



”

# week7

Bi-LSTM, Bi-RNN, Seq2Seq, Attention



# 발제 순서

”

01

Bi-directional?

02

Bi-RNN

03

Bi-LSTM

04

Seq2Seq

05

Attention

06

실습!



”

  
**Bi-Directional**

**01**

양방향?

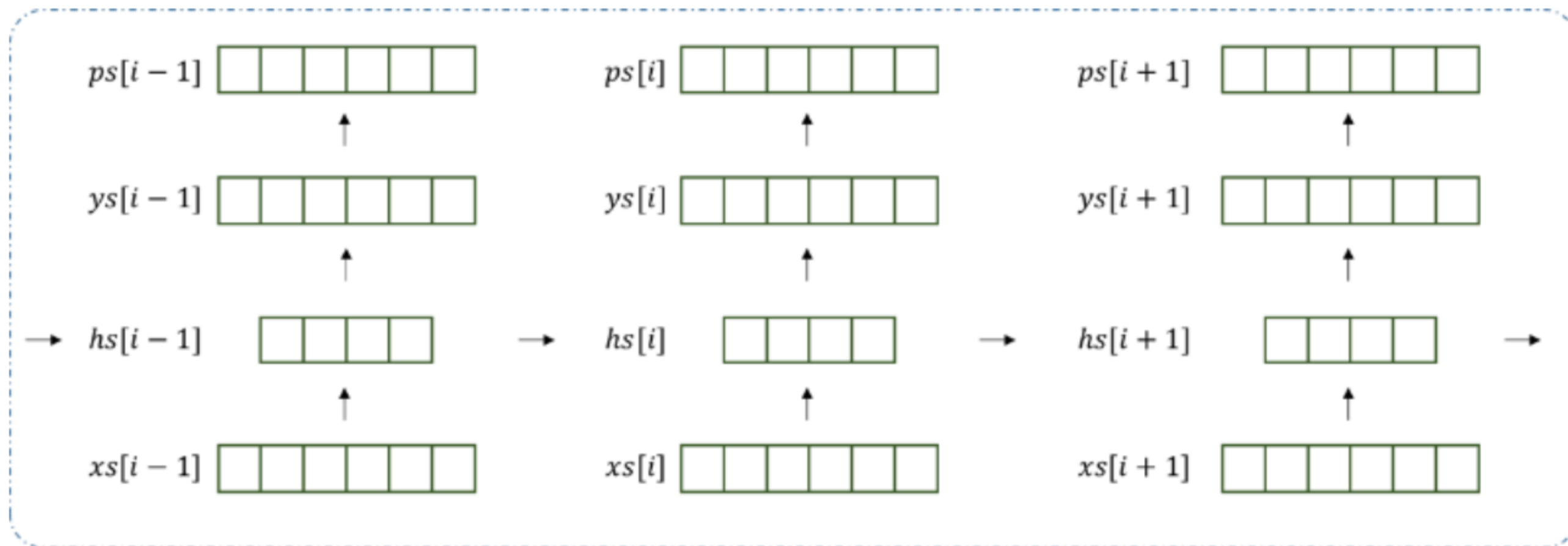


## Bi- Directional



나는 \_\_\_\_\_ 를 뒤집어쓰고 펑펑 울었다.

# 기존의 RNN



현시점보다 미래 시점인  
데이터는 추론하지 못한다는 단점 있음.

$$ps[i] = \text{softmax}(ys[i])$$

$V \times 1 \qquad V \times 1$

$$ys[i] = W_{hy} hs[i] + by$$

$V \times 1 \quad V \times H \quad H \times 1 \quad V \times 1$

$$hs[i] = \tanh(W_{xh} xs[i] + W_{hh} h[i-1] + bh)$$

$H \times 1 \quad H \times V \quad V \times 1 \quad H \times H \quad H \times 1 \quad H \times 1$

Hidden Size = H  
Vocabulary size = V



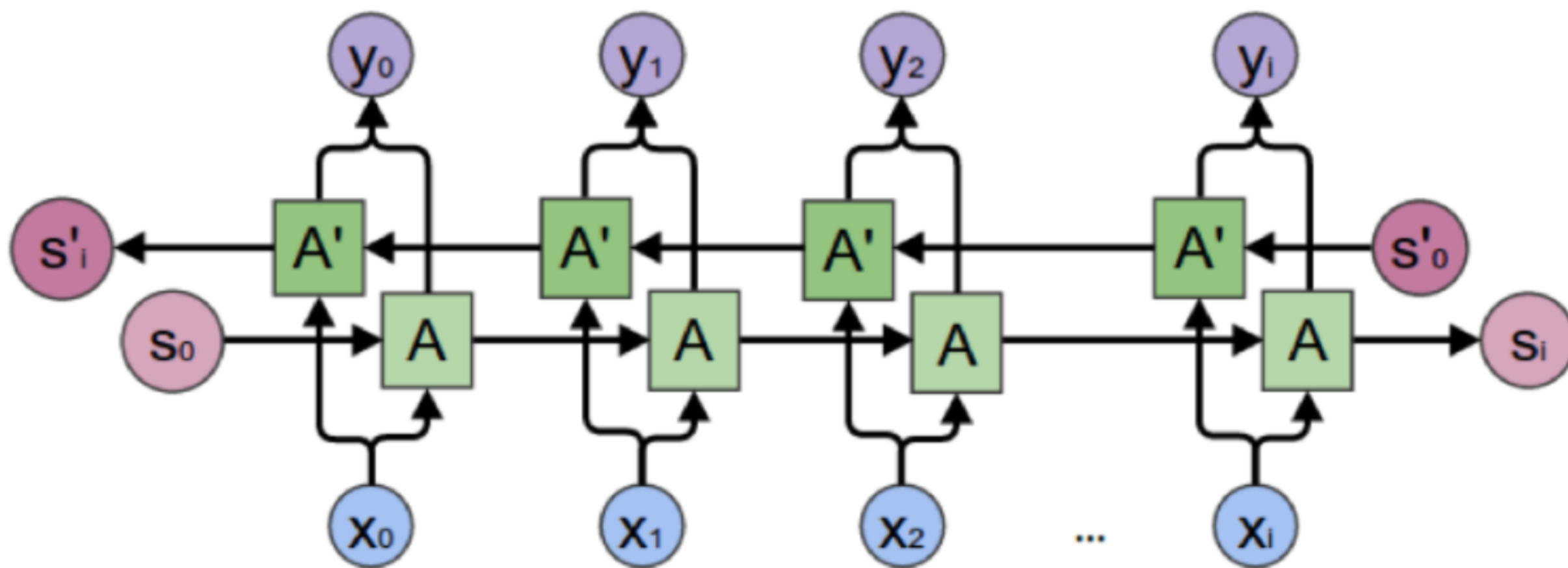
”

**Bi-RNN**

02

# Bi-directional RNN

두 개의 RNN 을 서로 합친 모델.  
한 RNN 에 대해서는 정방향으로 입력,  
다른 RNN에 대해서는 역방향으로 입력되어 합쳐진 후 출력.





”

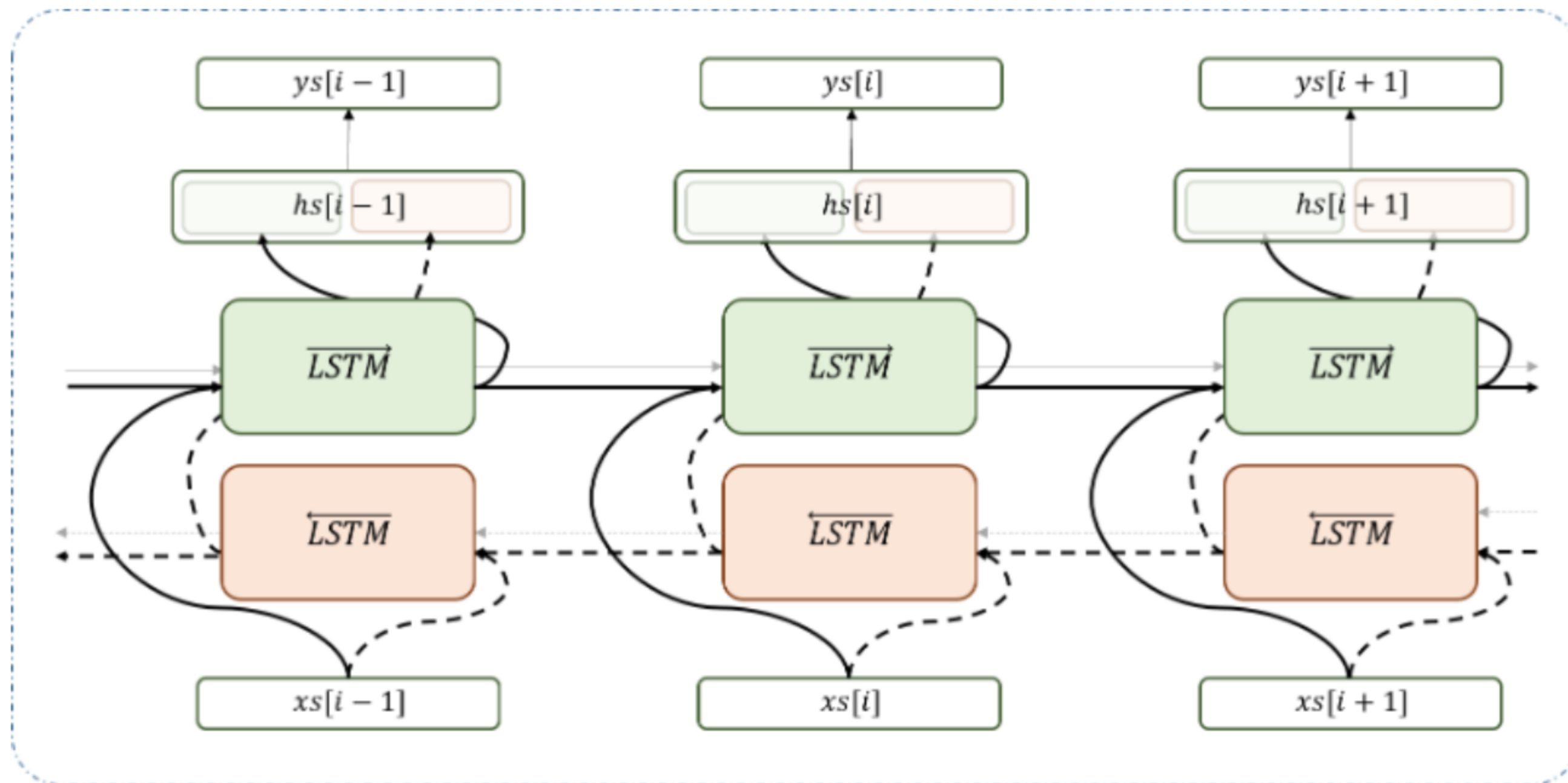
**Bi-LSTM**

03





# Bi-directional LSTM





”

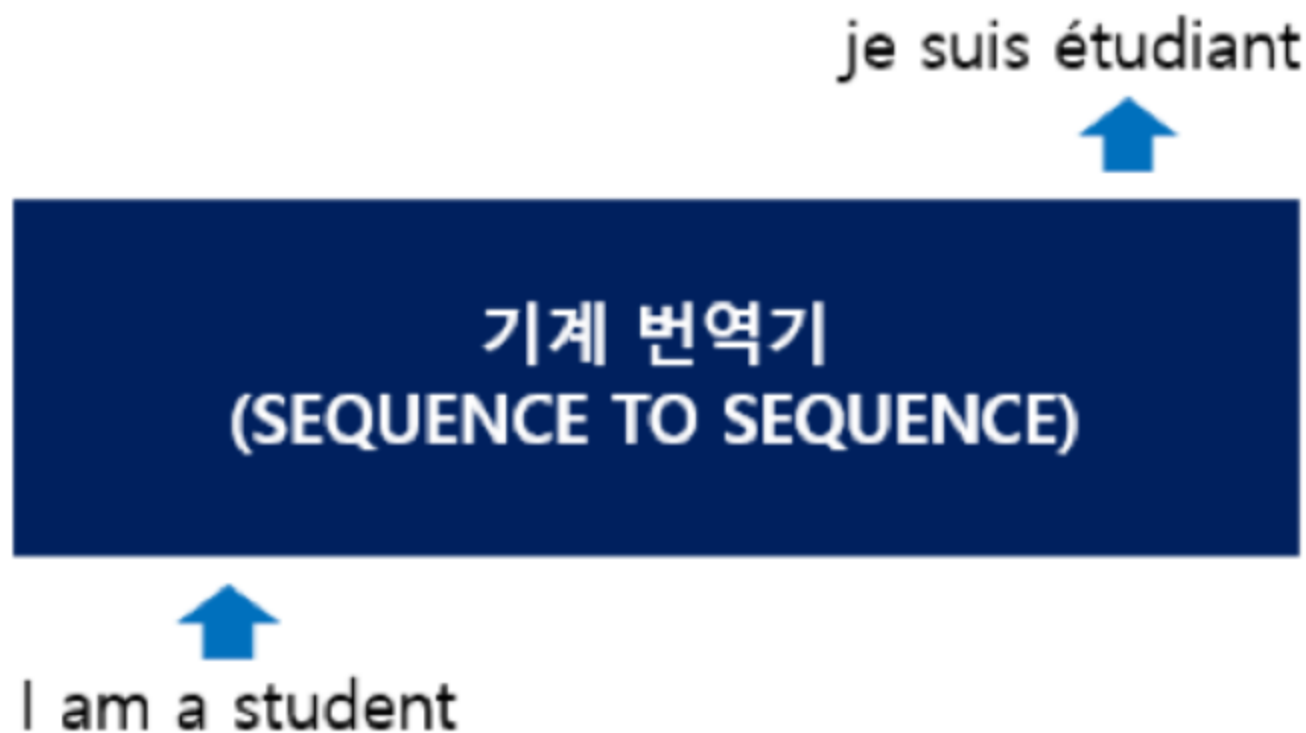
Seq2Seq

04

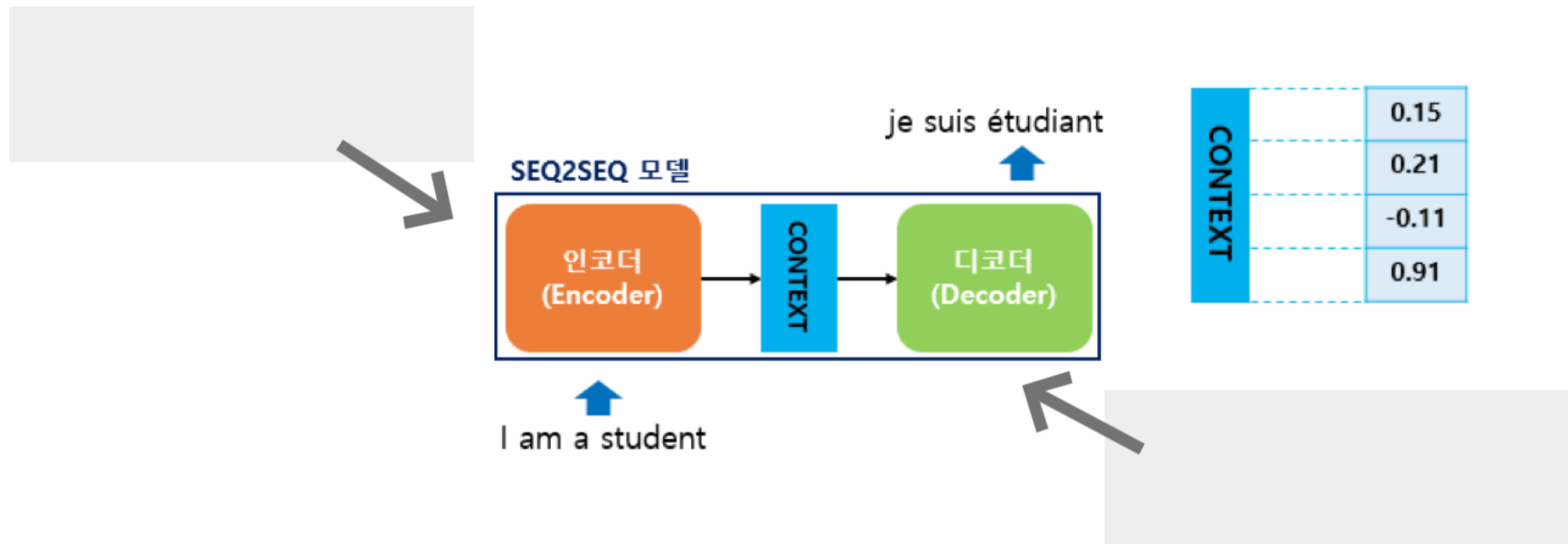


# Attention? 우선 Seq2Seq 부터!

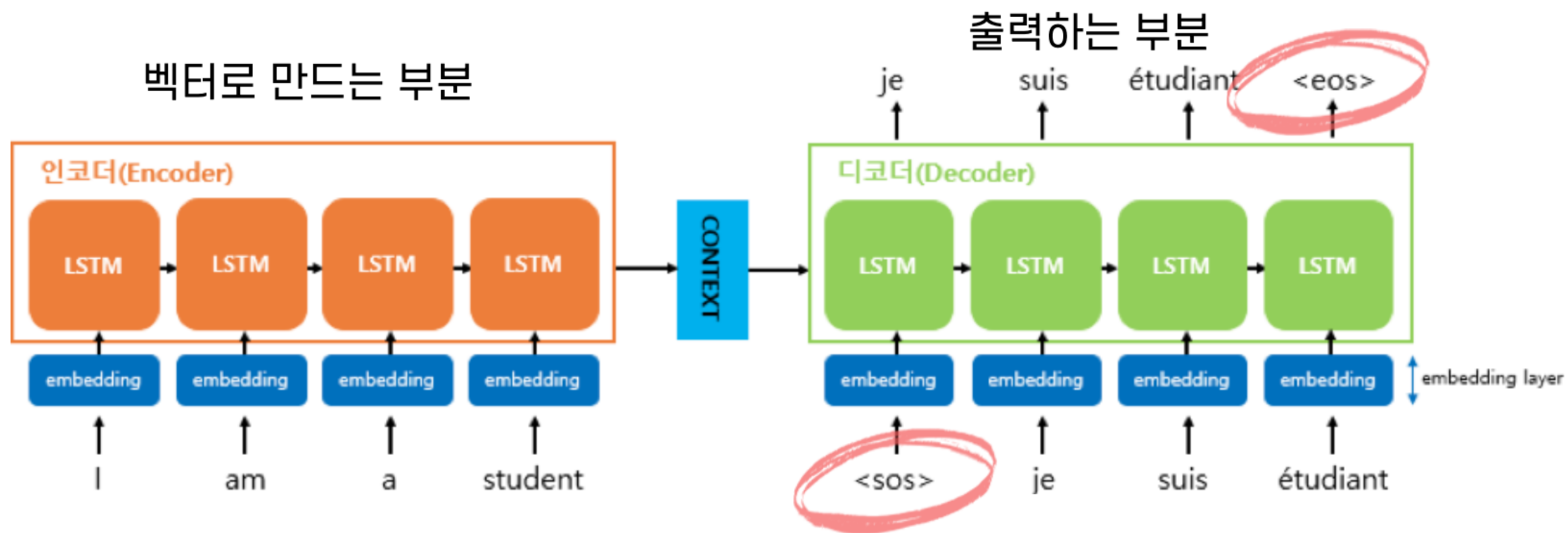
Attention + Bi LSTM 을 사용하는 것이 가장 성능이 좋다고 알려져 있음.

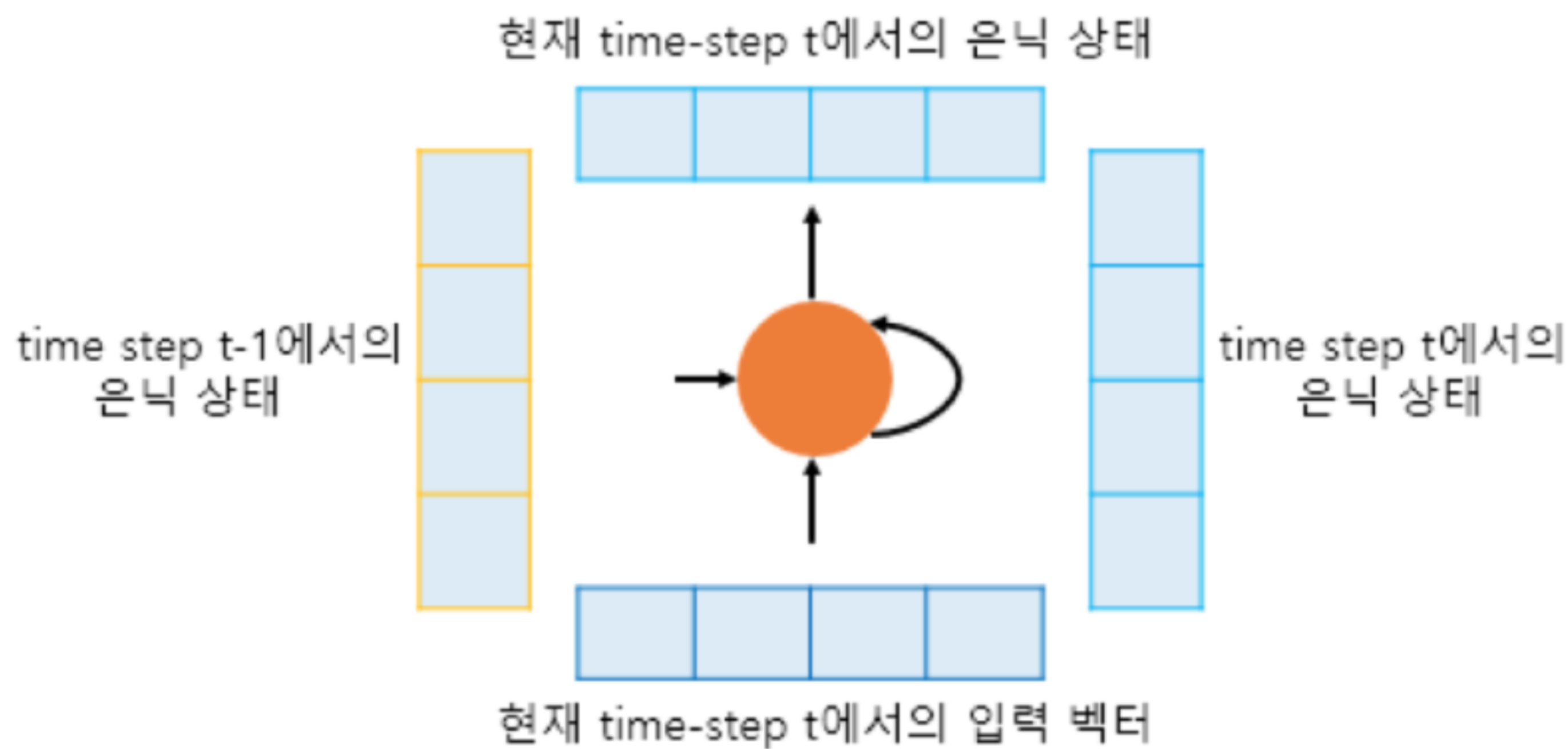


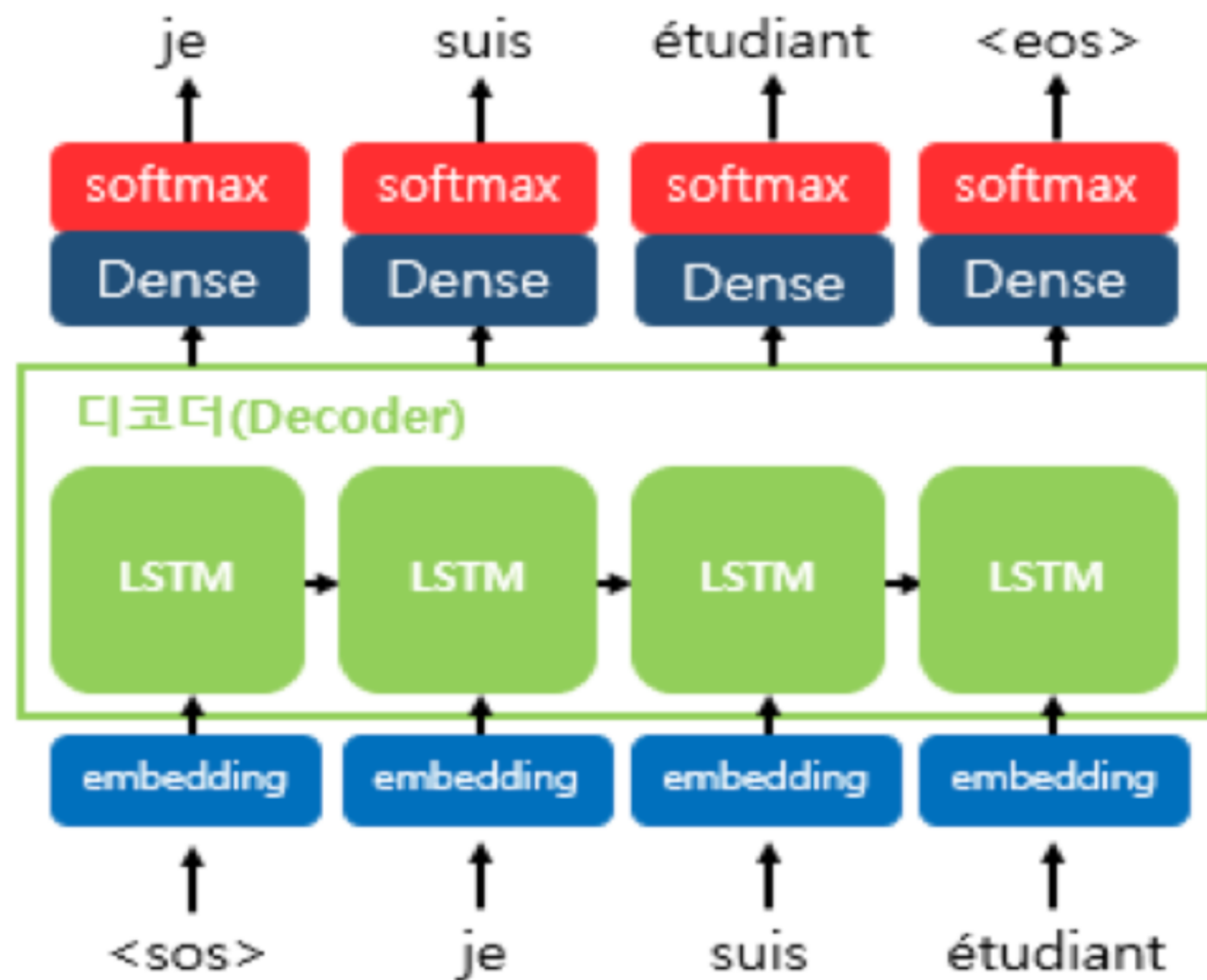
# Encoder & Decoder



# train / test 과정이 다름!









”

**Attention**

05





## seq2seq의 문제점

1. 정해진 크기의 벡터에 정보를 압축하다 보니 인코더에 정보손실 발생.  
이렇게 되면 디코더는 인코더가 압축한 정보를 초반 예측에만 활용하는 경향 존재.  
-> bottle neck 문제.

2. RNN의 고질적인 Vanishing Gradient 문제.

-> Attention의 필요성.



# Attention

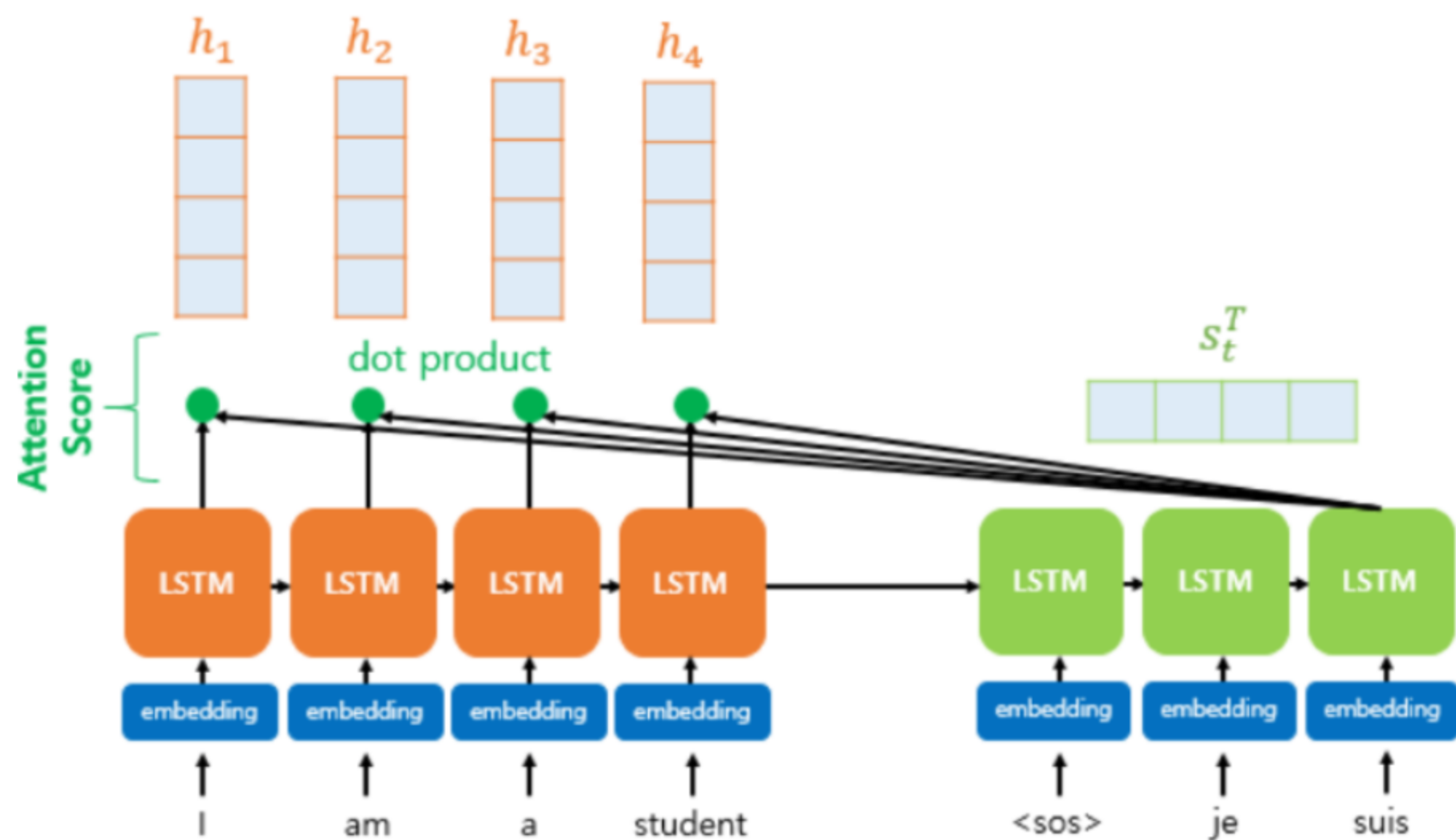
디코더에서 출력 단어를 예측하는 매 시점(time step)마다,  
인코더에서의 전체 입력 문장을 다시 한 번 참고하는 매커니즘.

가정) 인코더가 'bier'를 받아서 벡터로 만든 결과(인코더 출력)는  
디코더가 'beer'를 예측할 때 쓰는 벡터(디코더 입력)와 유사할 것

중요한것만 집중하게 만들자!

# Attention

## 1. Attention Score 구한다.



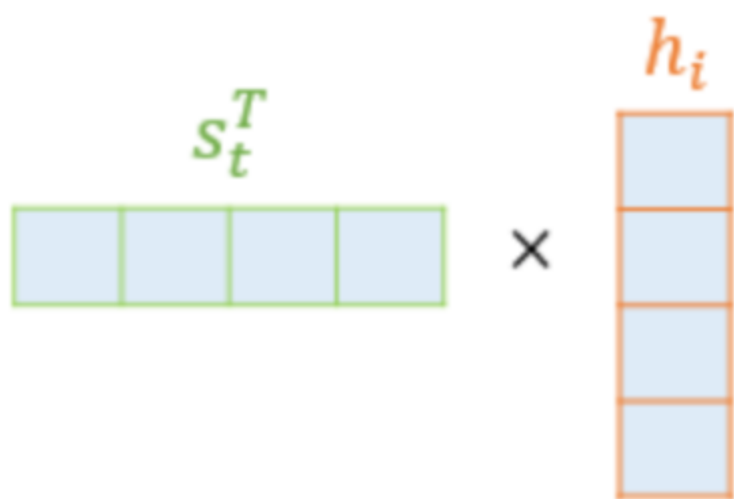
## Attention Score)

현재 디코더의 시점 $t$ 에서  
단어를 예측하기 위해  
인코더의 모든 은닉상태 각각이  
디코더의 은닉상태와 얼마나  
**유사한지** 판단하는 스코어.



# Attention

Attention Score 구하기 위해 내적 수행.

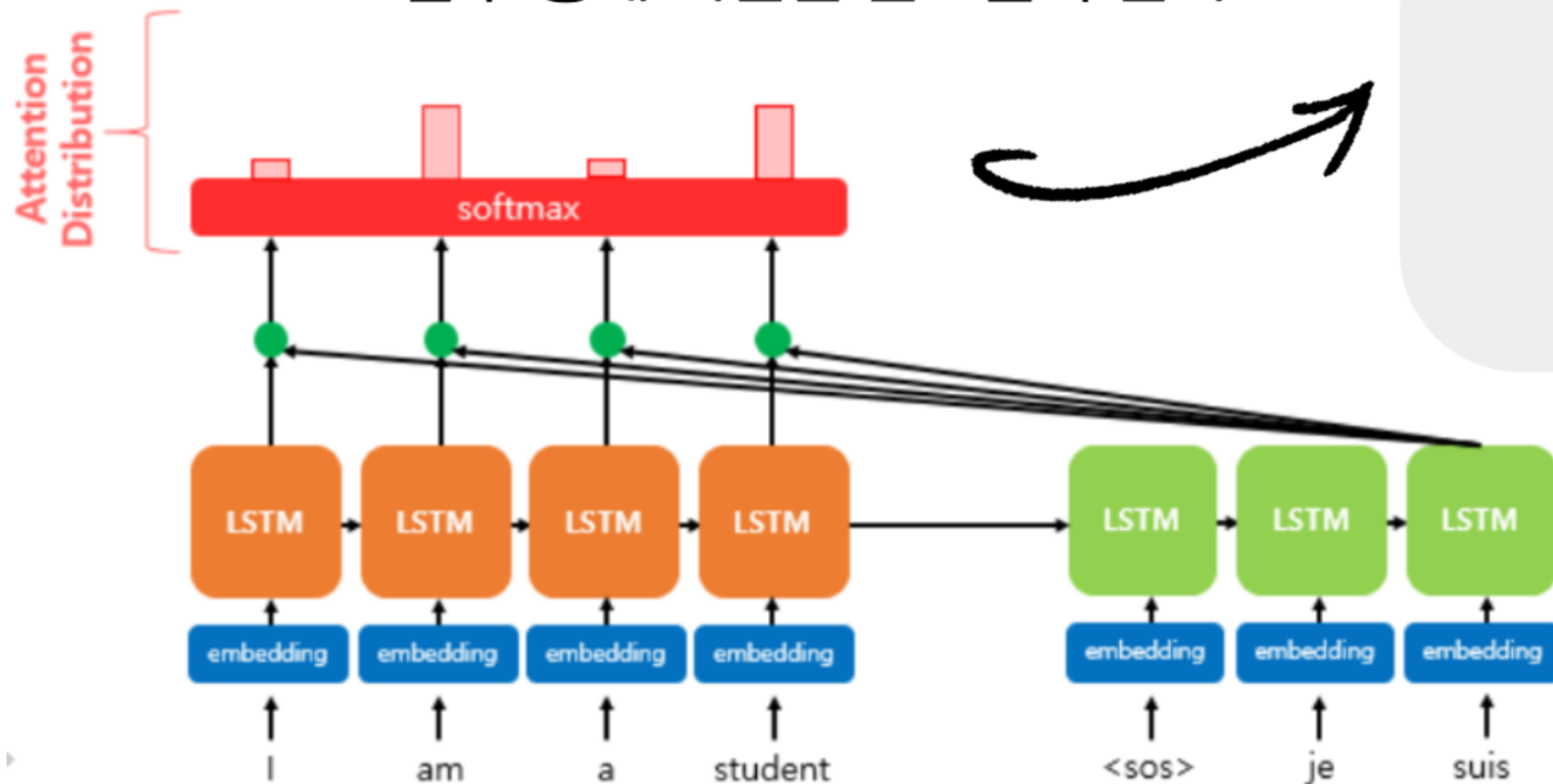


Attention score의 모음값

$$e^t = [s_t^T h_1, \dots, s_t^T h_N]$$

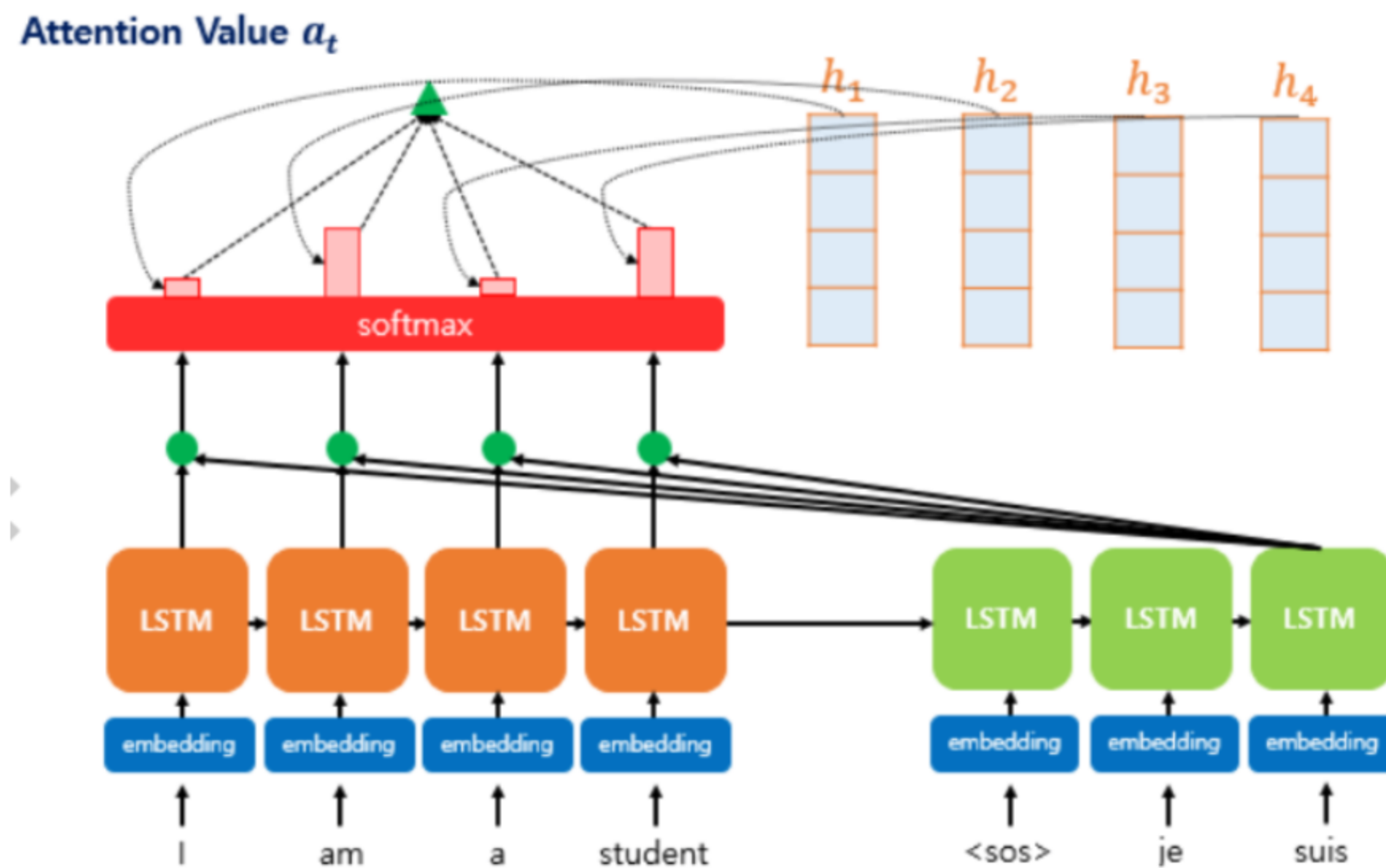
# Attention distribution by Softmax

2. softmax 함수 통해 어텐션 분포를 구한다.



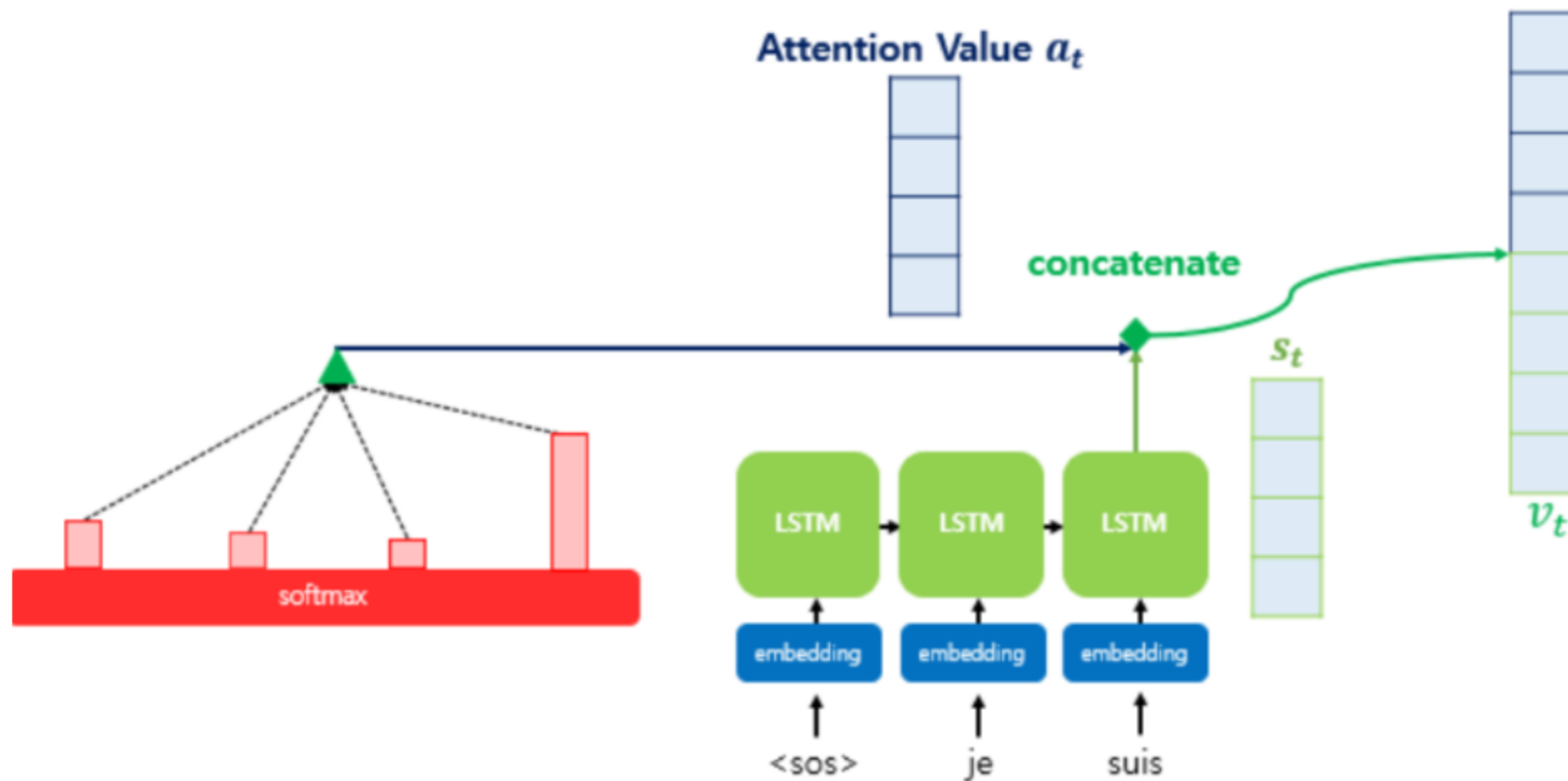
# Attention distribution by Softmax

3. 각 인코더의 어텐션 가중치와 은닉상태를 가중합하여 Attention Value 구한다.



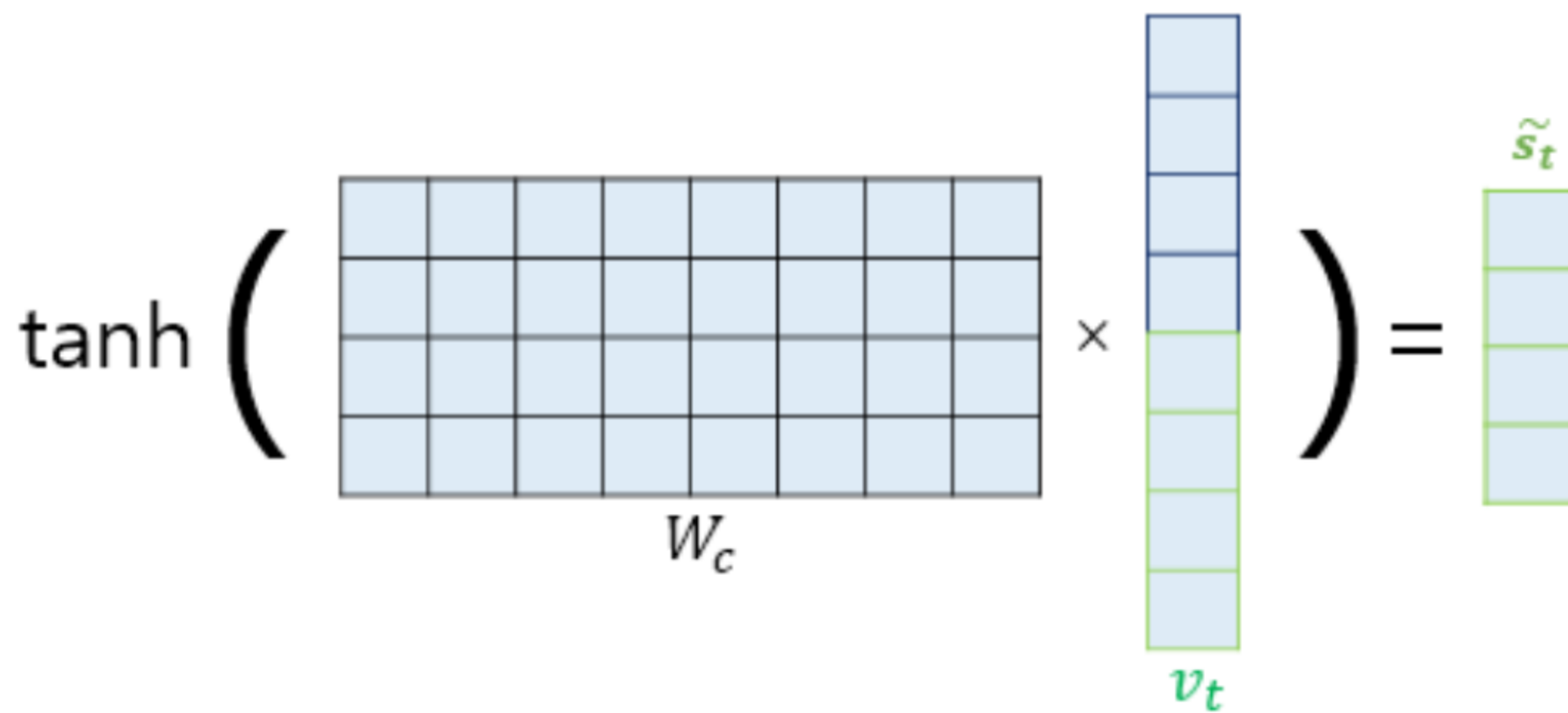
# Attention distribution by Softmax

## 4. 어텐션 값과 디코더의 t 시점의 은닉상태를 연결



# Attention distribution by Softmax

5. 출력층 연산의 입력값이 되는 것을 계산함.

$$\tanh \left( W_c \times v_t \right) = \tilde{s}_t$$


The diagram shows a matrix  $W_c$  (4 rows by 8 columns) and a vector  $v_t$  (10 rows by 1 column). The matrix  $W_c$  is represented by a 4x8 grid of light blue squares. The vector  $v_t$  is represented by a 10x1 column of light blue squares. The result of the multiplication is a 4x1 column vector  $\tilde{s}_t$ , represented by a 4x1 column of light blue squares. The  $\tanh$  function is applied to the product of  $W_c$  and  $v_t$ .





## Attention distribution by Softmax

6. softmax 식위 출력층의 결과, 즉 예측 벡터를 얻습니다.

즉, 전체 과정을 수식으로 나타내 보자면,

