

방학 4주차

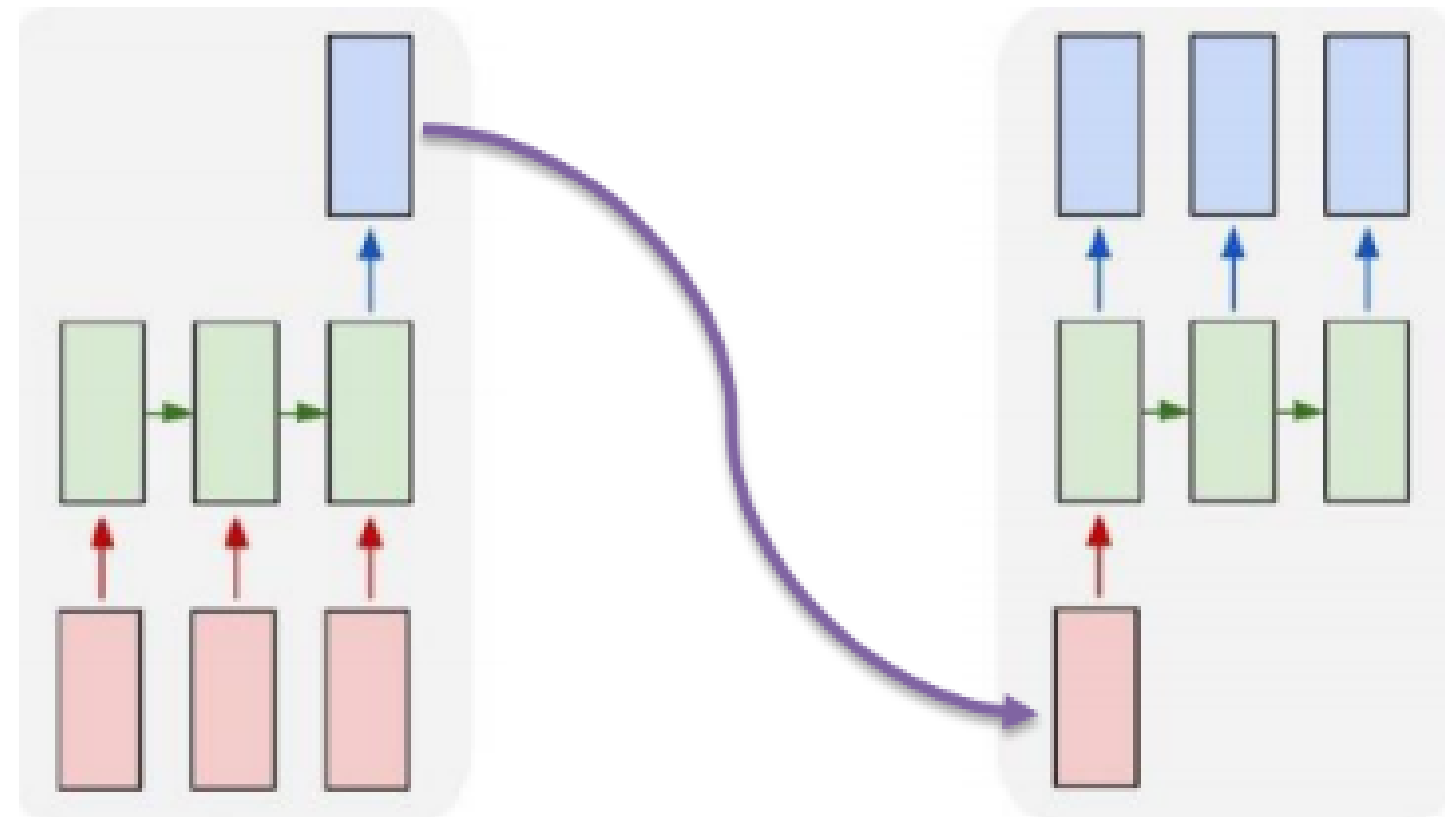
# TRANSFORMER & BERT

NEKA

# seq2seq

seq2seq 모델이란

인코더



디코더

# seq2seq

## seq2seq 모델의 한계

1. 하나의 고정된 크기의 벡터에 모든 정보를 압축하려고 하면서  
정보 손실이 발생함
2. Vanishing Gradient가 존재함

# Attention Mechanism

Attention Mechanism 이란

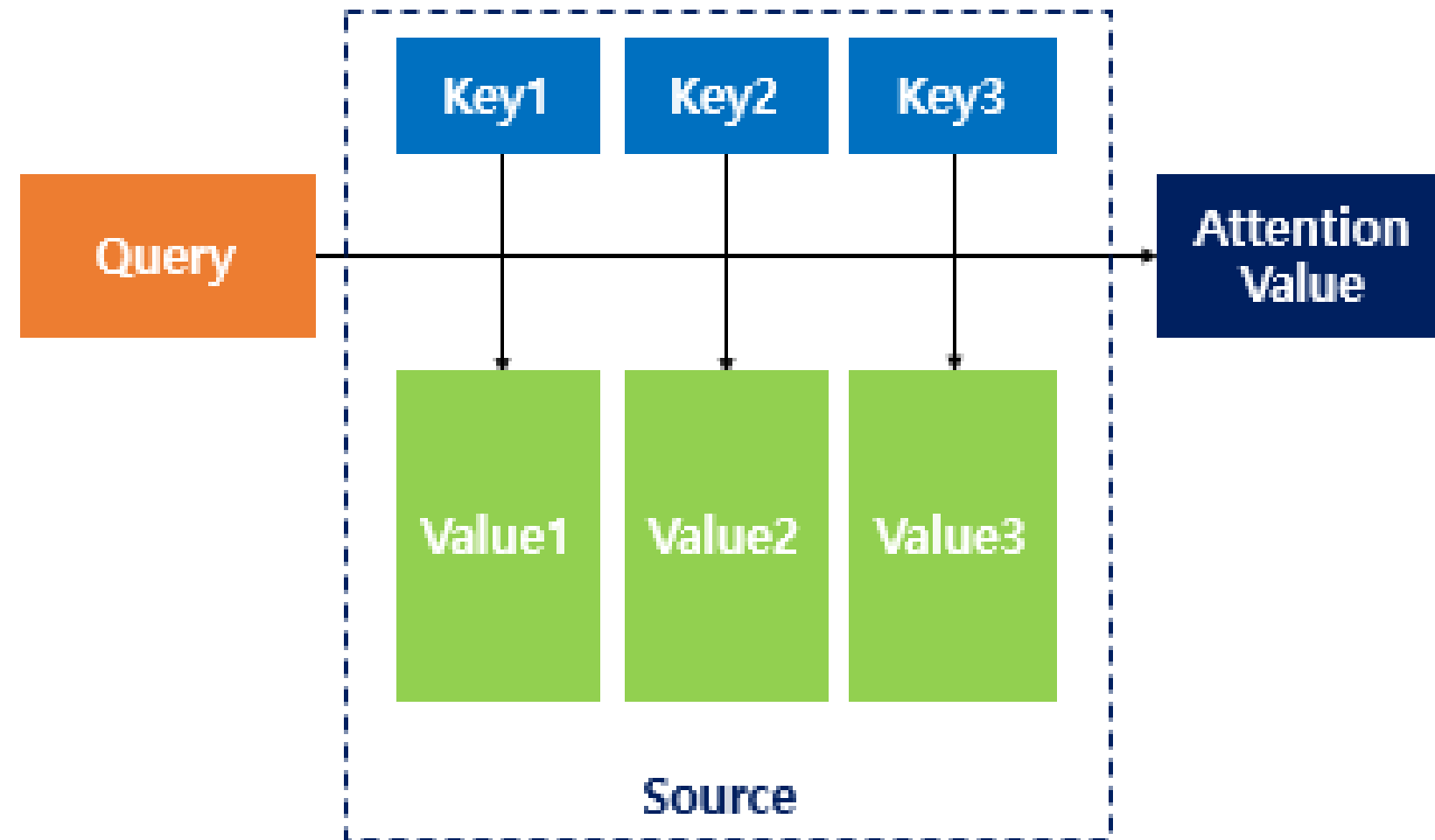
"디코더에서 매 시점마다 인코더에서의 전체 문장을 참고한다!"

"똑같이 참고하는 것이 아니라 연관성이 더 높은 곳에 집중한다!"

두가지 아이디어를 기반으로 만들어진 기법

# Attention Mechanism

## Attention Network



Query

=  $Q$

=  $t$  시점에서 디코더 셀의 은닉 상태

Keys

=  $K$

= 모든 시점에서의 인코더 셀의 은닉 상태

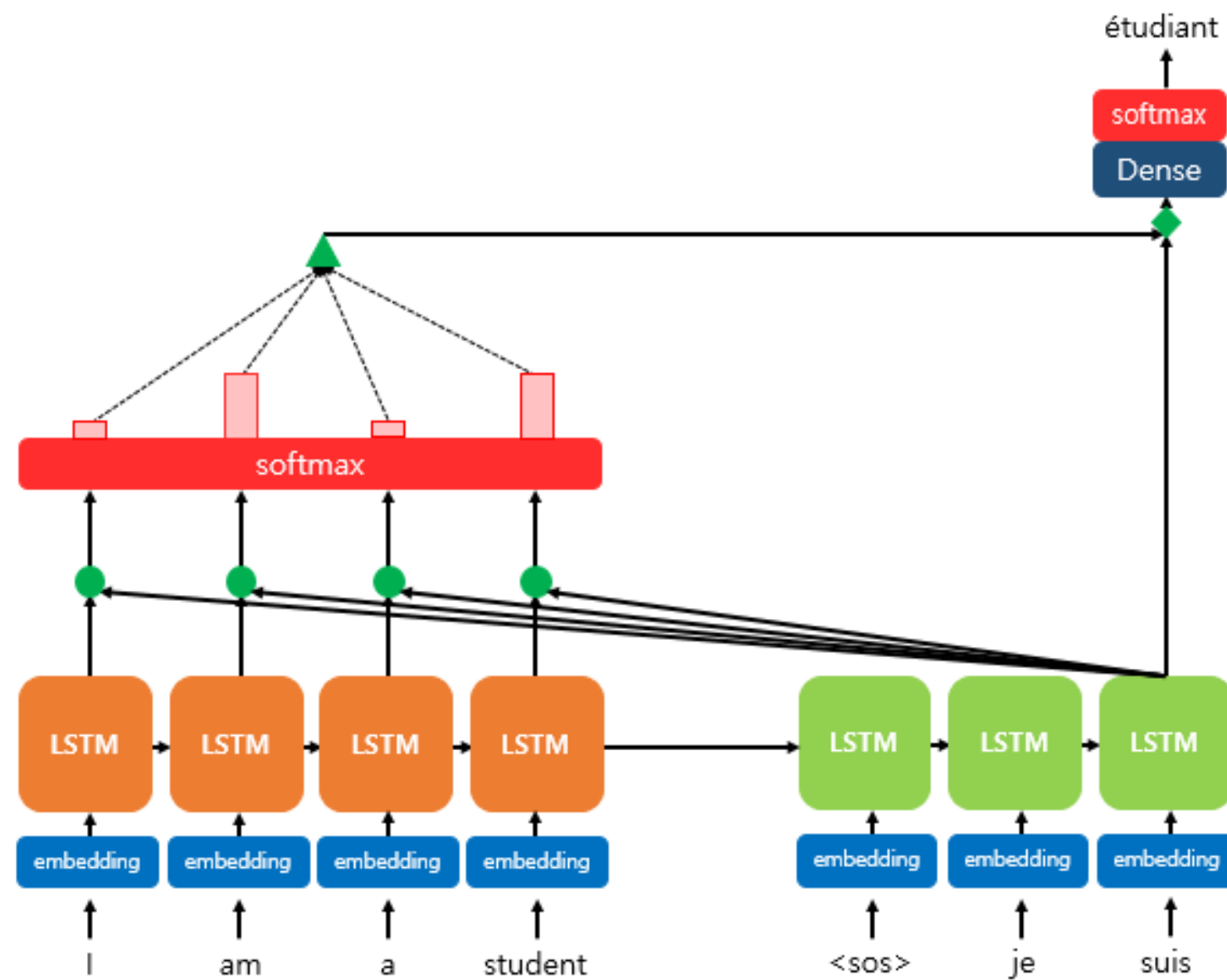
Values

=  $V$

= 모든 시점에서의 인코더 셀의 은닉 상태

# Attention Mechanism

## Attention Mechanism의 과정



- (1) Attention Score 구하기
- (2) Attention Distribution 구하기
- (3) Attention Value 구하기
- (4) Attention Value와 hidden state 연결하기
- (5) 출력층의 입력이 될  $S_t$  계산하기
- (6) Prediction Vector 구하기

# Attention Mechanism

## Attention Mechanism의 과정 (Attention Score)

Attention Score : 디코더의 현재 시점  $t$ 에서 단어를 예측하기 위해 인코더의 각 은닉 상태가 디코더의 현재 시점의 은닉 상태인  $S_t$ 와 얼마나 유사한지를 판단하는 값

### 종류

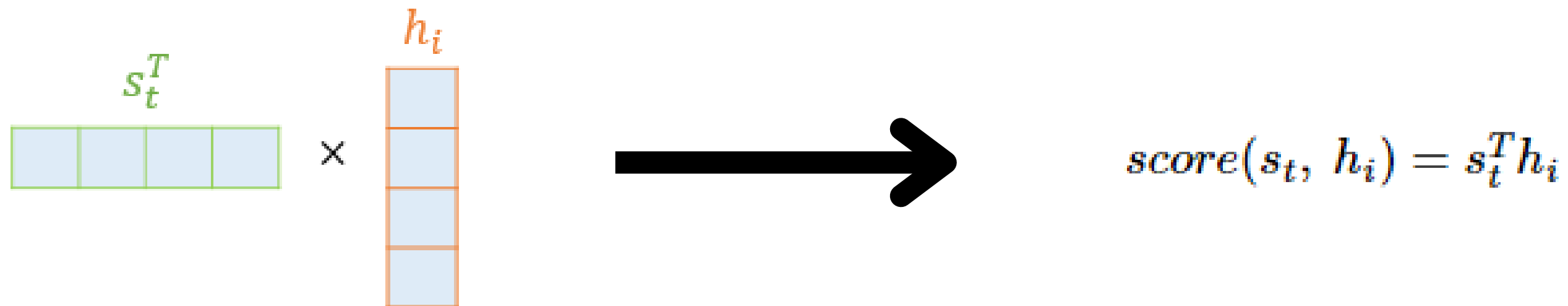
1. Dot-Product Attention
2. Scaled Dot-Product Attention
3. Bahdanau Attention
4. Multiplicative Attention
5. 기타 등등

# Attention Mechanism

## Attention Mechanism의 과정 (Attention Score)

### (1) Dot-Product Attention

transpose한  $S_t$ 와 인코더의 은닉 상태  $h_i$ 에 대해 내적을 수행함  
=> 결과값은 스칼라 값


$$s_t^T \times h_i \rightarrow \text{score}(s_t, h_i) = s_t^T h_i$$



# Attention Mechanism

## Attention Mechanism의 과정 (Attention Score)

### (2) Scaled Dot-Product Attention

transpose한  $S_t$ 와 인코더의 은닉 상태  $h_i$ 에 대해 내적을 수행한 후 차원 개수로 나눔  
 => 결과값은 스칼라 값

$$\text{softmax} \left( \frac{Q \times K^T}{\sqrt{d_k}} \right) \times V = \text{Attention Value Matrix } a$$

The diagram shows the following components and operations:

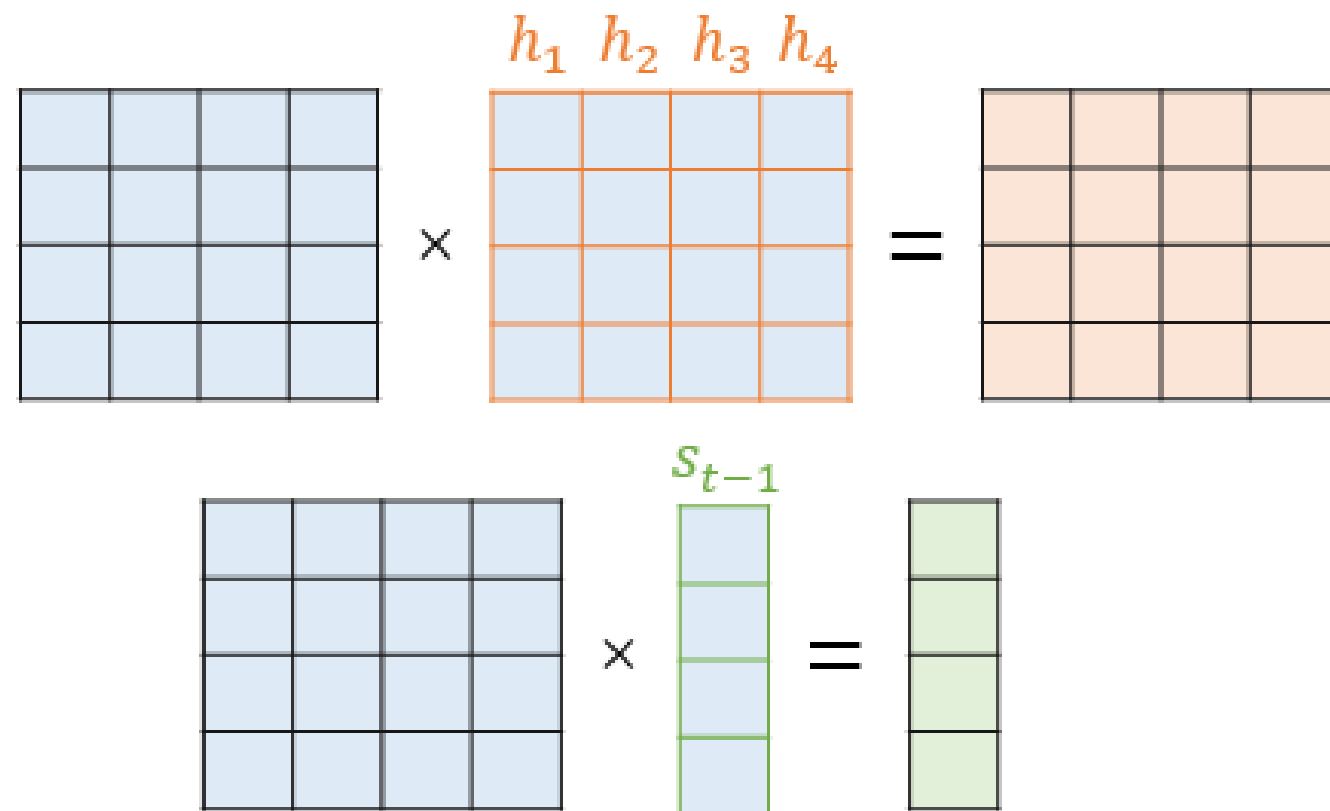
- Q**: A yellow 4x2 matrix representing the query.
- $K^T$** : An orange 2x4 matrix representing the transposed key.
- Operation**:  $Q \times K^T$  is performed, followed by division by  $\sqrt{d_k}$  (where  $d_k = 4$ ).
- V**: A green 4x2 matrix representing the value.
- Result**: The final product is the **Attention Value Matrix  $a$** , shown as a blue 4x2 matrix.

# Attention Mechanism

## Attention Mechanism의 과정 (Attention Score)

### (3) Additive Attention = Bahdanau Attention

Q가 t 시점이 아니라 t-1 시점의 디코더 셀의 은닉 상태가 됨  
 $s_{t-1}$ 과  $h_i$ 를 각각 가중치에 곱하고, 둘을 더한 값에 tanh를 씌우고 가중치를 곱함



$$W_a^T \tanh(W_b s_{t-1} + W_c H)$$

# Attention Mechanism

Attention Mechanism의 과정 (Attention Score)

(4) Multiplicative Attention = General Attention

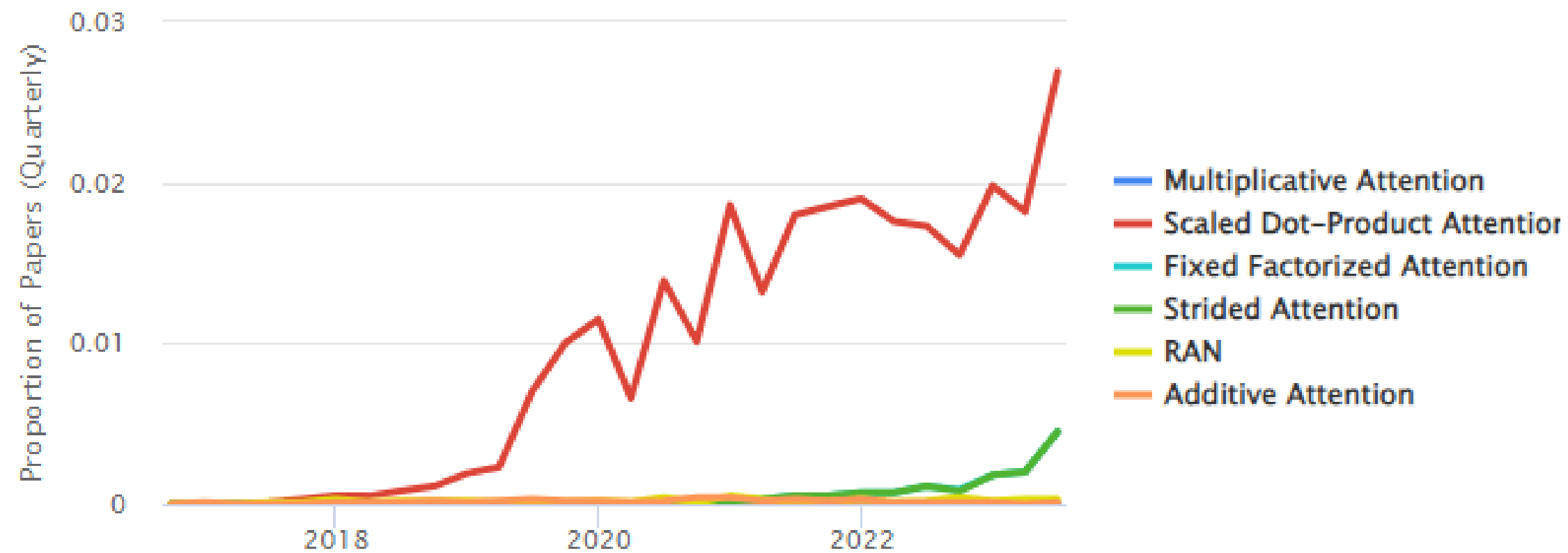
dot-product attention에 가중치  $W$ 를 붙인 형태

$$\mathbf{h}_i^T \mathbf{W}_a \mathbf{s}_j$$

# Attention Mechanism

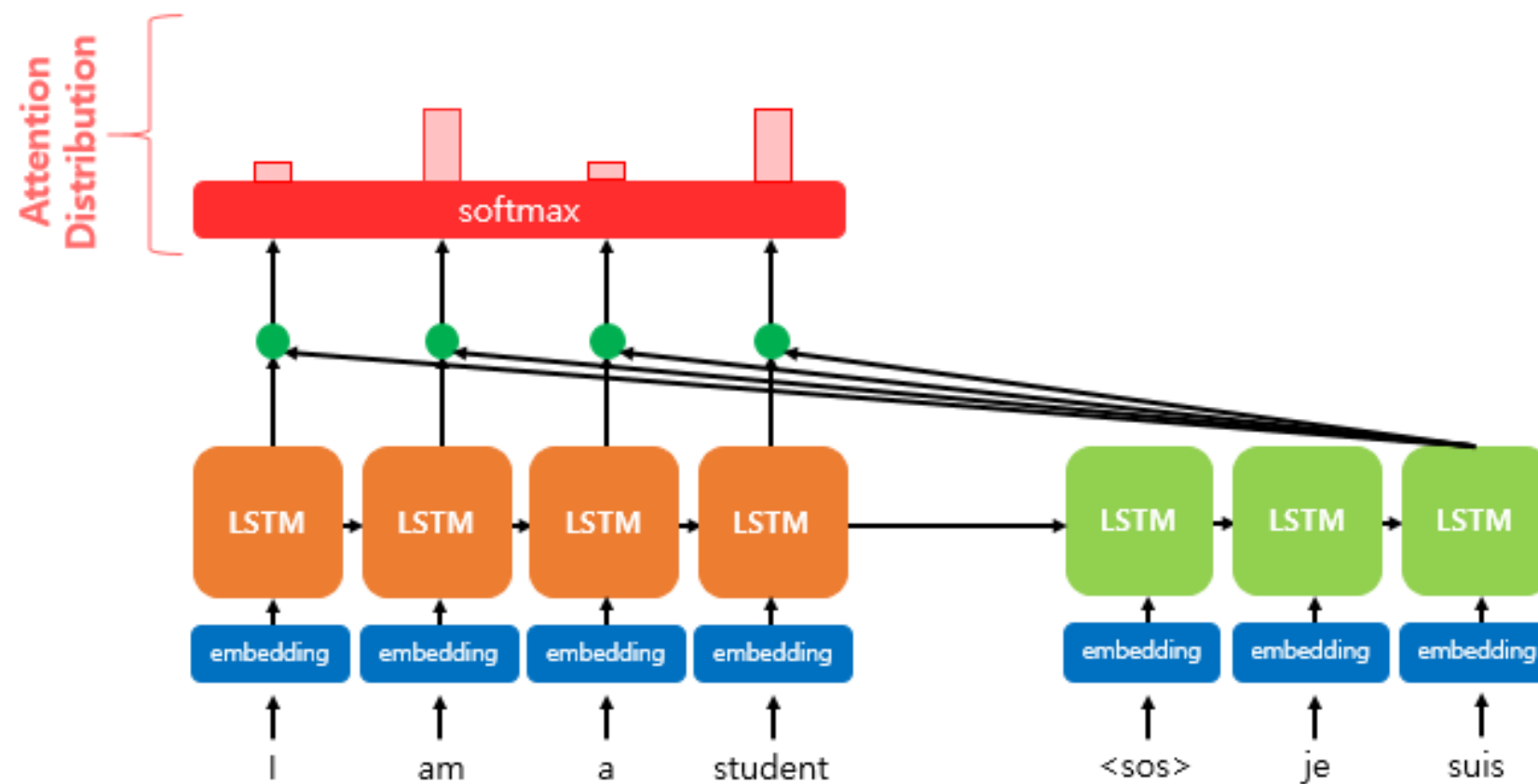
Attention Mechanism의 과정 (Attention Score)

Attention 종류 별 사용 추세



# Attention Mechanism

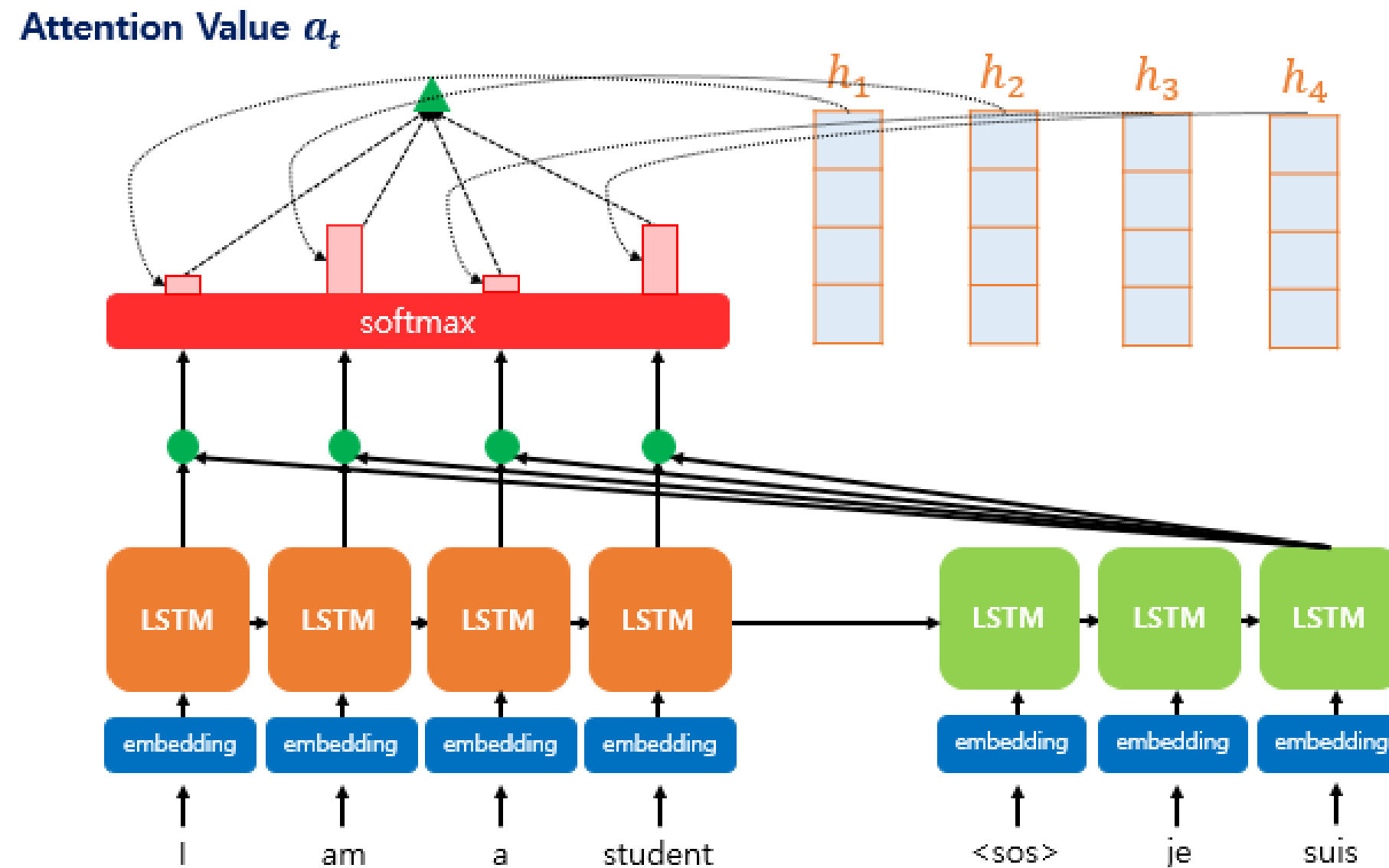
## Attention Mechanism의 과정 (Attention Distribution)



$$\alpha^t = \text{softmax}(e^t)$$

# Attention Mechanism

## Attention Mechanism의 과정 (Attention Value)

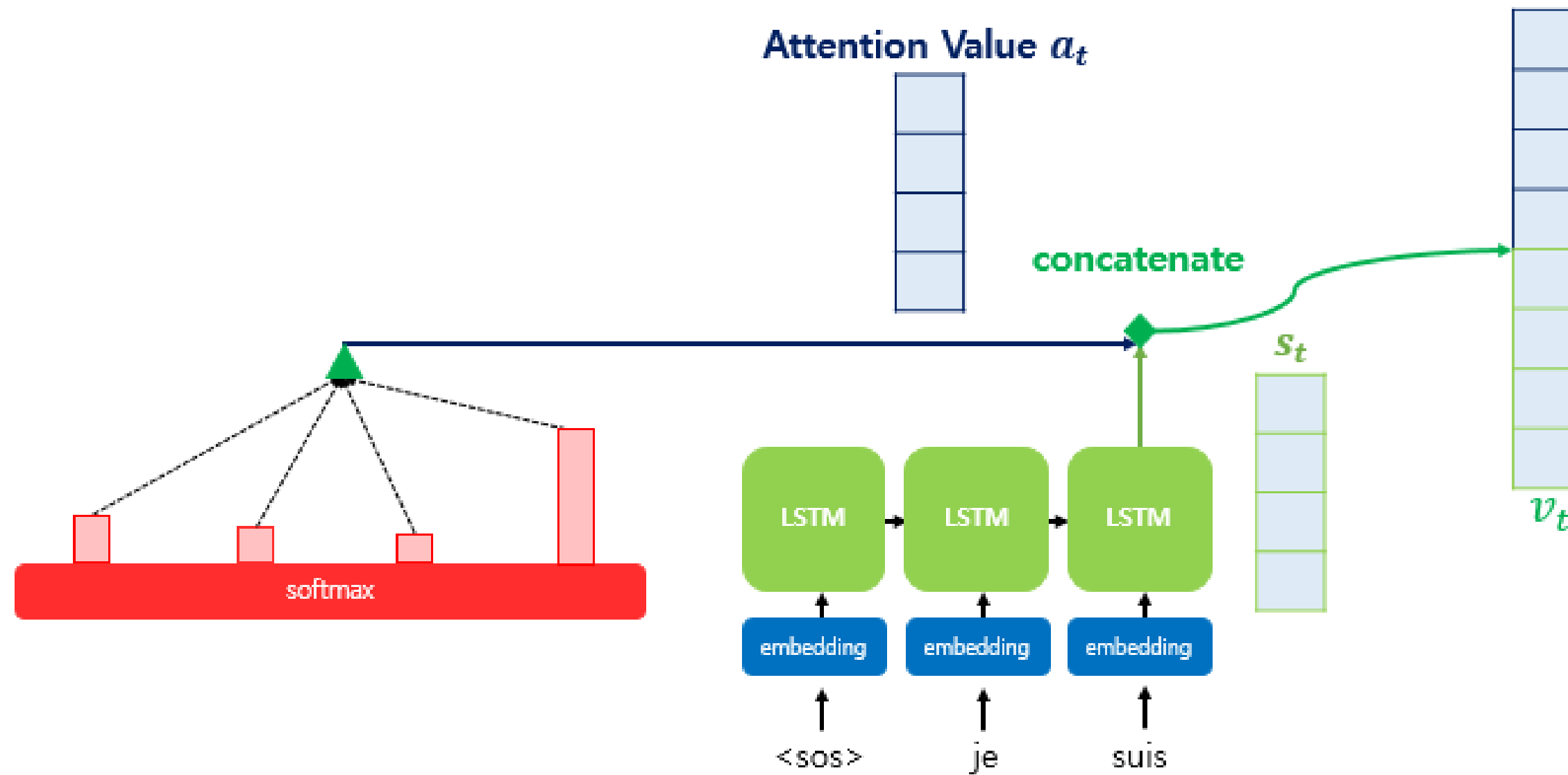


$$a_t = \sum_{i=1}^N \alpha_i^t h_i$$

이 때 attention value를  
context vector라고 하기도 함

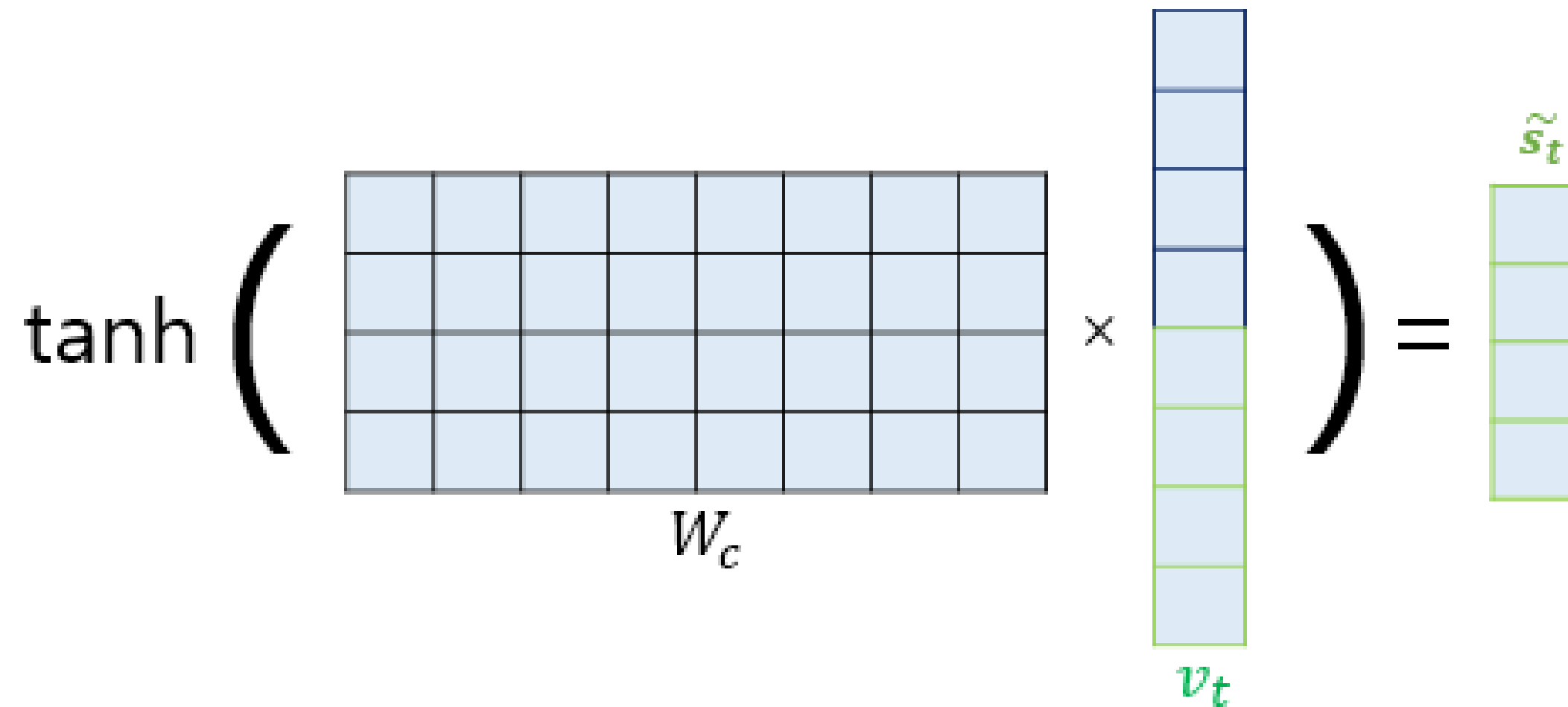
# Attention Mechanism

## Attention Mechanism의 과정 (Concatenation)



# Attention Mechanism

Attention Mechanism의 과정 (tanh)



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



# Attention Mechanism

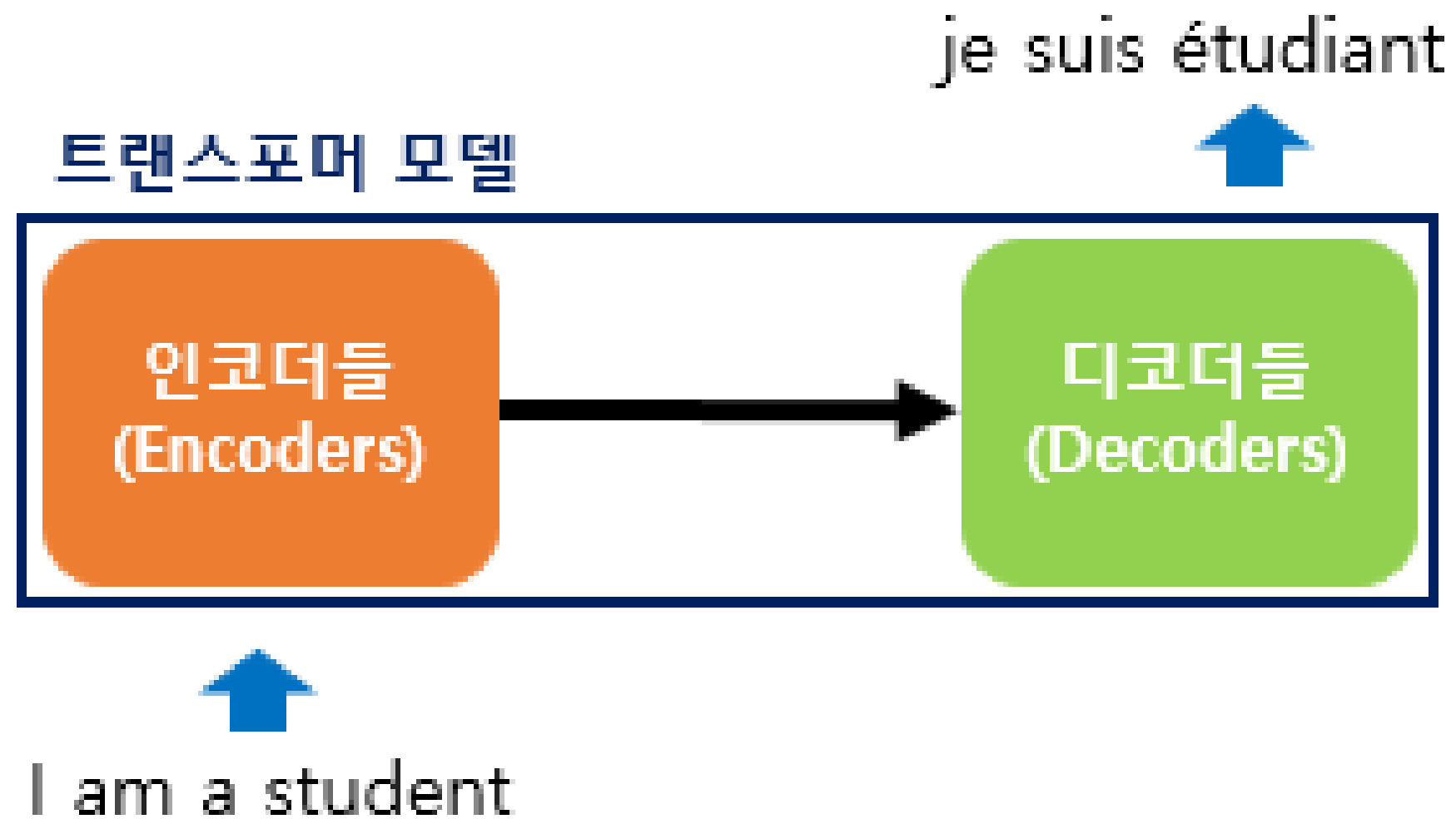
Attention Mechanism의 과정 (Prediction Vector)

$$\hat{y}_t = \text{Softmax}(W_y \tilde{s}_t + b_y)$$

# Transformer

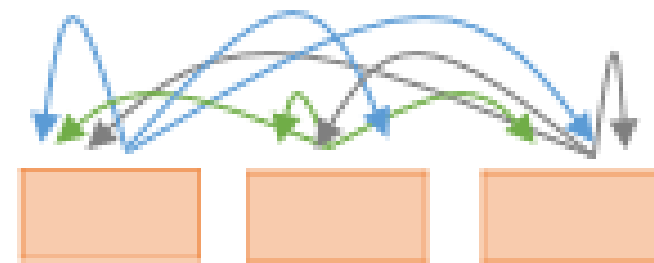
## Transformer란

문장 속 단어와 같은 순차 데이터 내의 관계를 추적해 맥락과 의미를 학습하는 신경망



# Transformer

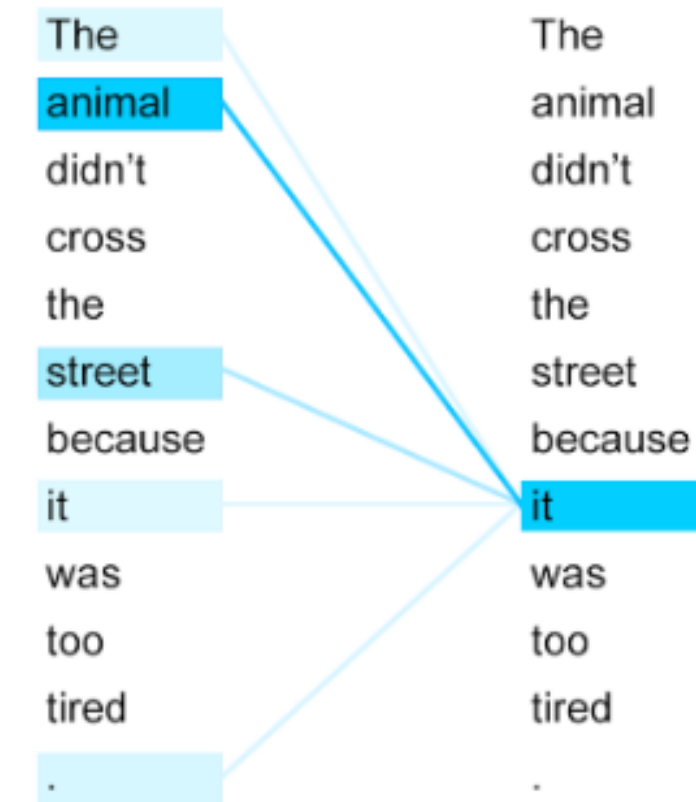
## Transformer의 특별한 Attention : Self-Attention



Encoder Self-Attention

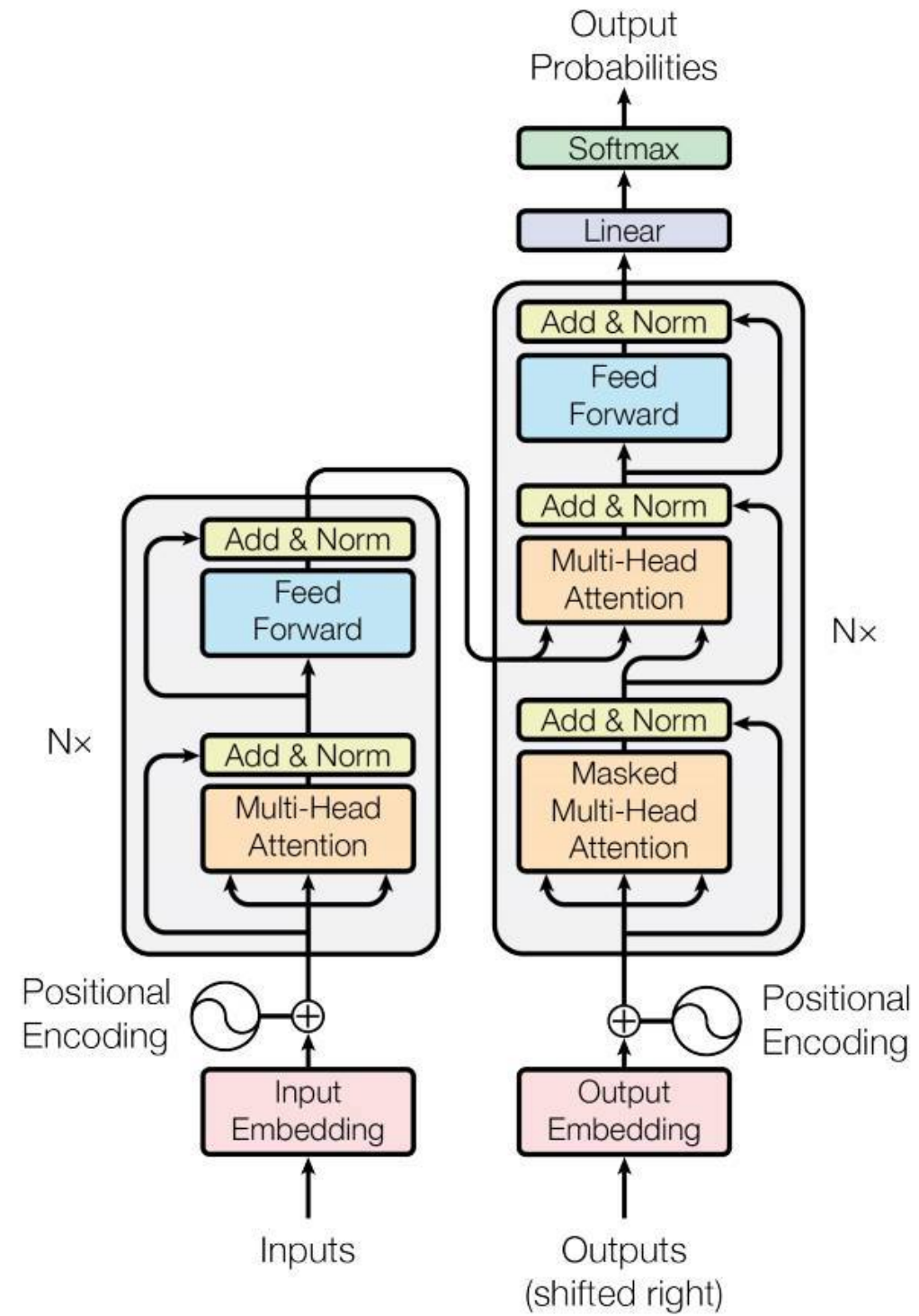
Q, V, W : 입력 문장의  
모든 단어 벡터들

하나의 입력 문장 내에서  
특정 단어 간 유사도를 구함



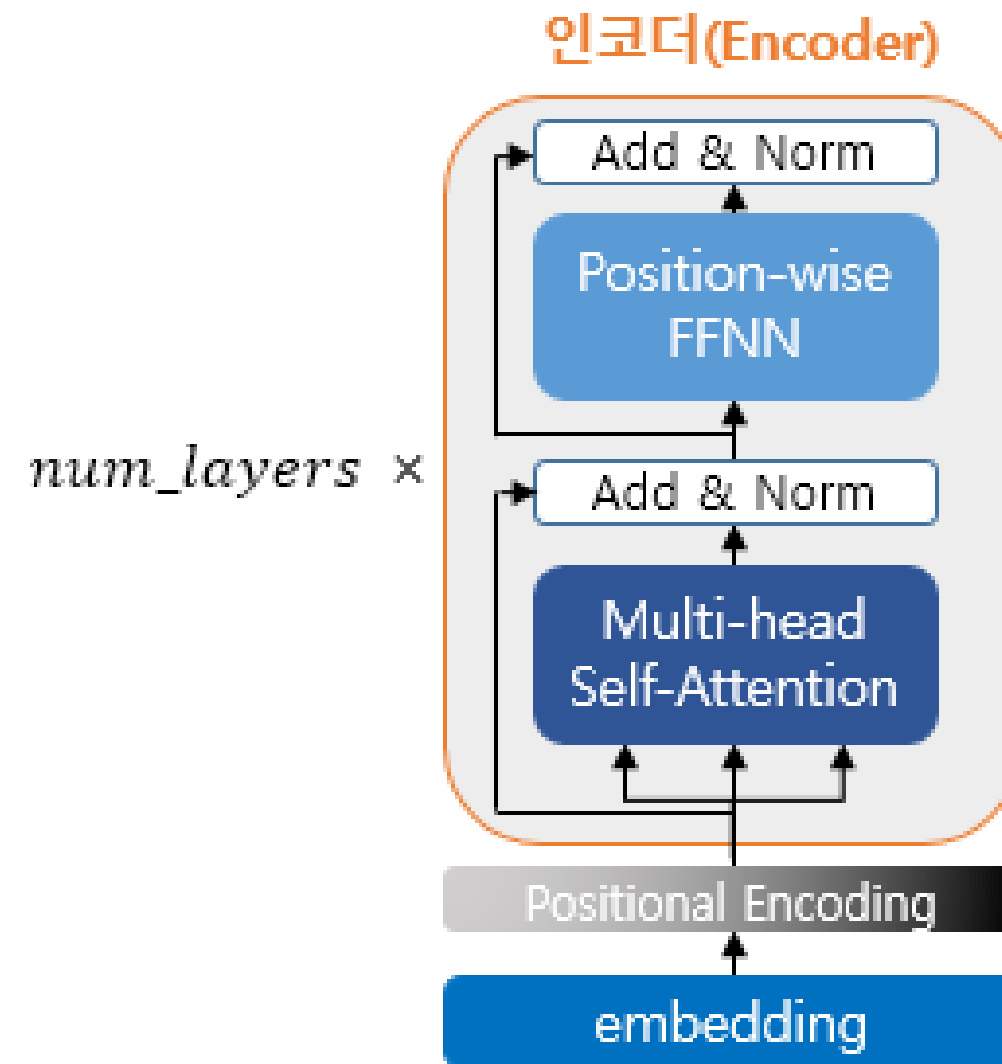
# Transformer

## Transformer의 구조



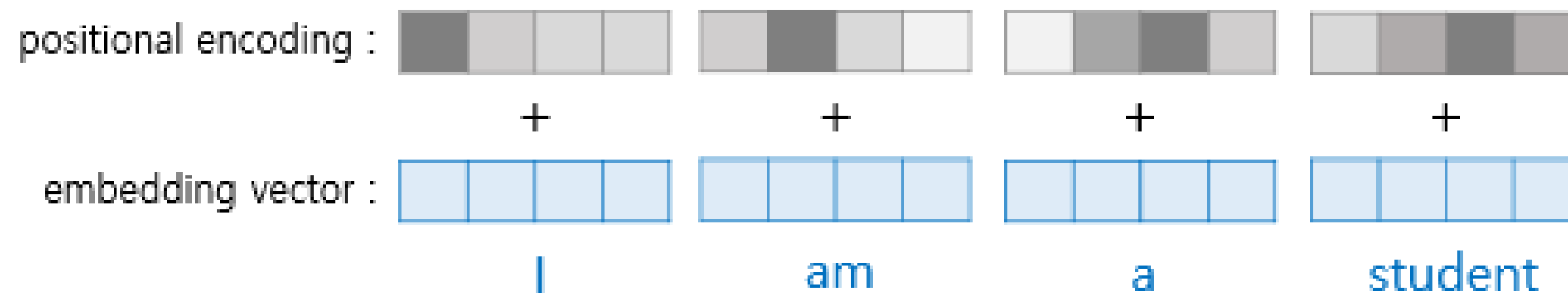
# Transformer

## Transformer의 구조 (Encoder)



# Transformer

## Transformer의 구조 (Encoder) – Positional Encoding

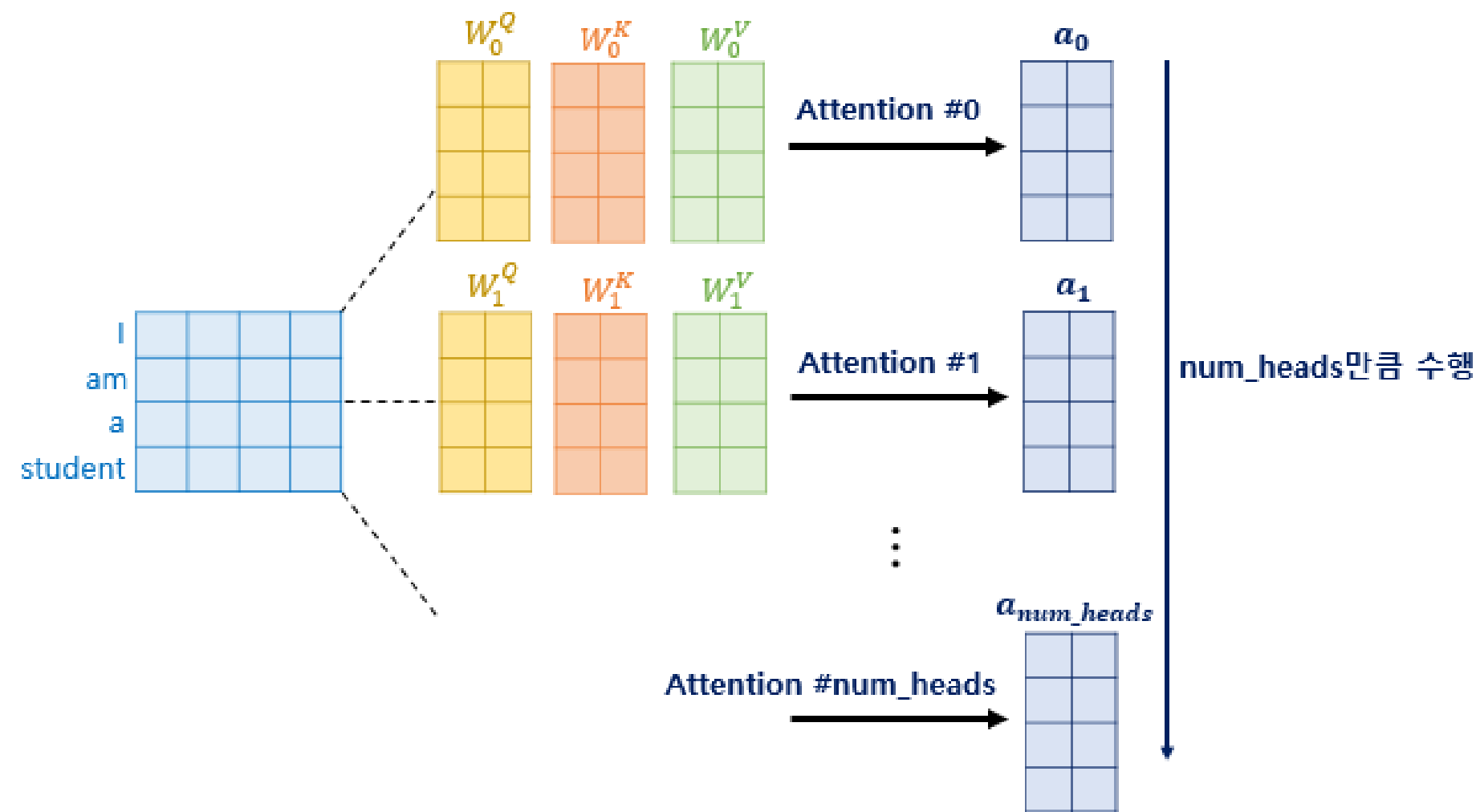


$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{model}})$$

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

# Transformer

## Transformer의 구조 (Encoder) – Multi-head Self-Attention



# Transformer

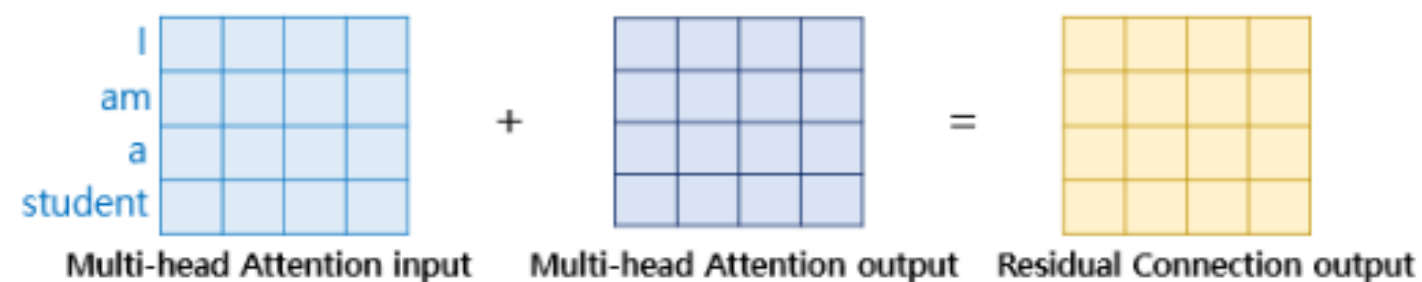
## Transformer의 구조 (Encoder) – Add & Norm

Add : Residual Connection

Norm : Layer Normalization

$$H(x) = x + \text{Multi-head Attention}(x)$$

$$LN = \text{LayerNorm}(x + \text{Sublayer}(x))$$



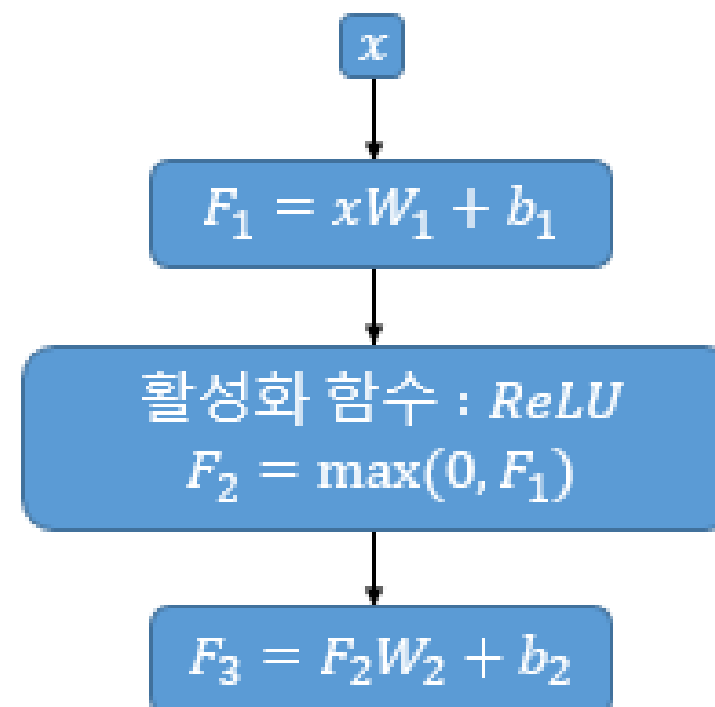
$$\hat{x}_{i,k} = \frac{x_{i,k} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}$$



# Transformer

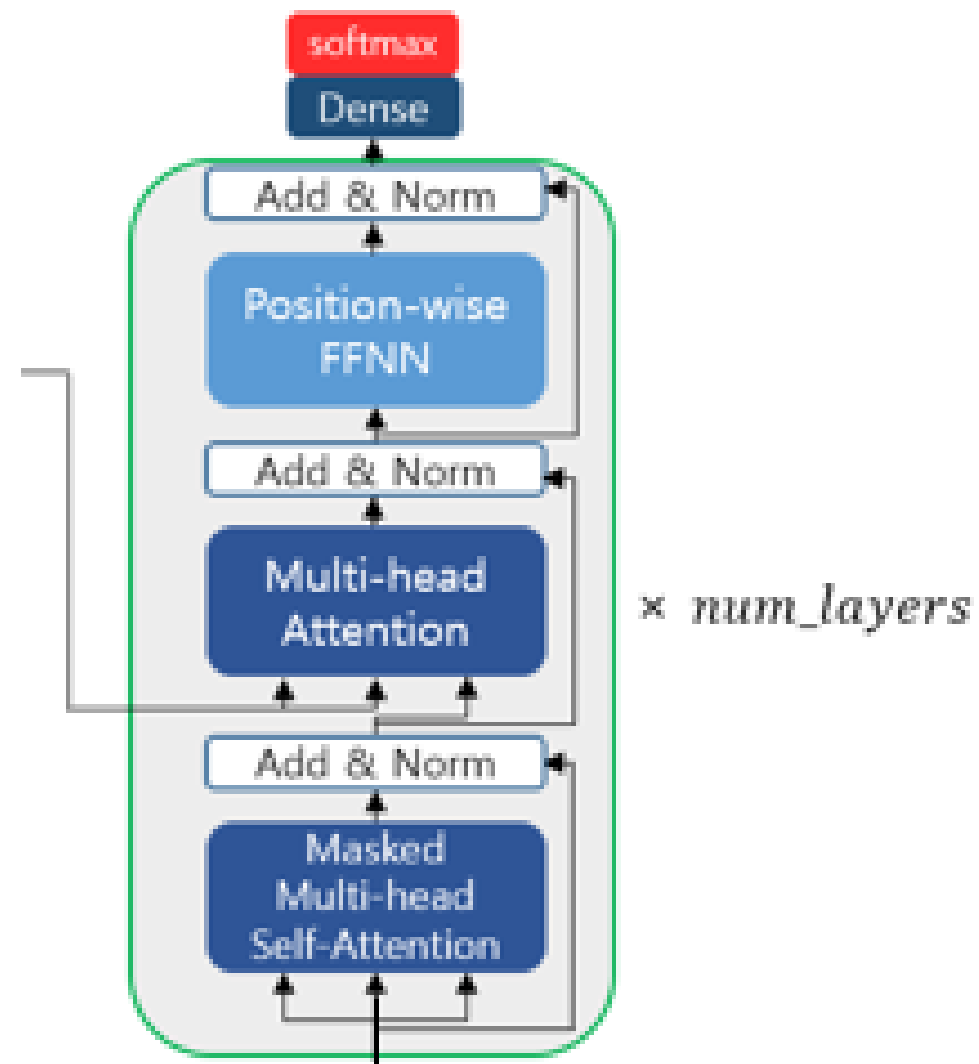
## Transformer의 구조 (Encoder) – Position-wise FFNN

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



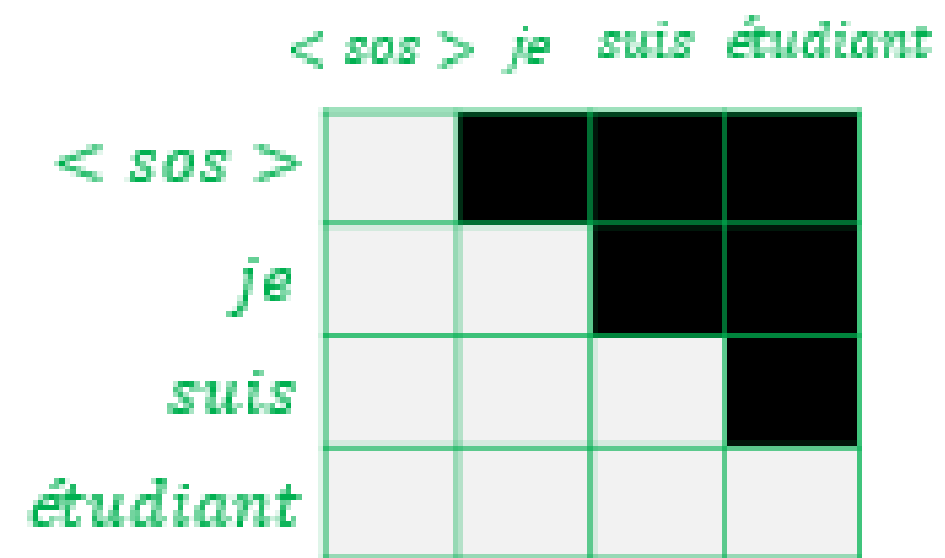
# Transformer

## Transformer의 구조 (Decoder)



# Transformer

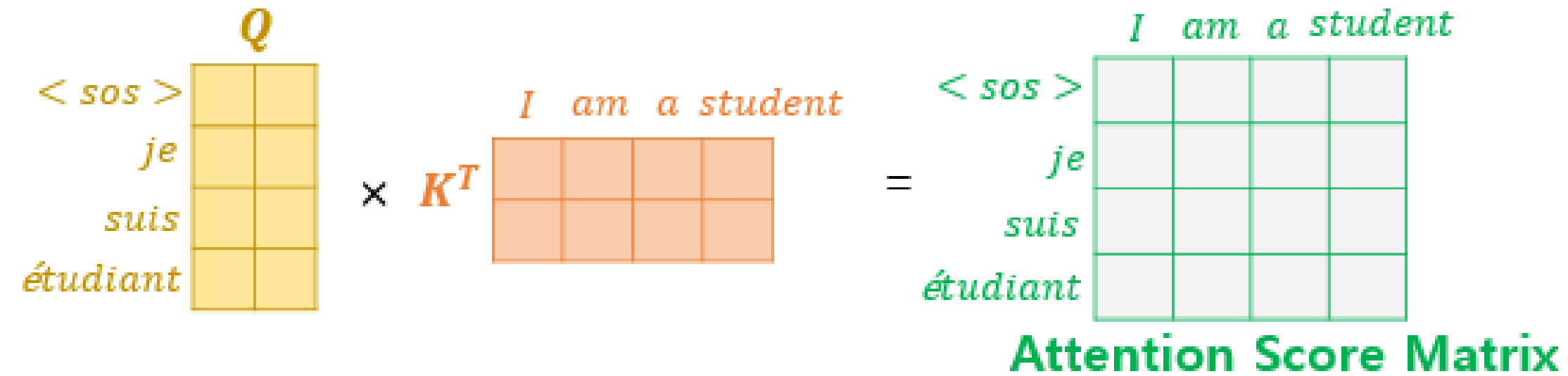
## Transformer의 구조 (Decoder) – Masked Multi-head Self-Attention



Attention Score Matrix

# Transformer

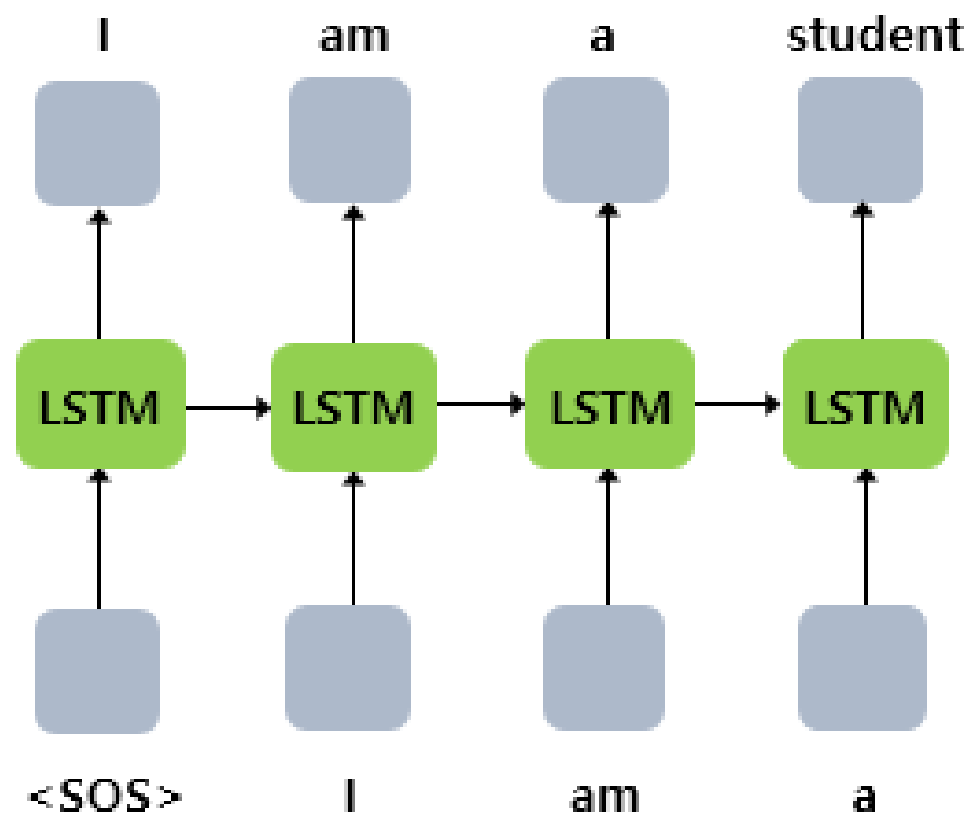
## Transformer의 구조 (Decoder) – Multi-head Attention



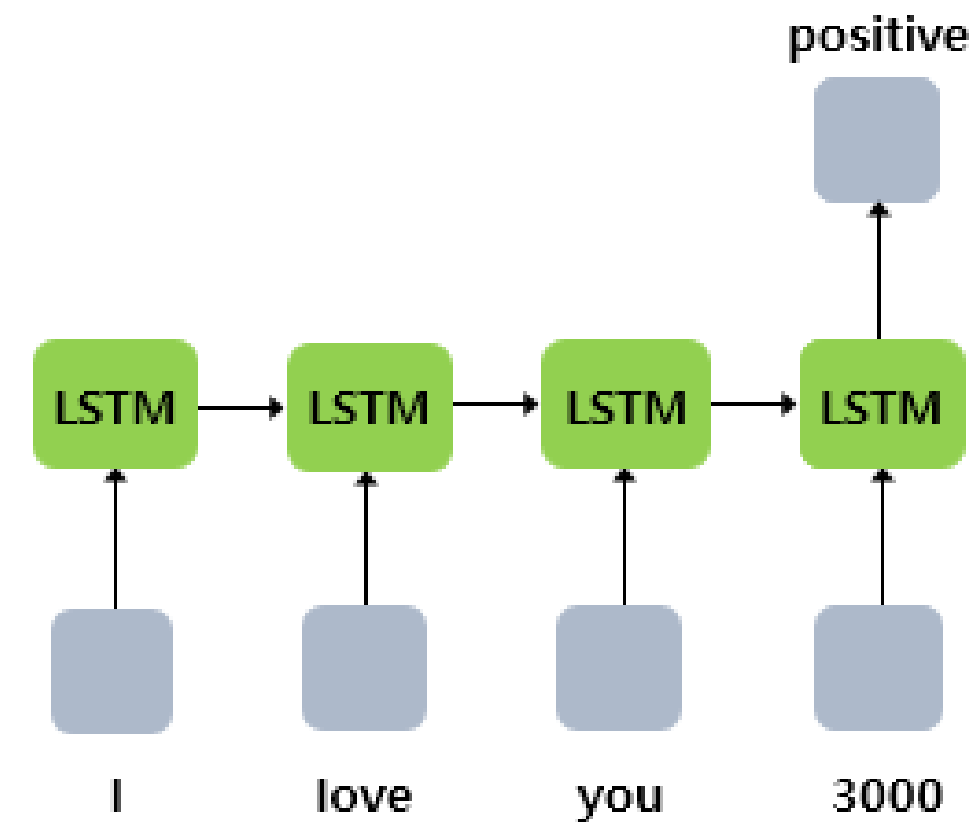
# BERT

## Pretrained Model

Semi-Supervised Sequence Learning, Google, 2015



LSTM 언어 모델을 사전 훈련



분류 문제에 파인 튜닝

# BERT

## Tokenizer

문장의 단어나 형태소 등을 기준으로 각각 토큰으로 만들어주는 것

BERT에서는 단어보다 더 작은 단위로 쪼개는 서브워드 토크나이즈를 사용함

```
result = tokenizer.tokenize('Here is the sentence I want embeddings  
for.')  
print(result)
```

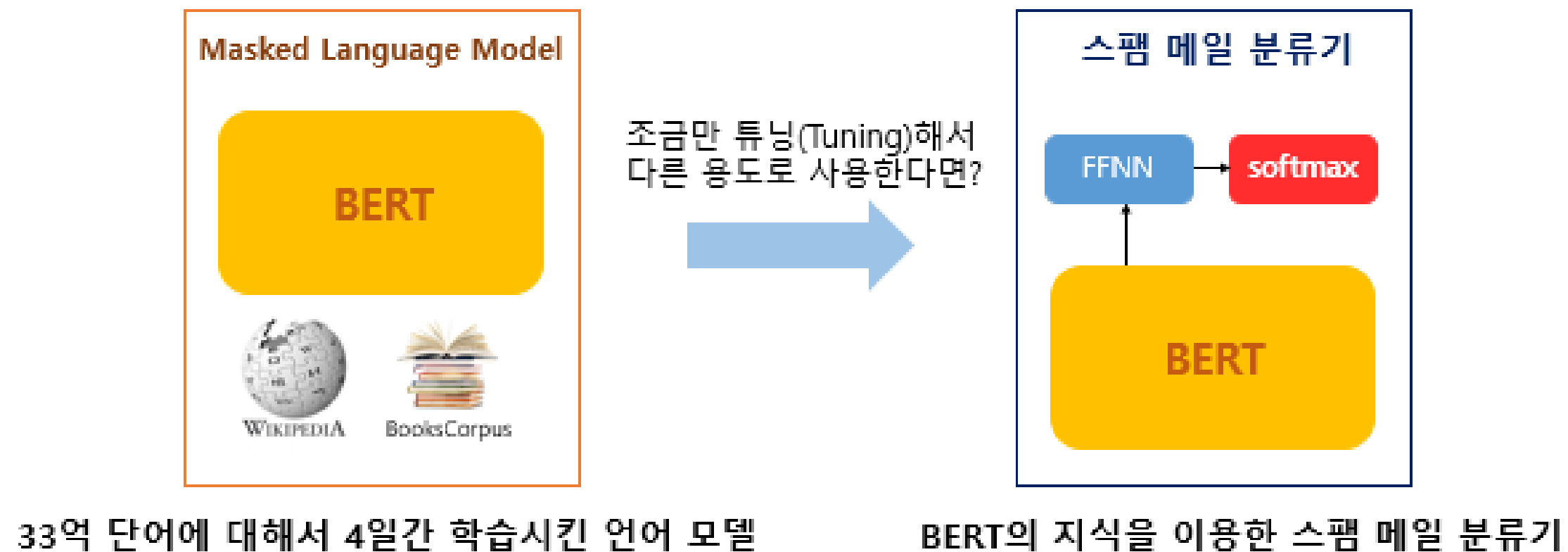
```
['here', 'is', 'the', 'sentence', 'i', 'want', 'em', '##bed', '##di  
ng', '##s', 'for', '.']
```

# BERT

## BERT란

Bidirectional Encoder Representations from Transformers

2018년에 구글이 공개한 pre-trained model로, SOTA의 지위를 획득했었음



# BERT

## Contextual Embedding

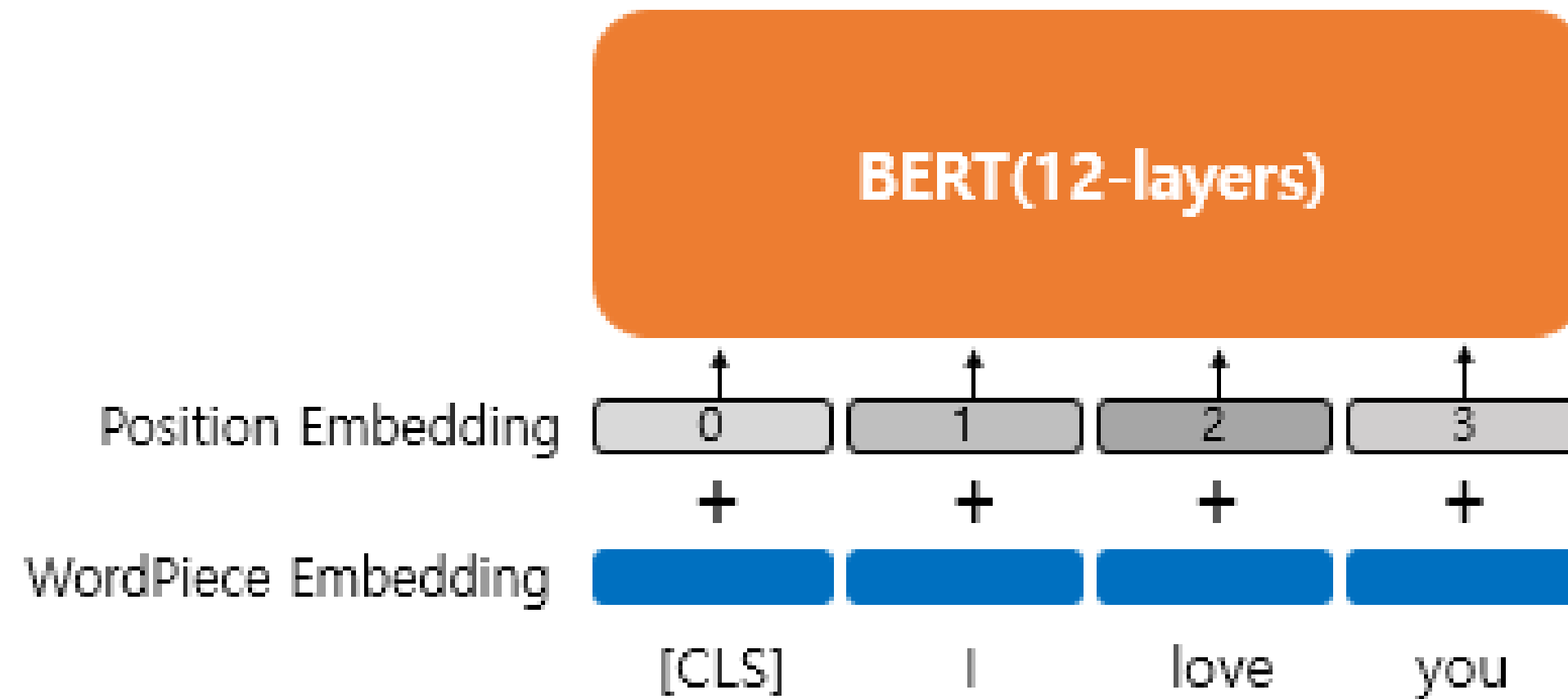
문맥을 반영한 임베딩으로, self-attention을 통해 얻게 됨



# BERT

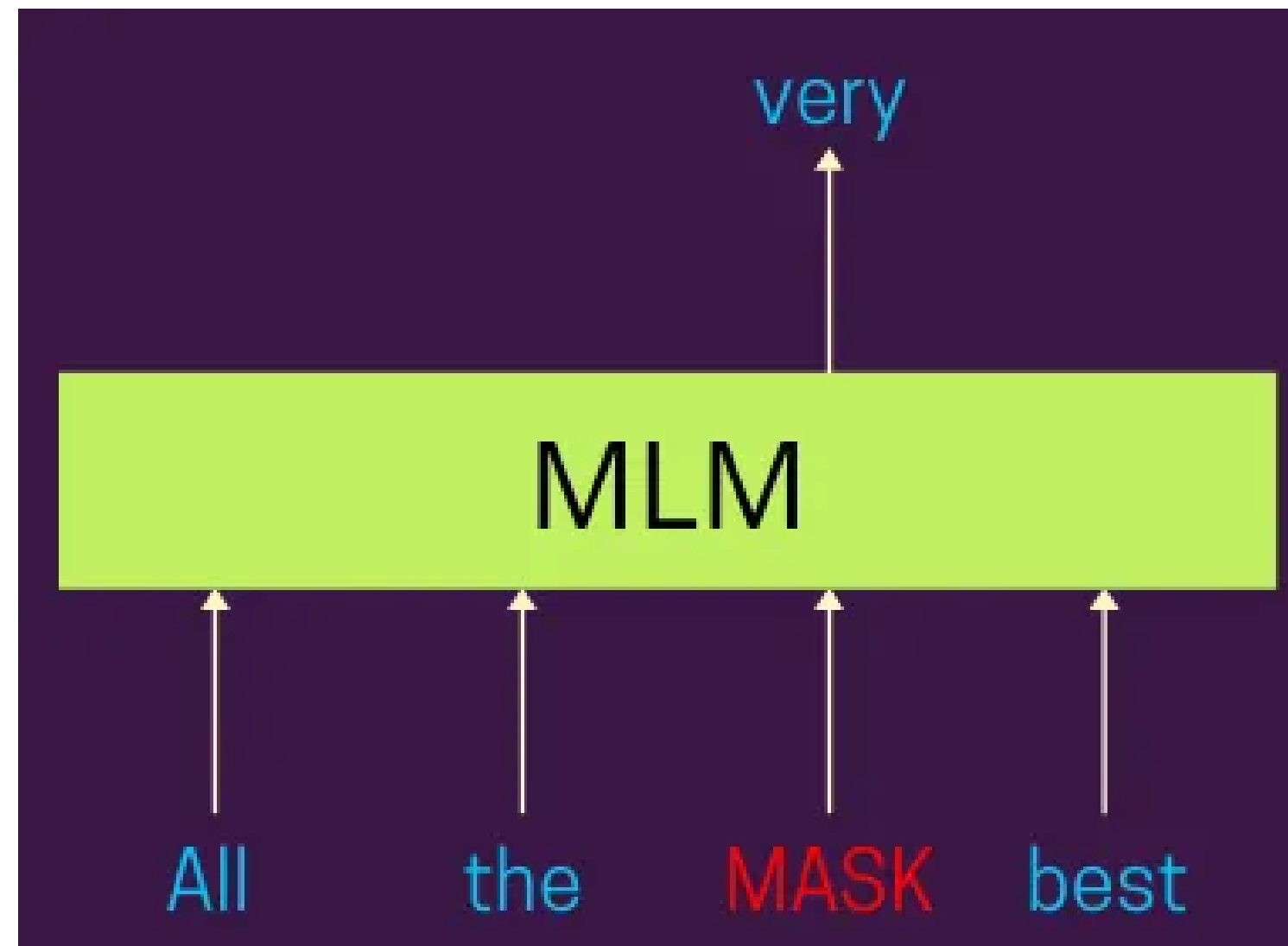
## Position Embedding

transformer의 Positional Encoding과는 다름.  
단어 임베딩에 위치 정보를 추가해줌



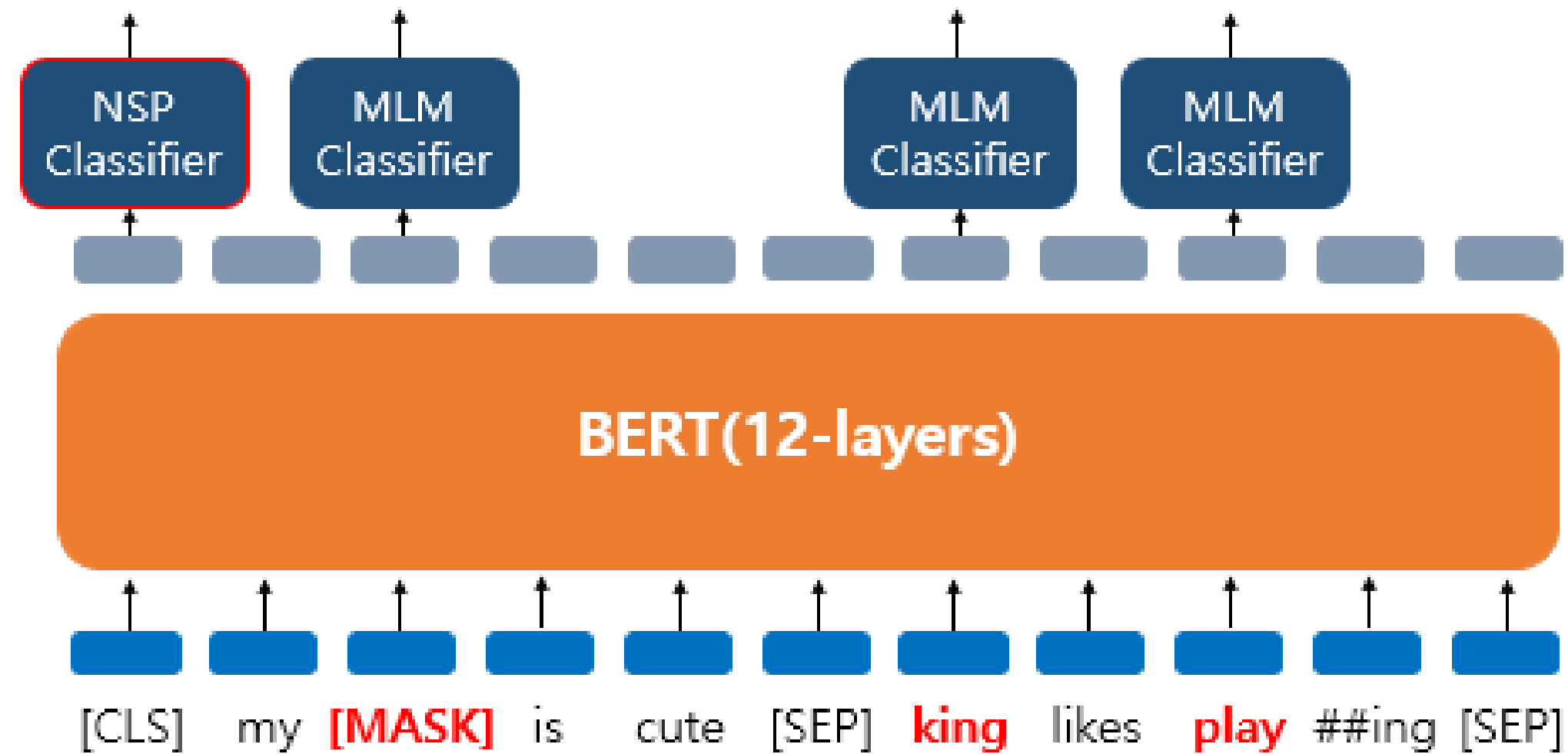
# BERT

## Masked Language Model (MLM)



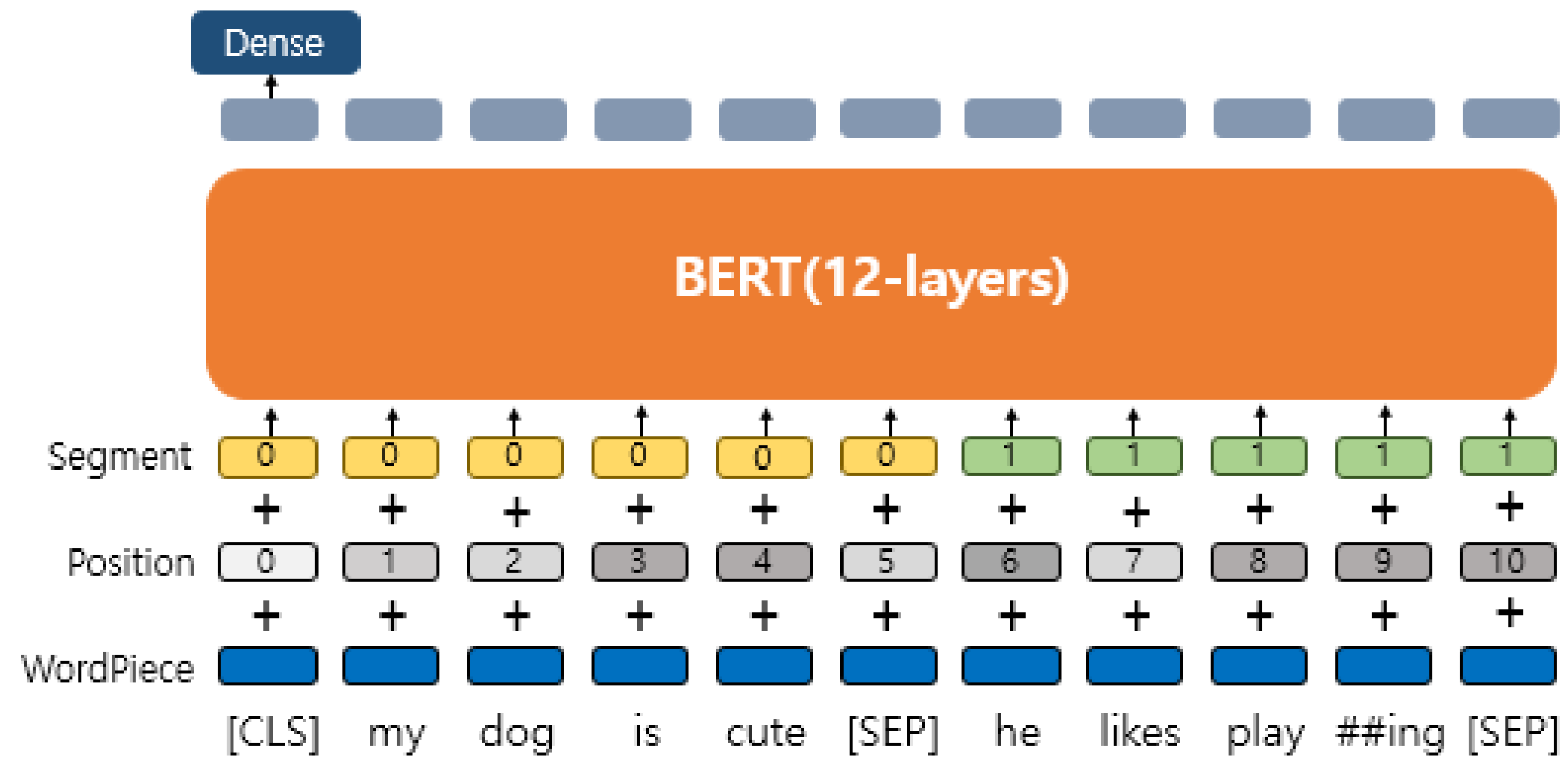
# BERT

## Next Sentence Prediction (NSP)



# BERT

## Segment Embedding



# BERT

## Fine-tuning

다른 작업에 대해서 파라미터 재조정을 위한 추가 훈련 과정

# 과제

오늘 배운

**Attention, Transformer, BERT**  
**정리하고 요약하기**

NEKA

**THANK YOU**