Attention Mechanism

(Seq2seq + Attention)

Winter Vacation Capstone Study

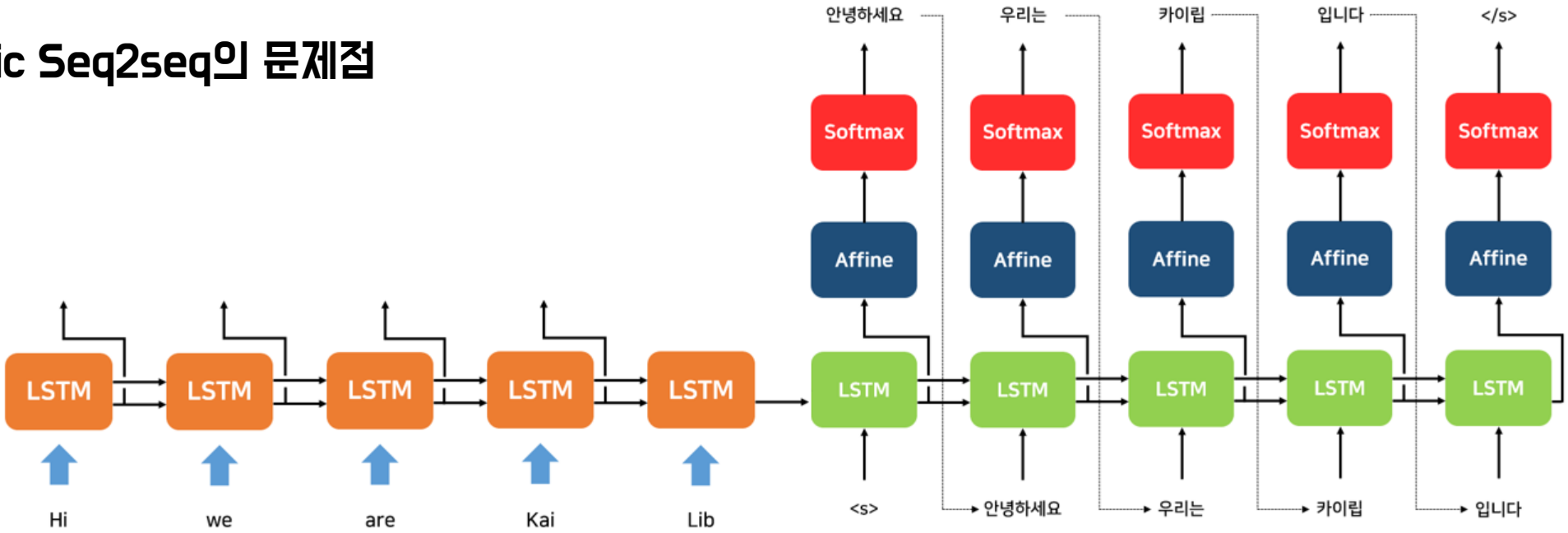
TEAM Kai.Lib

발표자 : 김수환

2020.01.20 (MON)

Seq2seq Review

- **Basic Seq2seq의 문제점**



앞서 배운 seq2seq 모델은 **인코더**에서 입력 시퀀스를 고정된 크기의 벡터로 압축하고, 디코더는 이 벡터를 통해서 출력 시퀀스를 만들어냈다. 하지만 이러한 RNN에 기반한 seq2seq 모델에는 크기 2가지 문제점이 있다.

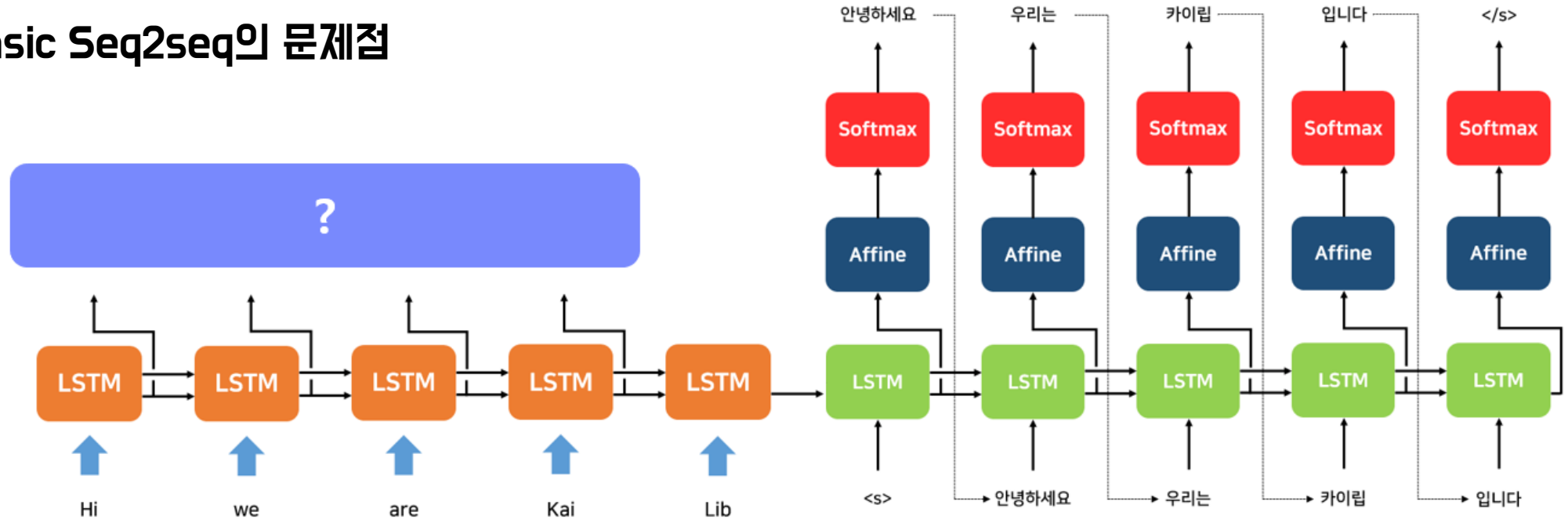
첫째, 하나의 고정된 크기의 벡터에 모든 정보를 압축하려고 하니까 정보 손실이 발생한다.

둘째, RNN의 고질적인 문제인 기울기 소실(Vanishing Gradient) 문제가 존재한다.

이러한 문제점들은 입력 시퀀스가 길어지면 성능이 저하되는 현상으로 이어졌다. 이를 위한 대안으로 입력 시퀀스가 길더라도, 정확도가 떨어지는 것을 보정해주기 위해 등장한 것이 어텐션(Attention) 기법이다.

Attention의 아이디어

- **Basic Seq2seq의 문제점**



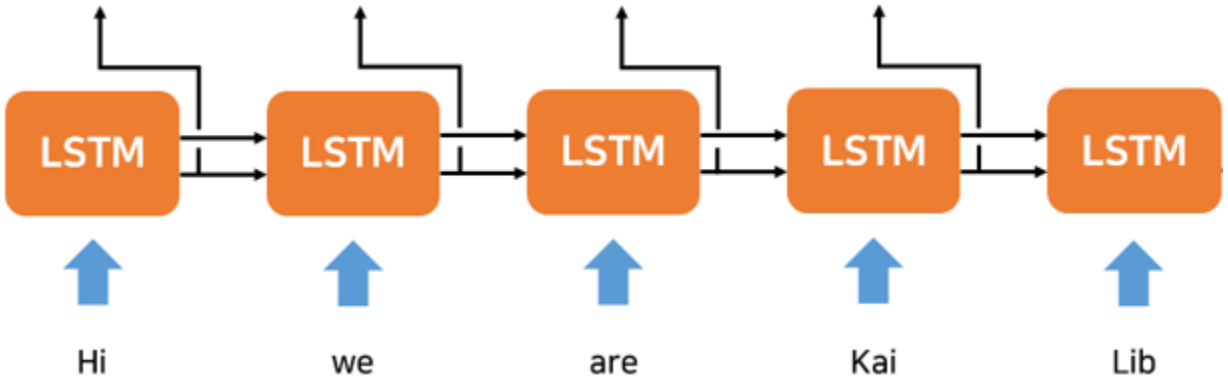
기본적인 seq2seq의 구조를 다시 살펴보자. 위의 구조에서는 마지막 RNN의 Hidden State만을 기반으로 **디코더**에서 출력 시퀀스를 만들어낸다. 그렇다면 나머지 인코더 RNN들의 Hidden State들은 어디에 쓰일까??

기본적인 seq2seq 구조에서는 전혀 사용되지 않는다. 어텐션은 바로 이 **사용되지 않은 Hidden State을 이용한 아이디어**이다.

어텐션의 기본 아이디어는 디코더에서 출력 단어를 예측하는 매 시점(time step)마다, 인코더에서의 **전체 입력 문장을 다시 한 번 참고한다는 아이디어**이다. 단, 전체 입력 문장을 전부 동일한 비율로 참고하는 것이 아니라, **해당 시점에서 예측해야할 단어와 연관이 있는 부분을 좀 더 집중(Attentnion)**해서 보는 방법이다.

Attention의 아이디어

- **Encoder in Seq2seq**

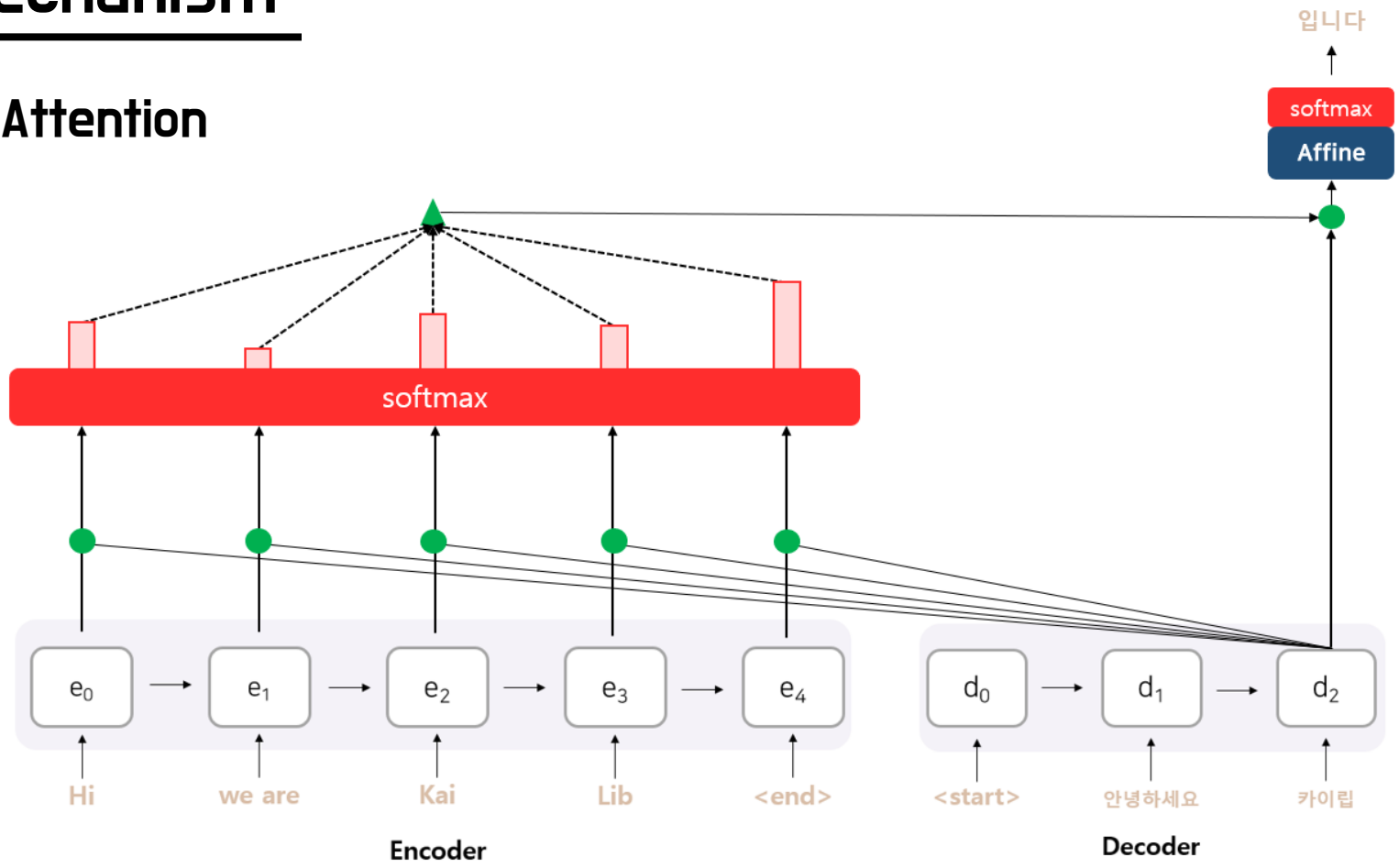


다시 Seq2seq의 인코더를 살펴보자. 주목할 것은 LSTM 계층의 Hidden State의 '내용'이다.
시각별 LSTM 계층의 은닉 상태에는 당연히 직전에 입력된 단어데 대한 정보가 많이 포함되어 있을 것이다.
예를 들면 'we' 라는 단어를 입력했을 때의 LSTM 계층의 Hidden State는 'we'의 성분이 많이 들어간 벡터라고 생각할 수 있다.

이렇게 생각하면 각 인코더의 Hidden State들은 입력된 각 단어에 대한 벡터라고도 생각할 수 있다.
어텐션의 기본적인 아이디어는 바로 여기서 시작된다.

Attention Mechanism

- **Dot-Product Attention**



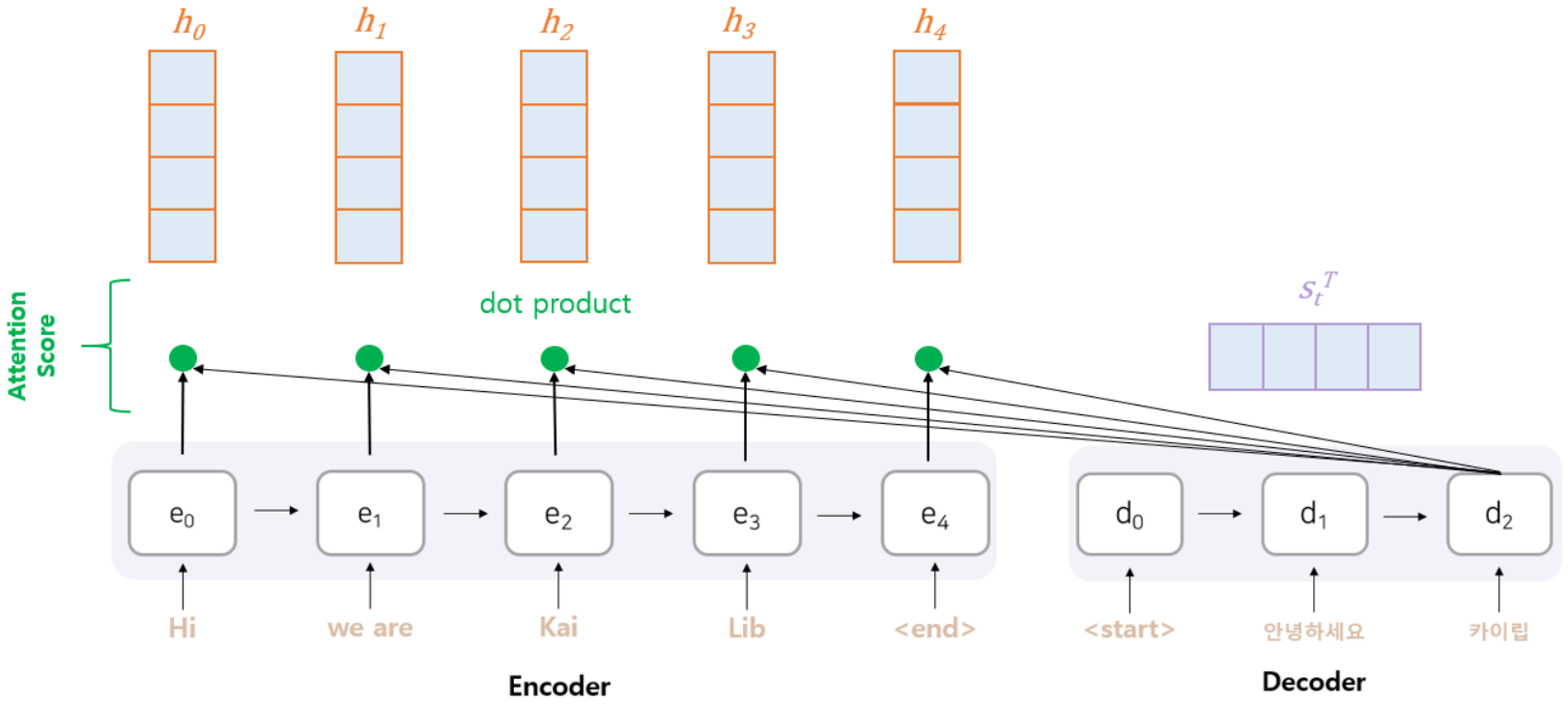
어텐션은 다양한 종류가 있는데 그 중에서도 가장 이해하기 쉬운 **Dot-Product Attention**을 통해 어텐션을 이해해보자.

위 그림은 디코더의 세번째 LSTM 셀에서 출력 단어를 예측할 때, 어텐션 매커니즘을 사용하는 모습이다.

이제부터 하나씩 어떤 식으로 어텐션이 적용되는지를 살펴보자.

Attention Mechanism

- **Attention Score (Cont.)**

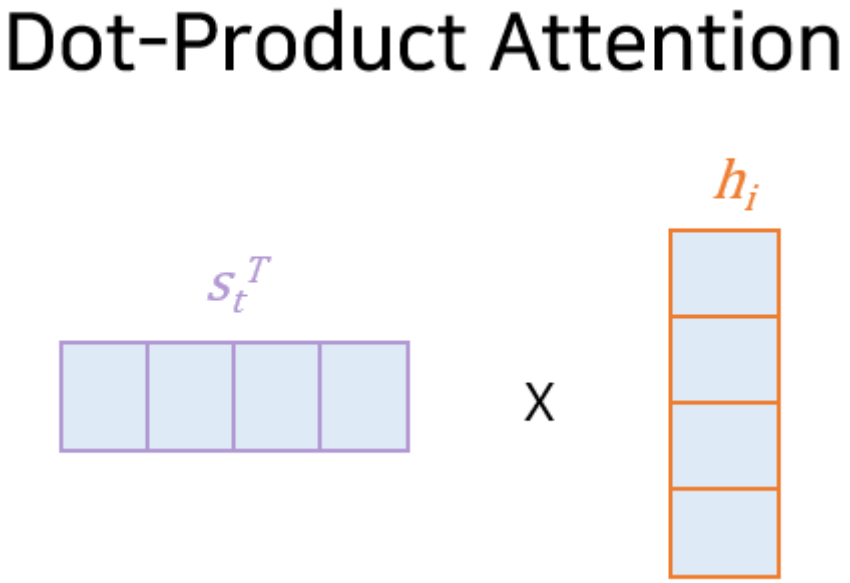


가장 먼저 어텐션 스코어(Attention Score)를 구해야한다. 어텐션 스코어란 현재 디코더의 시점 t에서 단어를 예측하기 위해, 인코더의 모든 Hidden State들 각각이 디코더의 현 시점의 Hidden State와 얼마나 유사한지를 점수로 매긴 값이다.

Dot-Product Attention에서는 이 어텐션 스코어를 매길 때 내적(dot)을 이용한다.

Attention Mechanism

- **Attention Score**

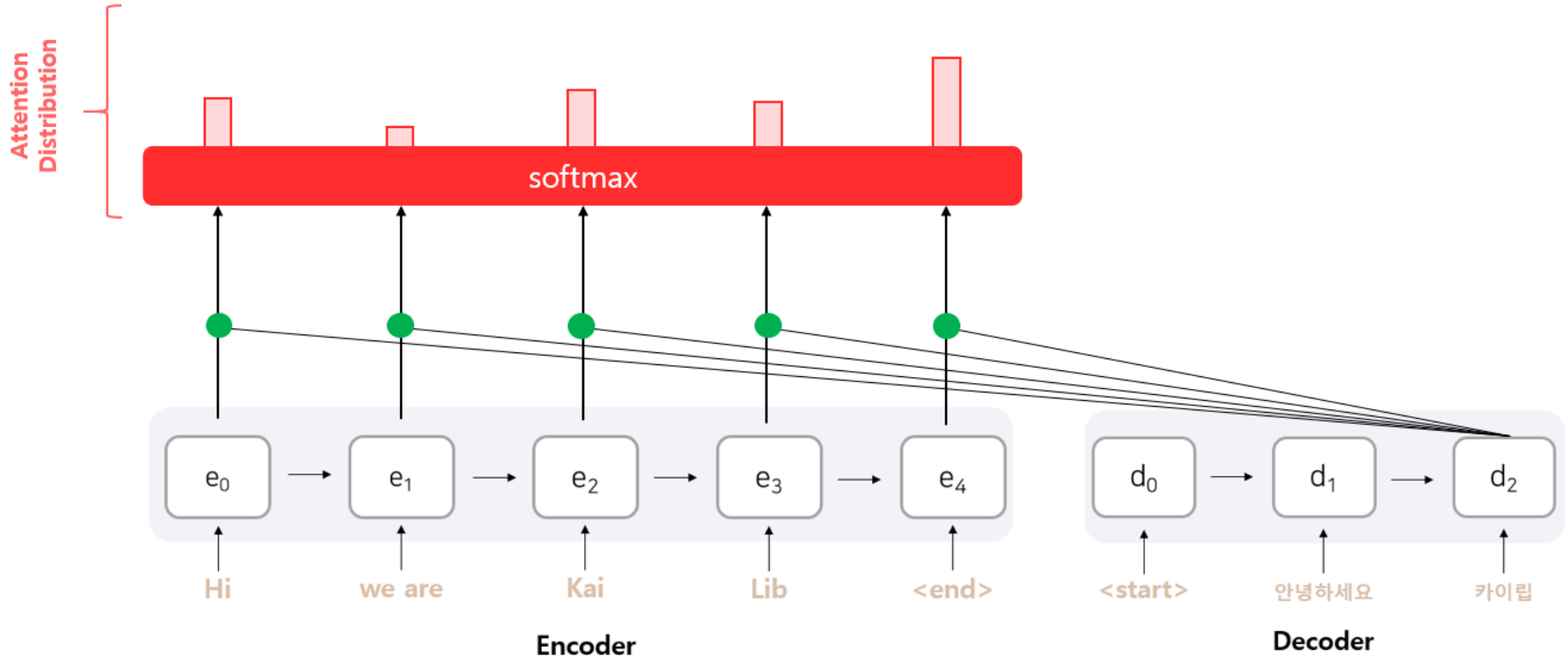


Dot-Product Attention에서는 이 스코어 값을 구하기 위해 $\mathbf{s_t}$(디코더 Hidden State)를 전치(transpose)하고 모든 인코더의 Hidden State와 내적을 수행한다. 벡터의 내적이므로 모든 어텐션 스코어 값은 스칼라이다.

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

Attention Mechanism

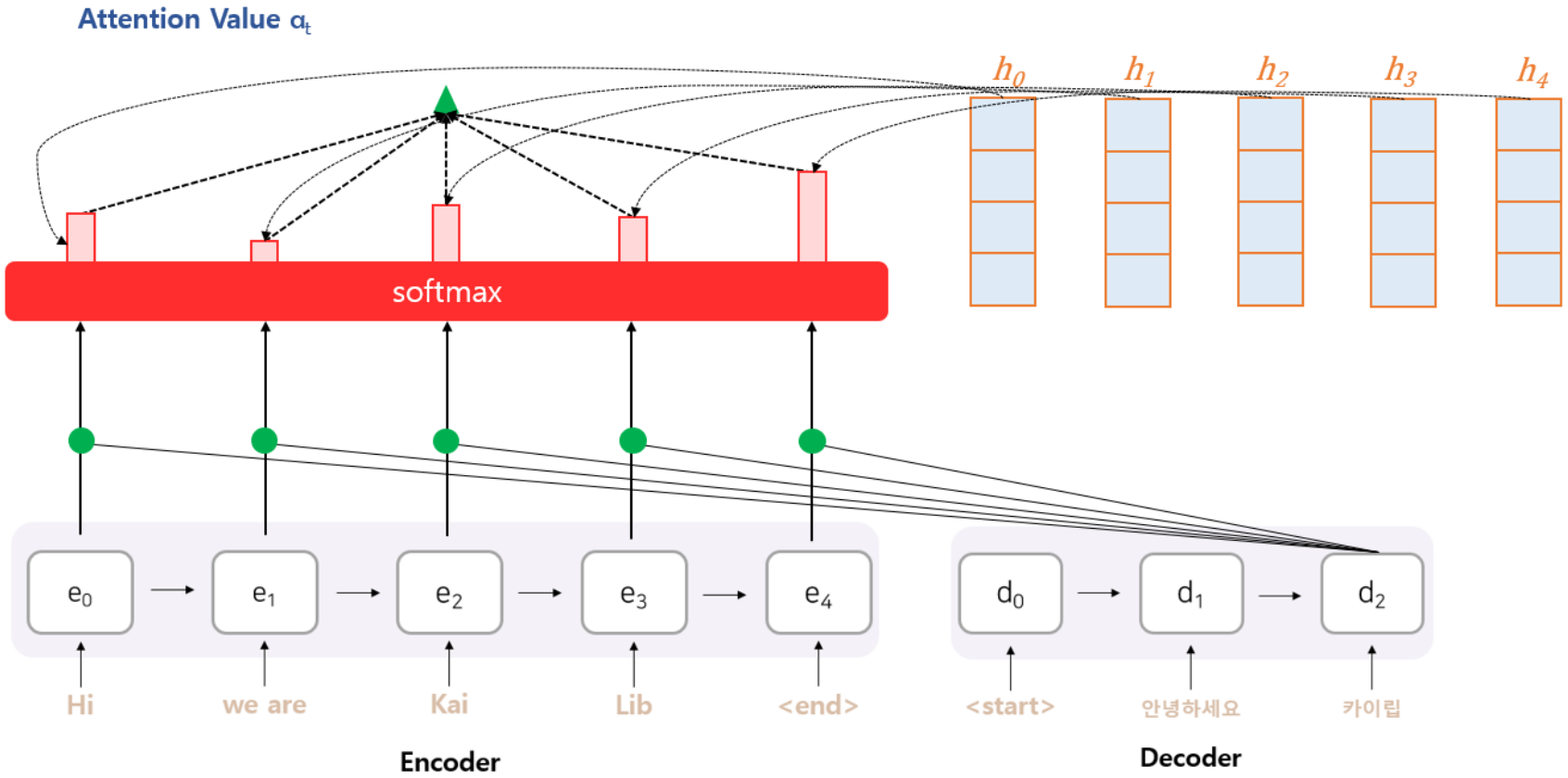
- **Attention Distribution**



다음으로 내적을 통해 얻은 각 어텐션 스코어 값을 소프트맥스 함수를 적용하게 되면 모든 값의 합이 1이 되는 점을 이용해서 어텐션 분포(Attention Distribution)을 구한다.

Attention Mechanism

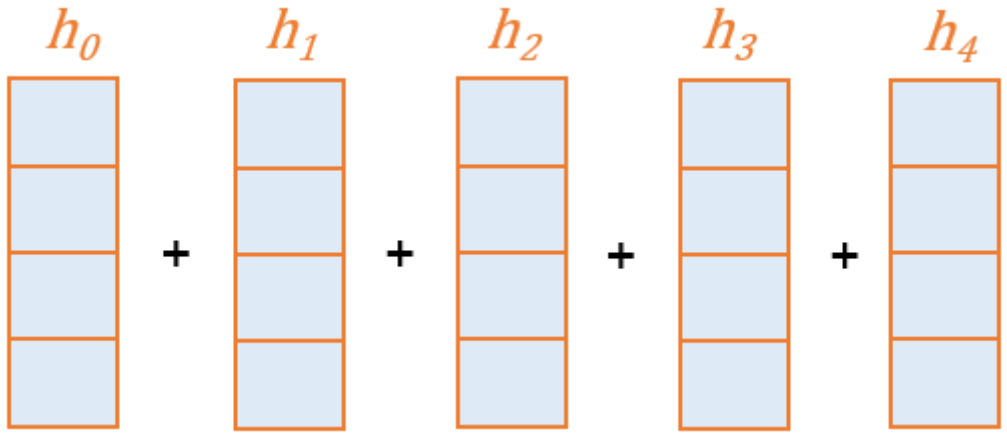
- **Attention Distribution X Encoder Hidden State**



소프트맥스를 통해 얻은 어텐션 분포를 각 인코더의 Hidden State와 곱해준다.

Attention Mechanism

- **Weight Sum**



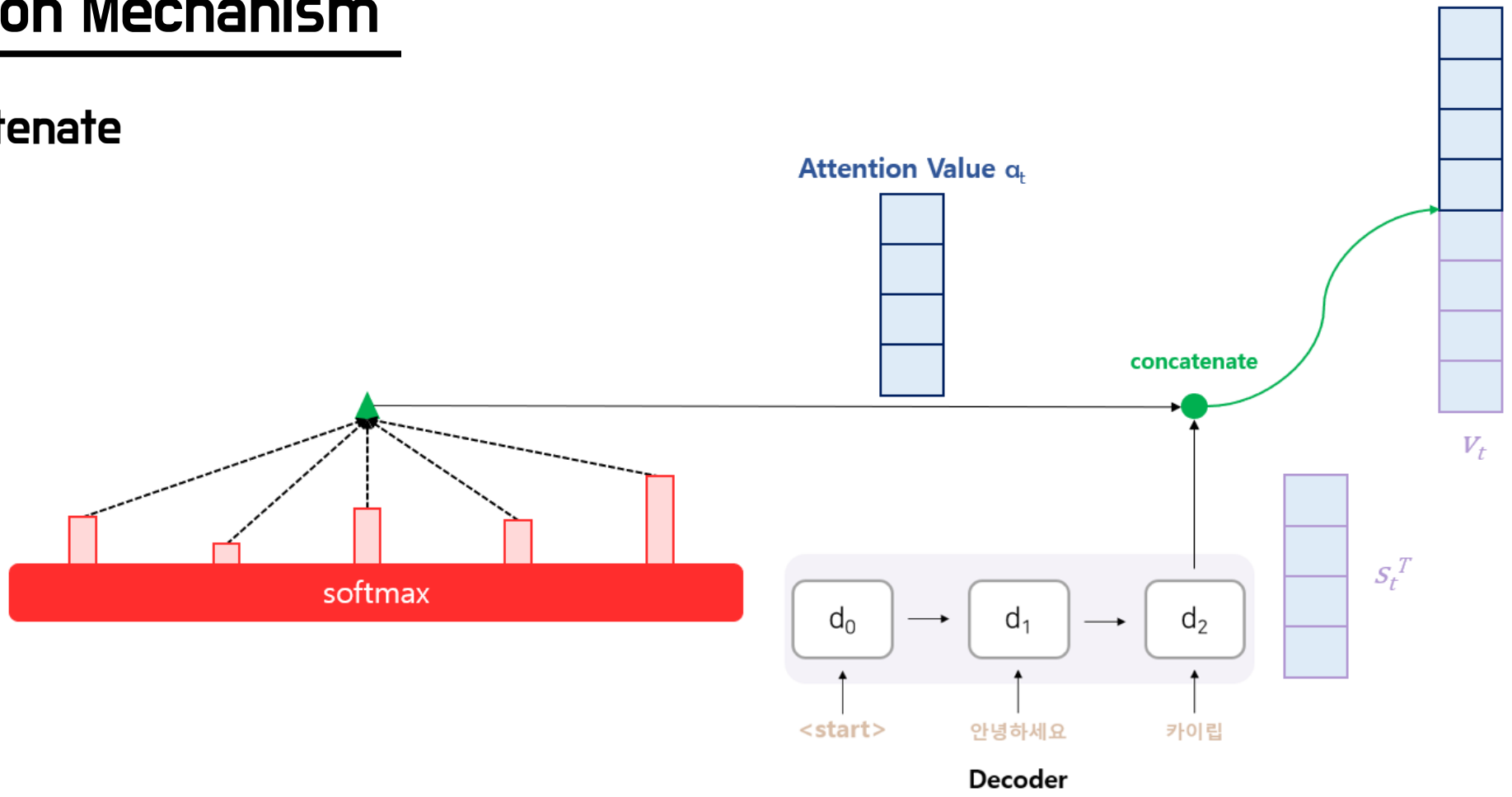
각 어텐션 분포와의 곱을 통해 얻어진 Hidden State들을 전부 더한다.

(element-wise sum)

※ 인코더의 Hidden State와 어텐션 분포의 곱과 가중합(Weight Sum)의 예시는 부록 참고 ※

Attention Mechanism

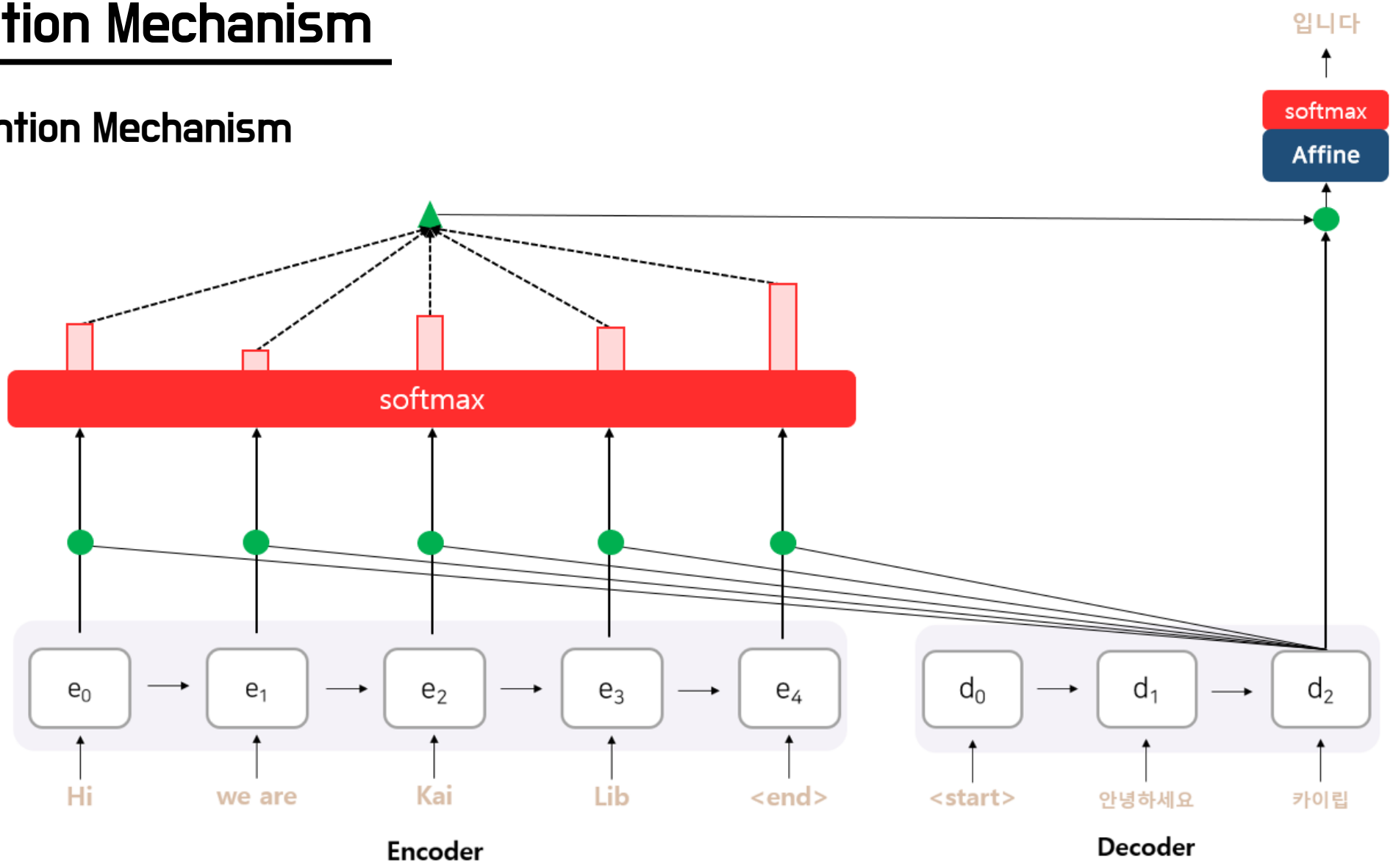
- **Concatenate**



그렇게 구한 어텐션 값과 t시점의 디코더의 Hidden State를 연결한다

Attention Mechanism

- **Attention Mechanism**



Attention Mechanism

- **Attention Mechanism (feat. Kai)**

각 디코딩 스텝마다 어텐션 스코어를 계산해서 추론 시 점수에 반영한다

Attention Mechanism

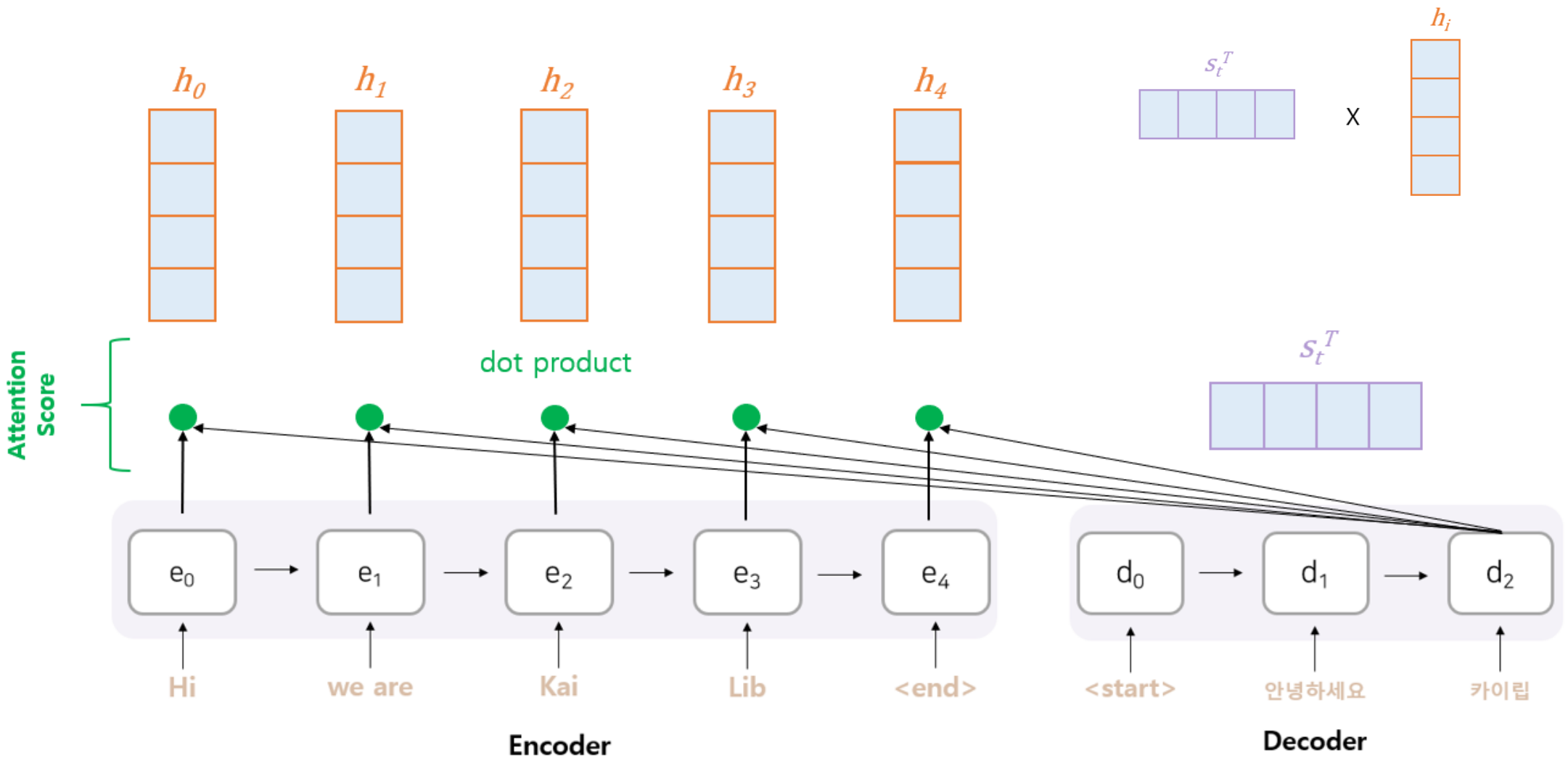
- **다양한 종류의 어텐션**

이름	스코어 함수
dot	$score(s_t,\ h_i) = s_t^T h_i$
$scaled\ dot$	$score(s_t,\ h_i) = \frac{s_t^T h_i}{\sqrt{n}}$
$general$	$score(s_t,\ h_i) = s_t^T W_a h_i$ // 단, W_a는 학습 가능한 가중치 행렬
$concat$	$score(s_t,\ h_i) = v_a^T\ tanh(W_a[s_t; h_i])$
$location - base$	$\alpha_t = softmax(W_a s_t)$ // α_t 산출 시에 s_t만 사용하는 방법.

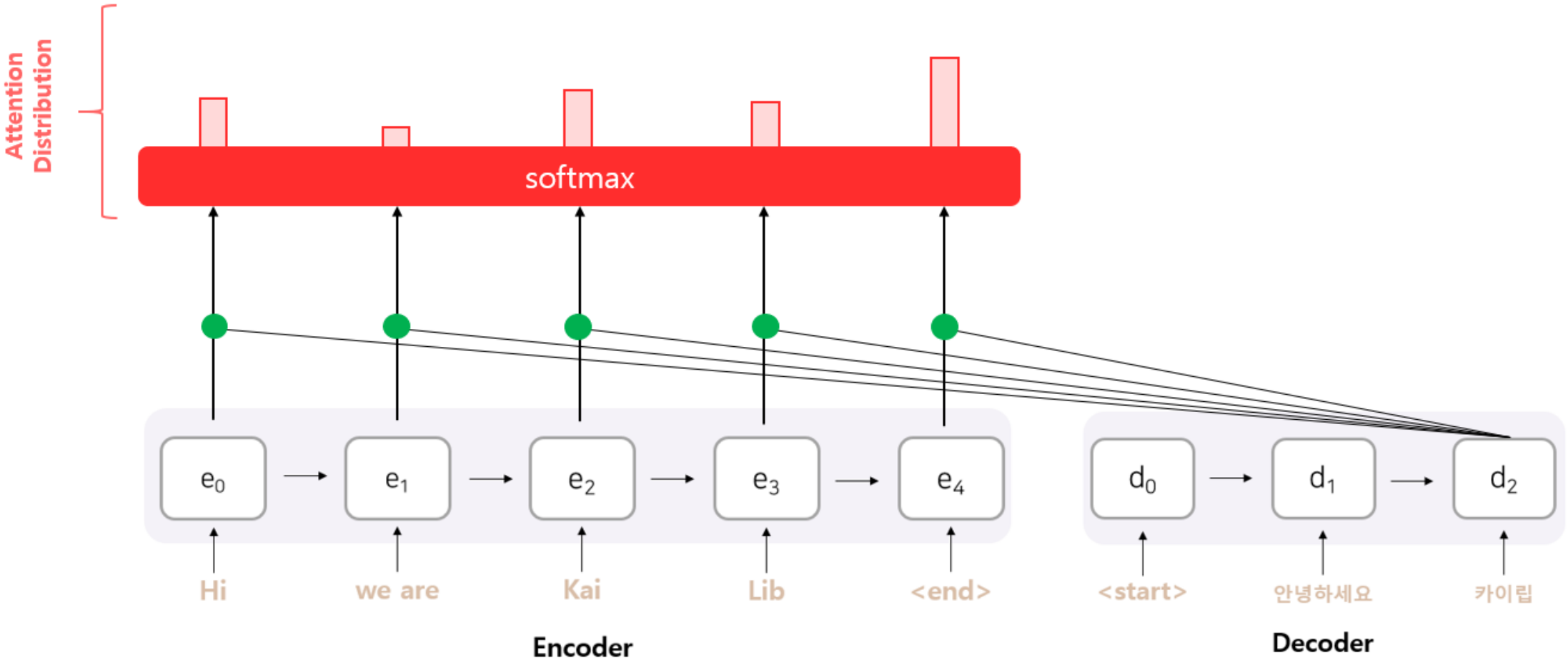
앞에서까지 Seq2seq + Attention 모델에 쓰일 수 있는 다양한 어텐션 종류 중 Dot-Product를 예시로 어텐션을 살펴봤다.
닷-프로덕트 어텐션과 다른 어텐션들의 차이는 중간 수식의 차이일 뿐이다. 여기서 말하는 중간 수식은 어텐션 스코어 함수를 말한다. 앞에서까지 본 어텐션이 닷-프로덕트 어텐션인 이유는 어텐션 스코어를 구하는 방법이 내적이였기 때문이다.
이외에도 어텐션 스코어를 구하는 방법은 여러가지가 제시되어 있으며 위의 표 이외에도 다양한 기법이 존재한다.

① Attention Score 계산

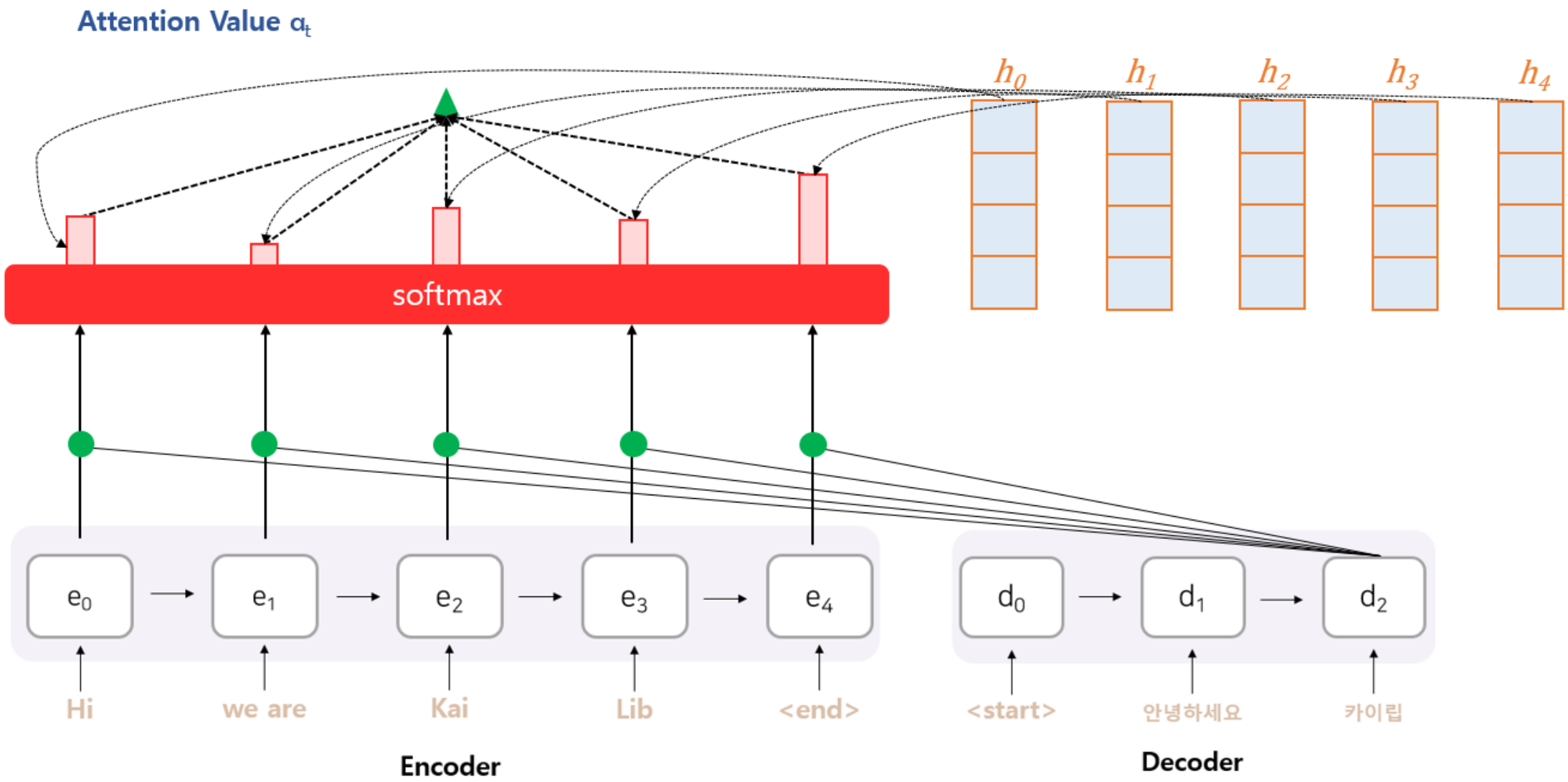
Dot-Product Attention



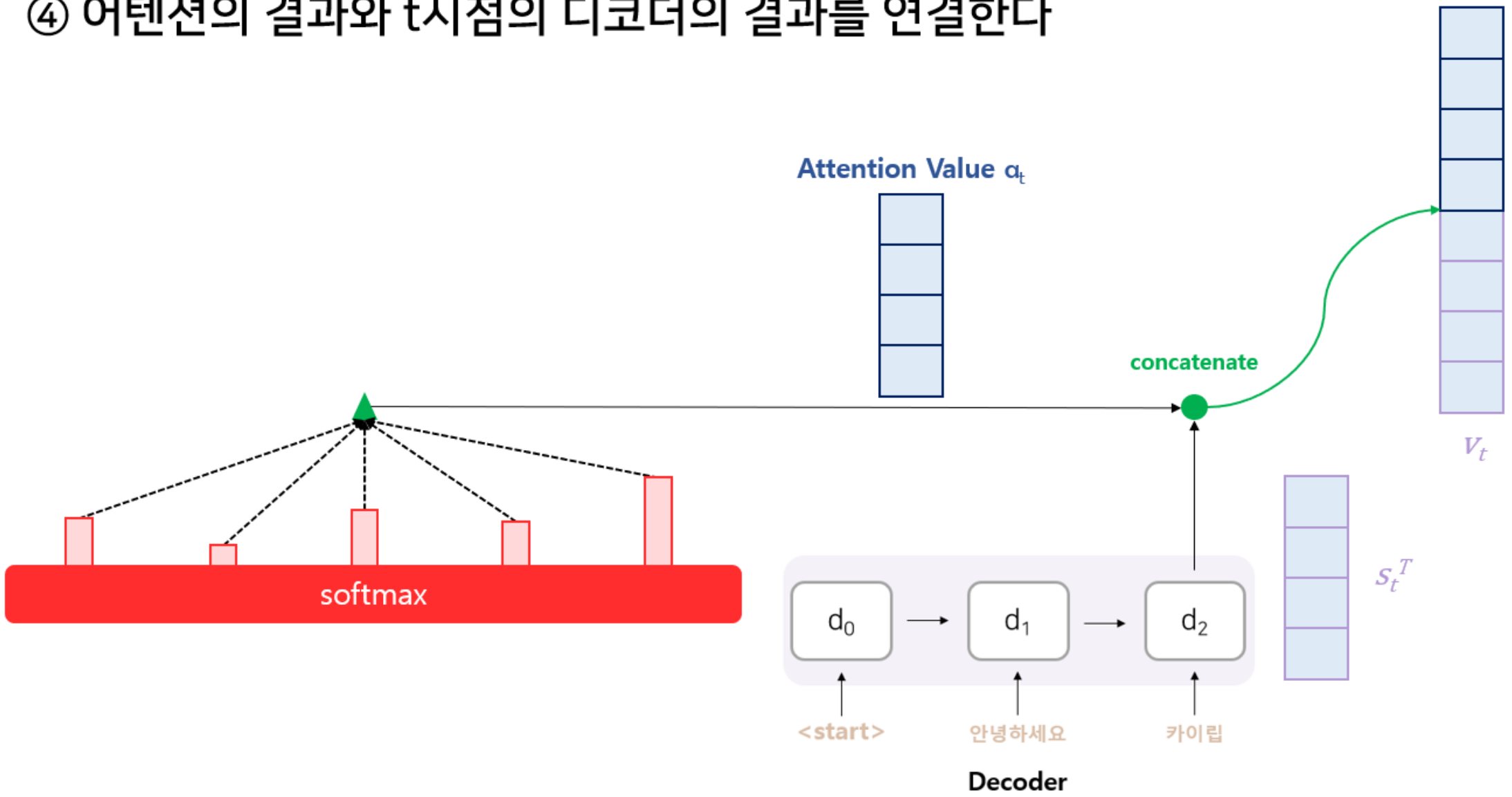
② 소프트맥스 함수를 통해 Attention Distribution을 구한다.



③ Attention Distribution과 인코더의 Hidden State들을 각각 곱한다

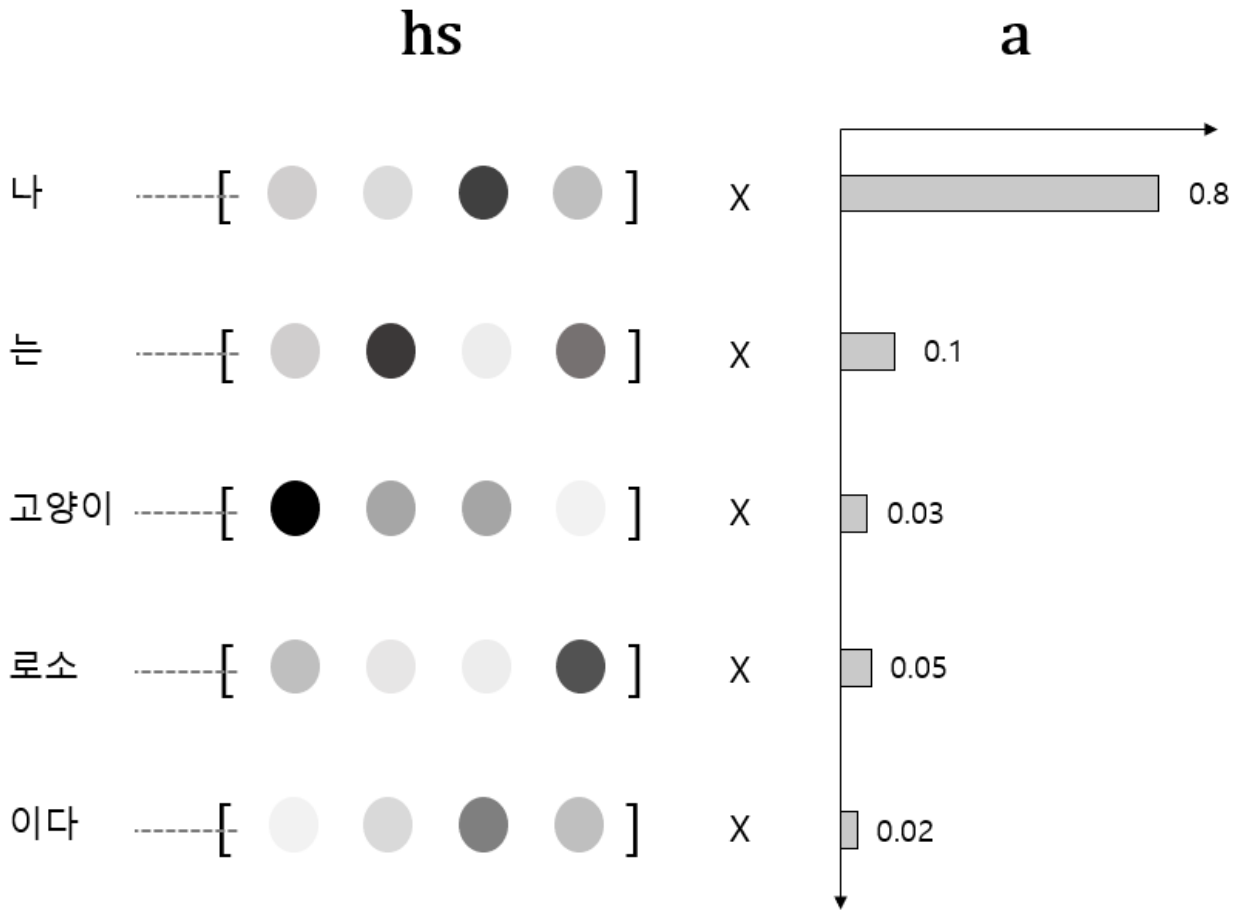


④ 어텐션의 결과와 t시점의 디코더의 결과를 연결한다



부록

- **Attention Distribution X Encoder Hidden State**



부록

- **Weight Sum**