딥 러닝을 이용한 자연어 처리 입문

15. 어텐션 메커니즘 (Attention Mechanism)

15-01 어텐션 메커니즘 (Attention Mechanism)

RNN에 기반한 seq2seq 모델 두가지 문제

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

둘째, RNN의 고질적인 문제인 기울기 소실(vanishing gradient) 문제

15-01 어텐션 메커니즘 (Attention Mechanism)

1. 어텐션(Attention)의 아이디어

인코더- 입력 문장을 고정된 크기의 벡터로 변환

디코더- 이 벡터를 활용하여 출력 문장을 한 단어씩 생성

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

**단, 전체 입력 문장을 전부 다 동일한 비율로 참고하는 것이 아니라,
해당 시점에서 예측해야할 단어와 연관이 있는
입력 단어 부분을 좀 더 집중(attention)**

15-01 어텐션 메커니즘 (Attention Mechanism)

2. 어텐션 함수(Attention Function)

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
```

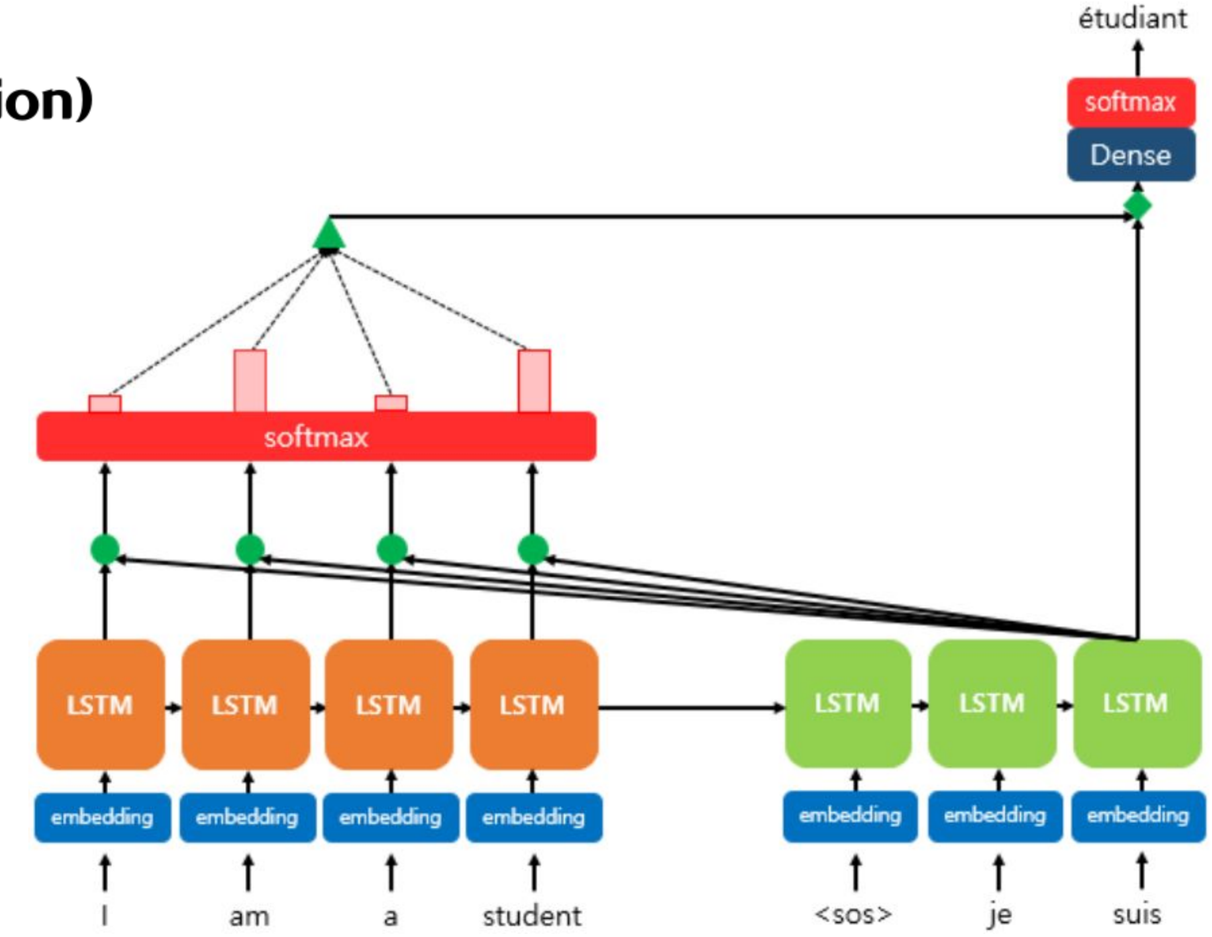
'쿼리(Query)'와 '키(Key)'의 유사도를 각각 구하기
> 구해낸 이 유사도를 키와 맵핑되어있는 각각의 '값(Value)'에 반영
>그리고 유사도가 반영된 '값(Value)'을 모두 더해서 리턴

15-01 어텐션 메커니즘 (Attention Mechanism)

3. 닷-프로덕트 어텐션(Dot-Product Attention)

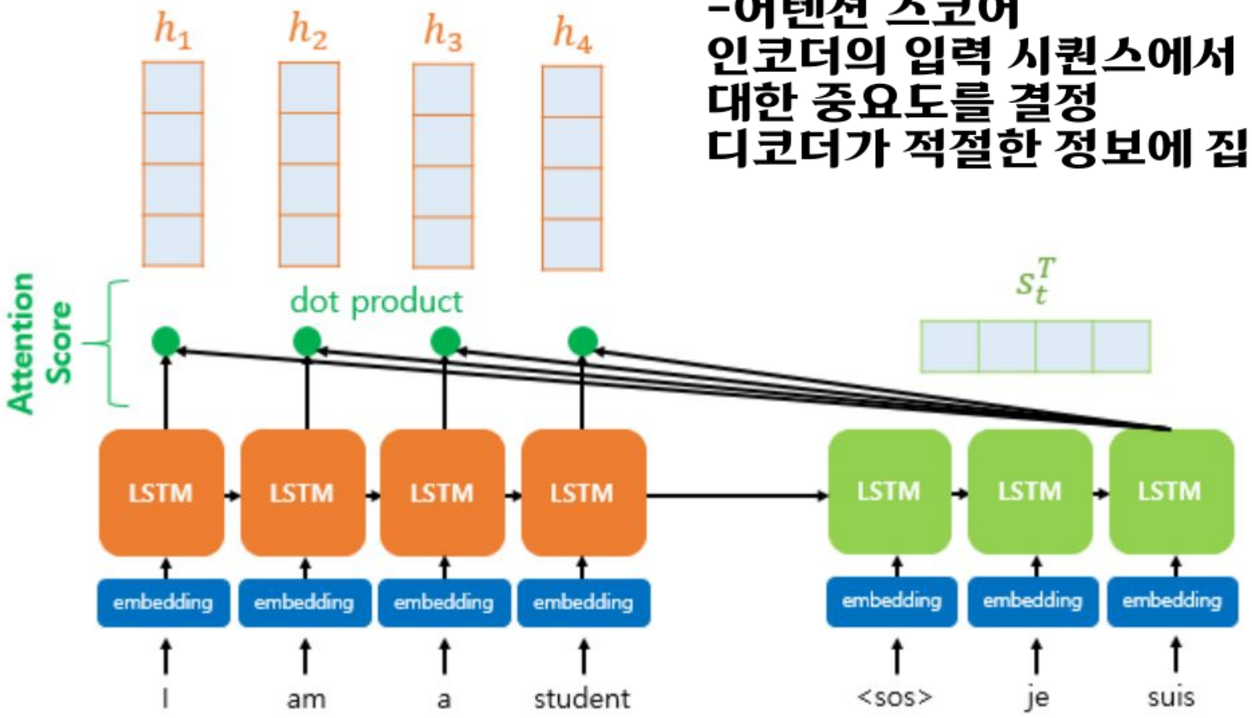
쿼리 벡터와 키 벡터 사이의 내적을 이용하여 유사도를 계산

내적은 두 벡터가 얼마나 비슷한 방향을 가지고 있는지를 나타내는 값으로 사용



15-01 어텐션 메커니즘 (Attention Mechanism)

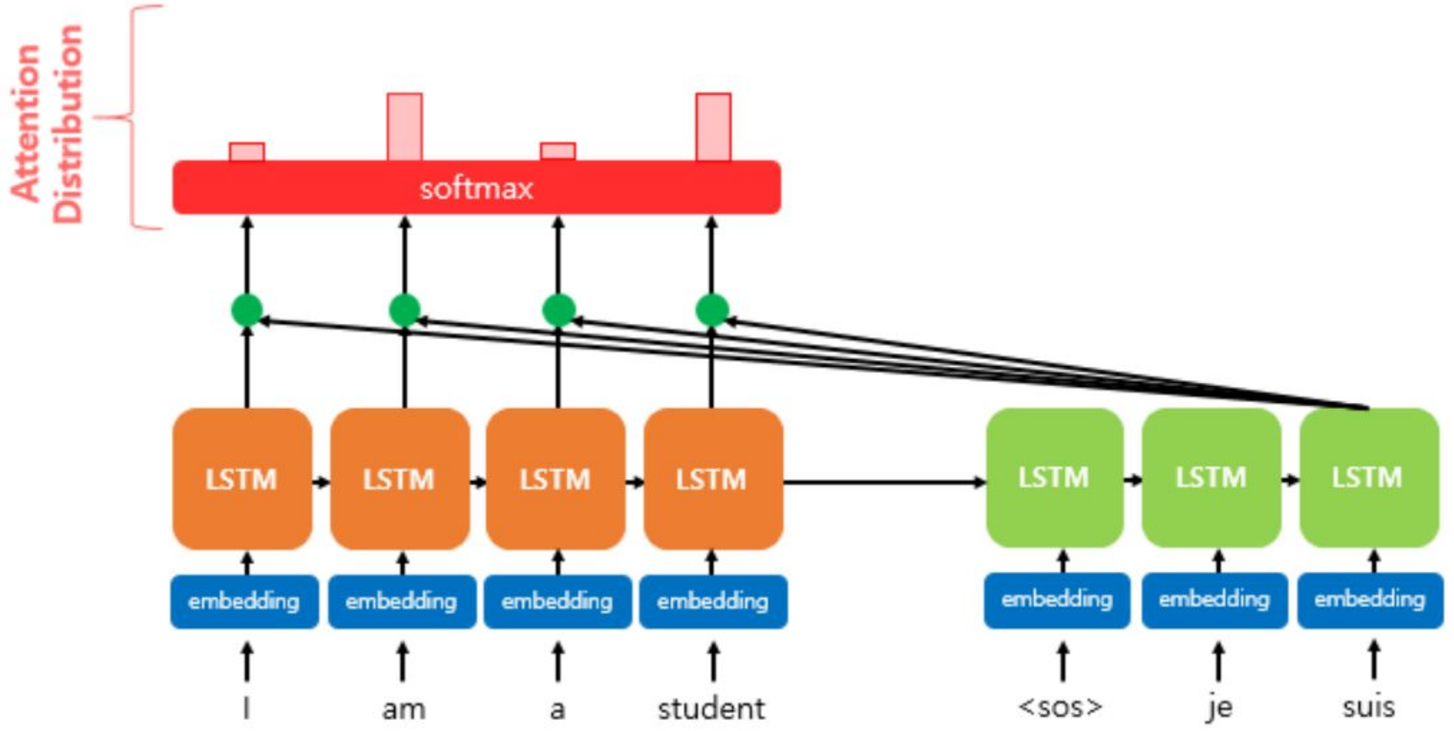
3. 닷-프로덕트 어텐션(Dot-Product Attention)
1) 어텐션 스코어(Attention Score)를 구한다.



-어텐션 스코어
인코더의 입력 시퀀스에서 디코더의 현재 예측 위치에 대한 중요도를 결정
디코더가 적절한 정보에 집중하여 출력 시퀀스를 생성

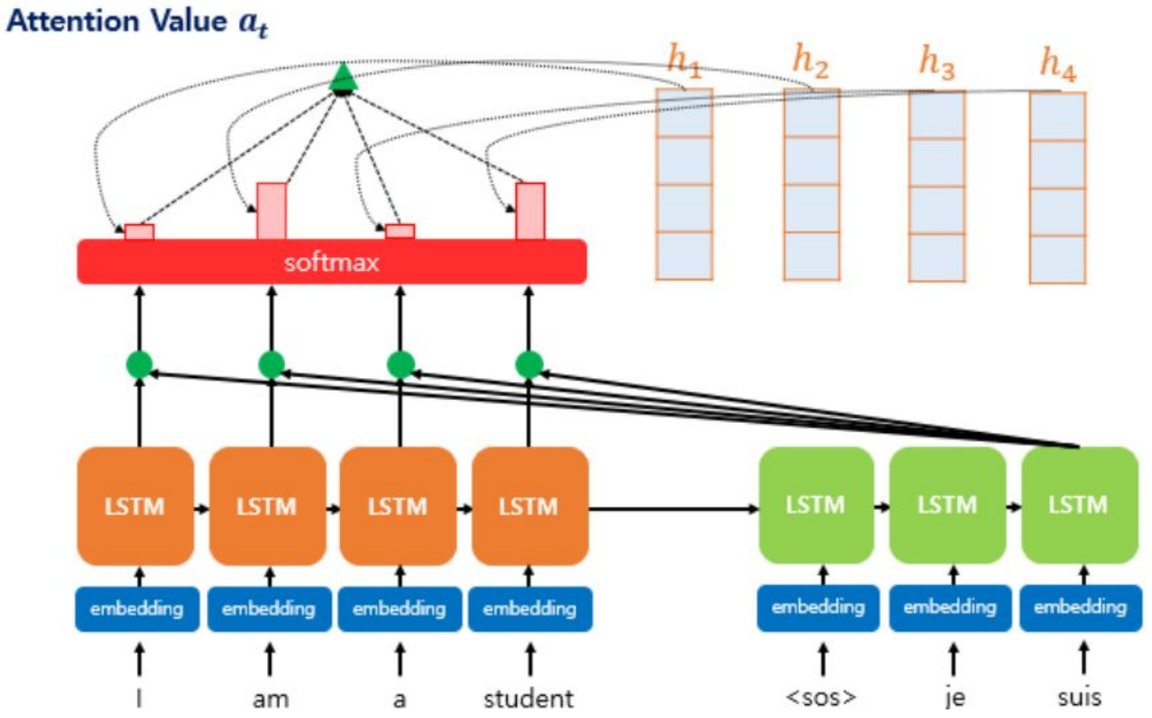
15-01 어텐션 메커니즘 (Attention Mechanism)

3. 닷-프로덕트 어텐션(Dot-Product Attention)
2) 소프트맥스(softmax) 함수를 통해 어텐션 분포(Attention Distribution)를 구한다.



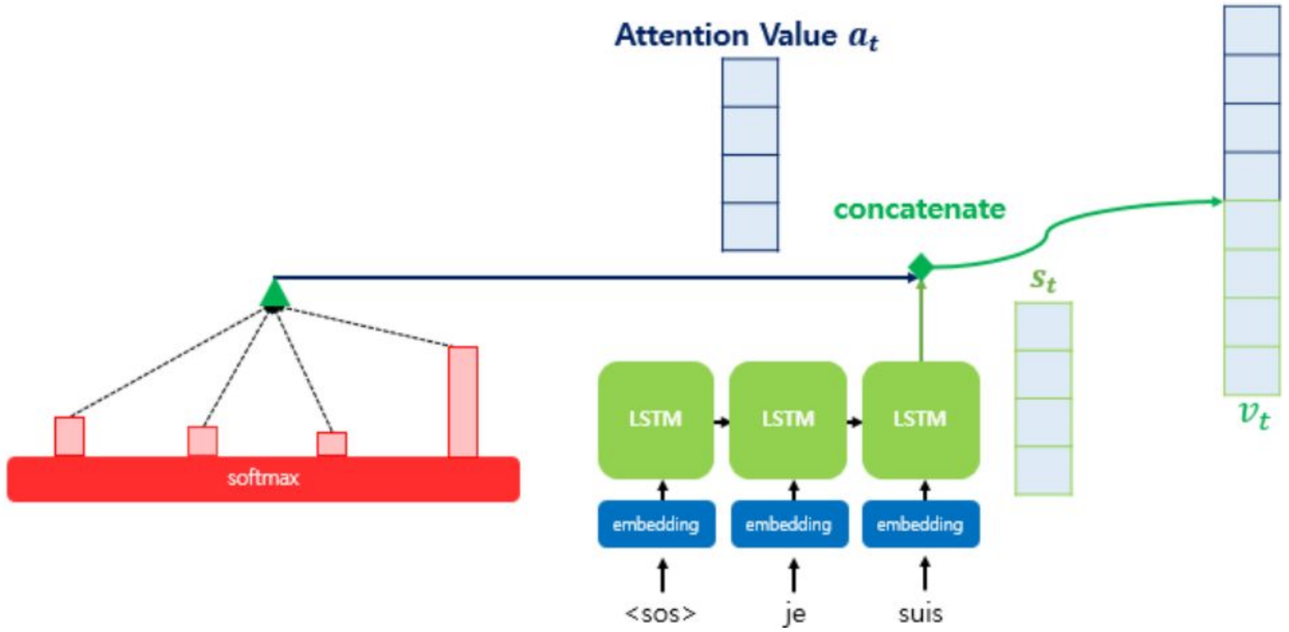
15-01 어텐션 메커니즘 (Attention Mechanism)

3. 닷-프로덕트 어텐션(Dot-Product Attention)
3) 각 인코더의 어텐션 가중치와 은닉 상태를 가중합하여 어텐션 값(Attention Value)을 구한다.



15-01 어텐션 메커니즘 (Attention Mechanism)

3. 닷-프로덕트 어텐션(Dot-Product Attention)
4) 어텐션 값과 디코더의 t 시점의 은닉 상태를 연결한다.(Concatenate)



15-02 바다나우 어텐션 (Bahdanau Attention)

1. 바다나우 어텐션 함수(Bahdanau Attention Function)

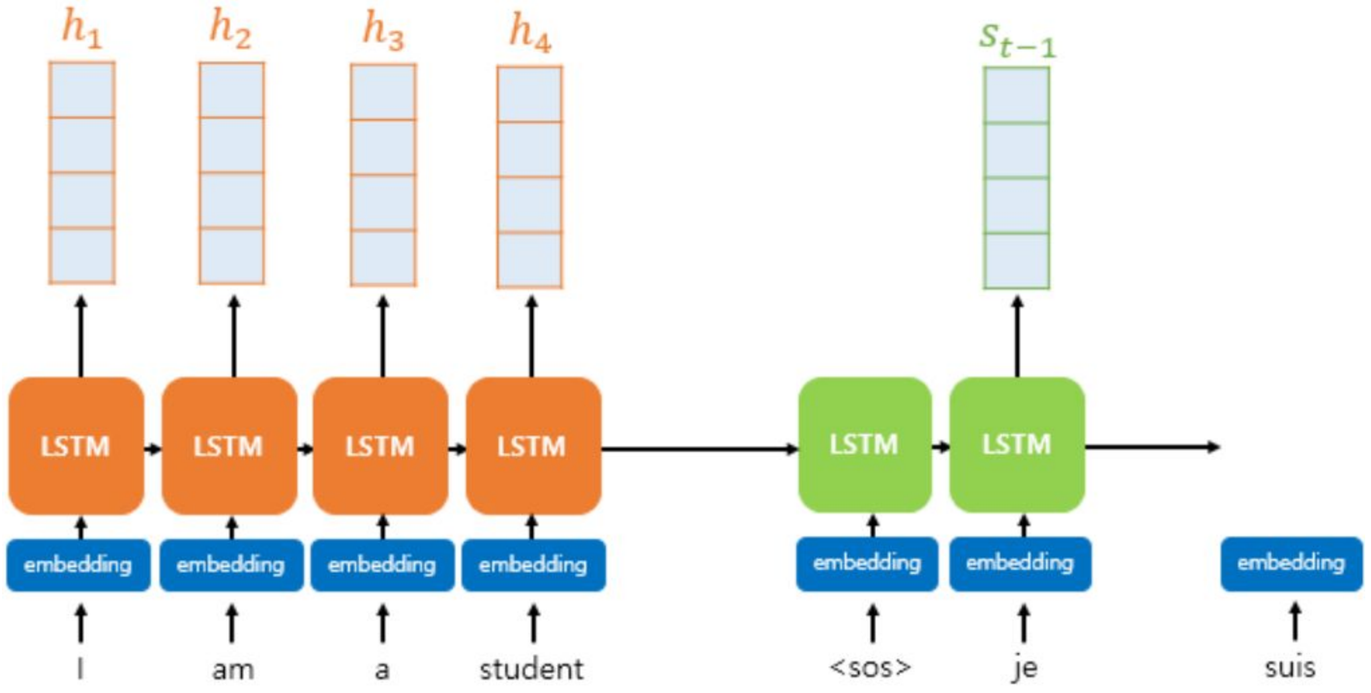
$$\text{Attention}(Q, K, V) = \text{Attention Value}$$

```
t = 어텐션 메커니즘이 수행되는 디코더 셀의 현재 시점을 의미.

Q = Query : t-1 시점의 디코더 셀에서의 은닉 상태
K = Keys : 모든 시점의 인코더 셀의 은닉 상태들
V = Values : 모든 시점의 인코더 셀의 은닉 상태들
```

15-02 바다나우 어텐션 (Bahdanau Attention)

1. 바다나우 어텐션 함수(Bahdanau Attention Function)



15-02 바다나우 어텐션 (Bahdanau Attention)

2. 바다나우 어텐션(Bahdanau Attention)

```
class BahdanauAttention(tf.keras.Model):
  def __init__(self, units):
    super(BahdanauAttention, self).__init__()
    self.W1 = Dense(units)
    self.W2 = Dense(units)
    self.V = Dense(1)

  def call(self, values, query): # 단, key와 value는 같음
    # query shape == (batch_size, hidden size)
    # hidden_with_time_axis shape == (batch_size, 1, hidden size)
    # score 계산을 위해 뒤에서 할 덧셈을 위해서 차원을 변경해줍니다.
    hidden_with_time_axis = tf.expand_dims(query, 1)

    # score shape == (batch_size, max_length, 1)
    # we get 1 at the last axis because we are applying score to self.V
    # the shape of the tensor before applying self.V is (batch_size, max_length, units)
    score = self.V(tf.nn.tanh(
        self.W1(values) + self.W2(hidden_with_time_axis)))

    # attention_weights shape == (batch_size, max_length, 1)
    attention_weights = tf.nn.softmax(score, axis=1)

    # context_vector shape after sum == (batch_size, hidden_size)
    context_vector = attention_weights * values
    context_vector = tf.reduce_sum(context_vector, axis=1)

    return context_vector, attention_weights
```

15-03 양방향 LSTM과 어텐션 메커니즘(BiLSTM with Attention mechanism)

텍스트 분류에서 어텐션 메커니즘을 사용하는 이유

RNN은 시퀀스 데이터를 처리할 수 있는 모델로,
각 단어를 순차적으로 입력으로 받아들이고 이전 단계의 은닉 상태를 활용하여
현재 단계의 출력을 생성

> RNN의 마지막 은닉 상태는 전체 시퀀스를 요약하는 정보로 사용되며,
이를 통해 텍스트 분류 작업을 수행

그러나 RNN은 시퀀스가 길어질수록 문제가 발생!
> 이를 해결하기 위해 어텐션 메커니즘
어텐션 메커니즘은 입력 시퀀스의 모든 단어를 참고하여 출력 시퀀스를 생성
= 모델이 입력 시퀀스의 모든 정보를 중요한 부분부터 차례대로 집중하여 활용