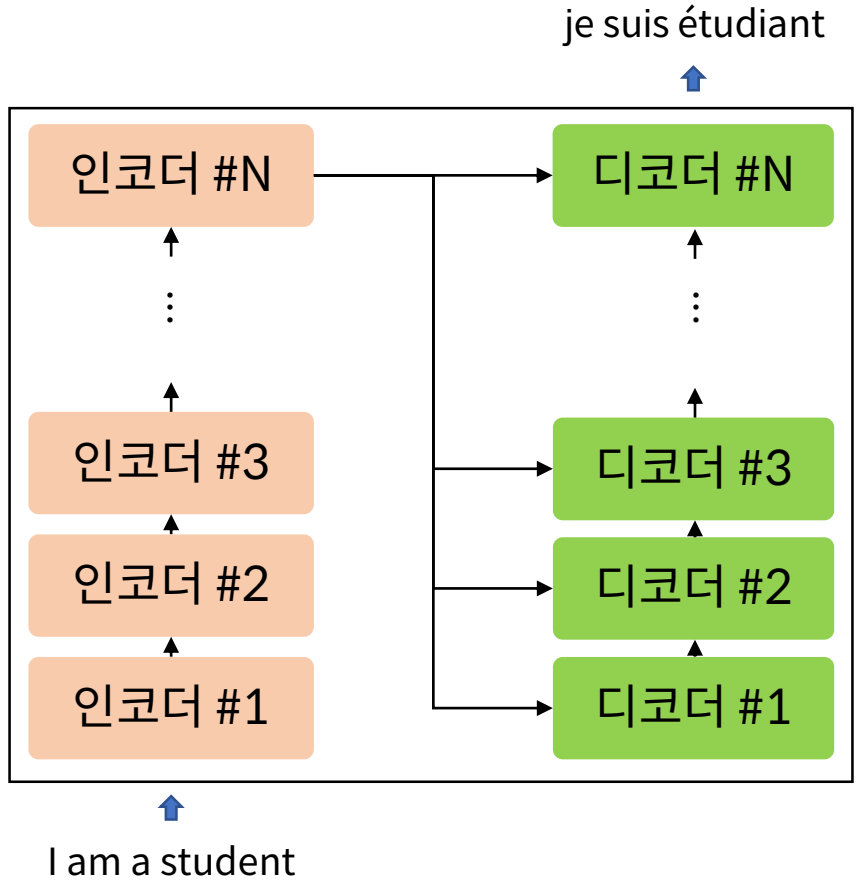
Transformer 모델 디테일



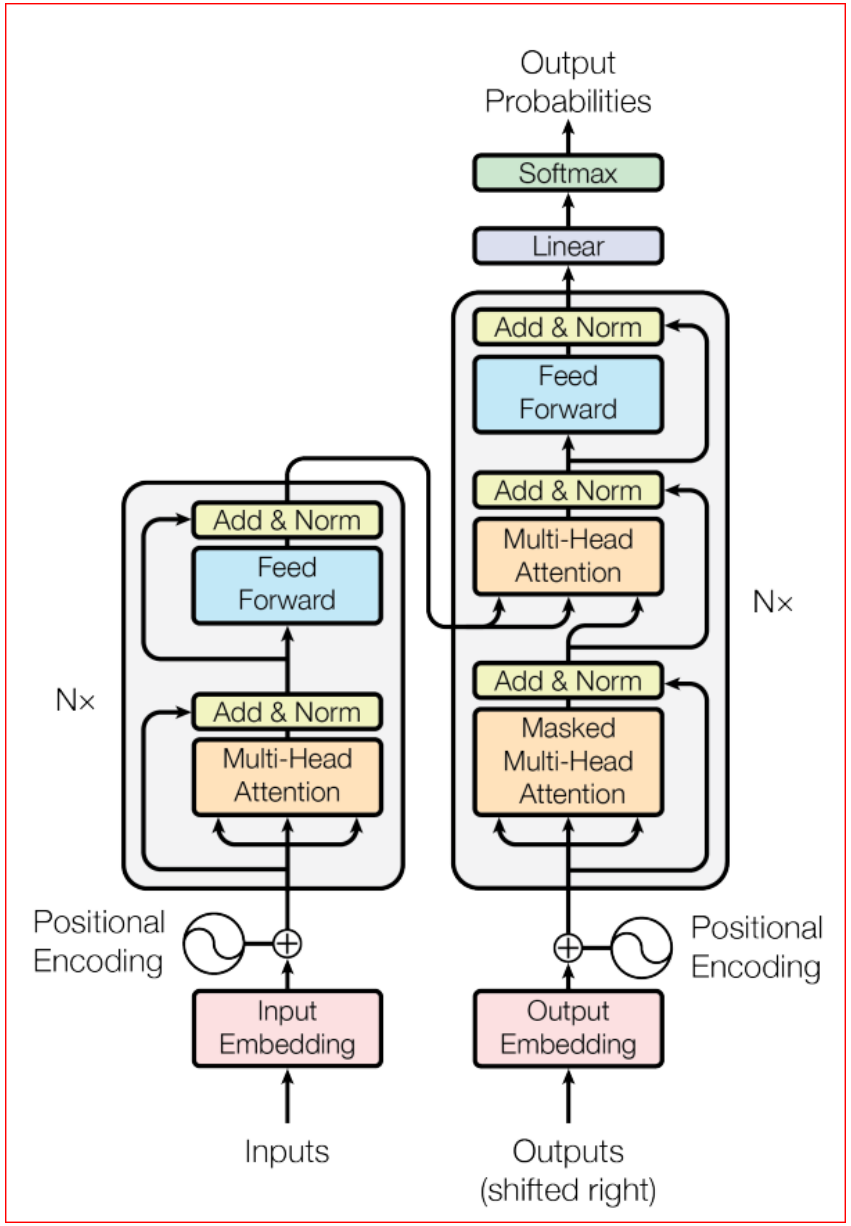
Transformer 모델 디테일



Transformer 모델 디테일

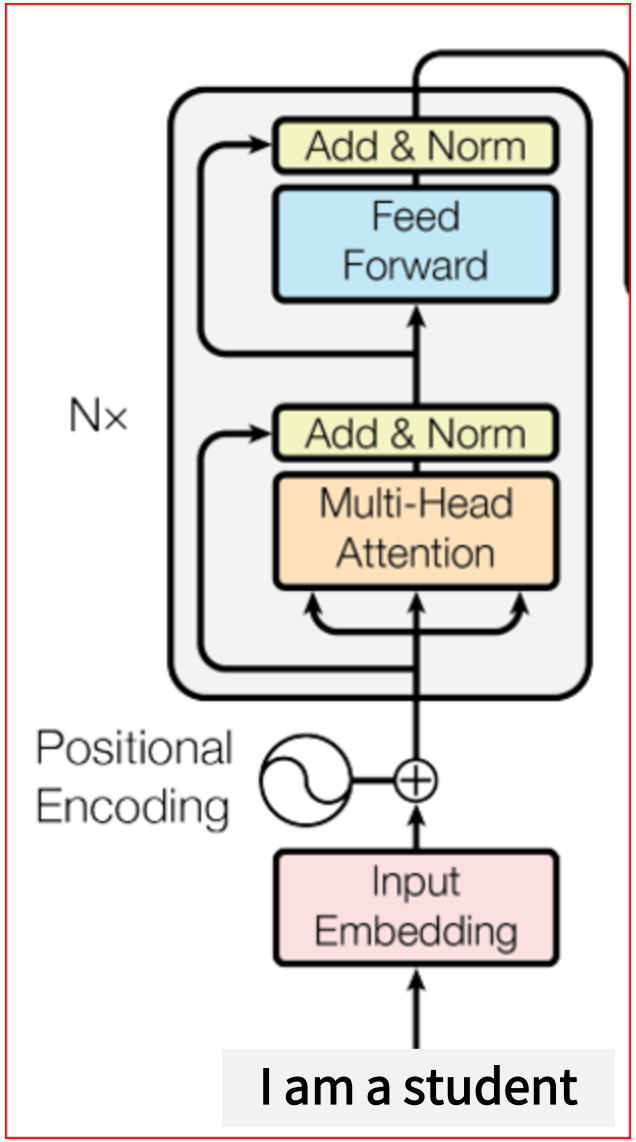


Transformer 모델 디테일

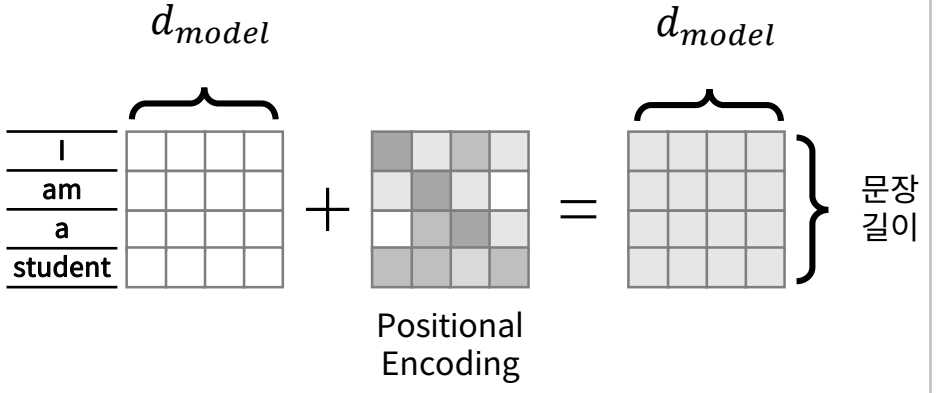


- 인코더
  - Multi-head self-attn.
- 디코더
  - Masked Multi-head self-attn.
  - 인코더-디코더 Multi-head attn.
- 공통
  - Positional Encoding
  - Feed-Forward 네트워크
  - Residual Connection
  - Layer Normalization

Transformer 모델 디테일
인코더

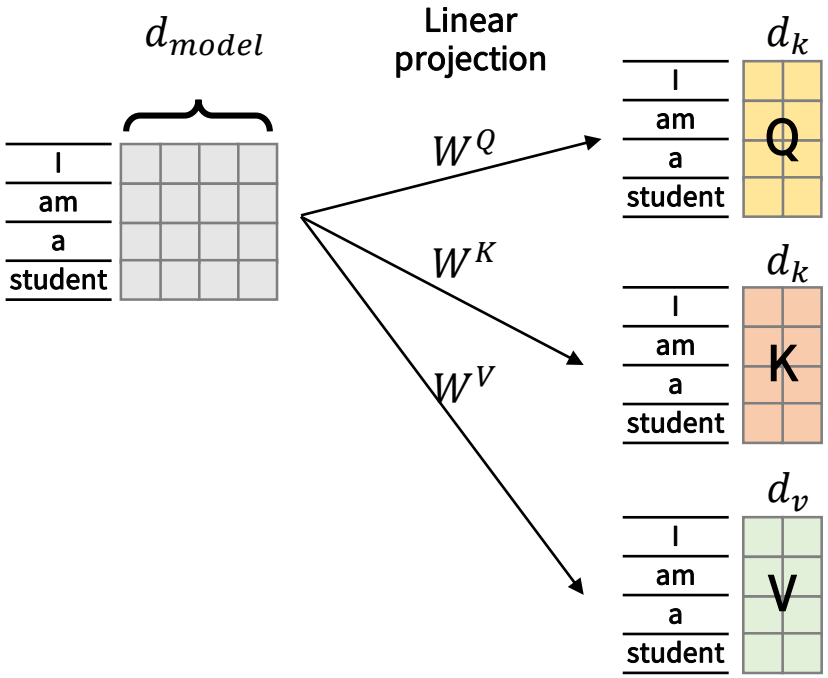


$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Transformer 모델 디테일

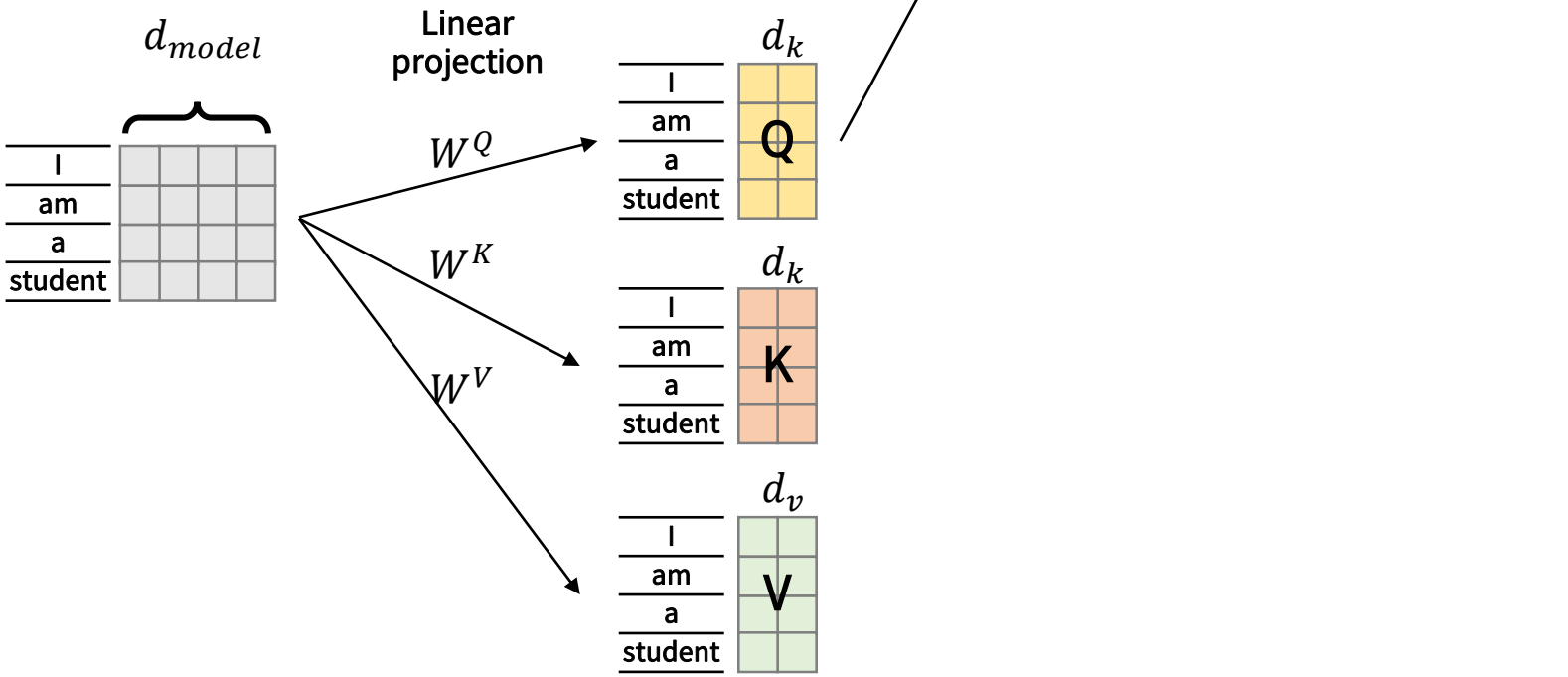
[인코더] Multi-head Self Attn.

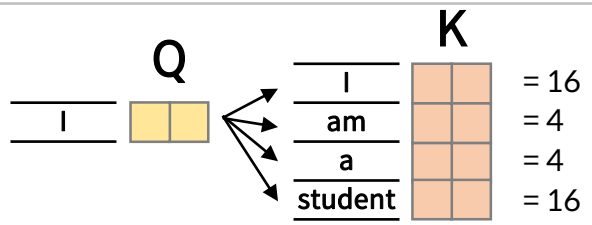


$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Transformer 모델 디테일

[인코더] Multi-head Self Attn.



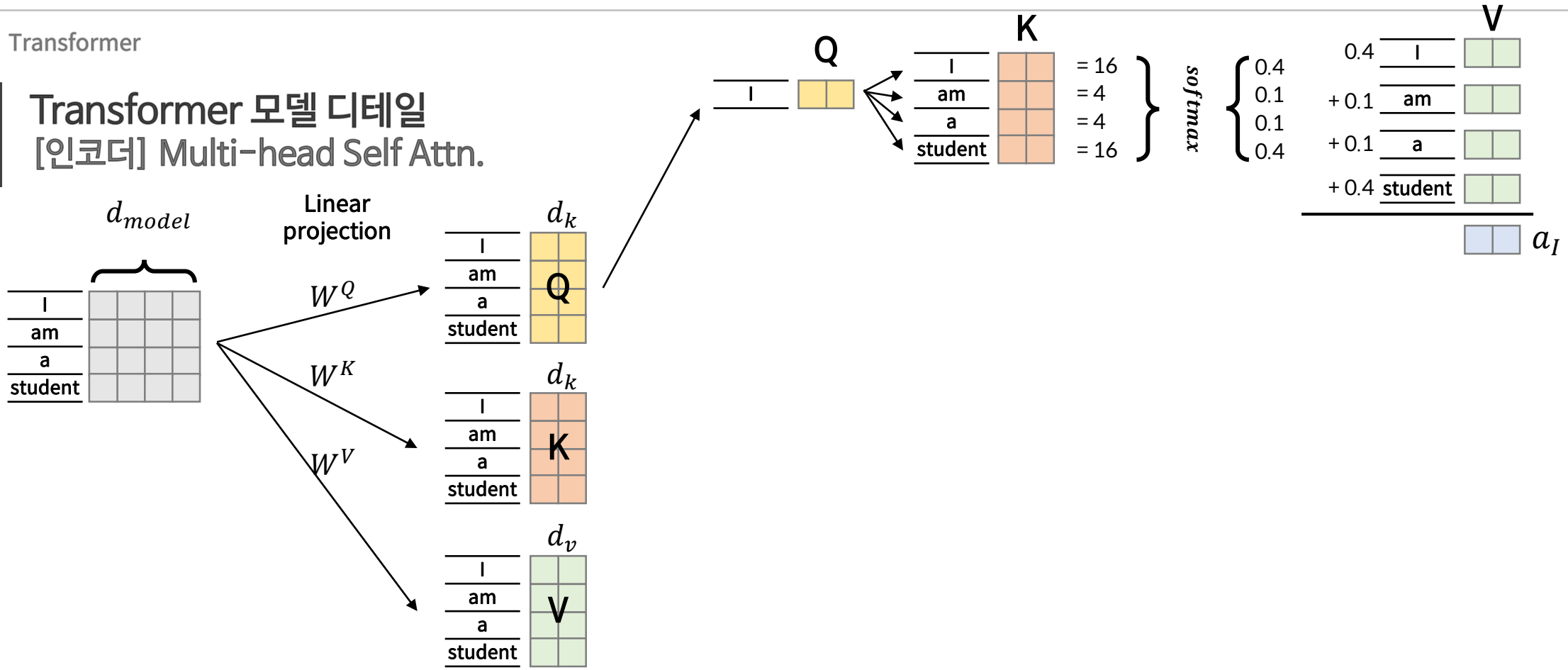


$$score(Q,K) = \frac{QK^T}{\sqrt{d_k}}$$

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

Transformer 모델 디테일

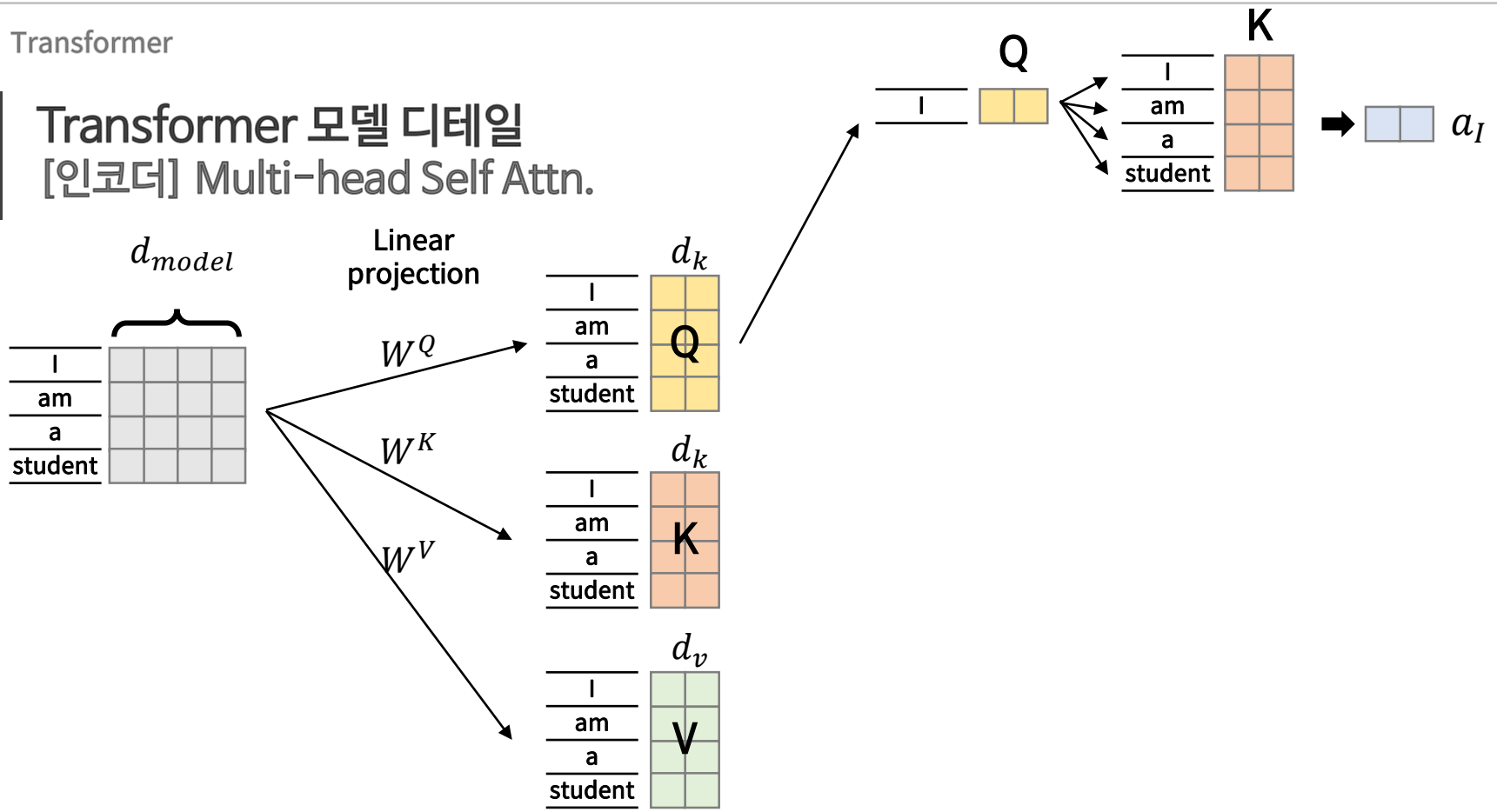
[인코더] Multi-head Self Attn.



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

Transformer 모델 디테일

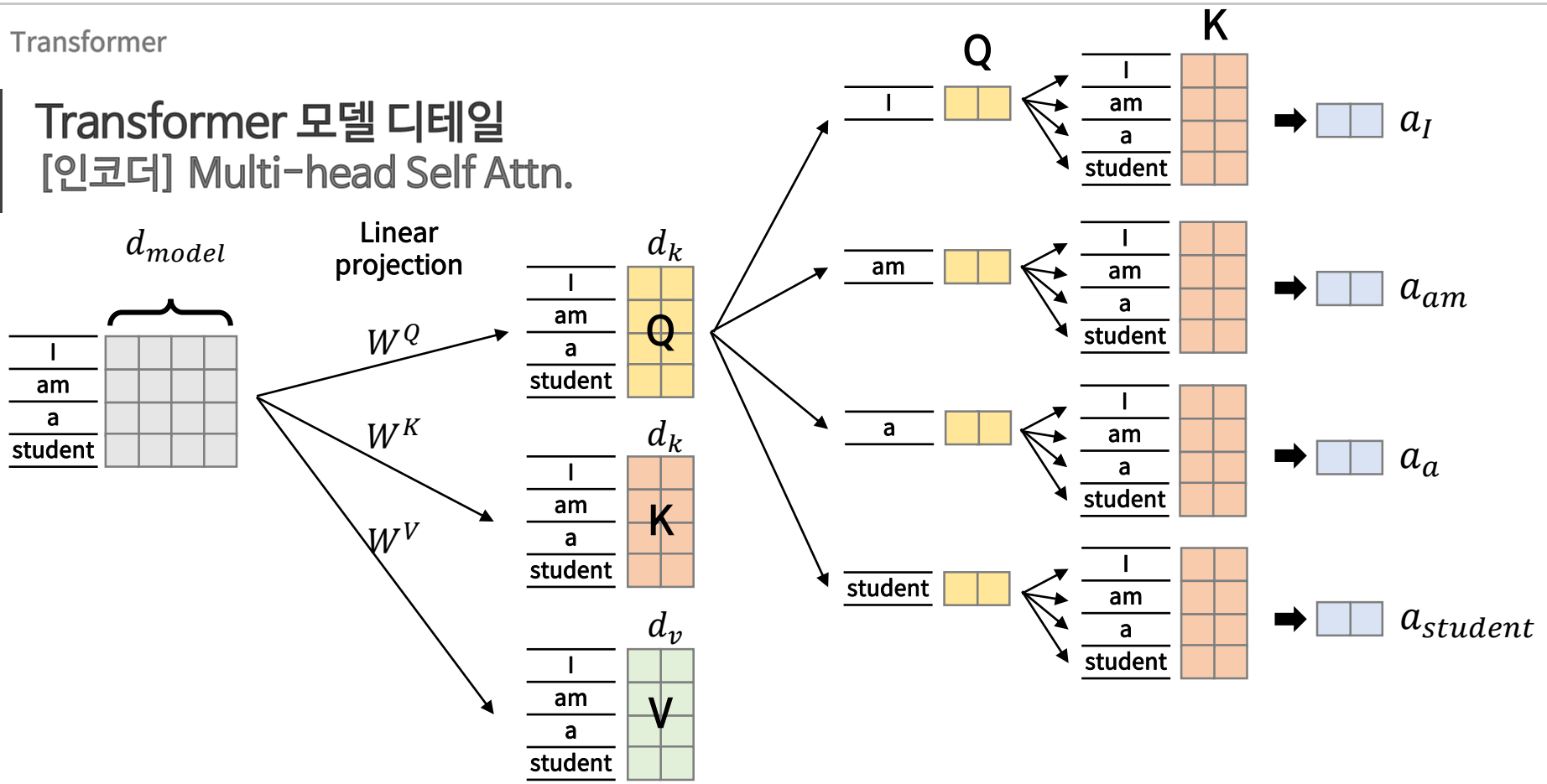
[인코더] Multi-head Self Attn.



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

Transformer 모델 디테일

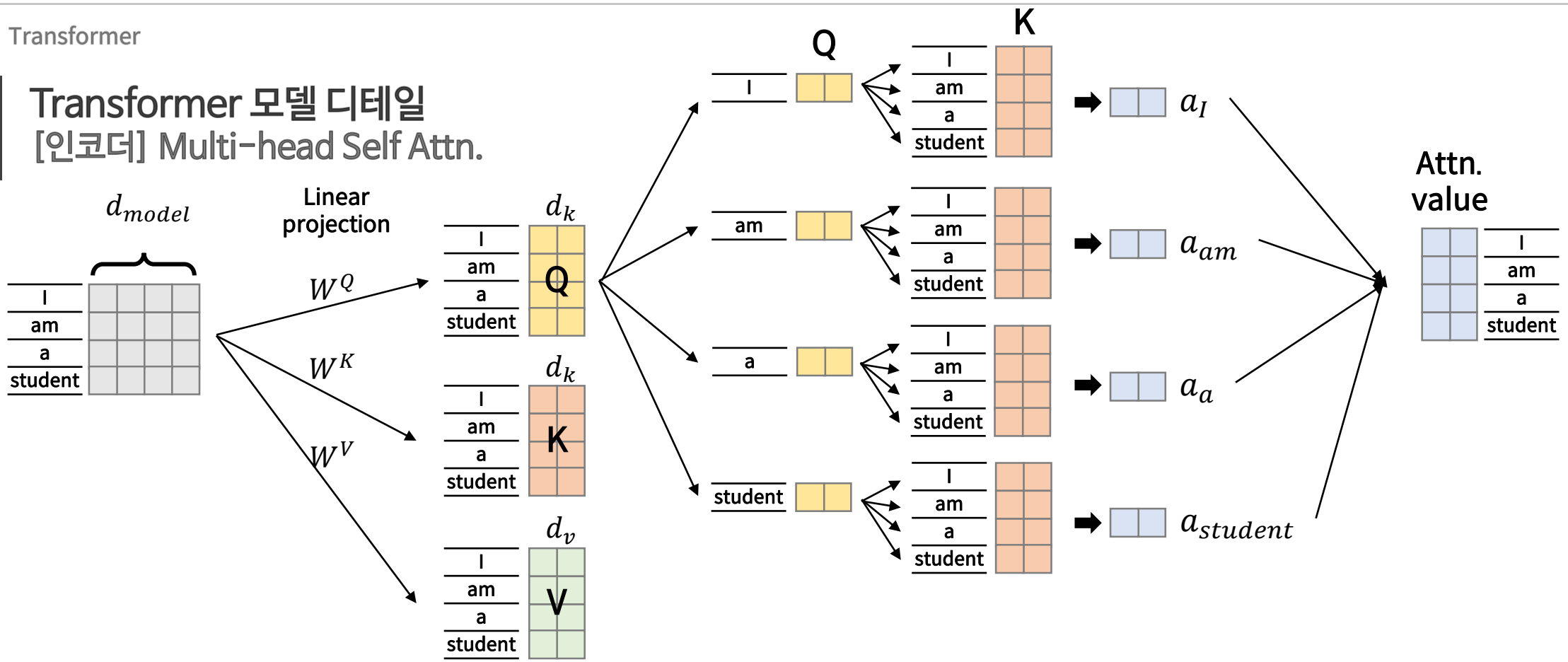
[인코더] Multi-head Self Attn.



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

Transformer 모델 디테일

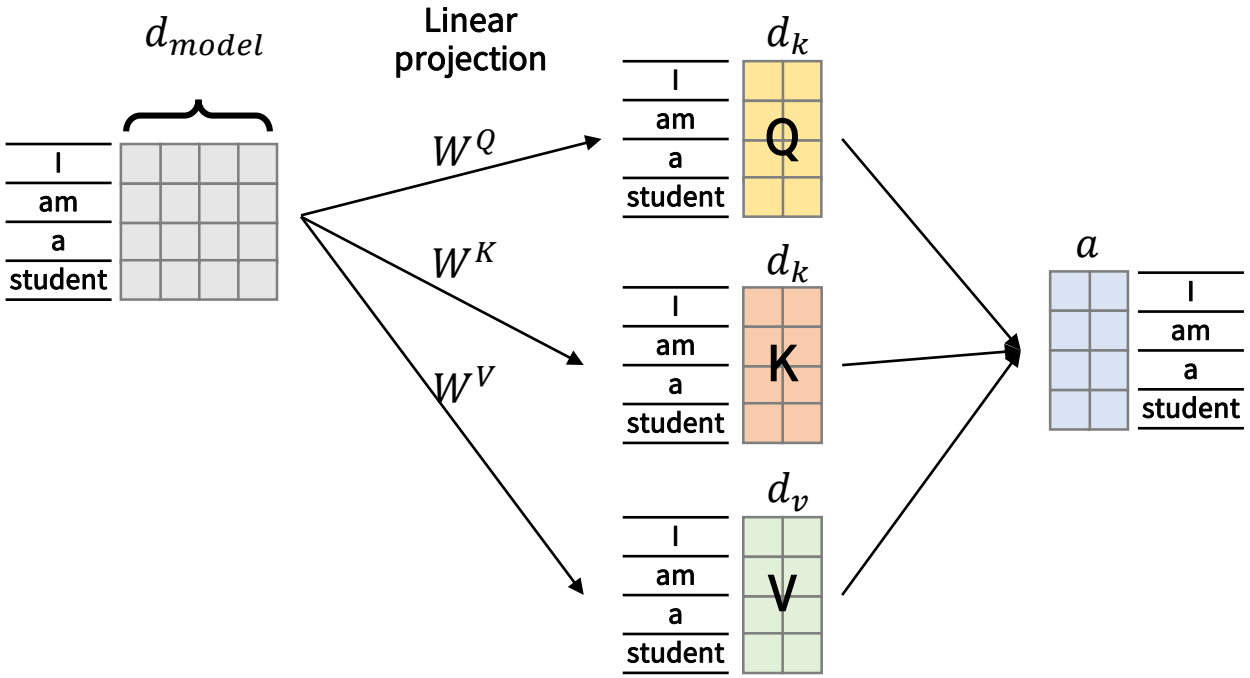
[인코더] Multi-head Self Attn.



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

Transformer 모델 디테일

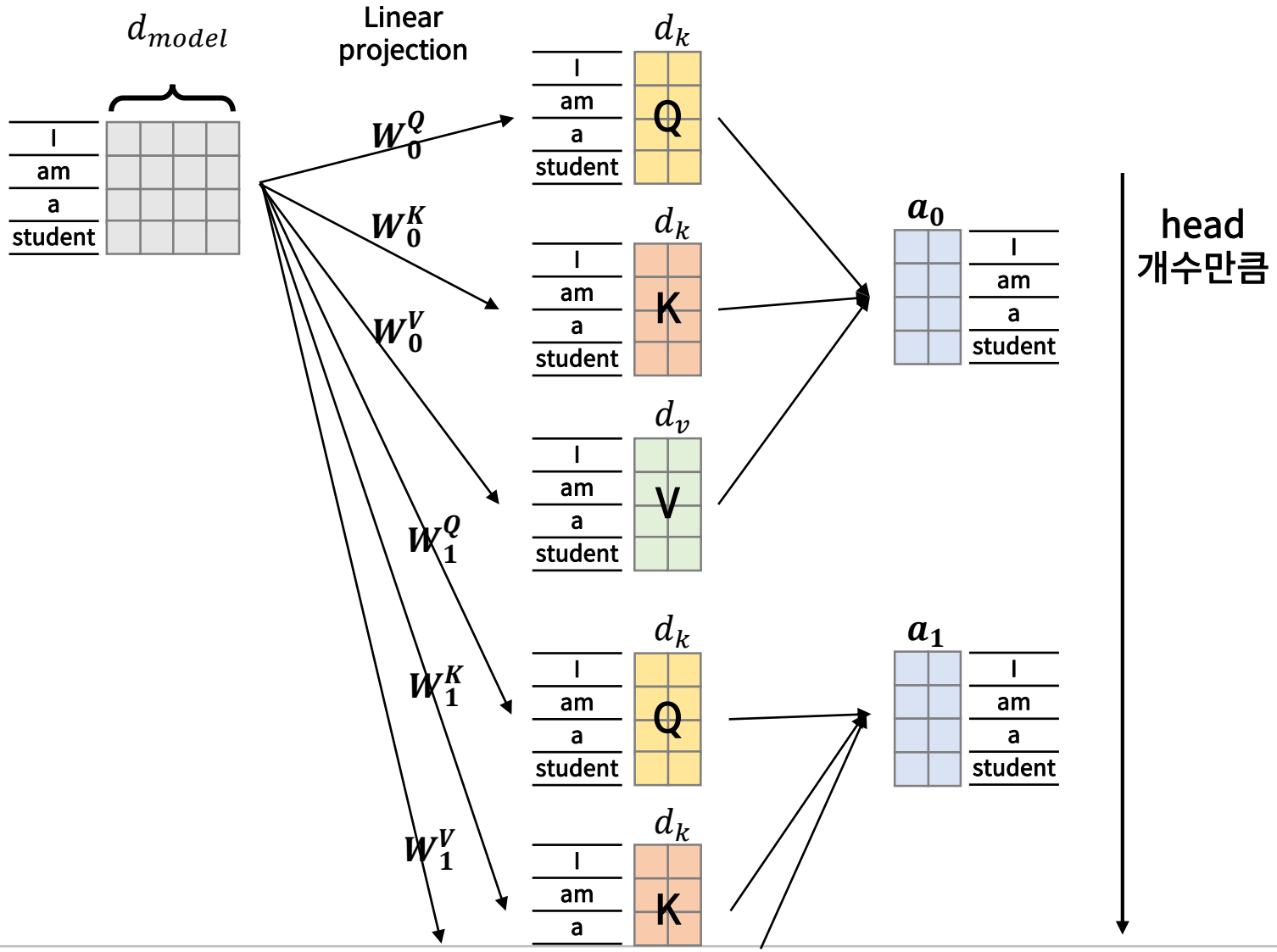
[인코더] Multi-head Self Attn.



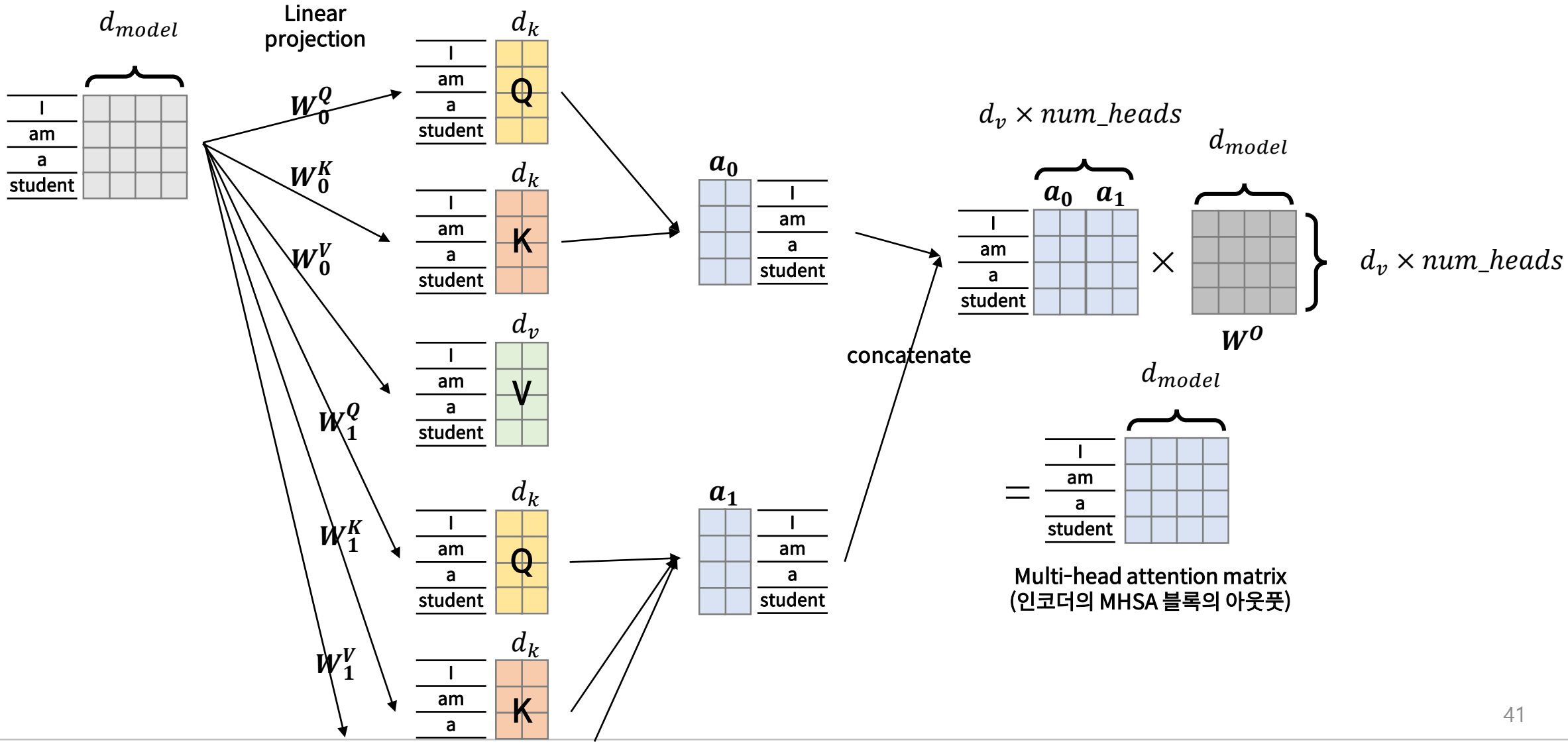
$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Transformer 모델 디테일

[인코더] Multi-head Self Attn.

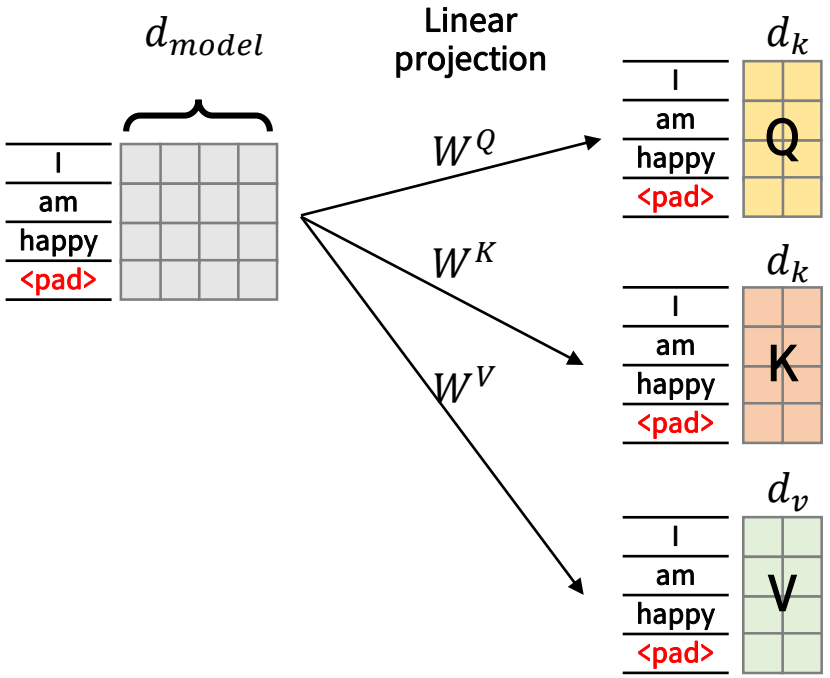


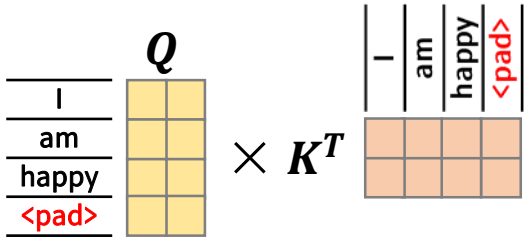
Transformer 모델 디테일
[인코더] Multi-head Self Attn.



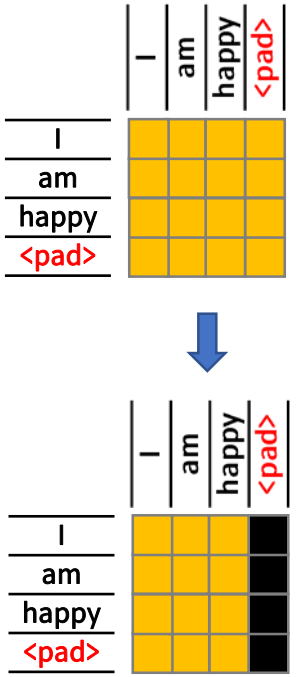
Transformer 모델 디테일

[인코더] Multi-head Self Attn.





Attention Score Matrix

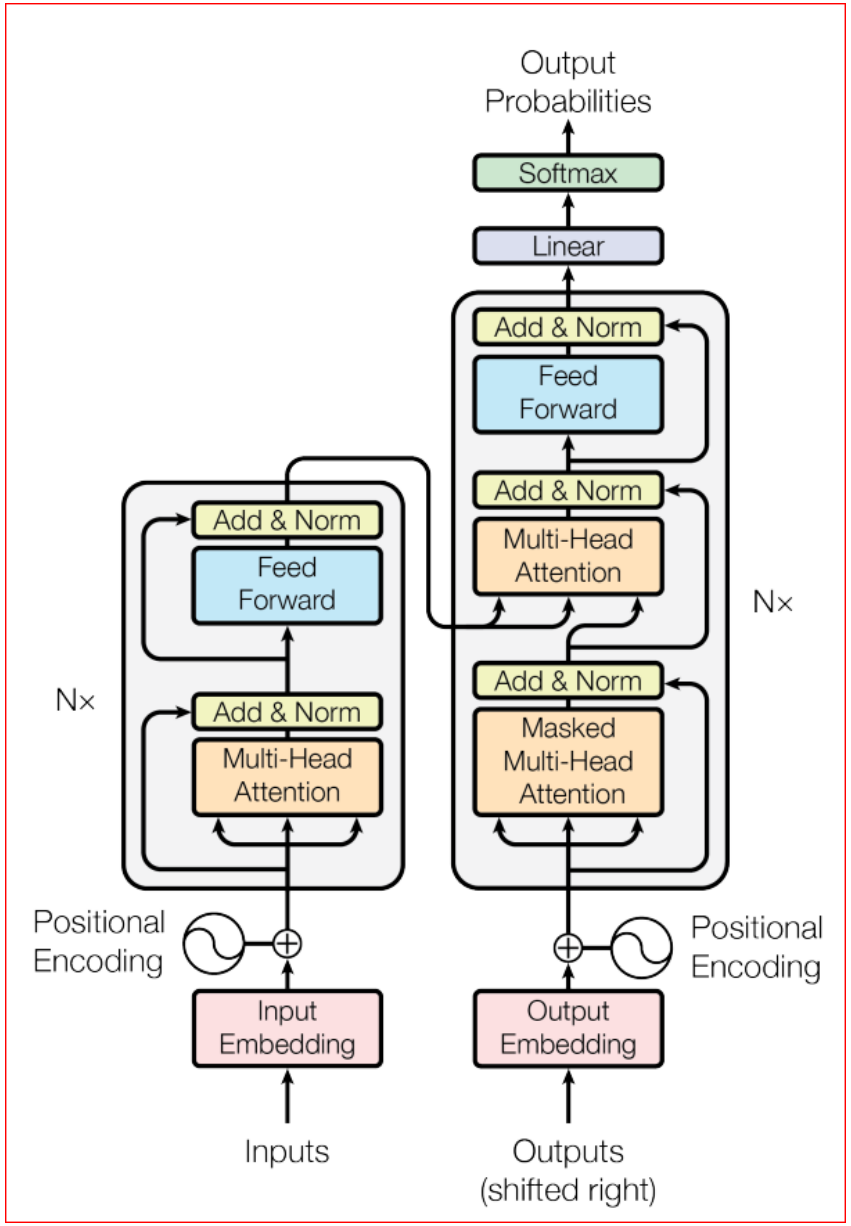


<pad> 토큰 자리에 -∞ 값을 넣어서 소프트맥스를 거치면 0이 나오도록 처리

모델에 입력되는 문장의 길이가 항상 같은 것이 아니기 때문에 입력에 <pad> 토큰이 들어가는 경우가 있음.

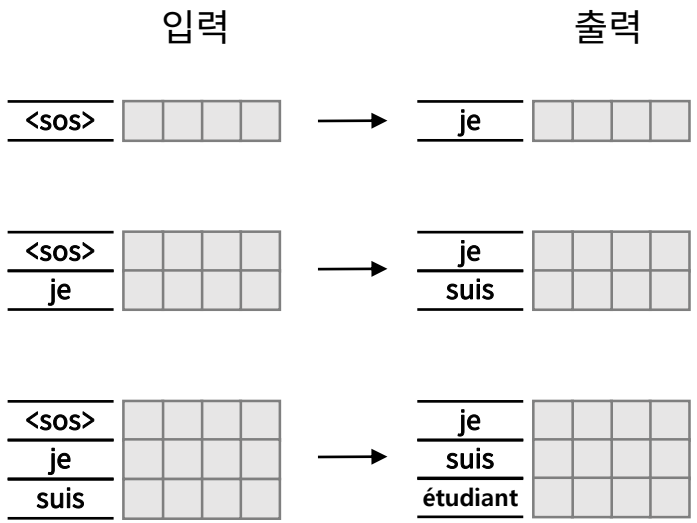
$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Transformer 모델 디테일

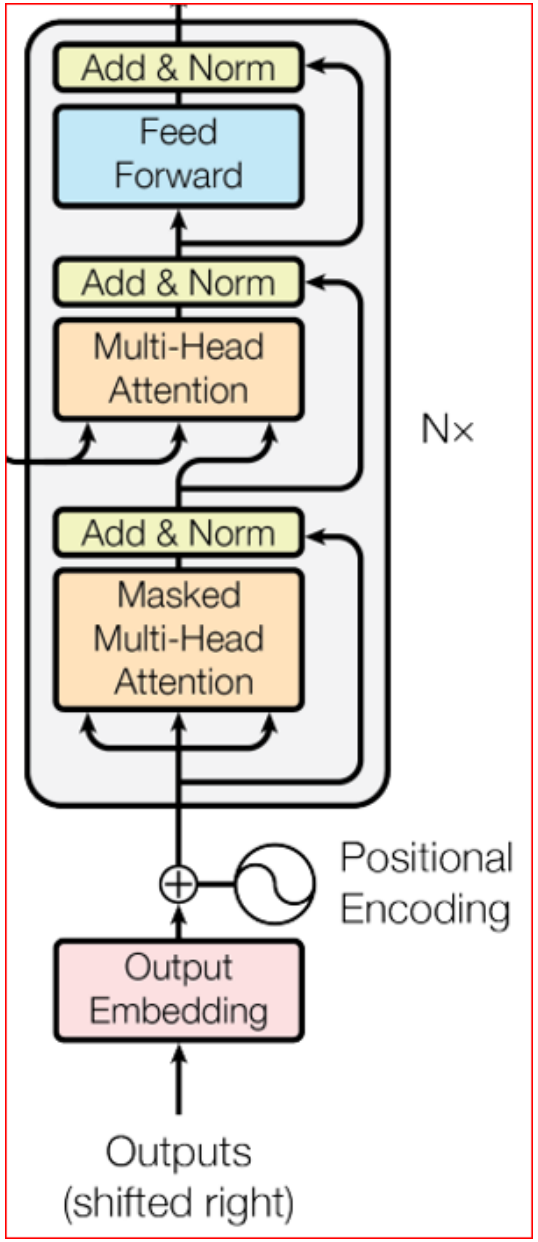


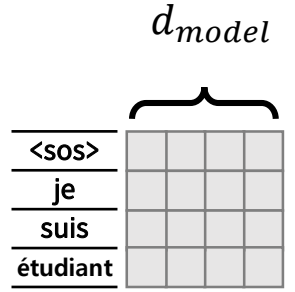
- 인코더
  - ~~Multi-head self-attn.~~
- 디코더
  - Masked Multi-head self-attn.
  - 인코더-디코더 Multi-head attn.
- 공통
  - Positional Encoding
  - Feed-Forward 네트워크
  - Residual Connection
  - Layer Normalization

Transformer 모델 디테일
[디코더] Masked Multi-head Self Attn.



학습이 완료된 디코더에 기대하는 것

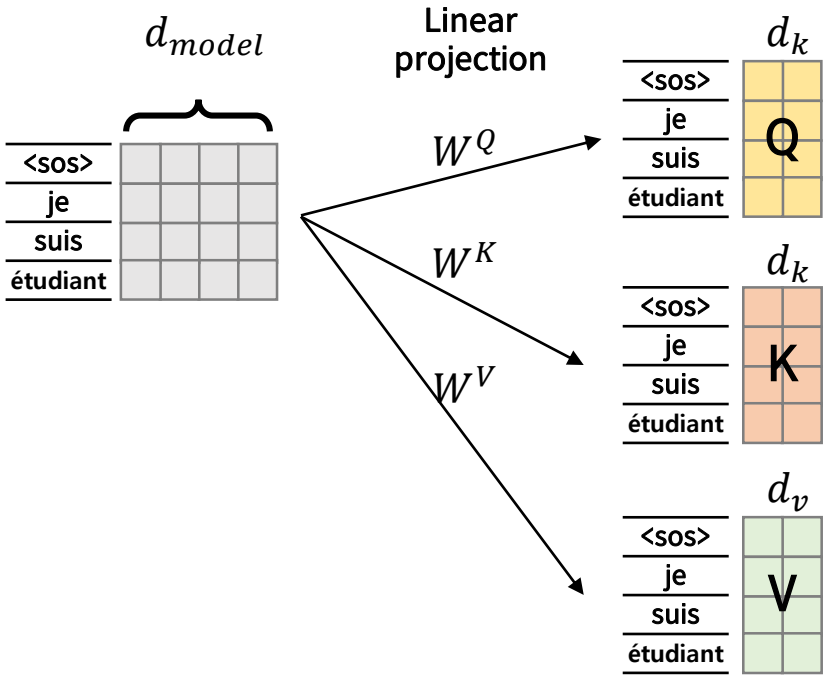


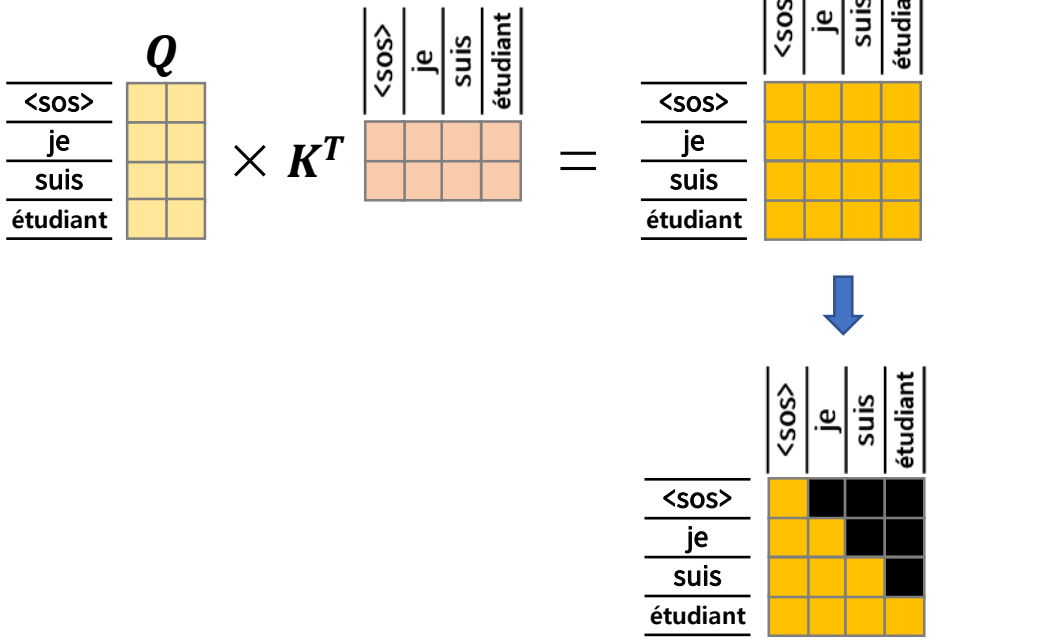


학습할 때 넣어주는 입력

Transformer 모델 디테일

[디코더] Masked Multi-head Self Attn.

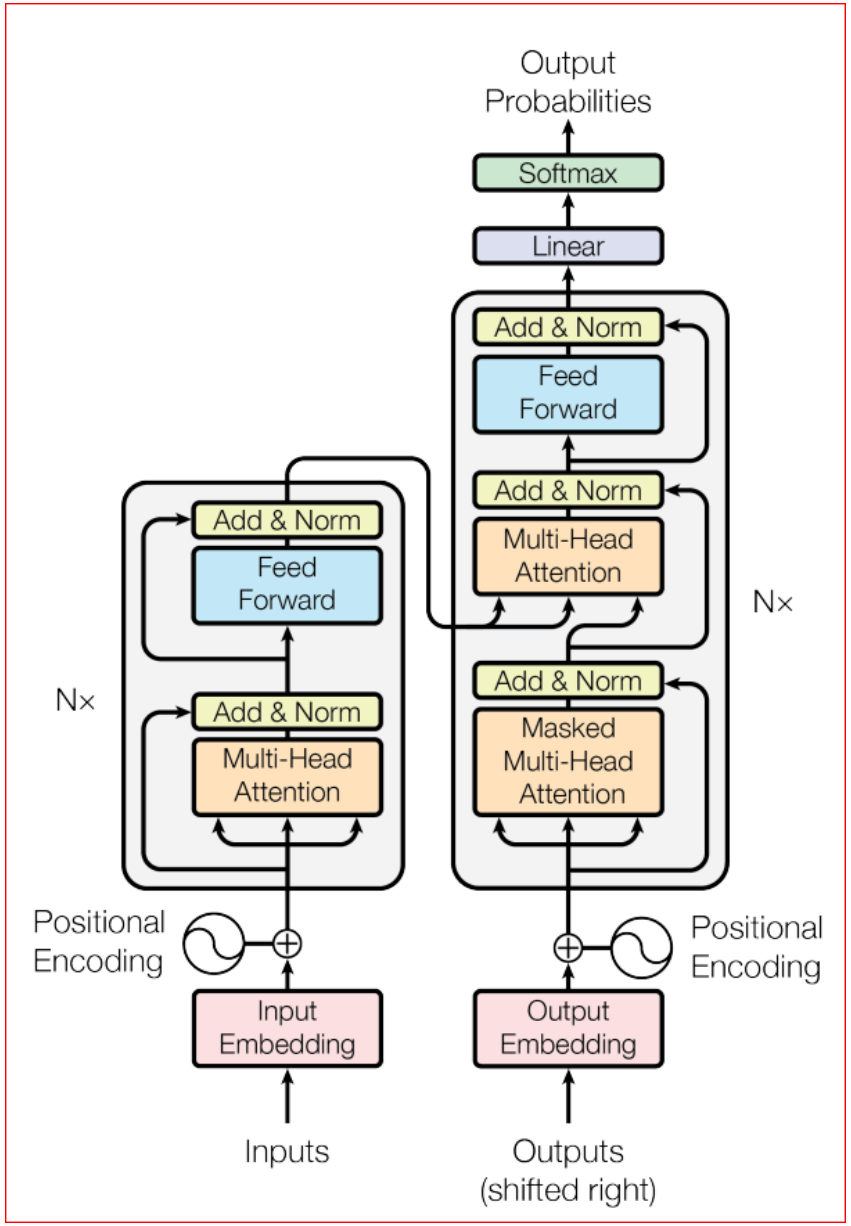




자신보다 이후 시점의 단어들을 참고하지 못하도록
마스킹을 해줌

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

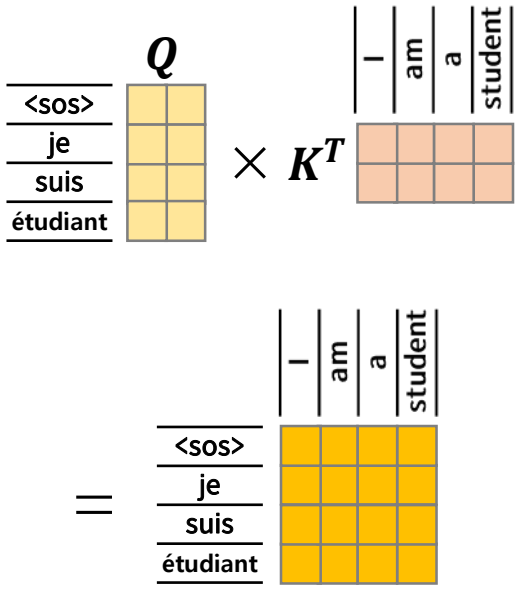
Transformer 모델 디테일



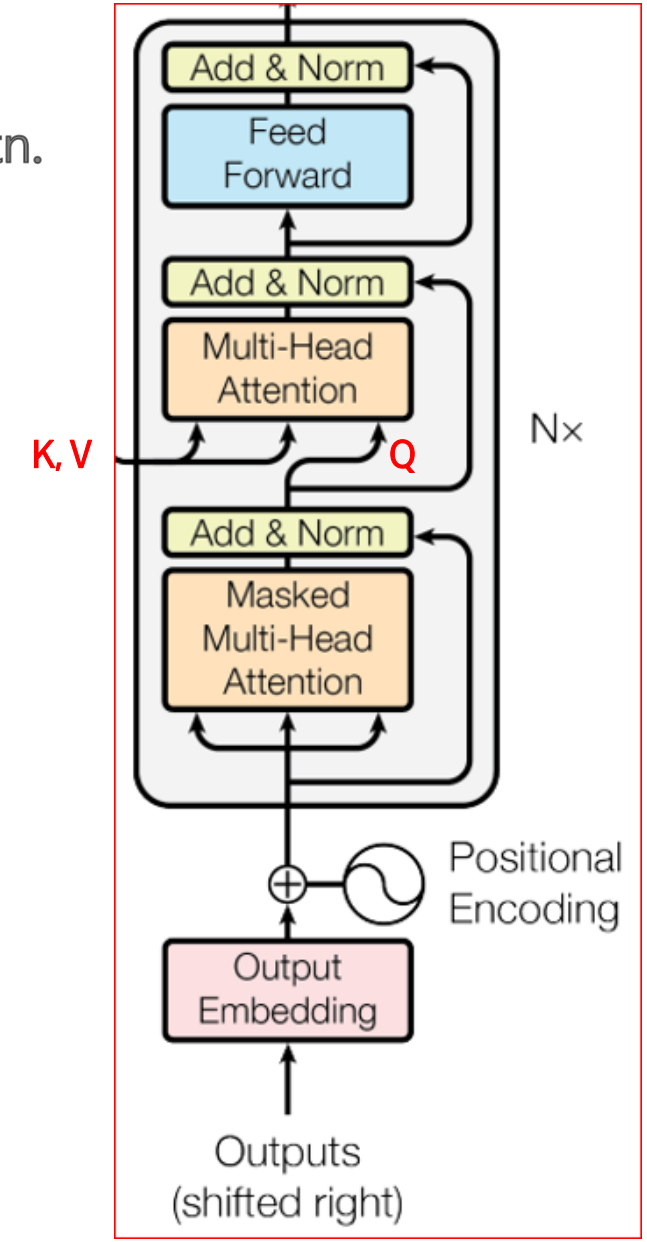
- 인코더
  - ~~Multi-head self-attn.~~
- 디코더
  - ~~Masked Multi-head self-attn.~~
  - 인코더-디코더 Multi-head attn.
- 공통
  - Positional Encoding
  - Feed-Forward 네트워크
  - Residual Connection
  - Layer Normalization

Transformer 모델 디테일

[디코더] 인코더-디코더 Multi-head Attn.



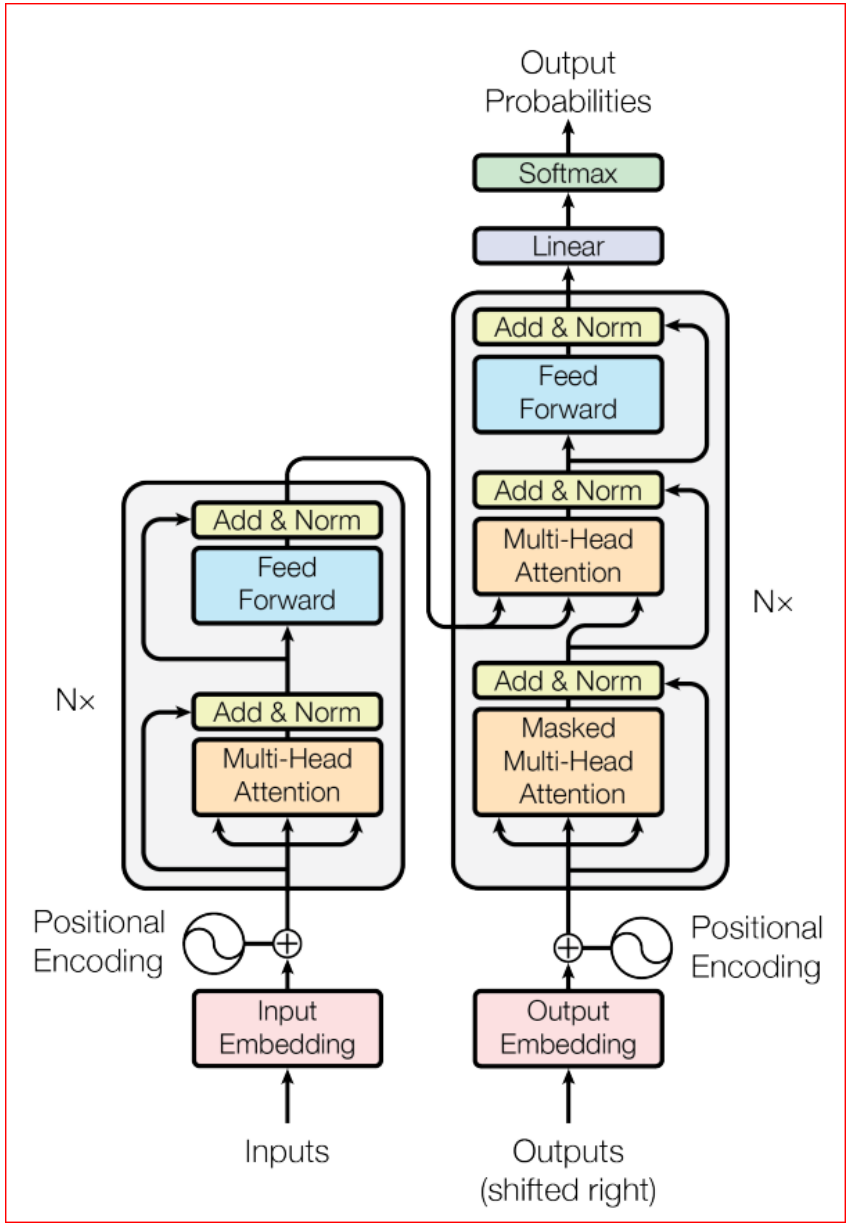
Attention Score Matrix



- Q(쿼리)는 디코더에서 옴
- K(키)와 V(밸류)는 인코더에서 넘어옴

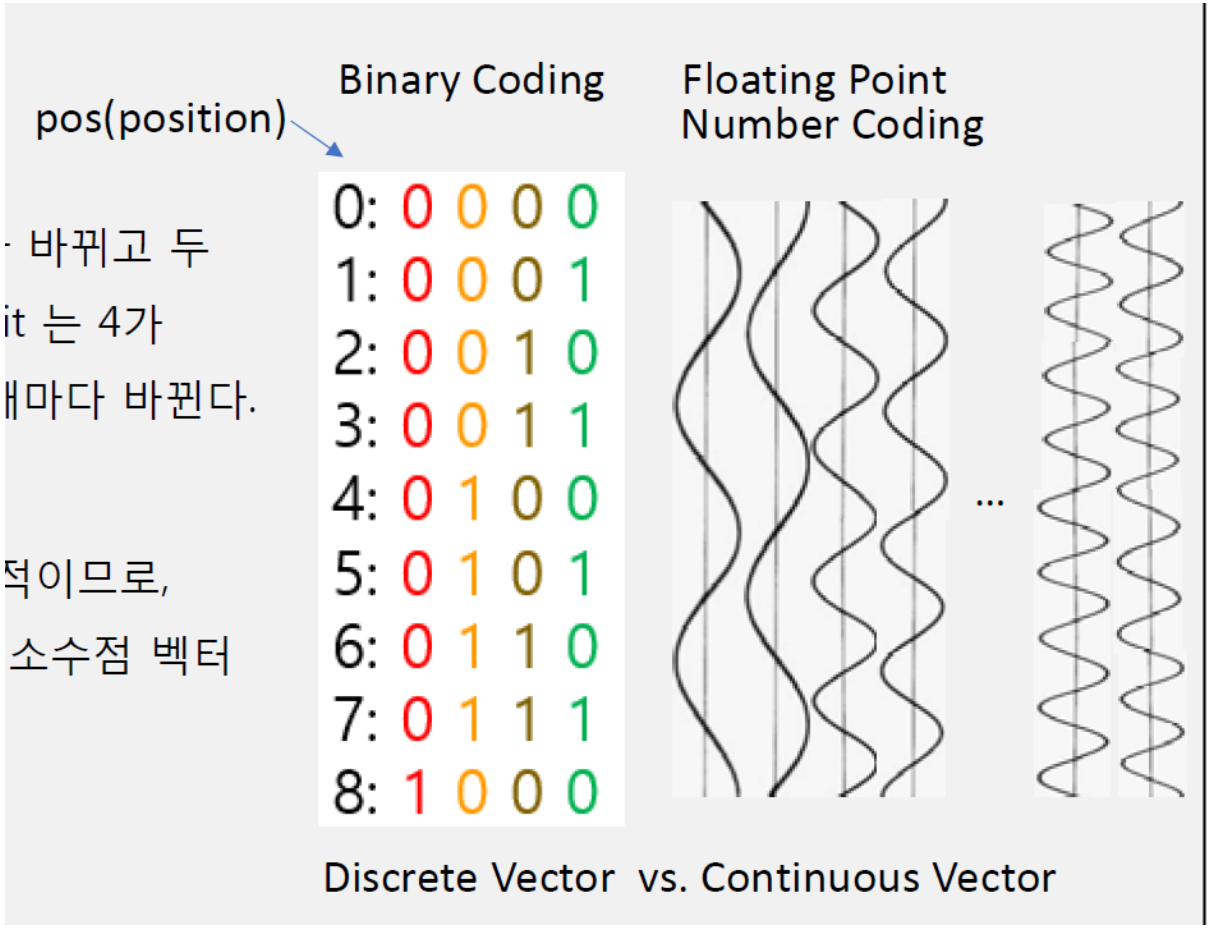
- 이 어텐션은 seq2seq 모델에 어텐션 적용할 때와 상당히 유사함

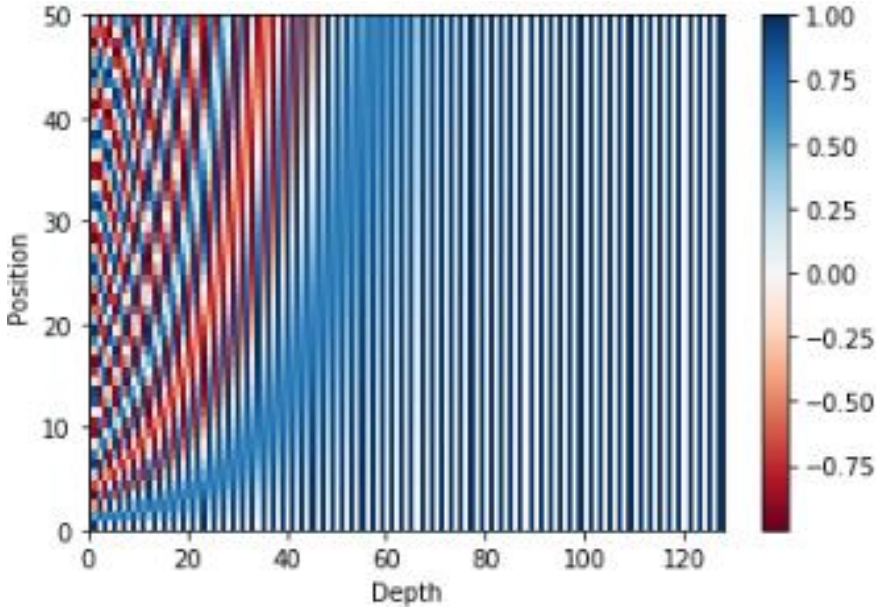
Transformer 모델 디테일



- 인코더
  - ~~Multi-head self-attn.~~
- 디코더
  - ~~Masked Multi-head self-attn.~~
  - ~~인코더-디코더 Multi-head attn.~~
- 공통
  - Positional Encoding
  - Feed-Forward 네트워크
  - Residual Connection
  - Layer Normalization

Transformer 모델 디테일
[공통] Positional Encoding

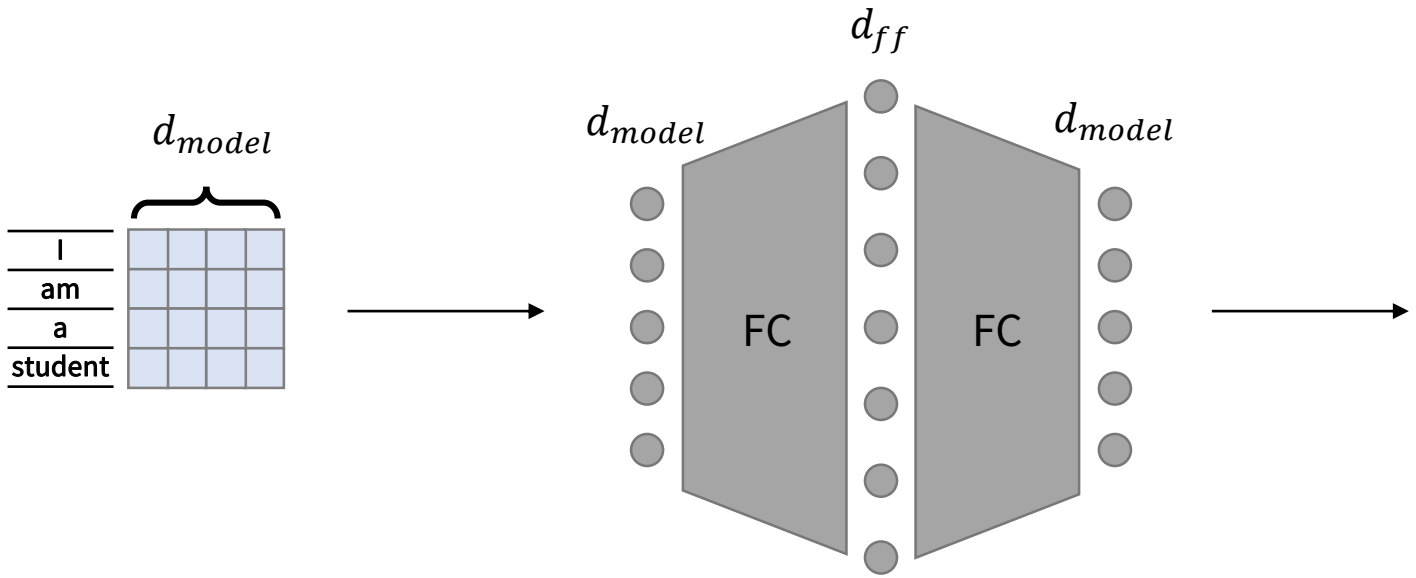




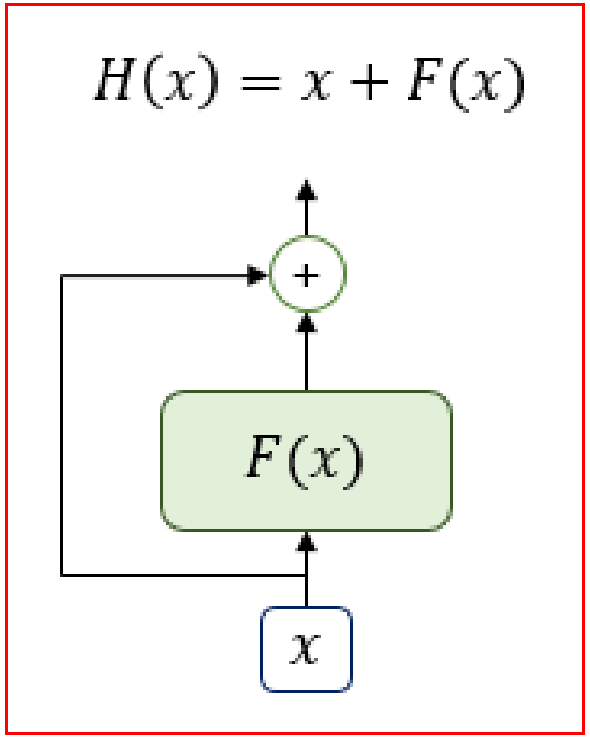
- 세로축 : 문장내에서 단어 위치 (문장길이가 최대 50)
- 가로축 : 임베딩벡터 차원 (=d_model)

Transformer 모델 디테일
[공통] Feed Forward 네트워크

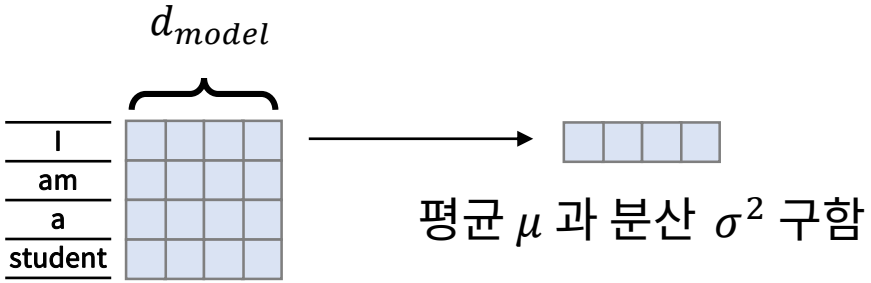
$$FFNN(x) = ReLU(xW_1 + b_1)W_2 + b_2$$



Transformer 모델 디테일
[공통] Residual Connection



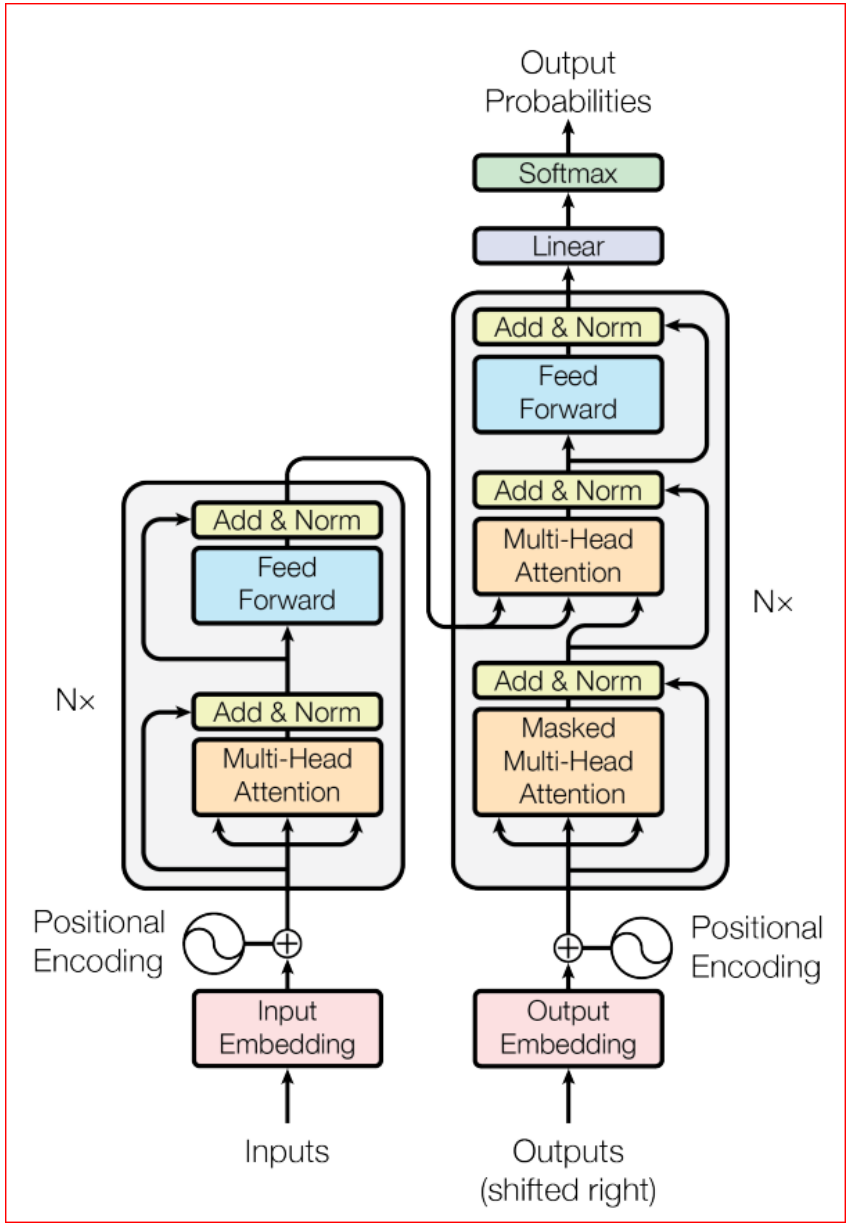
Transformer 모델 디테일
[공통] Layer Normalization



정규화 수식

$$z = \frac{x-\mu}{\sigma} \approx \frac{x-\mu}{\sqrt{\sigma^2+\epsilon}}$$

Transformer 모델 디테일



- 인코더
  - ~~Multi-head self-attn.~~
- 디코더
  - ~~Masked Multi-head self-attn.~~
  - ~~인코더-디코더 Multi-head attn.~~
- 공통
  - ~~Positional Encoding~~
  - ~~Feed-Forward 네트워크~~
  - ~~Residual Connection~~
  - ~~Layer Normalization~~

QA

참고자료

- 교수님 강의영상 : 16강 Transformer
- 논문 : Attention is all you need
- 자연어 처리 무료 eBook
  - 딥 러닝을 이용한 자연어 처리 입문 - 15. 어텐션 메커니즘
  - 딥 러닝을 이용한 자연어 처리 입문 - 16. 트랜스포머

Appendix

Model variations

PPL: Perplexity 낮을 수록 좋은 번역
BLEU : 높은 수록 좋은 번역

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	positional embedding instead of sinusoids									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213