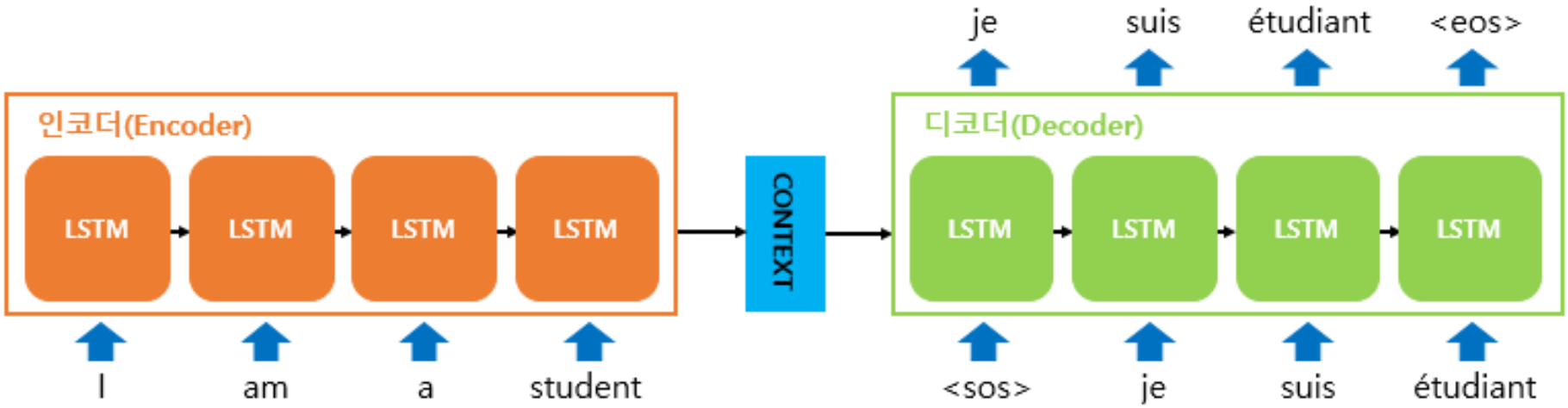
16. Attention Mechanism

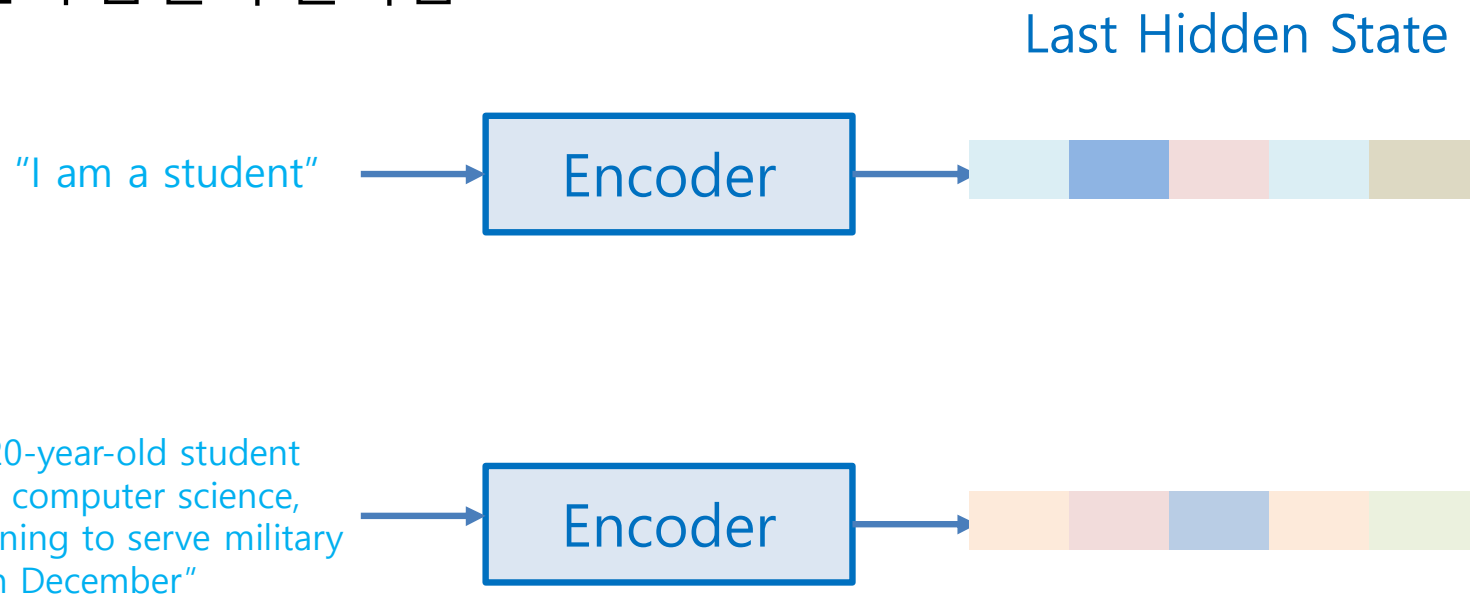
Encoder-decoder architecture

- 인코더와 디코더는 각각 RNN 아키텍처로 구현됨
- 입력 문장은 단어 단위로 쪼개지고 각 단어는 RNN셀의 입력이 됨
- 모든 단어를 입력받은 후 **RNN 셀의 마지막 은닉상태를** **Context vector**라고 함



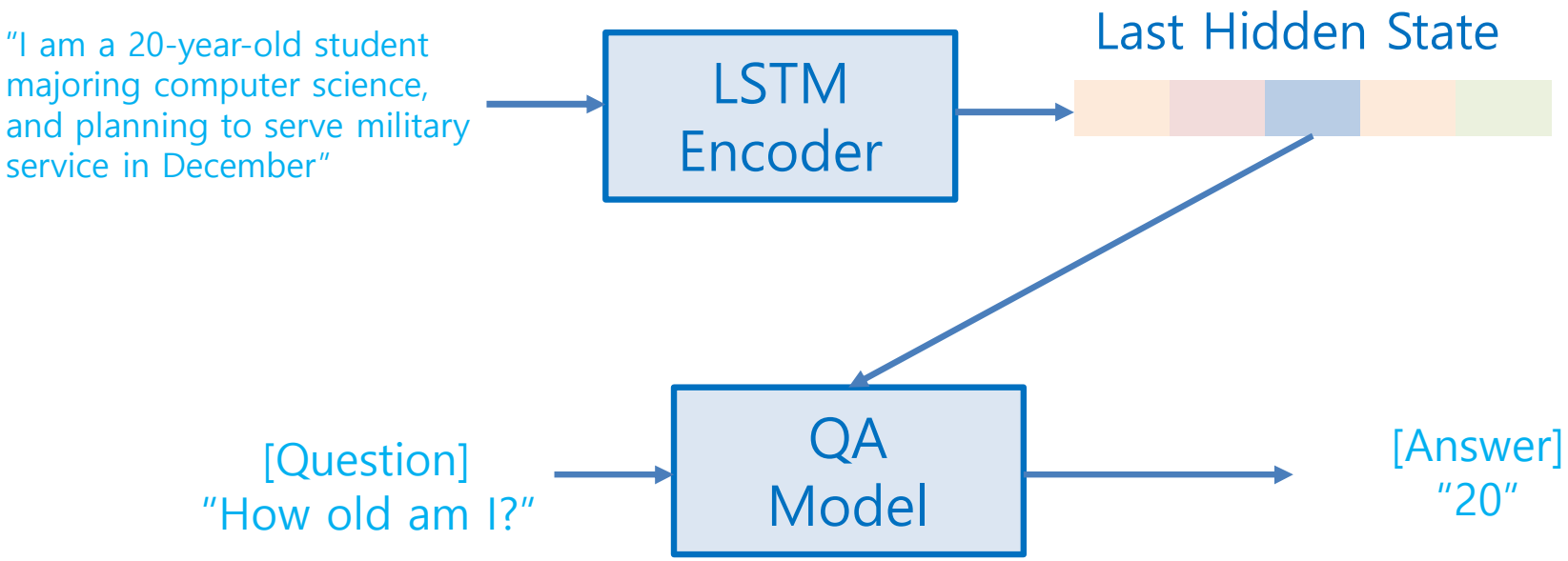
seq2seq 모델의 한계

- 고정된 크기로 문장 정보를 압축하려고 하니 정보 손실이 발생
- RNN의 문제인 기울기 소실(vanishing gradient) 문제가 존재: 입력 문장이 길면 번역 품질이 떨어짐



seq2seq encoder는 모든 문장을 같은 차원의 벡터로 표현 => 길이가 길면 문장 정보를 모두 담지 못할 수 있음

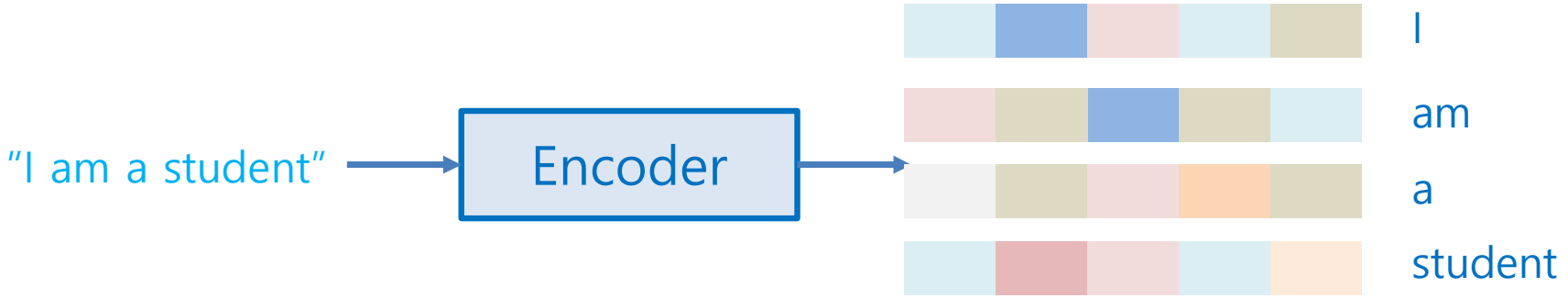
기계 독해 모델에서의 문제점



LSTM의 은닉 상태는 문장의 앞 쪽에 있는
정보를 제대로 가지지 못할 수 있음

어텐션 메커니즘의 개념 1: 입력 정보를 나누어 표현

- 정보 소실 문제는 전체 문장을 하나의 벡터로 인코딩하는 데서 발생 => 입력 데이터의 모든 부분의 정보를 따로 인코딩



D. Bahdanau, K. Cho, and Y. Bengio, Neural machine translation by jointly learning to align and translate, *arXiv preprint*, 2014.

어텐션 메커니즘의 개념 2: 위치별 관심 정도를 계산

- 입력의 각 위치가 **"질문(Query)"**과 어떤 연관성을 가지는지 계산 =>
  각 위치에서 Attention score를 계산

$$\text{attention} = f(Q, K, V)$$

Attention Score 계산

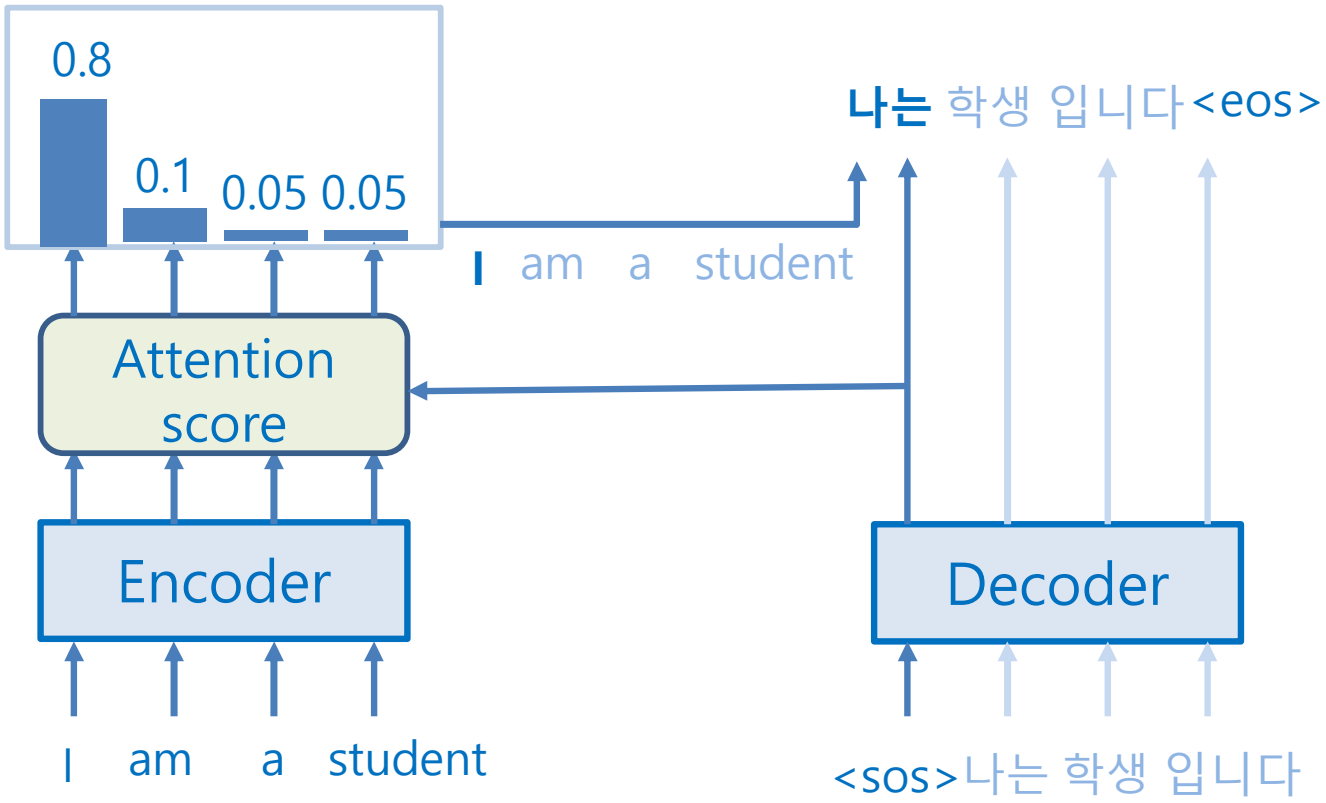
- 디코더에서 " 질문(Query)"의 상태 벡터를 Q로 표현
- 인코더의 각 단어 위치에서의 상태 벡터를 K와 V로 표현
- Q, K, V 등 세 벡터를 이용하여 인코더의 각 위치에서 Attention을 계산

$$\text{attention} = f(Q, K, V)$$

구분	명칭	내용
Q	Query	Decoder에서 각 단어 위치에서의 상태
K	Key	Encoder에서 각 단어 위치에서의 상태
V	Value	Encoder에서 각 단어 위치에서의 상태

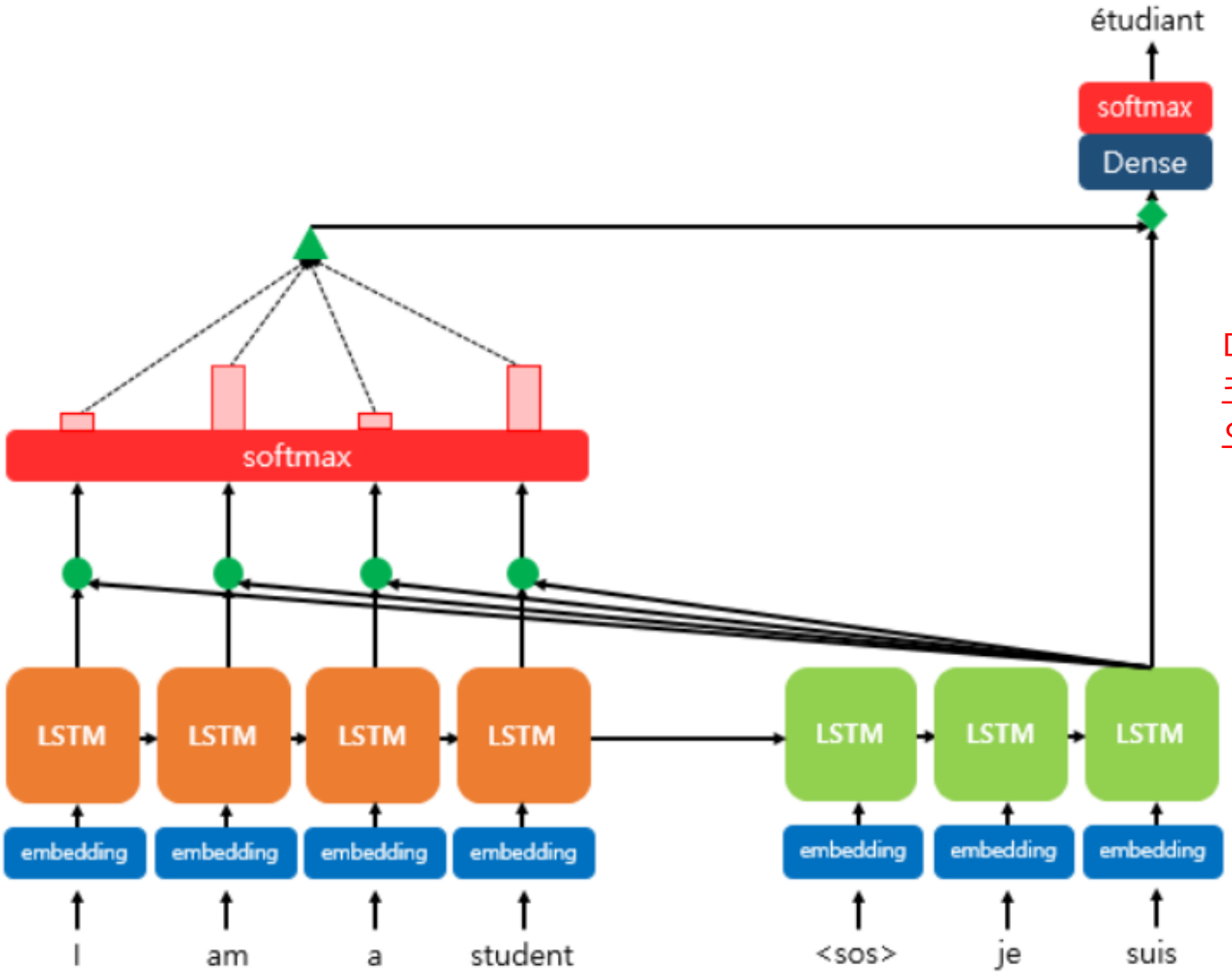
seq2seq 번역기에서의 attention mechanism

- 디코더에서 각 단어가 나올 때의 상태 벡터(Q)를 인코더로 전달
- 인코더에서는 각 단어를 처리할 때의 상태 벡터(K)와 Q를 이용하여 attention score를 계산하고 점수를 번역에 활용 (활용 방법은 추후 설명)



Dot-product attention

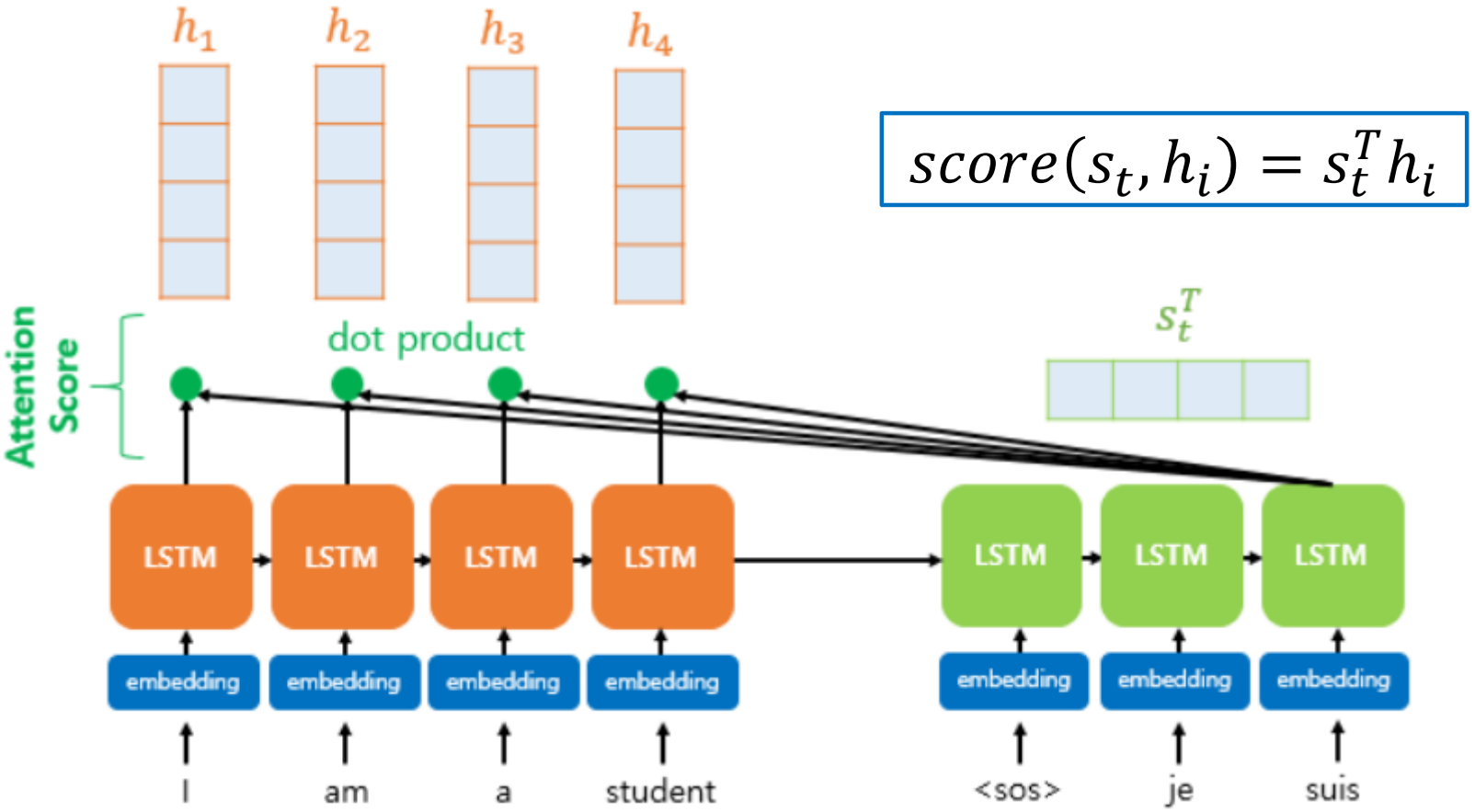
- 상태 벡터간의 내적(dot product)으로 attention을 계산



디코더의 현재 상태와 인코더의 각 상태와의 내적으로 attention을 계산

Attention Score (Dot product)

- $h_1 \sim h_4$: 각 단어에서의 encoder state, s_t: t 시점의 decoder state
- 디코더의 각 시점에서 attention을 새로 계산함: 디코더의 각 단어가 인코더의 어느 부분과 관계되는지 알 수 있음



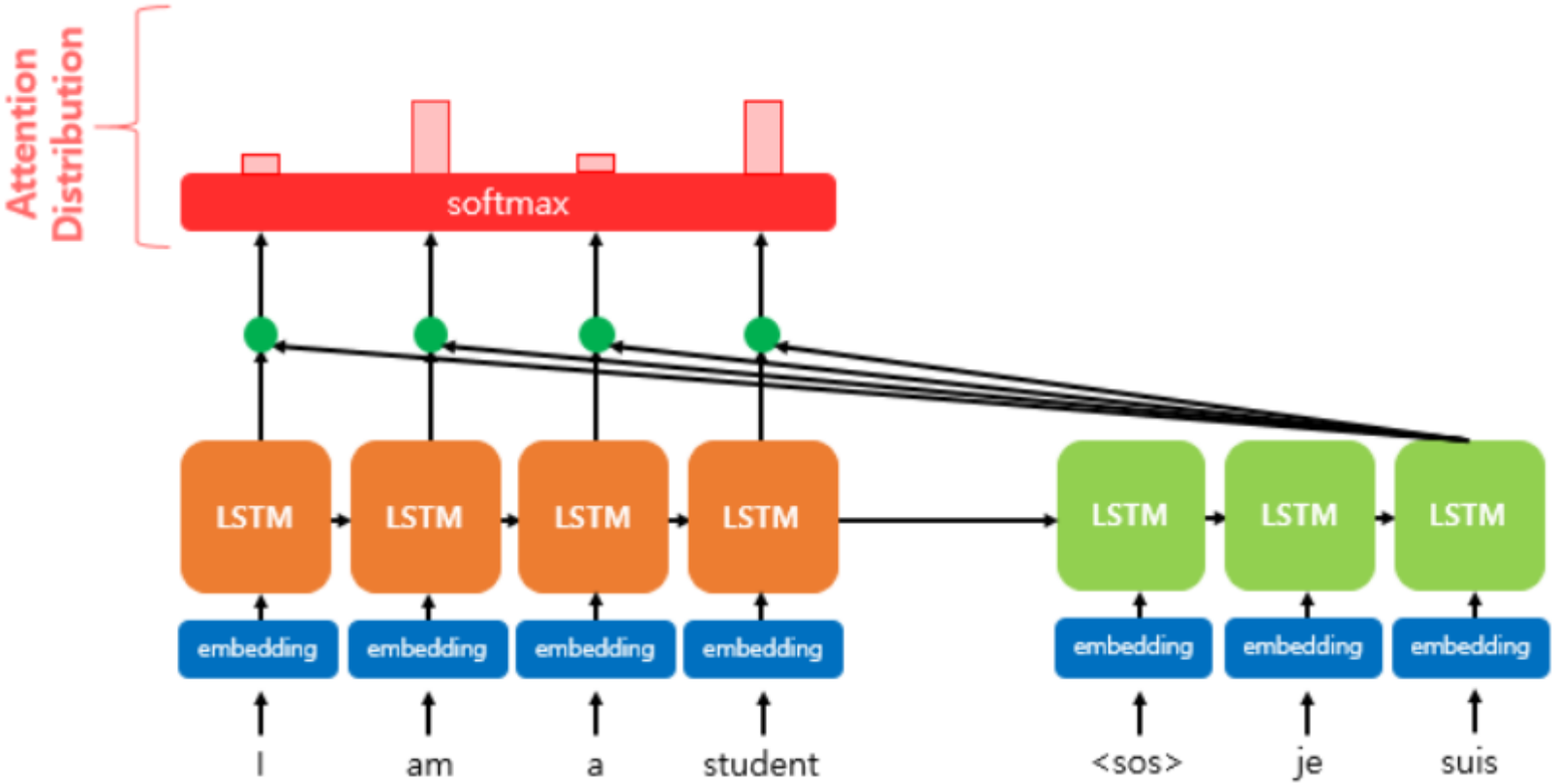
Attention Distribution

- s_t와 인코더의 모든 은닉 상태에서의 어텐션 모음값을 e^t로 정의

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

- Softmax 함수를 통해 어텐션 분포(attention distribution)를 구함

$$\alpha^t = softmax(e^t)$$



다양한 종류의 어텐션

- 어텐션 점수를 구하는 다양한 방식이 제시되어 있음

이름	스코어 함수
dot	$score(s_t, h_i) = s_t^T h_i$
scaled dot	$score(s_t, h_i) = \frac{s_t^T h_i}{\sqrt{n}}$
general	$score(s_t, h_i) = s_t^T W_a h_i$
concat	$score(s_t, h_i) = W_a^T \tanh(W_b s_t + W_c h_i)$

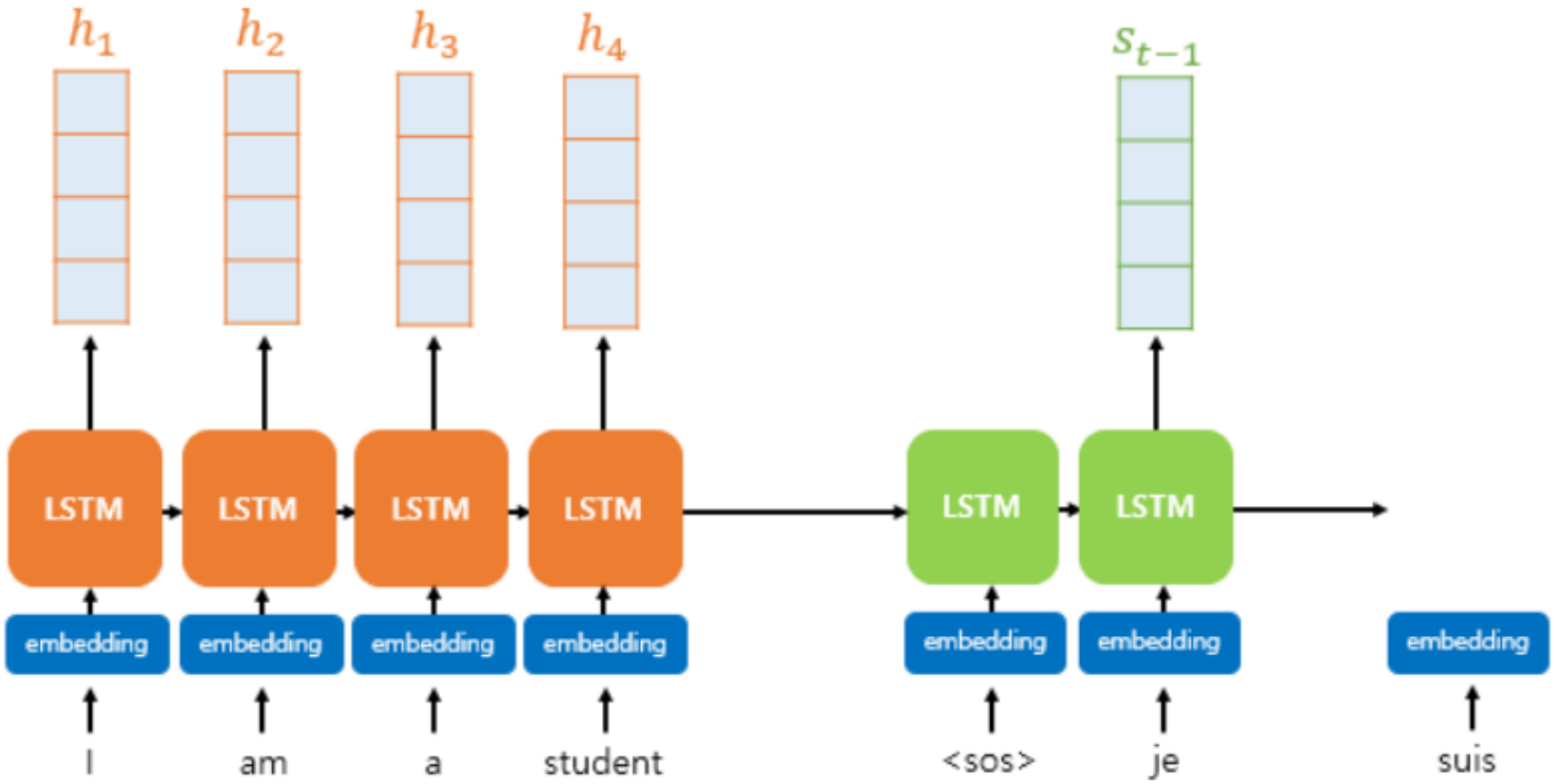
Bahdanau 어텐션이라고 함

Bahdanau 어텐션

- $h_1 \sim h_4$: 각 단어에서의 encoder state, s_t: t 시점의 decoder state

$$score(s_{t-1}, h_i) = W_a^T \tanh(W_b s_{t-1} + W_c h_i)$$

W_a, W_b, W_c는 학습가능한 가중치 행렬



Bahdanau 어텐션 계산

1. $h_1 \sim h_4$를 H로 두면 다음과 같이 attention score vector를 구함

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

2. 여기에 softmax 함수를 적용하여 attention distribution을 구함

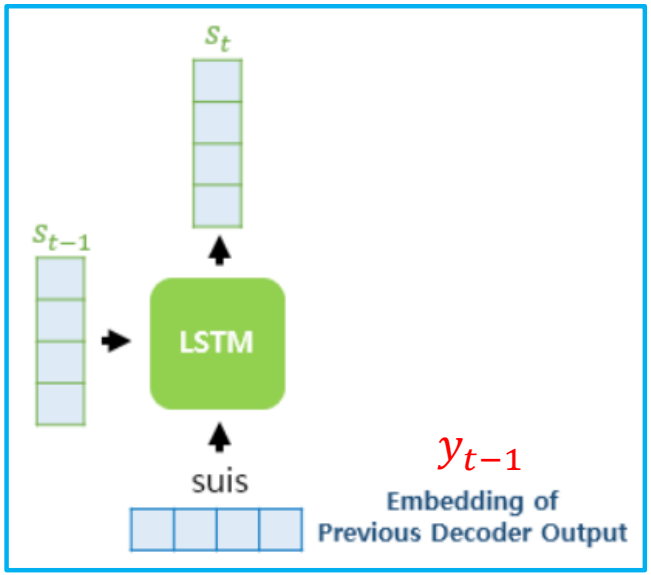
$$\alpha^t = softmax(e^t)$$

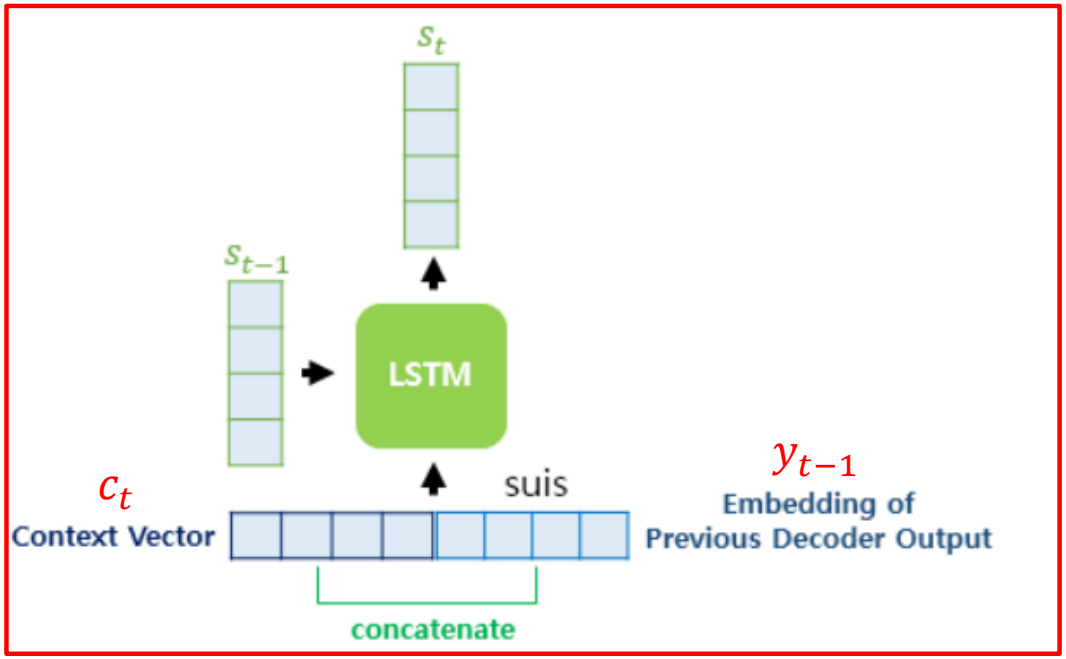
3. Context vector c^t를 구함

$$c^t = H\alpha^t$$

Decoder에서 context vector c_t를 활용

- 현재 시점에서 구한 c_t를 다음 단어 예측에 활용





Attention을 쓰지 않는 기존 구조

$$s_t = f(s_{t-1}, y_{t-1}, c)$$

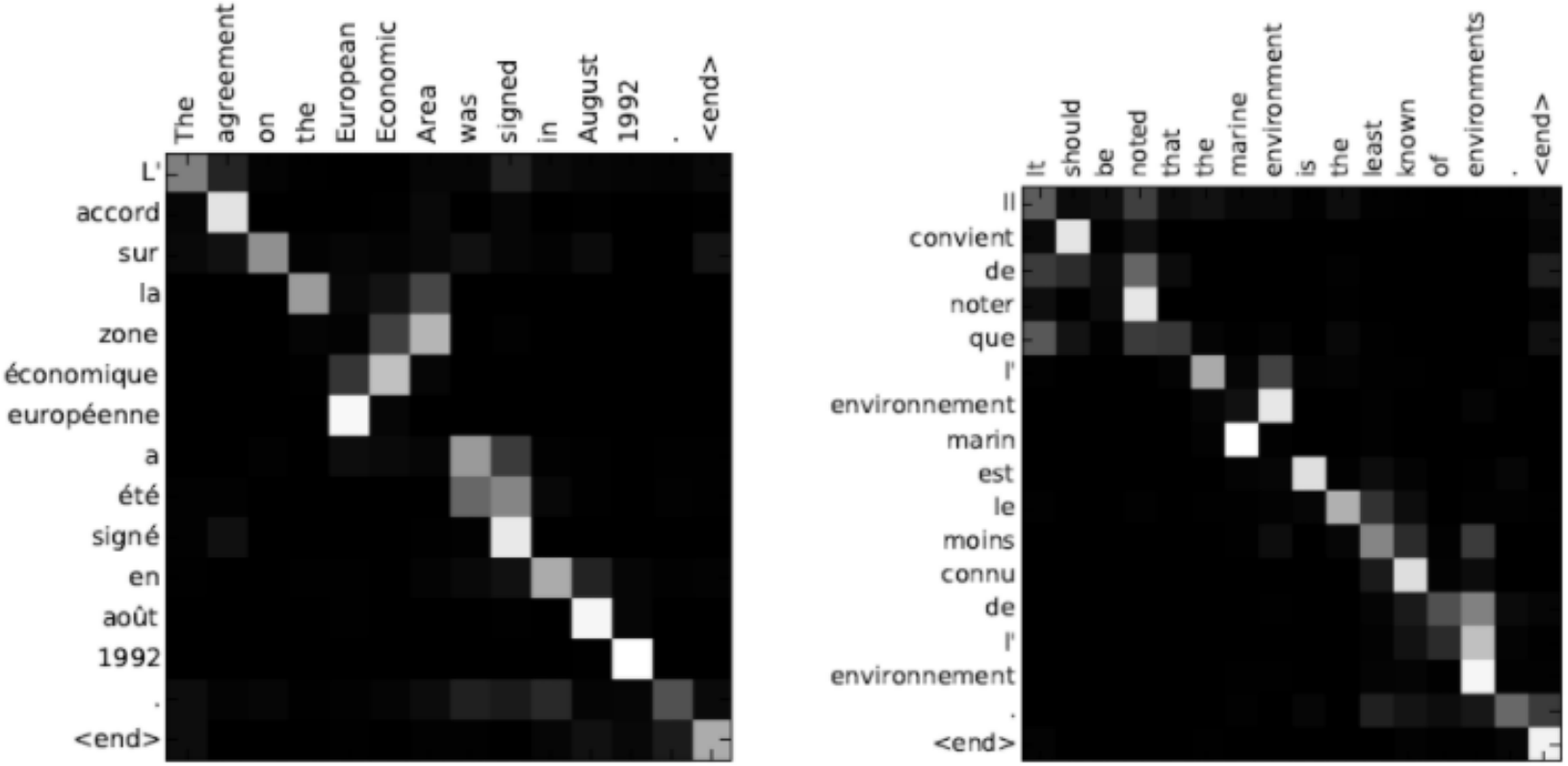
context vector가 고정되어 있음

Attention을 쓰는 새로운 구조:
단어를 생성할 때마다 새로운 컨텍스트 벡터를 생성

$$s_t = f(s_{t-1}, y_{t-1}, c_t)$$

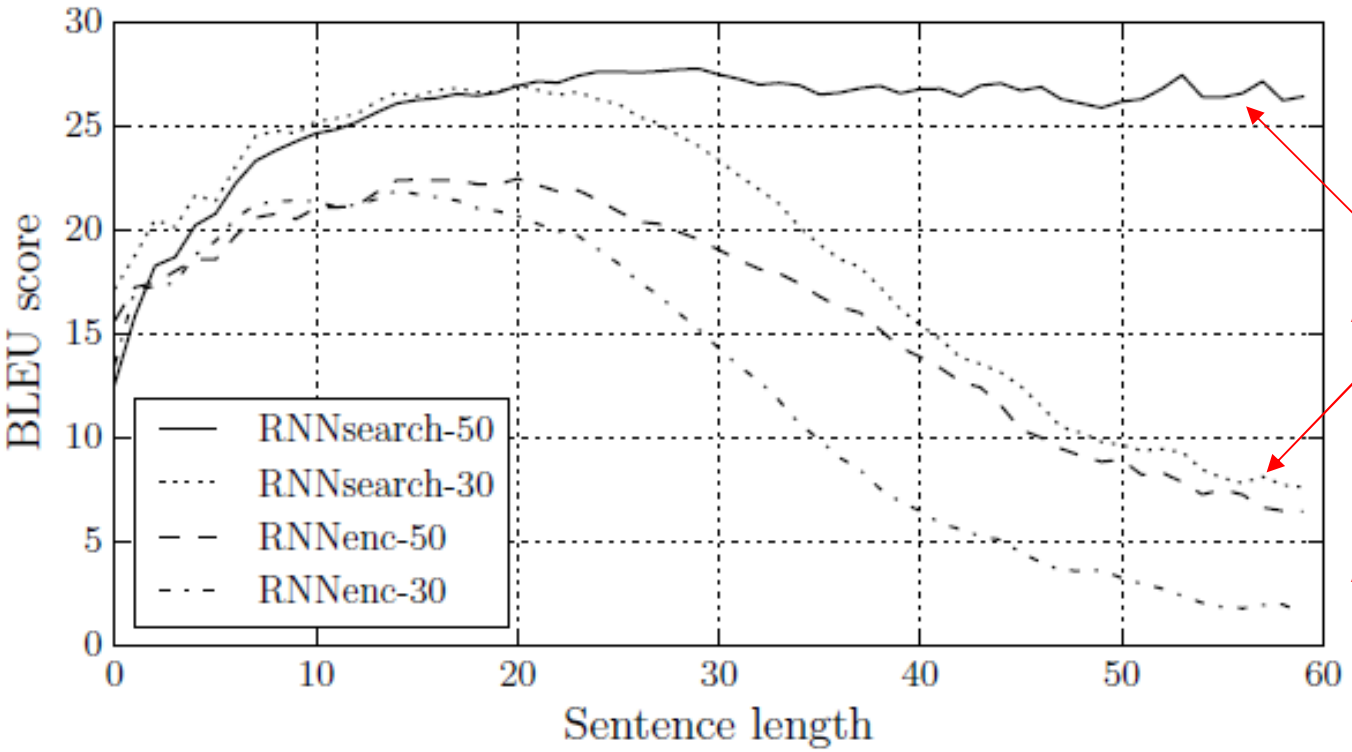
context vector를 계속 새로 계산

번역기에서 단어간의 attention 정합



영어와 프랑스 문장에서의 단어별 attention 점수

번역기에서의 attention의 기능



Attention 활용

Attention 비활용

Attention을 쓰지 않으면 문장 길이가 길어졌을 때 번역 성능이 떨어짐