# Attention 모델 설명

KUBIG NLP 분반

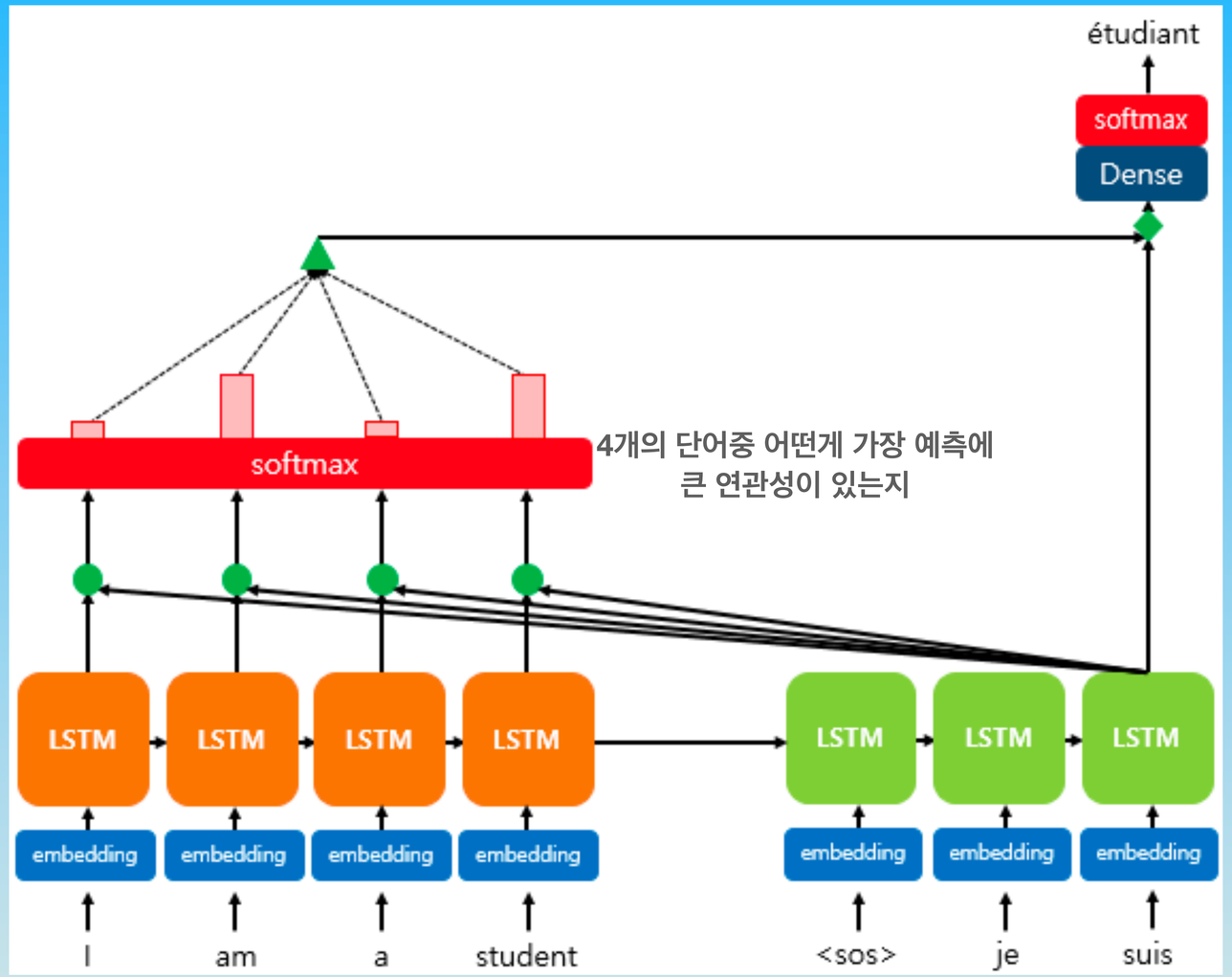
# RNN 기반 seq2seq 모델 단점

1. 하나의 고정된 크기의 벡터에 모든 정보를 압축하기 때문에 정보 손실이 발생
2. RNN의 고질적인 문제인 기울기 소실(vanishing gradient) 문제가 존재

# Attention?

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

모두 동일한 비율로 참고하는 것이 아니라, 해당 시점에서 예측해야 할 단어와 연관성 있는 입력 단어 부분을 좀 더 집중 (attention)



# Attention 함수

Attention(Q, K, V) = Attention Value

Q = Query : t 시점의 디코더 셀에서의 은닉 상태

K = Keys : 모든 시점의 인코더 셀의 은닉 상태들

V = Values : 모든 시점의 인코더 셀의 은닉 상태들

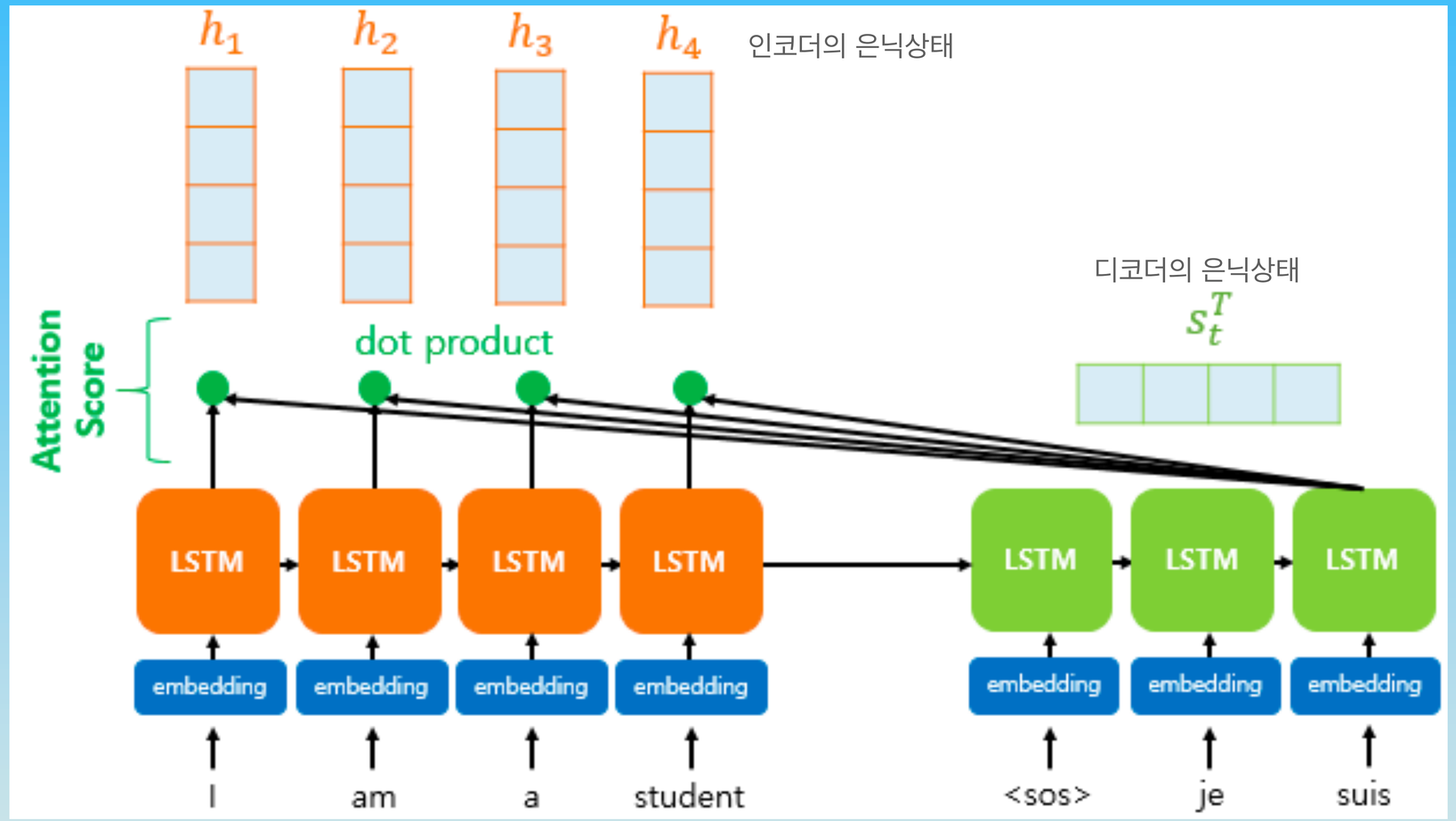
1. 쿼리에 대해 모든 키와의 유사도를 구함

2. 해당 유사도를 키와 매핑되어 있는 각각의 값(value)에 반영

3. 이렇게 유사도가 반영된 값을 모두 더해 리턴(Attention Value)

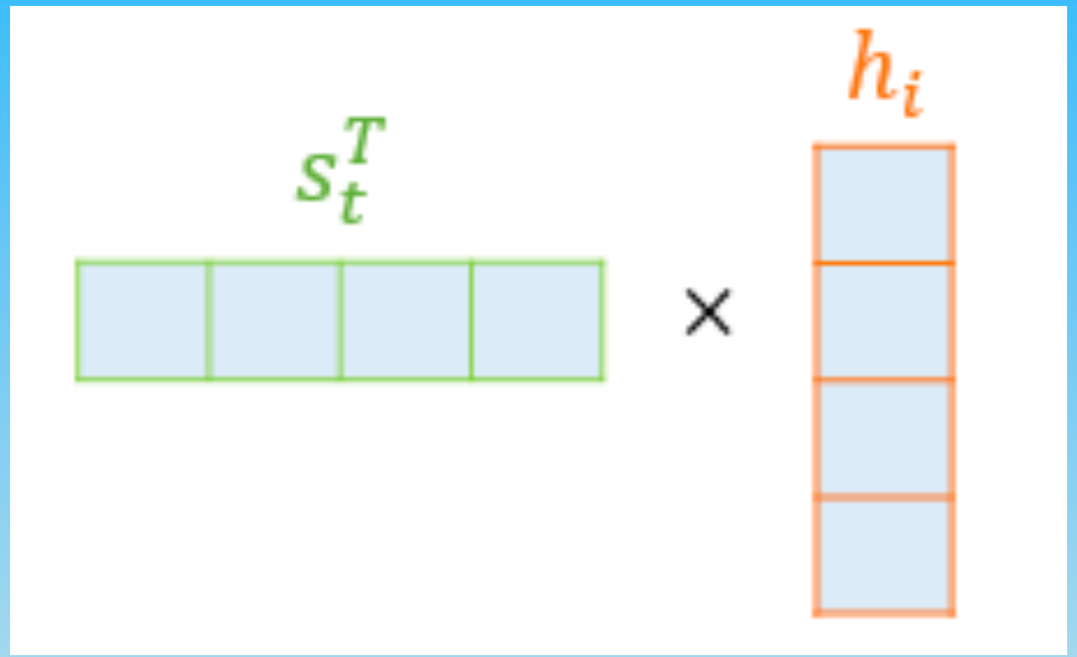
# Dot-Product Attention

1. attention score 구한다



# Dot-Product Attention

dot-product attention



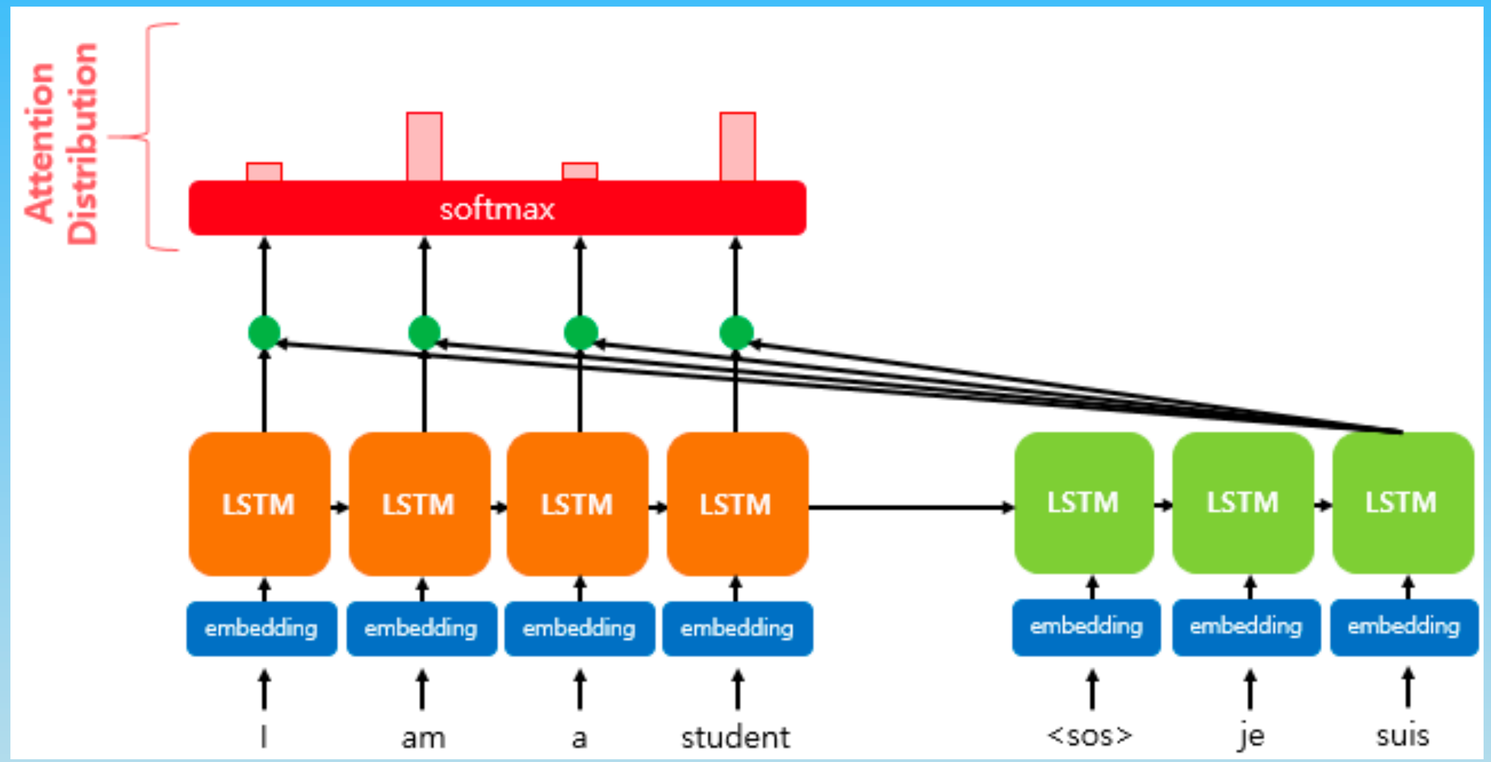

인코더의 모든 은닉 상태 각각이 디코더의 현 시점 은닉상태와
얼마나 유사한지 판단하는 스코어

어텐션 스코어 모음

et = [st*h1,⋯,st*hN]

# Dot-Product Attention

## 2. attention 분포 구하기

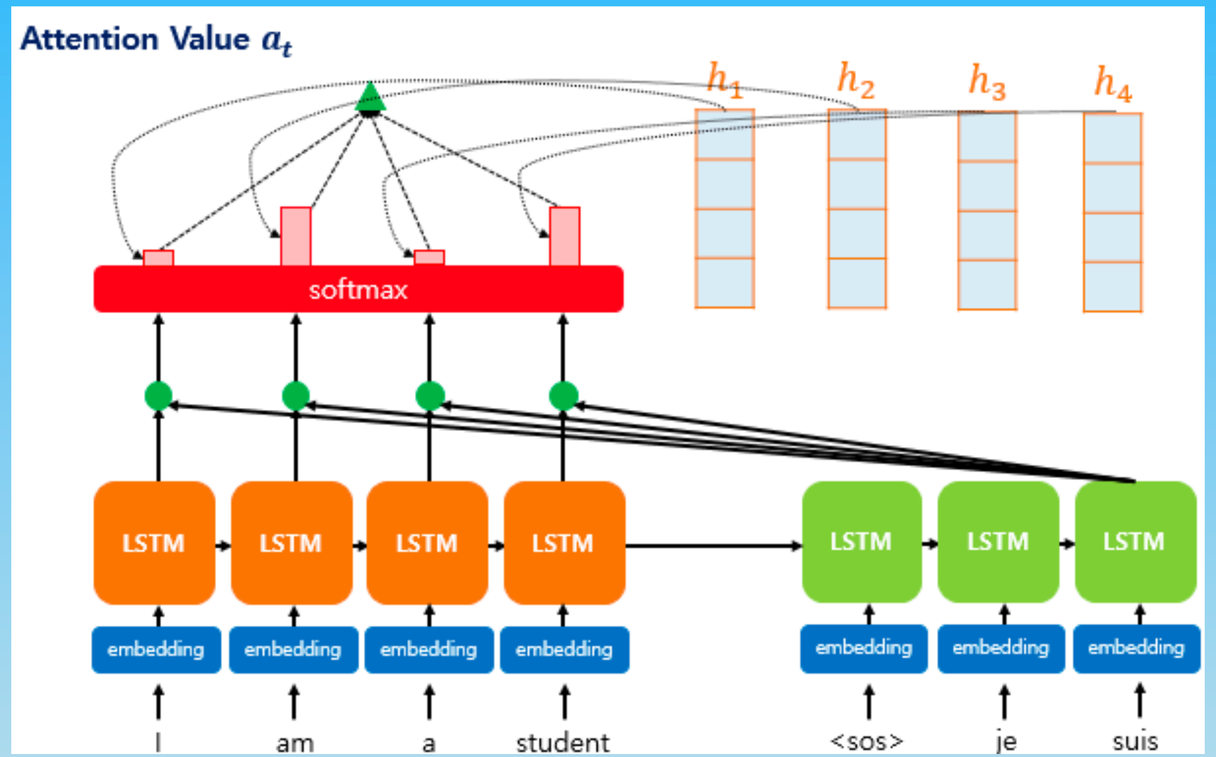



softmax 함수 이용하여 어텐션 분포 구함

$$\alpha t = softmax(et)$$

# Dot-Product Attention

## 3. 가중합 진행

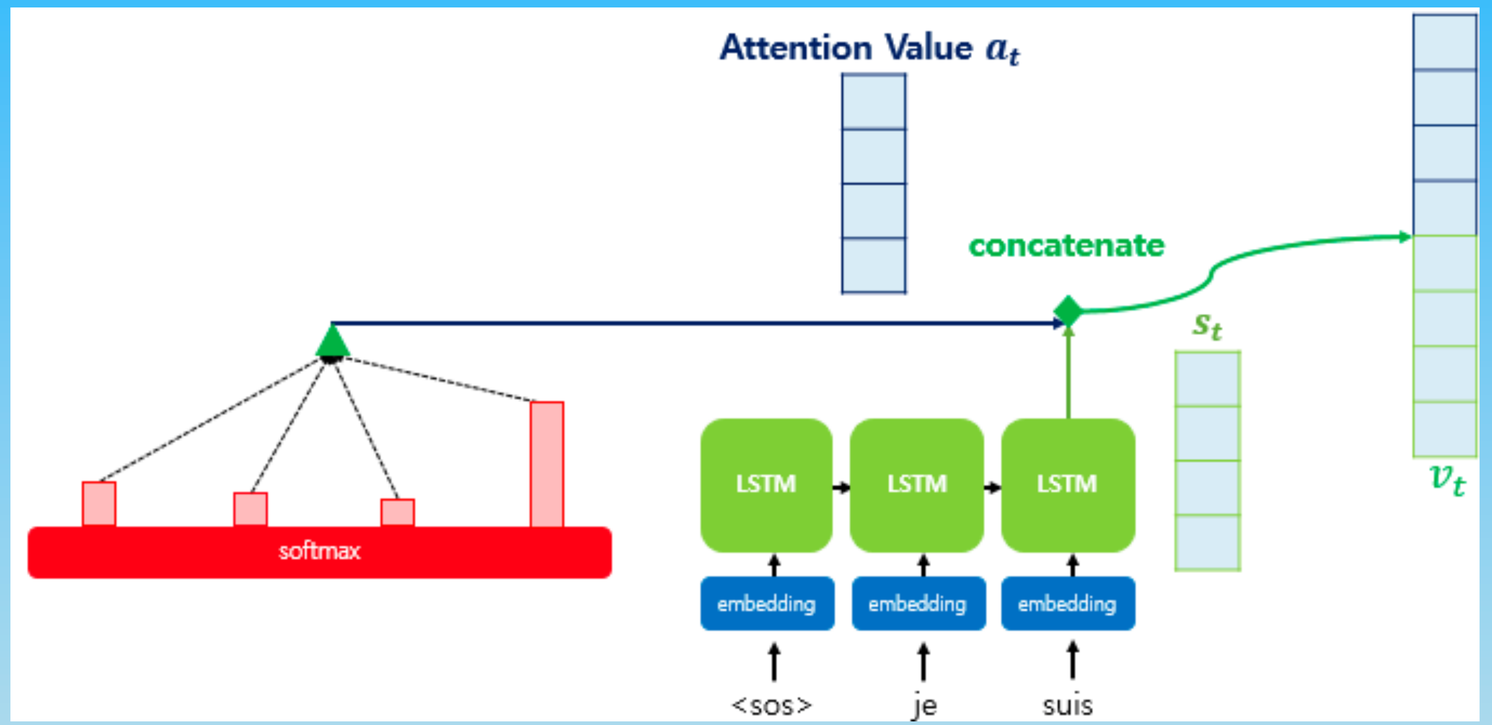



$$a_t = \sum_{i=1}^{N} \alpha_i^t h_i$$

인코더의 문맥을 포함하고 있다해서 컨텍스트 벡터

# Dot-Product Attention

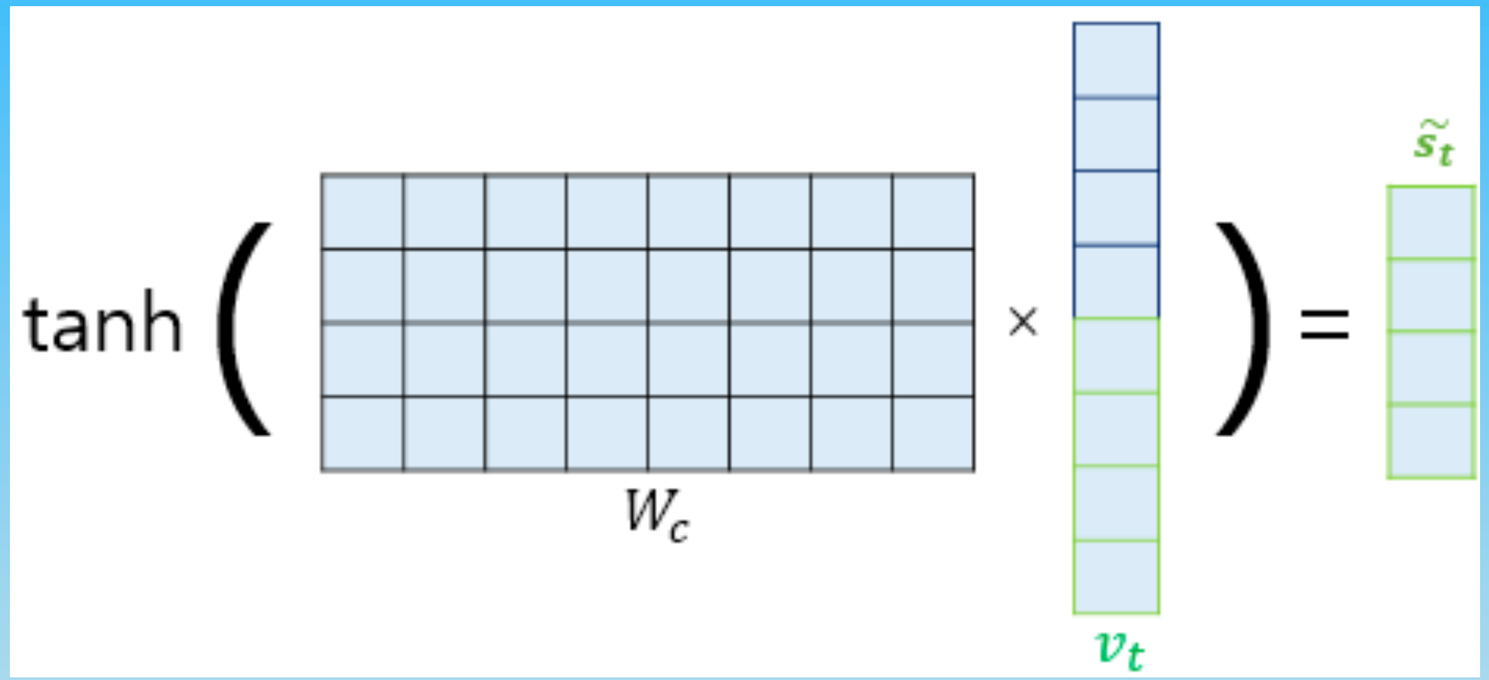
4. 어텐션 값과 디코더의 t시점 은닉 상태를 연결




at와 st 결합하여 vt
vt를 y예측 연산의 입력으로 사용

# Dot-Product Attention

5. 출력층 연산의 입력이 되는 st 계산, 출력층의 입력으로 사용




$$\tilde{s}_t = \tanh\left(\mathbf{W_c}[a_t; s_t] + b_c\right)$$

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