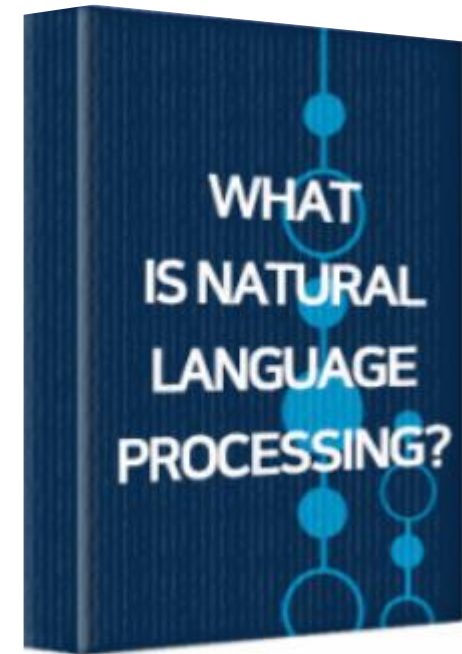


Tensorflow를 활용한 딥러닝 자연어 처리 입문. 7강

앞으로 배우게 될 내용

- Text preprocessing for NLP & Language Model
- Basic Tensorflow & Vectorization
- Word Embedding (Word2Vec, FastText, GloVe)
- Text Classification (using RNN & CNN)
- Chatbot with Deep Learning
- Sequence to Sequence
- **Attention Mechanism**
- Transformer & BERT



참고 자료 : <https://wikidocs.net/book/2155>

Attention Mechanism

What is Attention?

어학사전

영어사전

attention 미국·영국 [ə'tenʃn]  영국식  ★★ [다른 뜻\(4건\)](#) | [예문보기](#)

1. 주의 (집중), 주목 2. 관심, 흥미 3. (관심을 끌기 위한) 행동

프랑스어사전



attention [atãsjõ]  ★★ [다른 뜻\(2건\)](#) | [예문보기](#)

[여성명사] 1. 주의(력),조심,긴장,관심 2. 친절,정중,배려 = amabilité,empressement,prévenance

What is Attention?

어학사전

영어사전

attention 미국·영국 [ə'tenʃn]  영국식  ★★ [다른 뜻\(4건\)](#) | [예문보기](#)

1. 주의 (집중), 주목 2. 관심, 흥미 3. (관심을 끌기 위한) 행동

프랑스어사전



attention [atãsjõ]  ★★ [다른 뜻\(2건\)](#) | [예문보기](#)

[여성명사] 1. 주의(력),조심,긴장,관심 2. 친절,정중,배려 = amabilité,empressement,prévenance

What is Attention?

어학사전

영어사전

attention 미국·영국 [ə'tenʃn]  영국식  ★★ [다른 뜻\(4건\)](#) | [예문보기](#)

1. 주의 (집중), 주목 2. 관심, 흥미 3. (관심을 끌기 위한) 행동

프랑스어사전

attention [atãsjõ]  ★★ [다른 뜻\(2건\)](#) | [예문보기](#)

[여성명사] 1. 주의(력),조심,긴장,관심 2. 친절,정중,배려 = amabilité,empressement,prévenance

풀고자 하는 Task의 핵심이 되는 정보를 찾아서 집중!

Sentiment Analysis with Attention Mechanism

- 감성 분석의 결과에 영향을 주는 단어에 **Attention**.

훈련 데이터

예측 결과

리뷰1 : 와 이 영화 진심 존잼!!!

리뷰2 : 실망대망. 교과서적 연출로도 평타 이상은 가능할
좀비소재를 이렇게까지...

Sentiment Analysis with Attention Mechanism

- 감성 분석의 결과에 영향을 주는 단어에 **Attention**.

훈련 데이터

예측 결과

리뷰1 : 와 이 영화 진심 **존잼!!!**

리뷰2 : 실망대망. 교과서적 연출로도 평타 이상은 가능할
좀비소재를 이렇게까지...

Sentiment Analysis with Attention Mechanism

- 감성 분석의 결과에 영향을 주는 단어에 **Attention**.

훈련 데이터

리뷰1 : 와 이 영화 진심 **존잼!!!**

리뷰2 : 실망대망. 교과서적 연출로도 평타 이상은 가능할
좀비소재를 이렇게까지...

예측 결과

긍정

Sentiment Analysis with Attention Mechanism

- 감성 분석의 결과에 영향을 주는 단어에 **Attention**.

훈련 데이터

예측 결과

리뷰1 : 와 이 영화 진심 **존잼!!!**

긍정

리뷰2 : **실망대망**. 교과서적 연출로도 평타 이상은 가능할 좀비소재를 이렇게까지...

Sentiment Analysis with Attention Mechanism

- 감성 분석의 결과에 영향을 주는 단어에 **Attention**.

훈련 데이터

예측 결과

리뷰1 : 와 이 영화 진심 **존잼!!!**

긍정

리뷰2 : **실망대망**. 교과서적 연출로도 평타 이상은 가능할
좀비소재를 이렇게까지...

부정

Neural Machine Translation with Attention Mechanism

- 번역 문장을 만드는 과정에서 기존 문장에서 중요한 단어를 **Attention**.

I am a good student

Neural Machine Translation with Attention Mechanism

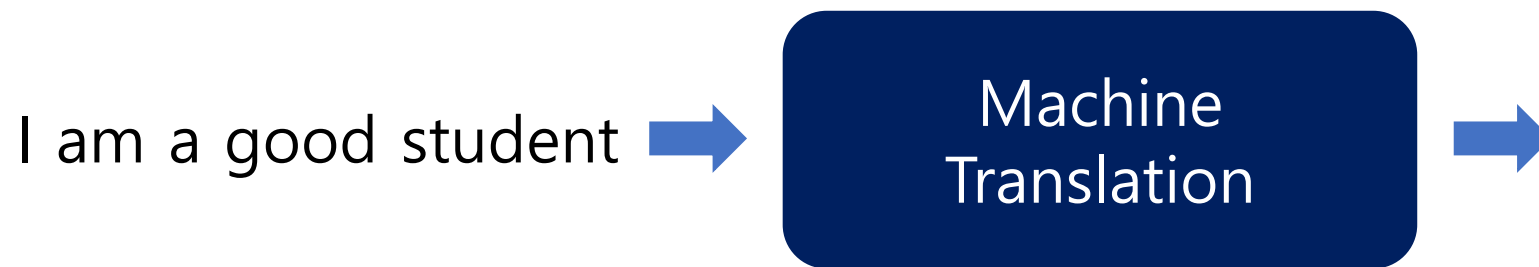
- 번역 문장을 만드는 과정에서 기존 문장에서 중요한 단어를 **Attention**.

I am a good student →

Machine
Translation

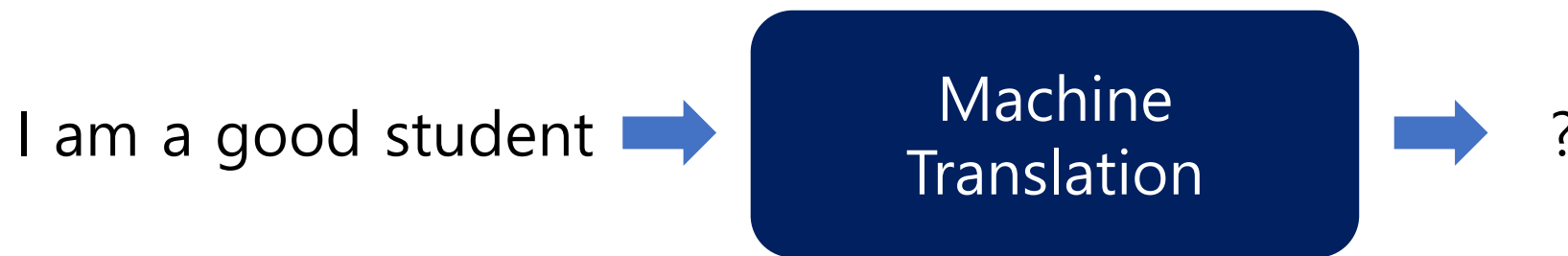
Neural Machine Translation with Attention Mechanism

- 번역 문장을 만드는 과정에서 기존 문장에서 중요한 단어를 **Attention**.



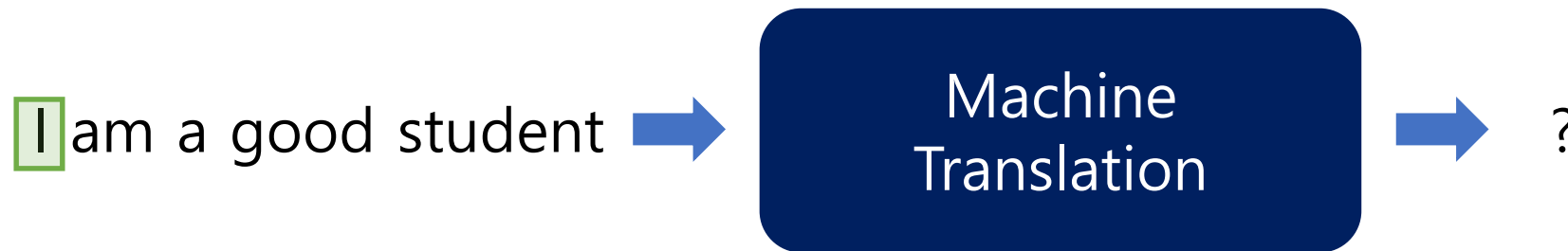
Neural Machine Translation with Attention Mechanism

- 번역 문장을 만드는 과정에서 기존 문장에서 중요한 단어를 **Attention**.



Neural Machine Translation with Attention Mechanism

- 번역 문장을 만드는 과정에서 기존 문장에서 중요한 단어를 **Attention**.



Neural Machine Translation with Attention Mechanism

- 번역 문장을 만드는 과정에서 기존 문장에서 중요한 단어를 **Attention**.



Neural Machine Translation with Attention Mechanism

- 번역 문장을 만드는 과정에서 기존 문장에서 중요한 단어를 **Attention**.



Neural Machine Translation with Attention Mechanism

- 번역 문장을 만드는 과정에서 기존 문장에서 중요한 단어를 **Attention**.



Neural Machine Translation with Attention Mechanism

- 번역 문장을 만드는 과정에서 기존 문장에서 중요한 단어를 **Attention**.



Neural Machine Translation with Attention Mechanism

- 번역 문장을 만드는 과정에서 기존 문장에서 중요한 단어를 **Attention**.



Neural Machine Translation with Attention Mechanism

- 번역 문장을 만드는 과정에서 기존 문장에서 중요한 단어를 **Attention**.



Neural Machine Translation with Attention Mechanism

- 번역 문장을 만드는 과정에서 기존 문장에서 중요한 단어를 **Attention**.



Neural Machine Translation with Attention Mechanism

- 번역 문장을 만드는 과정에서 기존 문장에서 중요한 단어를 **Attention**.



Neural Machine Translation with Attention Mechanism

- 번역 문장을 만드는 과정에서 기존 문장에서 중요한 단어를 **Attention**.



Neural Machine Translation with Attention Mechanism

- 번역 문장을 만드는 과정에서 기존 문장에서 중요한 단어를 **Attention**.



Neural Machine Translation with Attention Mechanism

- 번역 문장을 만드는 과정에서 기존 문장에서 중요한 단어를 **Attention**.



대표적인 모델 : seq2seq with attention mechanism
디코더의 모델 생성 과정에서 어텐션을 수행한다.

Question Answering with Attention Mechanism

- 질문의 정답을 찾는 과정에서 문서에서 중요한 단어를 **Attention**.

본문

다니엘이 정원으로 갔다.
그때 매리가 침실로 들어갔다.
산드라는 마침 화장실로 갔다.
매리가 다시 거실로 나왔다.
다니엘은 부엌으로 갔다.
산드라 또한 다기 거실로 왔다.

Question Answering with Attention Mechanism

- 질문의 정답을 찾는 과정에서 문서에서 중요한 단어를 **Attention**.

본문

다니엘이 정원으로 갔다.
그때 매리가 침실로 들어갔다.
산드라는 마침 화장실로 갔다.
매리가 다시 거실로 나왔다.
다니엘은 부엌으로 갔다.
산드라 또한 다기 거실로 왔다.

질문 : 부엌에 있는 사람은 누구인가요?

Question Answering with Attention Mechanism

- 질문의 정답을 찾는 과정에서 문서에서 중요한 단어를 **Attention**.

본문

다니엘이 정원으로 갔다.
그때 매리가 침실로 들어갔다.
산드라는 마침 화장실로 갔다.
매리가 다시 거실로 나왔다.
다니엘은 부엌으로 갔다.
산드라 또한 다기 거실로 왔다.

질문 : **부엌**에 있는 사람은 누구인가요?

Question Answering with Attention Mechanism

- 질문의 정답을 찾는 과정에서 문서에서 중요한 단어를 **Attention**.

본문

다니엘이 정원으로 갔다.
그때 매리가 침실로 들어갔다.
산드라는 마침 화장실로 갔다.
매리가 다시 거실로 나왔다.
다니엘은 **부엌**으로 갔다.
산드라 또한 다기 거실로 왔다.

질문 : **부엌**에 있는 사람은 누구인가요?

Question Answering with Attention Mechanism

- 질문의 정답을 찾는 과정에서 문서에서 중요한 단어를 **Attention**.

본문

다니엘이 정원으로 갔다.
그때 매리가 침실로 들어갔다.
산드라는 마침 화장실로 갔다.
매리가 다시 거실로 나왔다.
다니엘은 **부엌**으로 갔다.
산드라 또한 다기 거실로 왔다.

질문 : **부엌**에 있는 사람은 누구인가요?

다니엘

Natural Language Processing

- 자연어 처리는 크게 자연어 이해(NLU)와 자연어 생성(NLG)의 영역이 있다.
- 자연어 이해는 기계가 자연어를 이해하는 영역을 말한다.
- 자연어 생성은 기계가 텍스트를 스스로 생성하는 영역을 말한다.

$$\text{NLP} = \text{NLU} + \text{NLG}$$

Natural Language Processing

- 자연어 처리는 크게 자연어 이해(NLU)와 자연어 생성(NLG)의 영역이 있다.
- 자연어 이해는 기계가 자연어를 이해하는 영역을 말한다.
- 자연어 생성은 기계가 텍스트를 스스로 생성하는 영역을 말한다.

$$\text{NLP} = \text{NLU} + \text{NLG}$$

생성에서의 어텐션

Neural Machine Translation with Attention Mechanism

- 번역 문장을 만드는 과정에서 기존 문장에서 중요한 단어를 **Attention**.



자연어 생성 과정에서 어텐션을 수행하고 있다.

Natural Language Processing

- 자연어 처리는 크게 자연어 이해(NLU)와 자연어 생성(NLG)의 영역이 있다.
- 자연어 이해는 기계가 자연어를 이해하는 영역을 말한다.
- 자연어 생성은 기계가 텍스트를 스스로 생성하는 영역을 말한다.

$$\text{NLP} = \text{NLU} + \text{NLG}$$

이해에서의 어텐션

Sentiment Analysis with Attention Mechanism

- 감성 분석의 결과에 영향을 주는 단어에 **Attention**.

훈련 데이터

예측 결과

리뷰1 : 와 이 영화 진심 **존잼!!!**

긍정

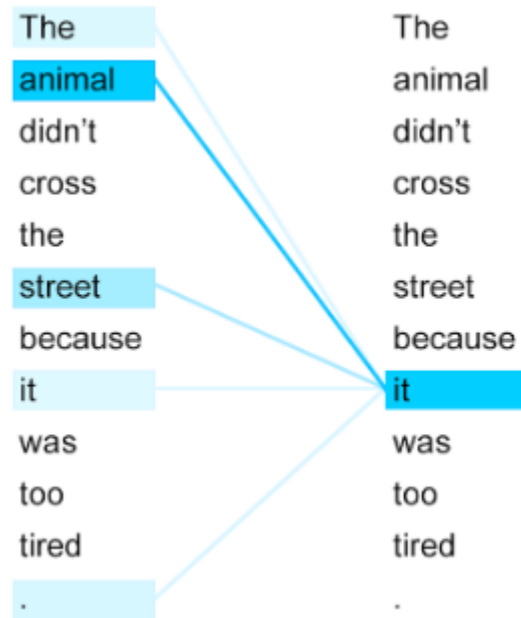
리뷰2 : **실망대망**. 교과서적 연출로도 평타 이상은 가능할
좀비소재를 이렇게까지...

부정

자연어 이해 과정에서 어텐션을 수행하고 있다.

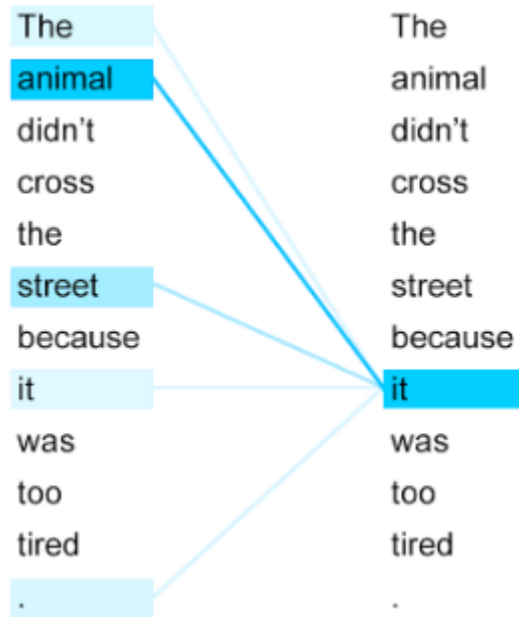
Natural Language Understanding with Self-Attention

- 주어진 자연어 문장을 이해하기 위한 어텐션 기법
- 하나의 문장 내 단어들 간 유사도를 구함으로써 어텐션을 수행



Natural Language Understanding with Self-Attention

- 주어진 자연어 문장을 이해하기 위한 어텐션 기법
- 하나의 문장 내 단어들 간 유사도를 구하므로써 어텐션을 수행



이 내용은 다음 주차 강의에서 다룹니다.

Attention Mechanism

어텐션을 사용하는 모델들.

1. BiLSTM + Attention : 감성 분류, 텍스트 분류
2. Seq2Seq + Attention : 기계 번역, 텍스트 요약
3. End-to-End Memory Network : 질의 응답

어텐션의 종류

1. Bahdanau Attention
2. Luong Attention

Attention Mechanism

어텐션을 사용하는 모델들.

1. BiLSTM + Attention : 감성 분류, 텍스트 분류
2. **Seq2Seq + Attention** : 기계 번역, 텍스트 요약
3. End-to-End Memory Network : 질의 응답

어텐션의 종류

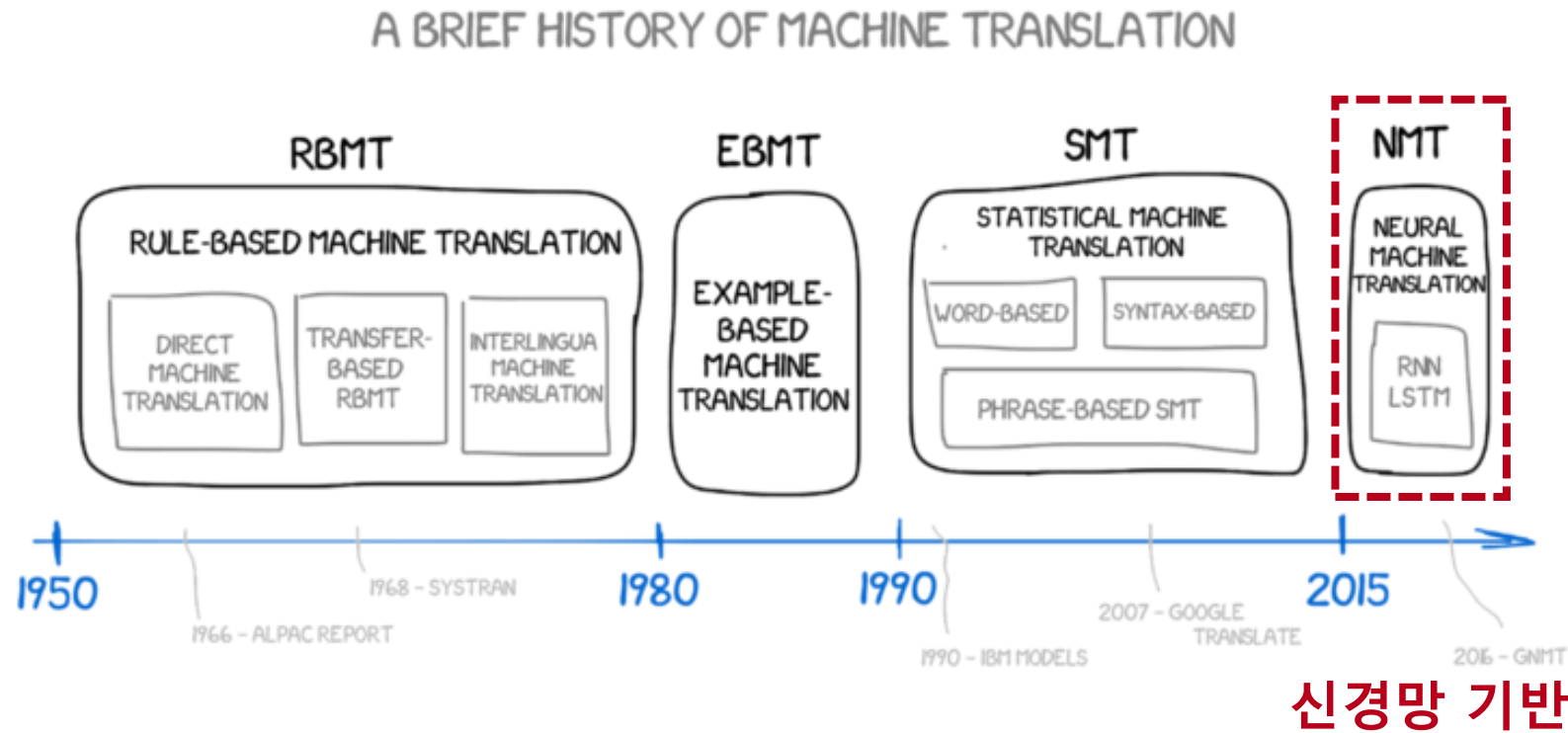
1. Bahdanau Attention
2. Luong Attention

Sequence-to-Sequence

Machine Translation

- 입력 문장(source text)을 번역한 출력 문장(target text)을 생성해내는 Task.

Neural Machine Translation 시대의 시작.



Neural Machine Translation

- 입력 문장(source text)을 번역한 출력 문장(target text)을 생성해내는 Task.

Neural Machine Translation 시대의 시작.

HOW?

1. Word Embedding으로 인한 Continuous Representation의 힘.
2. 기존 SMT가 여러 모듈이 결합된 결과였다면 이제는 End-to-End 모델의 시대.
3. Attention으로 인해 길이가 긴 문장 또한 좋은 성능을 보이기 시작!

Neural Machine Translation

- 입력 문장(source text)을 번역한 출력 문장(target text)을 생성해내는 Task.

Neural Machine Translation 시대의 시작.

HOW?

1. Word Embedding으로 인한 Continuous Representation의 힘.
2. 기존 SMT가 여러 모듈이 결합된 결과였다면 이제는 End-to-End 모델의 시대.
3. Attention으로 인해 길이가 긴 문장 또한 좋은 성능을 보이기 시작!

Neural Machine Translation

- 입력 문장(source text)을 번역한 출력 문장(target text)을 생성해내는 Task.

Neural Machine Translation 시대의 시작.

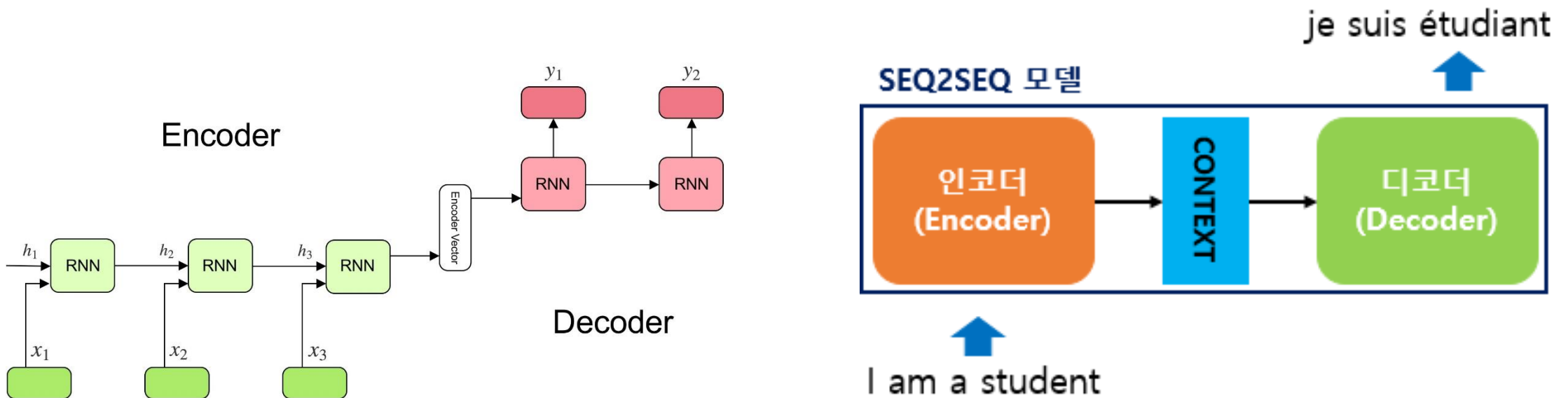
HOW?

1. Word Embedding으로 인한 Continuous Representation의 힘.
2. 기존 SMT가 여러 모듈이 결합된 결과였다면 이제는 End-to-End 모델의 시대.
3. Attention으로 인해 길이가 긴 문장 또한 좋은 성능을 보이기 시작!

왜 기존 seq2seq에 Attention이 필요했을까?

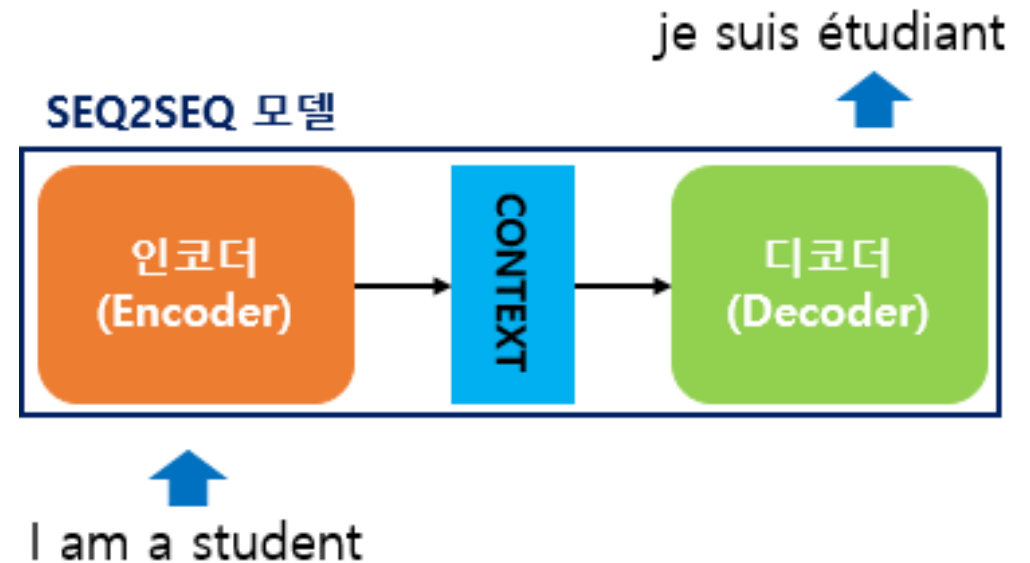
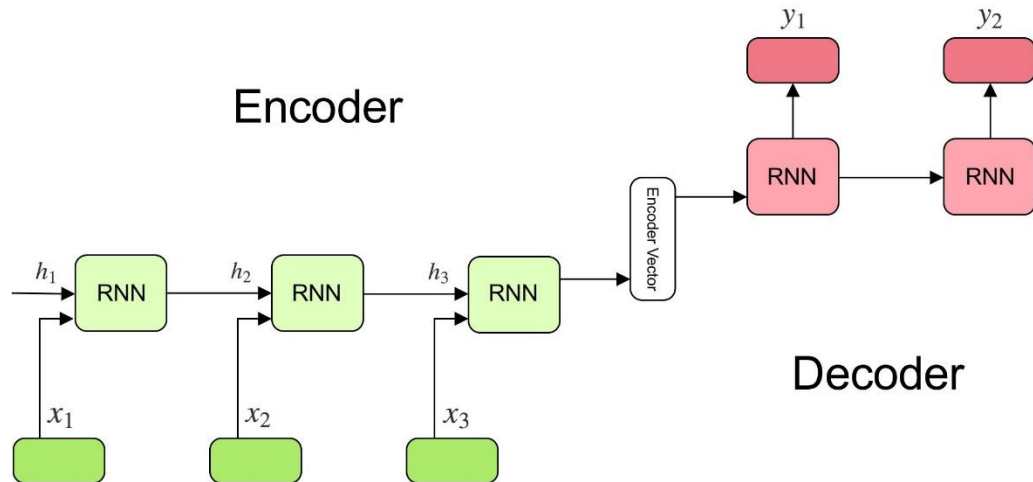
Sequence-To-Sequence, Seq2Seq

- Sequence-to-Sequence는 입력 문장의 정보를 하나의 벡터로 압축하여 디코더에게 전달한다.
- 이 벡터는 인코더의 마지막 시점의 은닉 상태이다.



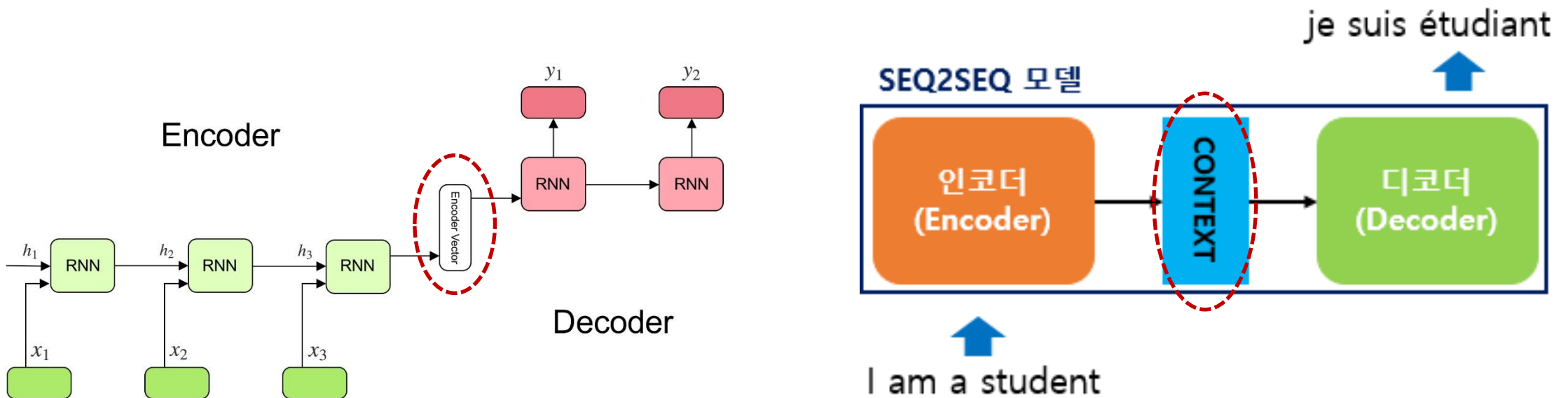
Sequence-To-Sequence, Seq2Seq

- Sequence-to-Sequence는 입력 문장의 정보를 하나의 벡터로 압축하여 디코더에게 전달한다.
- 이 벡터는 인코더의 마지막 시점의 은닉 상태이다.
- 인코더의 은닉 상태의 크기를 가지므로 **고정된 크기(fixed size vector)의 벡터** 이다.
- 이 벡터를 **컨텍스트 벡터(Context Vector)**라고 부른다.



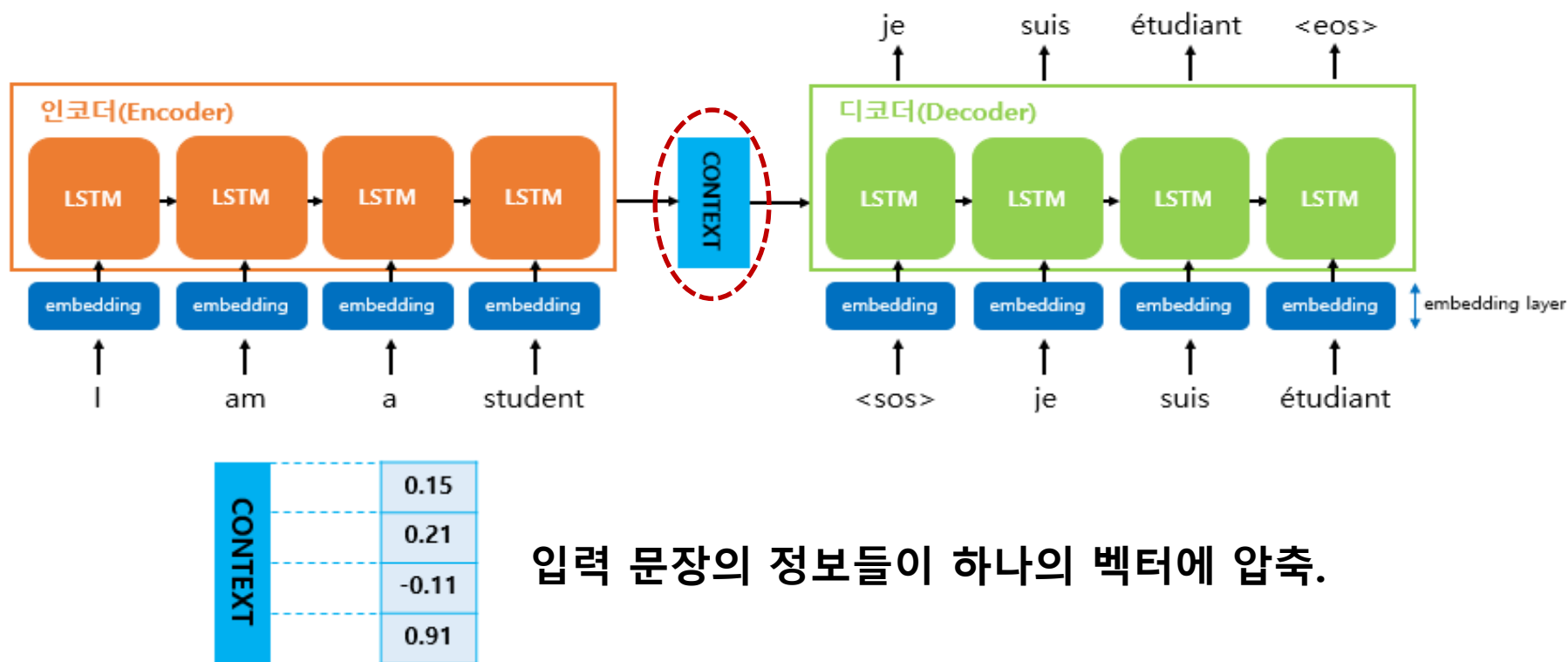
Sequence-To-Sequence, Seq2Seq

- Sequence-to-Sequence는 입력 문장의 정보를 하나의 벡터로 압축하여 디코더에게 전달한다.
- 이 벡터는 인코더의 마지막 시점의 은닉 상태이다.
- 인코더의 은닉 상태의 크기를 가지므로 **고정된 크기(fixed size vector)의 벡터** 이다.
- 이 벡터를 **컨텍스트 벡터(Context Vector)**라고 부른다.



Sequence-To-Sequence, Seq2Seq

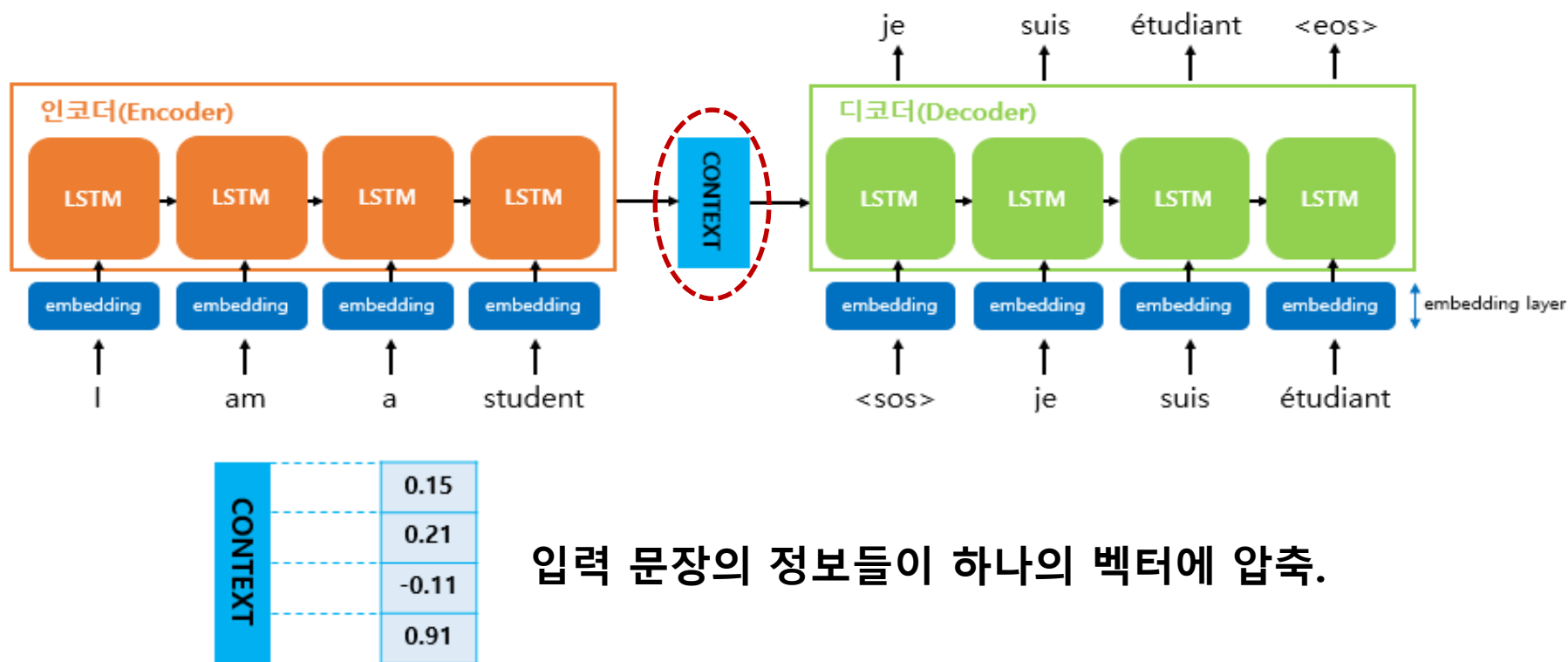
- 입력 문장의 길이와 상관없이 고정된 크기의 벡터에 정보를 모두 압축한다.



입력 문장의 정보들이 하나의 벡터에 압축.

Sequence-To-Sequence, Seq2Seq

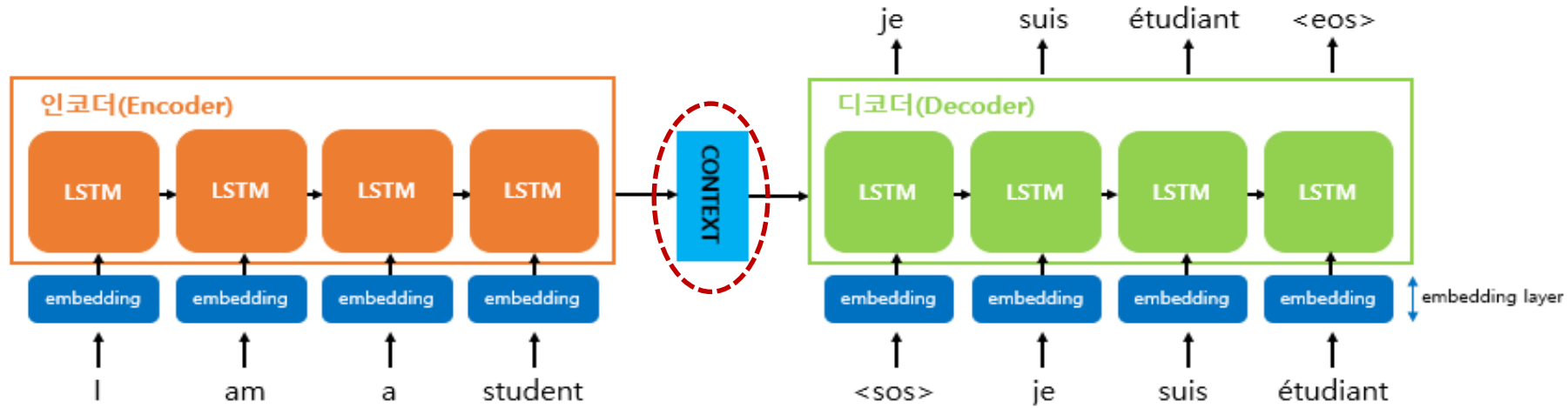
- 입력 문장의 길이와 상관없이 고정된 크기의 벡터에 정보를 모두 압축한다.
- 고정된 크기의 벡터에 정보가 다 압축되지 않아 정보 손실이 발생. (Bottleneck)



입력 문장의 정보들이 하나의 벡터에 압축.

Sequence-To-Sequence, Seq2Seq

- 입력 문장의 길이와 상관없이 고정된 크기의 벡터에 정보를 모두 압축한다.
- 고정된 크기의 벡터에 정보가 다 압축되지 않아 정보 손실이 발생. (Bottleneck)
- RNN 계열의 고질적인 장기 의존성 문제로 초기 정보가 손실되며 전달.

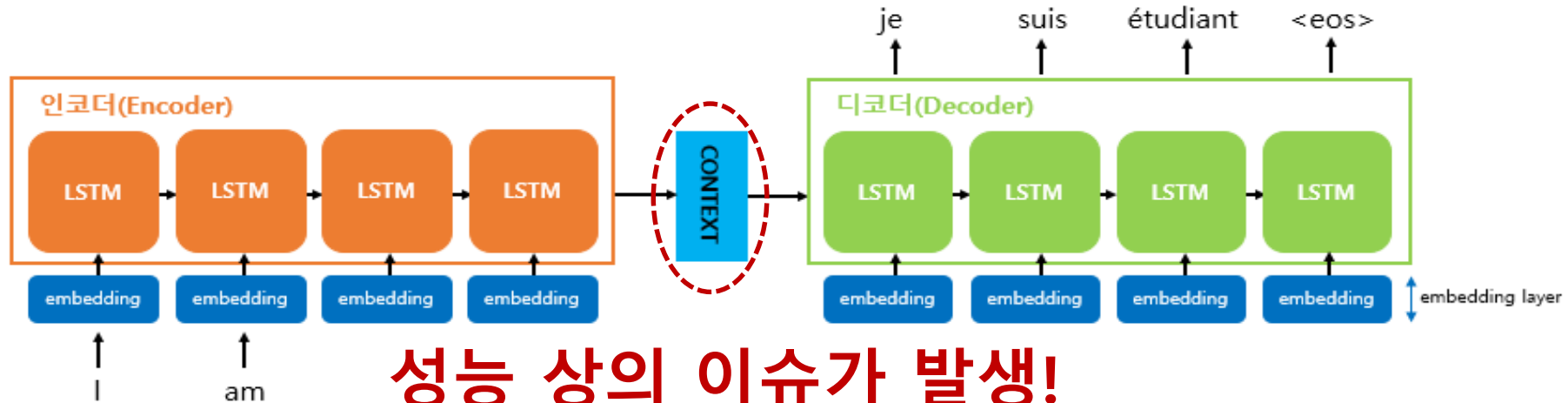


CONTEXT	0.15
	0.21
	-0.11
	0.91

입력 문장의 정보들이 하나의 벡터에 압축.

Sequence-To-Sequence, Seq2Seq

- 입력 문장의 길이와 상관없이 고정된 크기의 벡터에 정보를 모두 압축한다.
- 고정된 크기의 벡터에 정보가 다 압축되지 않아 정보 손실이 발생. (Bottleneck)
- RNN 계열의 고질적인 장기 의존성 문제로 초기 정보가 손실되며 전달.



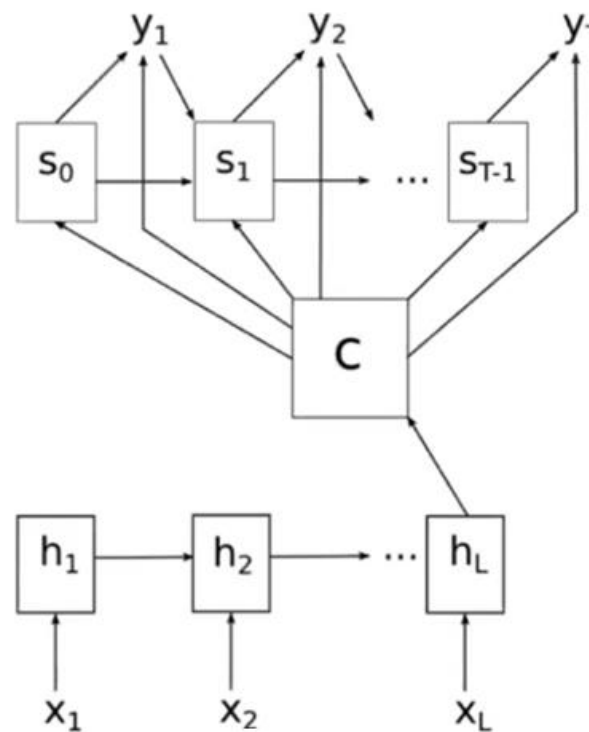
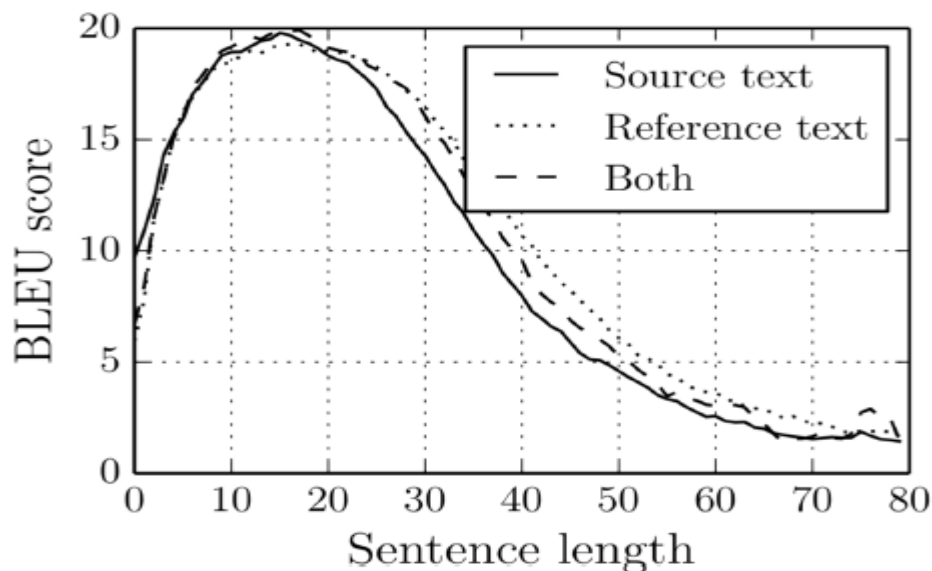
CONTEXT	0.15
	0.21
	-0.11
	0.91

입력 문장의 정보들이 하나의 벡터에 압축.

Sequence-To-Sequence, Seq2Seq

RNN Encoder-Decoder: Issues

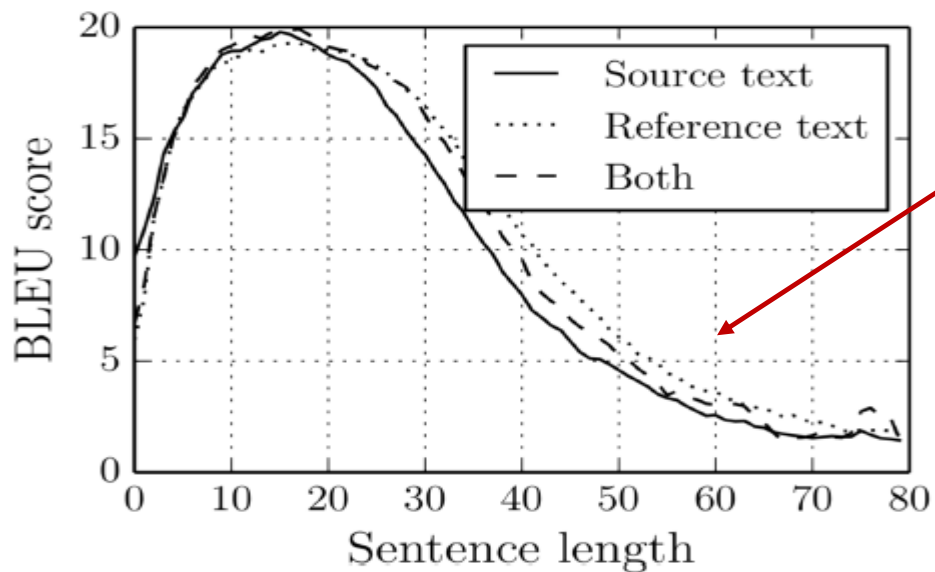
- has to remember the whole sentence Even with LSTM/GRU
- fixed size representation can be the bottleneck
- humans do it differently Need to 'attend' to each individual word



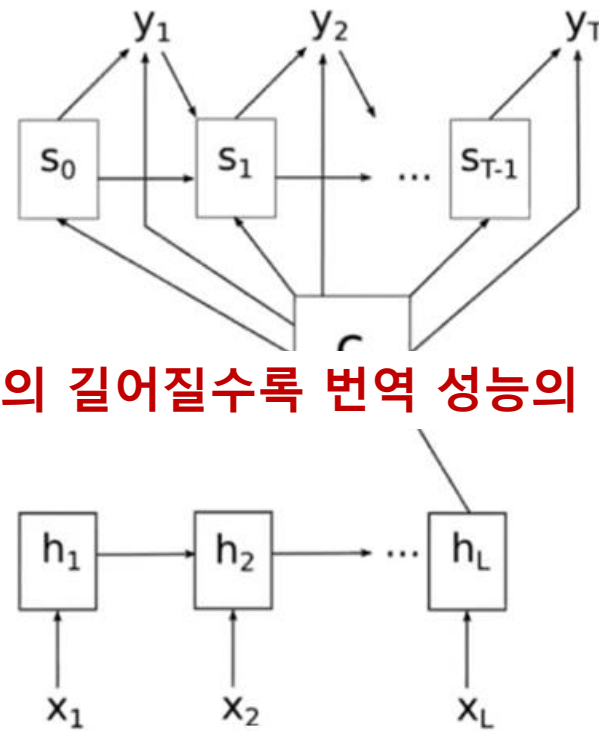
Sequence-To-Sequence, Seq2Seq

RNN Encoder-Decoder: Issues

- has to **장기 의존성 문제** \Rightarrow sentence Even with LSTM/GRU
- fixed size representation can be tl **보틀넥 발생**
- humans do it differently Need to 'attend' to each individual word



문장의 길어질수록 번역 성능의 급격한 저하



Sequence-To-Sequence, Seq2Seq

RNN Encoder-Decoder: Issues

밑줄이 번역이 제대로 되지 않은 부분

An admitting privilege is the right of a doctor to admit a patient to a hospital or a medical centre to carry out a diagnosis or a procedure, based on his status as a health care worker at a hospital.

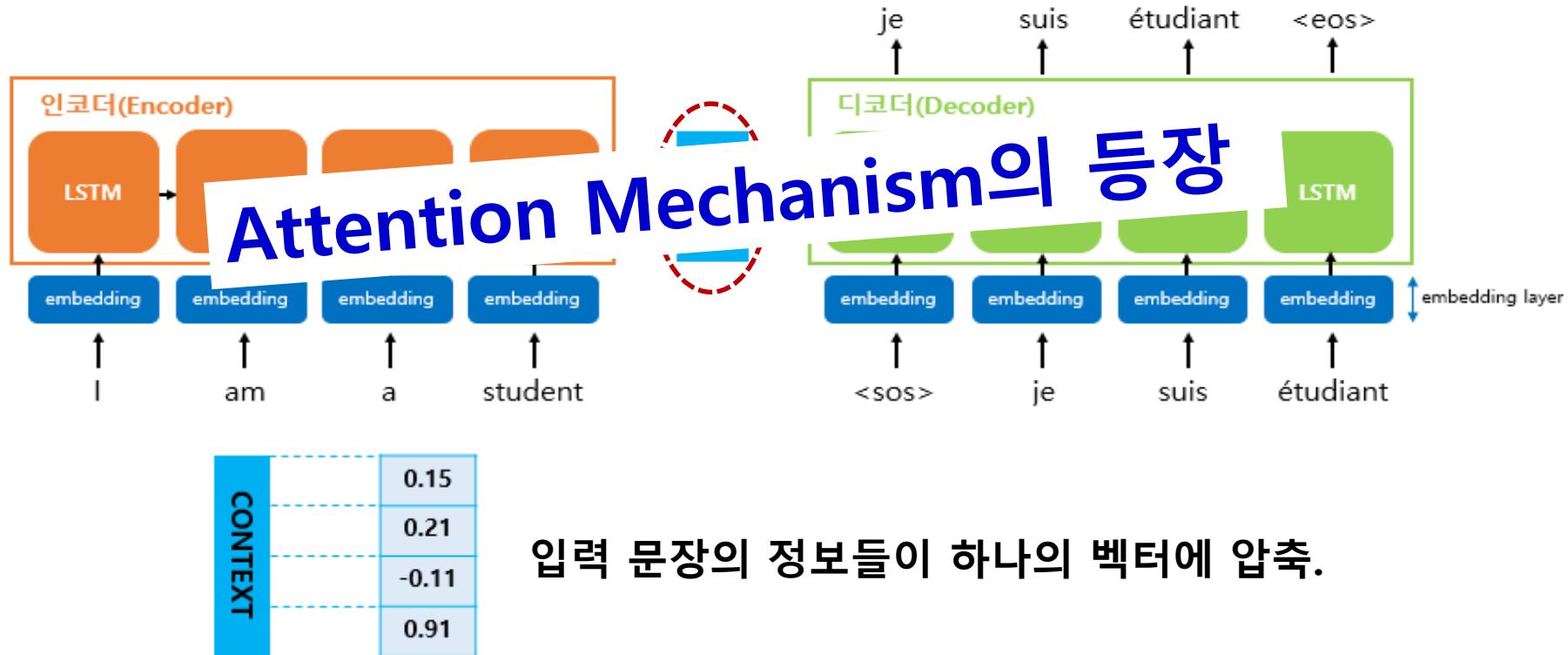
RNN Encoder-
Decoder

Un privilège d'admission est le droit d'un médecin de reconnaître un patient à l'hôpital ou un centre médical d'un diagnostic ou de prendre un diagnostic en fonction de son état de santé.

[based on his state of health???

Sequence-To-Sequence, Seq2Seq

- 입력 문장의 길이와 상관없이 고정된 크기의 벡터에 정보를 모두 압축한다.
- 고정된 크기의 벡터에 정보가 다 압축되지 않아 정보 손실이 발생. (Bottleneck)
- RNN 계열의 고질적인 장기 의존성 문제로 초기 정보가 손실되며 전달.



Seq2Seq + Attention

Python 자료형 : Dictionary

- 파이썬 자료형 딕셔너리는 Key와 Value를 한 쌍으로 갖는 자료형이다.

```
# 파이썬의 딕셔너리 자료형을 선언  
# 키(Key) : 값(value)의 형식으로 키와 값의 쌍(Pair)을 선언한다.  
dict = {"2017" : "Transformer", "2018" : "BERT"}
```

```
print(dict["2017"]) #2017이라는 키에 해당되는 값을 출력
```

Python 자료형 : Dictionary

- 파이썬 자료형 딕셔너리는 Key와 Value를 한 쌍으로 갖는 자료형이다.

```
# 파이썬의 딕셔너리 자료형을 선언  
# 키(Key) : 값(value)의 형식으로 키와 값의 쌍(Pair)을 선언한다.  
dict = {"2017" : "Transformer", "2018" : "BERT"}
```

```
print(dict["2017"]) #2017이라는 키에 해당되는 값을 출력
```



Python 자료형 : Dictionary

- 파이썬 자료형 딕셔너리는 Key와 Value를 한 쌍으로 갖는 자료형이다.

```
# 파이썬의 딕셔너리 자료형을 선언  
# 키(Key) : 값(value)의 형식으로 키와 값의 쌍(Pair)을 선언한다.  
dict = {"2017" : "Transformer", "2018" : "BERT"}
```

```
print(dict["2017"]) #2017이라는 키에 해당되는 값을 출력
```



Transformer

Python 자료형 : Dictionary

- 파이썬 자료형 딕셔너리는 Key와 Value를 한 쌍으로 갖는 자료형이다.

```
# 파이썬의 딕셔너리 자료형을 선언  
# 키(Key) : 값(value)의 형식으로 키와 값의 쌍(Pair)을 선언한다.  
dict = {"2017" : "Transformer", "2018" : "BERT"}
```

```
print(dict["2017"]) #2017이라는 키에 해당되는 값을 출력
```



Transformer

```
print(dict["2018"]) #2018이라는 키에 해당되는 값을 출력
```

Python 자료형 : Dictionary

- 파이썬 자료형 딕셔너리는 Key와 Value를 한 쌍으로 갖는 자료형이다.

```
# 파이썬의 딕셔너리 자료형을 선언  
# 키(Key) : 값(value)의 형식으로 키와 값의 쌍(Pair)을 선언한다.  
dict = {"2017" : "Transformer", "2018" : "BERT"}
```

```
print(dict["2017"]) #2017이라는 키에 해당되는 값을 출력
```



Transformer

```
print(dict["2018"]) #2018이라는 키에 해당되는 값을 출력
```



Python 자료형 : Dictionary

- 파이썬 자료형 딕셔너리는 Key와 Value를 한 쌍으로 갖는 자료형이다.

```
# 파이썬의 딕셔너리 자료형을 선언  
# 키(Key) : 값(value)의 형식으로 키와 값의 쌍(Pair)을 선언한다.  
dict = {"2017" : "Transformer", "2018" : "BERT"}
```

```
print(dict["2017"]) #2017이라는 키에 해당되는 값을 출력
```



Transformer

```
print(dict["2018"]) #2018이라는 키에 해당되는 값을 출력
```



BERT

Python 자료형 : Dictionary

- 파이썬 자료형 딕셔너리는 Key와 Value를 한 쌍으로 갖는 자료형이다.

```
# 파이썬의 딕셔너리 자료형을 선언  
# 키(Key) : 값(value)의 형식으로 키와 값의 쌍(Pair)을 선언한다.  
dict = {"2017" : "Transformer", "2018" : "BERT"}
```

```
print(dict["2017"]) #2017이라는 키에 해당되는 값을 출력
```



Transformer

```
print(dict["2018"]) #2018이라는 키에 해당되는 값을 출력
```



BERT

각 Key는 각 Key와 맵핑되는 Value를 가진다.

Python 자료형 : Dictionary

- 파이썬 자료형 딕셔너리는 Key와 Value를 한 쌍으로 갖는 자료형이다.

```
# 파이썬의 딕셔너리 자료형을 선언  
# 키(Key) : 값(value)의 형식으로 키와 값의 쌍(Pair)을 선언한다.  
dict = {"2017" : "Transformer", "2018" : "BERT"}
```

```
print(dict["2017"]) #2017이라는 키에 해당되는 값을 출력
```



Transformer

```
print(dict["2018"]) #2018이라는 키에 해당되는 값을 출력
```

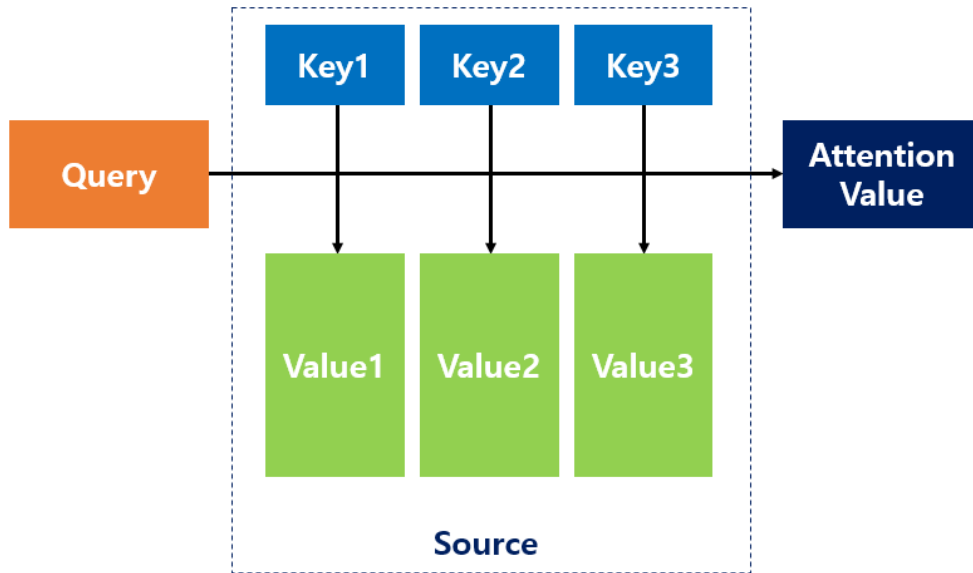


BERT

각 Key는 각 Key와 맵핑되는 Value를 가진다.
이 구조를 기억해둍시다.

Attention function(Q, K, V)

Attention(Q, K, V) = Attention Value

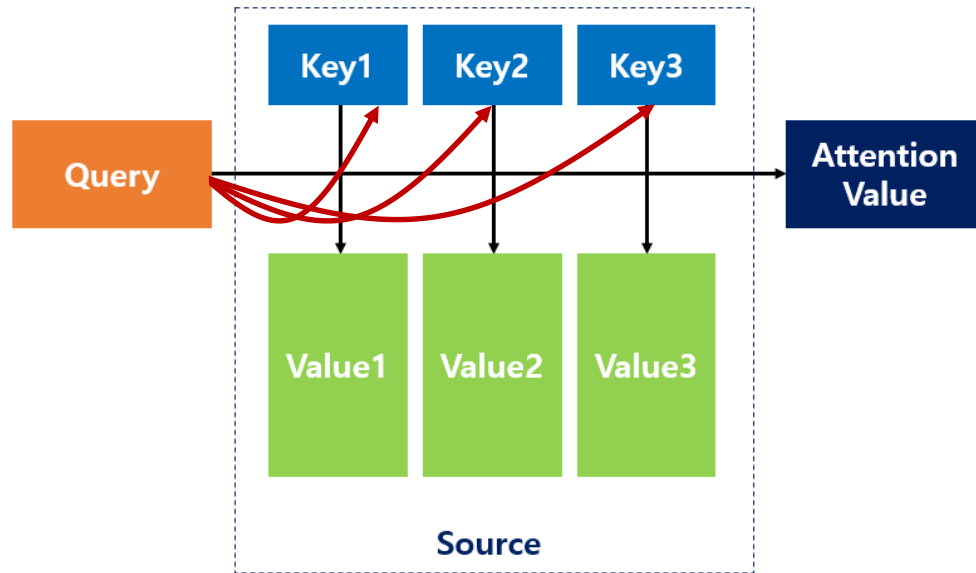


1. 어텐션 함수는 주어진 '쿼리(Query)'에 대해서 모든 '키(Key)'와의 유사도를 각각 구한다.
2. 구해낸 이 유사도를 키와 맵핑되어있는 각각의 '값(Value)'에 반영해준다.
3. 유사도가 반영된 '값(Value)'들을 모두 더해서 리턴한다.
4. 이를 어텐션 값(Attention Value)이라고 한다.

- 참고) Query, Key, Value에서 Key와 Value는 일반적으로 동일한 경우가 많음.

Attention function(Q, K, V)

Attention(Q, K, V) = Attention Value

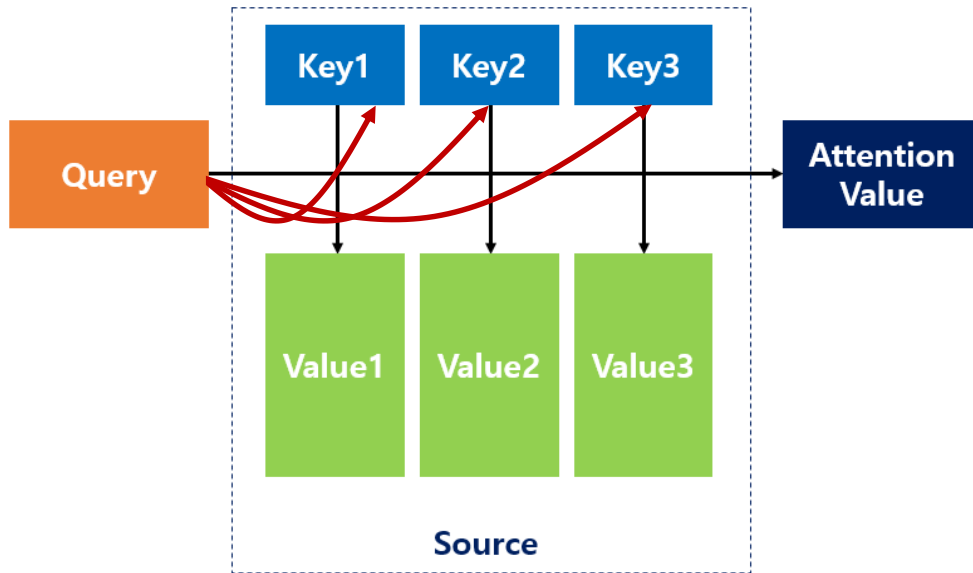


1. 어텐션 함수는 주어진 '쿼리(Query)'에 대해서 모든 '키(Key)'와의 **유사도**를 각각 구한다.
2. 구해낸 이 유사도를 키와 맵핑되어있는 각각의 '값(Value)'에 반영해준다.
3. 유사도가 반영된 '값(Value)'들을 모두 더해서 리턴한다.
4. 이를 어텐션 값(Attention Value)이라고 한다.

- 참고) Query, Key, Value에서 Key와 Value는 일반적으로 동일한 경우가 많음.

Attention function(Q, K, V)

Attention(Q, K, V) = Attention Value



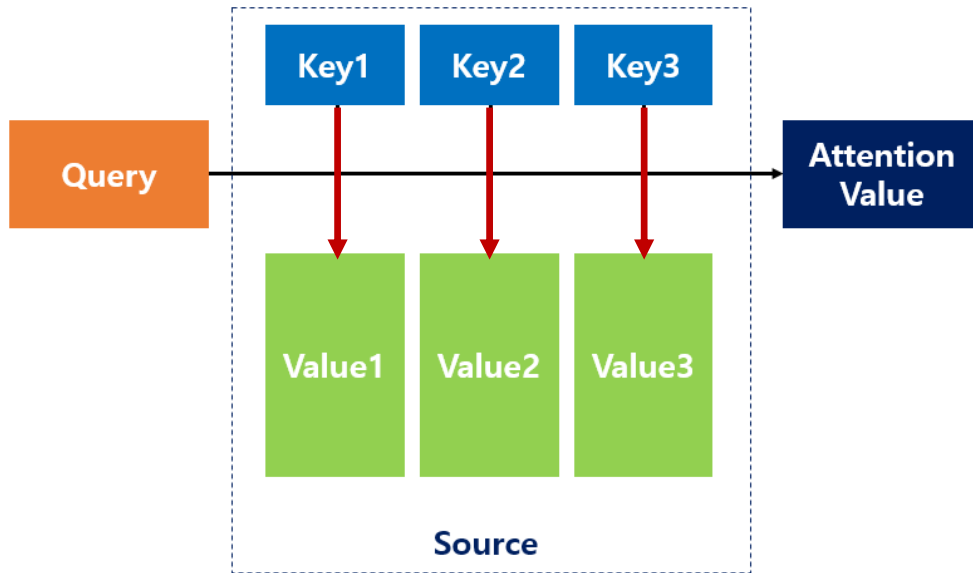
구하는 방법은 여러가지가 있다.
이 연산을 수행하는 함수를
어텐션 스코어 함수라고 부른다.

1. 어텐션 함수는 주어진 '쿼리(Query)'에 대해서 모든 '키(Key)'와의 유사도를 각각 구한다.
2. 구해낸 이 유사도를 키와 맵핑되어있는 각각의 '값(Value)'에 반영해준다.
3. 유사도가 반영된 '값(Value)'들을 모두 더해서 리턴한다.
4. 이를 어텐션 값(Attention Value)이라고 한다.

- 참고) Query, Key, Value에서 Key와 Value는 일반적으로 동일한 경우가 많음.

Attention function(Q, K, V)

Attention(Q, K, V) = Attention Value

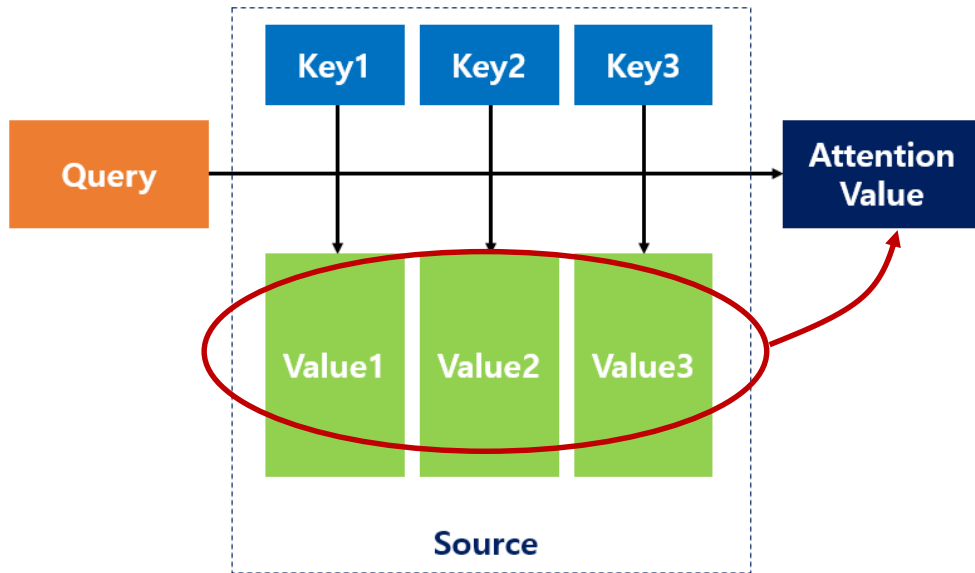


1. 어텐션 함수는 주어진 '쿼리(Query)'에 대해서 모든 '키(Key)'와의 유사도를 각각 구한다.
2. 구해낸 이 유사도를 키와 맵핑되어있는 각각의 '값(Value)'에 반영해준다.
3. 유사도가 반영된 '값(Value)'들을 모두 더해서 리턴한다.
4. 이를 어텐션 값(Attention Value)이라고 한다.

- 참고) Query, Key, Value에서 Key와 Value는 일반적으로 동일한 경우가 많음.

Attention function(Q, K, V)

Attention(Q, K, V) = Attention Value

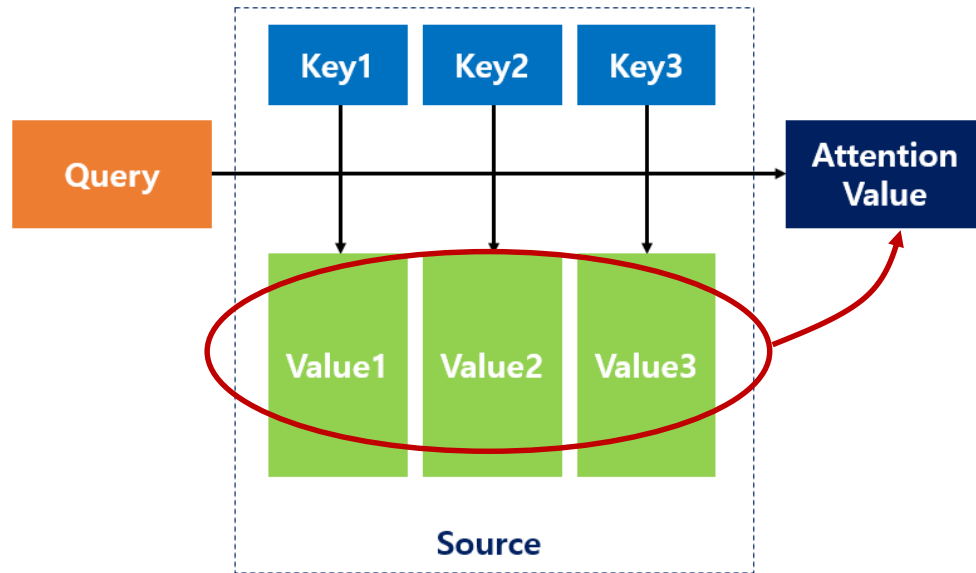


1. 어텐션 함수는 주어진 '쿼리(Query)'에 대해서 모든 '키(Key)'와의 **유사도**를 각각 구한다.
2. 구해낸 이 유사도를 키와 맵핑되어있는 각각의 '값(Value)'에 반영해준다.
3. 유사도가 반영된 '**값(Value)**'들을 모두 **더해서** 리턴한다.
4. 이를 어텐션 값(Attention Value)이라고 한다.

- 참고) Query, Key, Value에서 Key와 Value는 일반적으로 동일한 경우가 많음.

Attention function(Q, K, V)

Attention(Q, K, V) = Attention Value

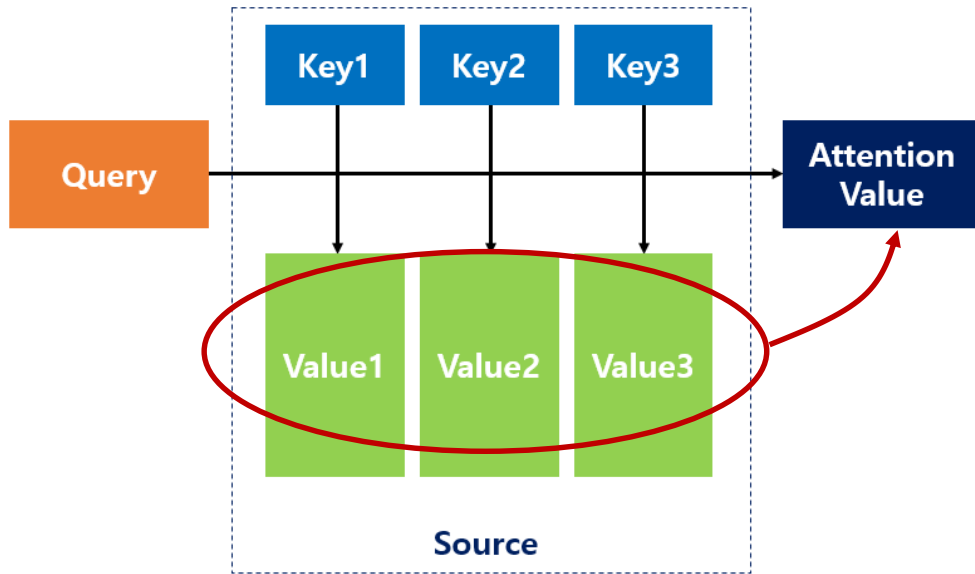


1. 어텐션 함수는 주어진 '쿼리(Query)'에 대해서 모든 '키(Key)'와의 **유사도**를 각각 구한다.
2. 구해낸 이 유사도를 키와 맵핑되어있는 각각의 '값(Value)'에 반영해준다.
3. 유사도가 반영된 '값(Value)'들을 모두 더해서 리턴한다.
4. 이를 **어텐션 값(Attention Value)**이라고 한다.

- 참고) Query, Key, Value에서 Key와 Value는 일반적으로 동일한 경우가 많음.

Attention function(Q, K, V)

Attention(Q, K, V) = Attention Value

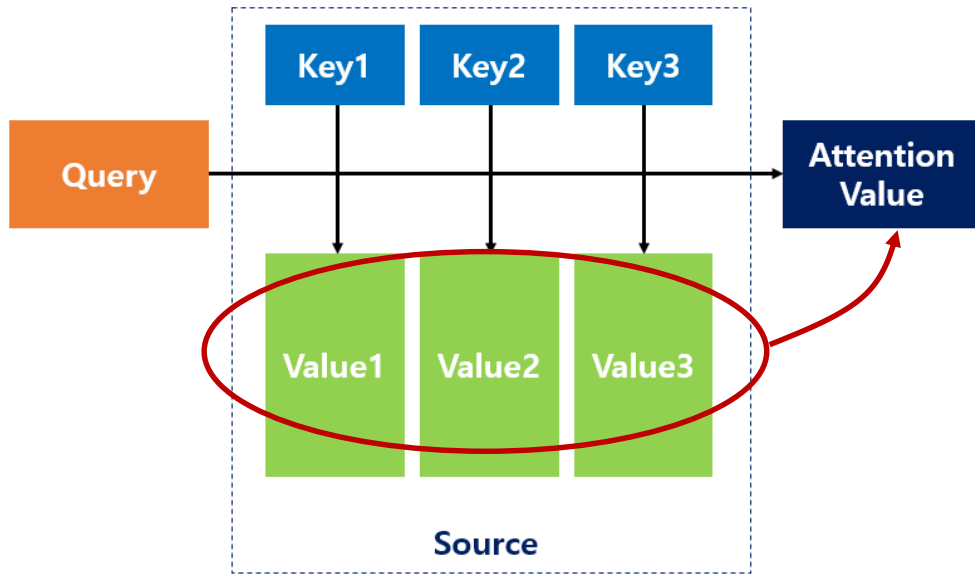


1. 어텐션 함수는 주어진 '쿼리(Query)'에 대해서 모든 '키(Key)'와의 **유사도**를 각각 구한다.
2. 구해낸 이 유사도를 키와 맵핑되어있는 각각의 '값(Value)'에 반영해준다.
3. 유사도가 반영된 '**값(Value)**'들을 **모두 더해서** 리턴한다.
4. 이를 어텐션 값(Attention Value)이라고 한다.

- 참고) Query, Key, Value에서 **Key와 Value는 일반적으로 동일한 경우가 많음.**

Attention function(Q, K, V)

Attention(Q, K, V) = Attention Value



1. 어텐션 함수는 주어진 '쿼리(Query)'에 대해서 모든 '키(Key)'와의 **유사도**를 각각 구한다.
2. 구해낸 이 유사도를 키와 맵핑되어있는 각각의 '값(Value)'에 반영해준다.
3. 유사도가 반영된 '**값(Value)**'들을 **모두 더해서** 리턴한다.
4. 이를 어텐션 값(Attention Value)이라고 한다.

- 참고) Query, Key, Value에서 **Key와 Value는 일반적으로 동일한 경우가 많음.**

가장 먼저 알아볼 어텐션은 닷 프로덕트 어텐션.

어텐션 메커니즘의 종류

어텐션 메커니즘은 다음 순서로 제안되었다. 이들의 큰 차이는 어텐션 스코어 함수가 다르다는 점이다.

1. 바다나우 어텐션 (Neural Machine Translation by Jointly Learning to Align and Translate)

- 콘캣 어텐션(Concat Attention) : $score(s_t, h_i) = W_a^T \tanh(W_b[s_t; h_i])$

2. 루옹 어텐션 (Effective Approaches to Attention-based Neural Machine Translation)

- 닷 프로덕트 어텐션(Dot-Product Attention) : $score(s_t, h_i) = s_t^T h_i$
- 제네럴 어텐션(General Attention) : $score(s_t, h_i) = s_t^T W_a h_i$

3. 스케일드 닷 프로덕트 어텐션(Attention is All you need)

- 스케일드 닷 프로덕트 어텐션(Scaled Dot-Product Attention) : $score(s_t, h_i) = \frac{s_t^T h_i}{\sqrt{n}}$

어텐션 메커니즘의 종류

어텐션 메커니즘은 다음 순서로 제안되었다. 이들의 큰 차이는 어텐션 스코어 함수가 다르다는 점이다.

처음에는 어텐션이라고 안 하고 Align이라고 불렀음.

1. 바다나우 어텐션 (Neural Machine Translation by Jointly Learning to Align and Translate)

- 콘캣 어텐션(Concat Attention) : $score(s_t, h_i) = W_a^T \tanh(W_b[s_t; h_i])$

2. 루옹 어텐션 (Effective Approaches to Attention-based Neural Machine Translation)

- 닷 프로덕트 어텐션(Dot-Product Attention) : $score(s_t, h_i) = s_t^T h_i$
- 제네럴 어텐션(General Attention) : $score(s_t, h_i) = s_t^T W_a h_i$

3. 스케일드 닷 프로덕트 어텐션(Attention is All you need)

- 스케일드 닷 프로덕트 어텐션(Scaled Dot-Product Attention) : $score(s_t, h_i) = \frac{s_t^T h_i}{\sqrt{n}}$

어텐션 메커니즘의 종류

어텐션 메커니즘은 다음 순서로 제안되었다. 이들의 큰 차이는 어텐션 스코어 함수가 다르다는 점이다.

1. 바다나우 어텐션 (Neural Machine Translation by Jointly Learning to Align and Translate)

- 콘캣 어텐션(Concat Attention) : $score(s_t, h_i) = W_a^T \tanh(W_b[s_t; h_i])$

2. 루옹 어텐션 (Effective Approaches to Attention-based Neural Machine Translation)

- **닷 프로덕트 어텐션(Dot-Product Attention)** : $score(s_t, h_i) = s_t^T h_i$ **가장 쉬우니까
닷 프로덕트 어텐션을 먼저 설명.**
- 제네럴 어텐션(General Attention) : $score(s_t, h_i) = s_t^T W_a h_i$

3. 스케일드 닷 프로덕트 어텐션(Attention is All you need)

- 스케일드 닷 프로덕트 어텐션(Scaled Dot-Product Attention) : $score(s_t, h_i) = \frac{s_t^T h_i}{\sqrt{n}}$

어텐션 메커니즘의 종류

어텐션 메커니즘은 다음 순서로 제안되었다. 이들의 큰 차이는 어텐션 스코어 함수가 다르다는 점이다.

1. 바다나우 어텐션 (Neural Machine Translation by Jointly Learning to Align and Translate)

- 콘캣 어텐션(Concat Attention) : $score(s_t, h_i) = W_a^T \tanh(W_b[s_t; h_i])$

2. 루옹 어텐션 (Effective Approaches to Attention-based Neural Machine Translation)

- **닷 프로덕트 어텐션(Dot-Product Attention)** : $score(s_t, h_i) = s_t^T h_i$ **가장 쉬우니까
닷 프로덕트 어텐션을 먼저 설명.**
- 제네럴 어텐션(General Attention) : $score(s_t, h_i) = s_t^T W_a h_i$

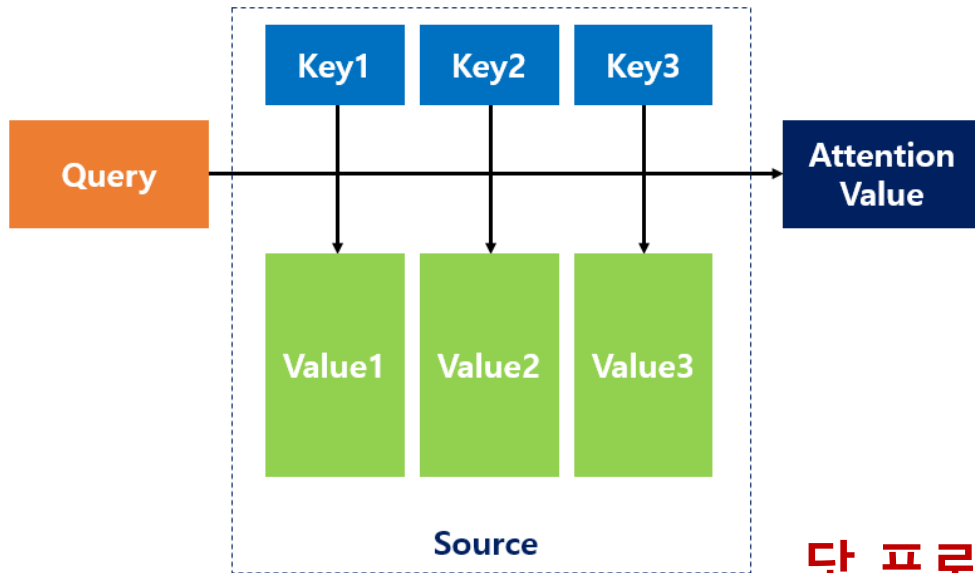
3. 스케일드 닷 프로덕트 어텐션(Attention is All you need)

- 스케일드 닷 프로덕트 어텐션(Scaled Dot-Product Attention) : $score(s_t, h_i) = \frac{s_t^T h_i}{\sqrt{n}}$

트랜스포머 논문에서 등장

Dot product Attention

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



유사도를 내적(dot product)으로 계산

1. 어텐션 함수는 주어진 '쿼리(Query)'에 대해서 모든 '키(Key)'와의 유사도를 각각 구한다.
2. 구해낸 이 유사도를 키와 맵핑되어있는 각각의 '값(Value)'에 반영해준다.
3. 유사도가 반영된 '**값(Value)**'들을 모두 더해서 리턴한다.
4. 이를 어텐션 값(Attention Value)이라고 한다.

닷 프로덕트 어텐션에서의 Q, K, V

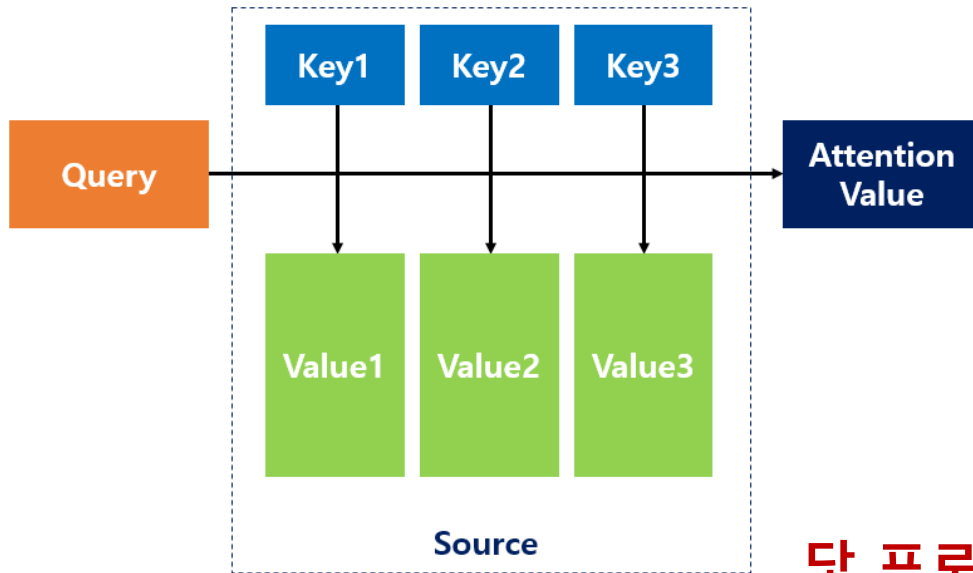
Q = Query : t 시점의 디코더 셀에서의 은닉 상태

K = Keys : 모든 시점의 인코더 셀의 은닉 상태들

V = Values : 모든 시점의 인코더 셀의 은닉 상태들

Dot product Attention

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



유사도를 내적(dot product)으로 계산

1. 어텐션 함수는 주어진 '쿼리(Query)'에 대해서 모든 '키(Key)'와의 유사도를 각각 구한다.
2. 구해낸 이 유사도를 키와 맵핑되어있는 각각의 '값(Value)'에 반영해준다.
3. 유사도가 반영된 '**값(Value)**'들을 모두 더해서 리턴한다.
4. 이를 어텐션 값(Attention Value)이라고 한다.

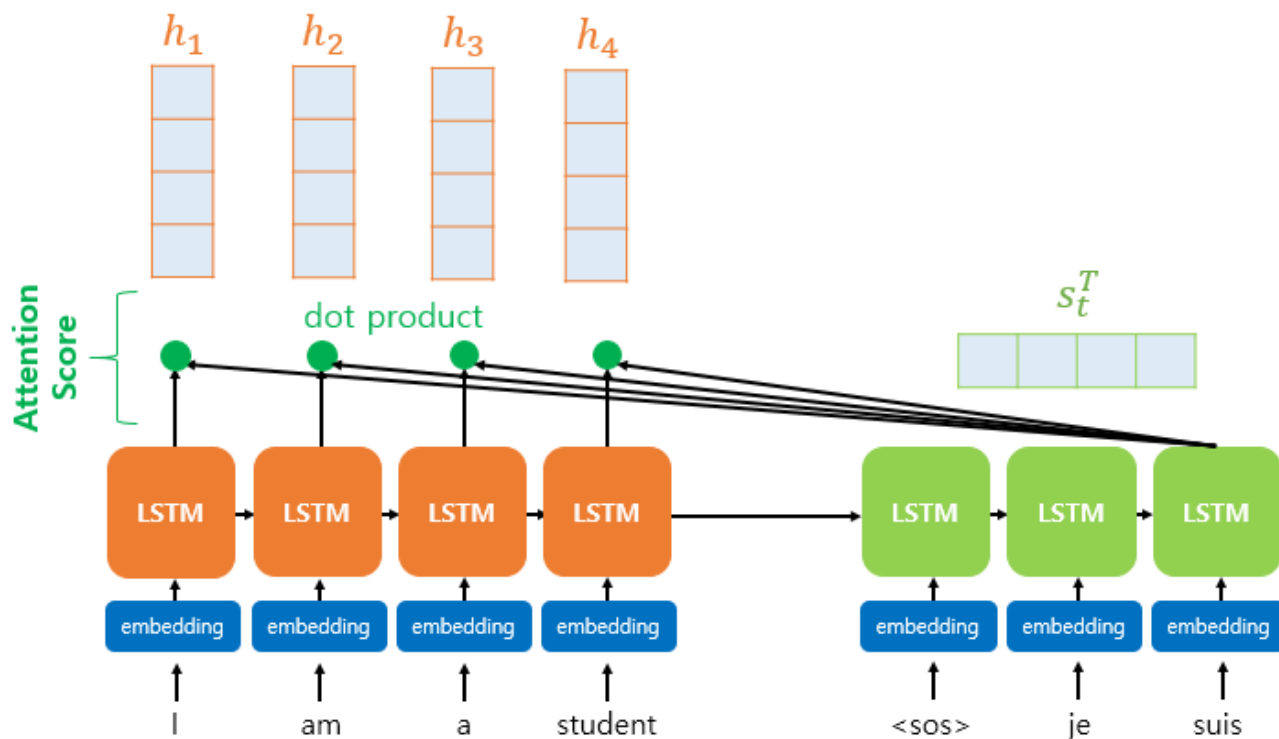
닷 프로덕트 어텐션에서의 Q, K, V

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

Key와 Value는
동일한 경우가 많음!

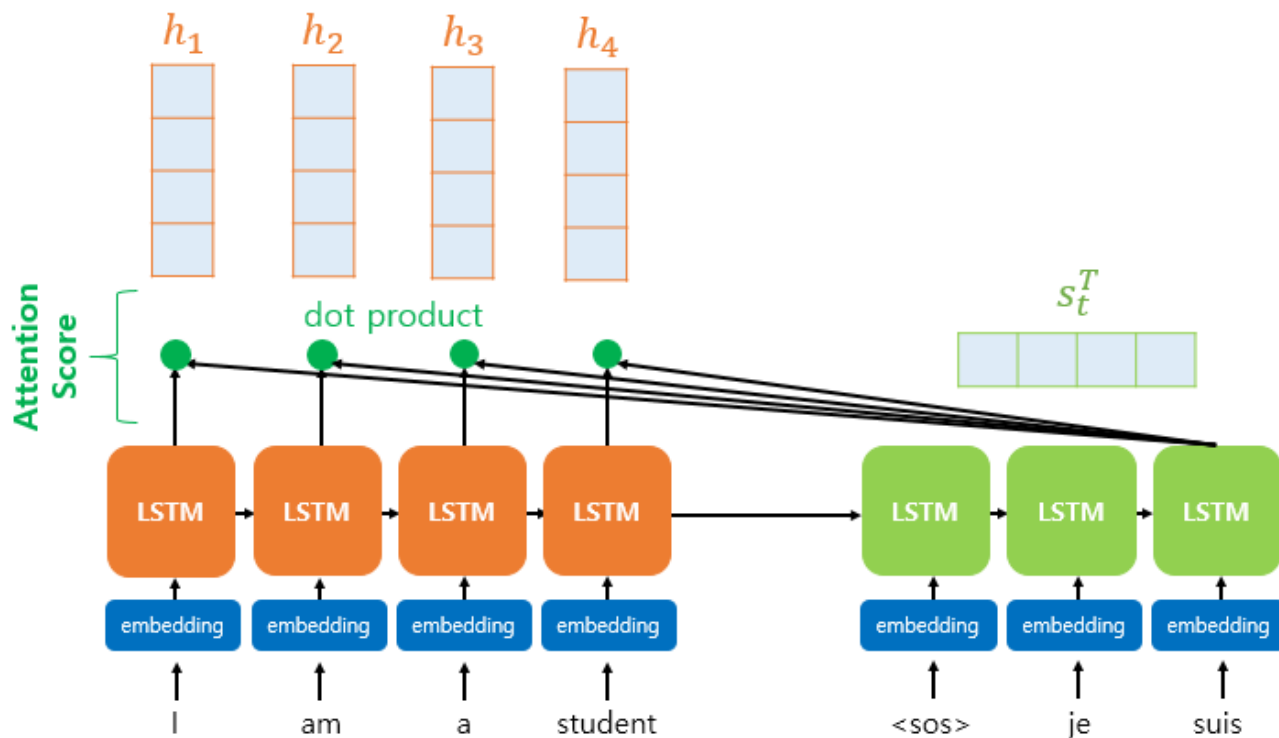
Dot product Attention

- 현재 상황은 t 번째 시점에서 단어를 예측하고자 하는 상태.
- 디코더의 t 번째 은닉 상태를 인코더의 모든 시점의 은닉 상태와 내적(Dot product)을 한다.
- 디코더의 t 번째 은닉 상태를 인코더의 모든 시점의 은닉 상태와 유사도를 구하는 것이다.



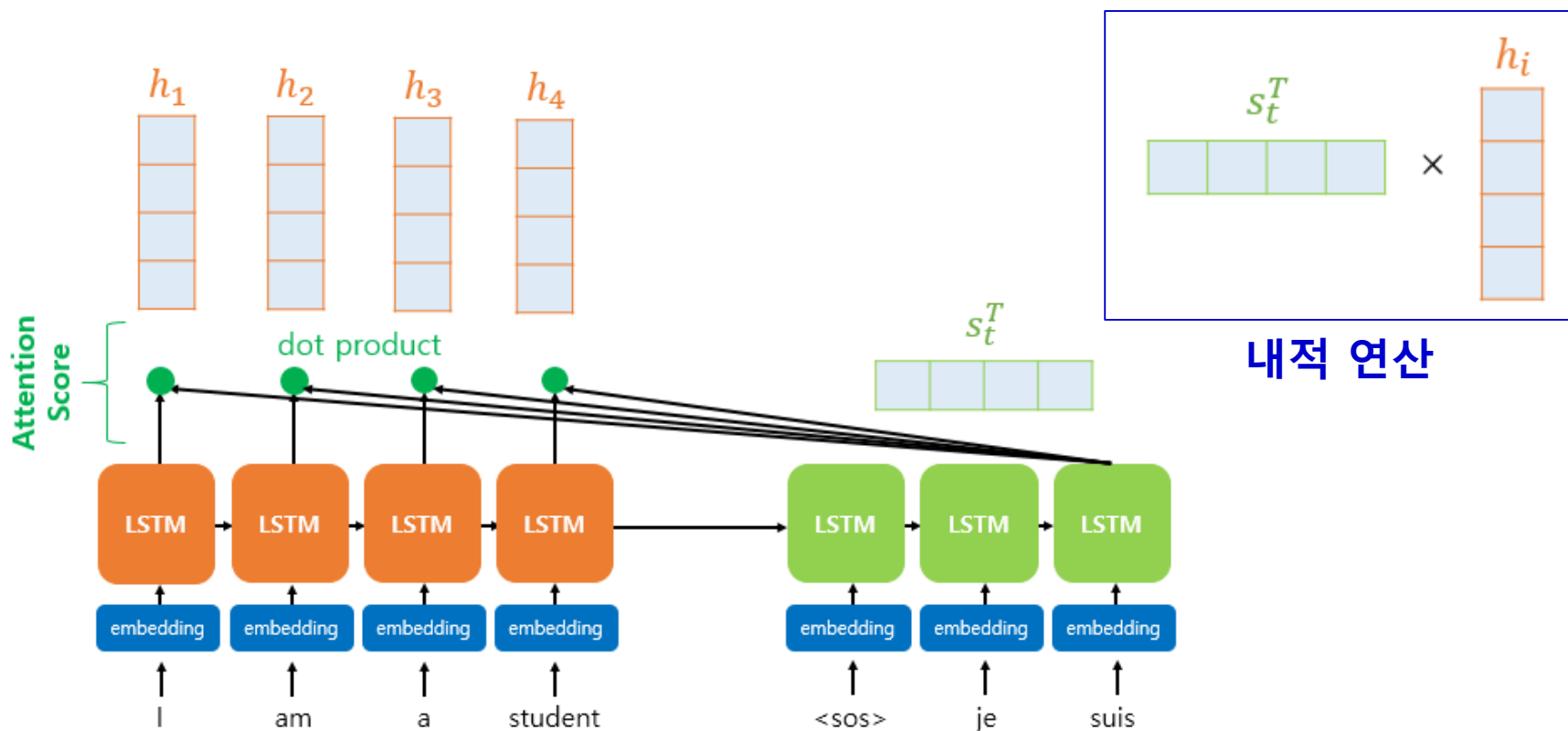
Dot product Attention

- 현재 상황은 **t번째 시점에서 단어를 예측하고자 하는 상태**.
- 디코더의 t번째 은닉 상태를 인코더의 모든 시점의 은닉 상태와 내적(Dot product)을 한다.
- 디코더의 t번째 은닉 상태를 인코더의 모든 시점의 은닉 상태와 유사도를 구하는 것이다.



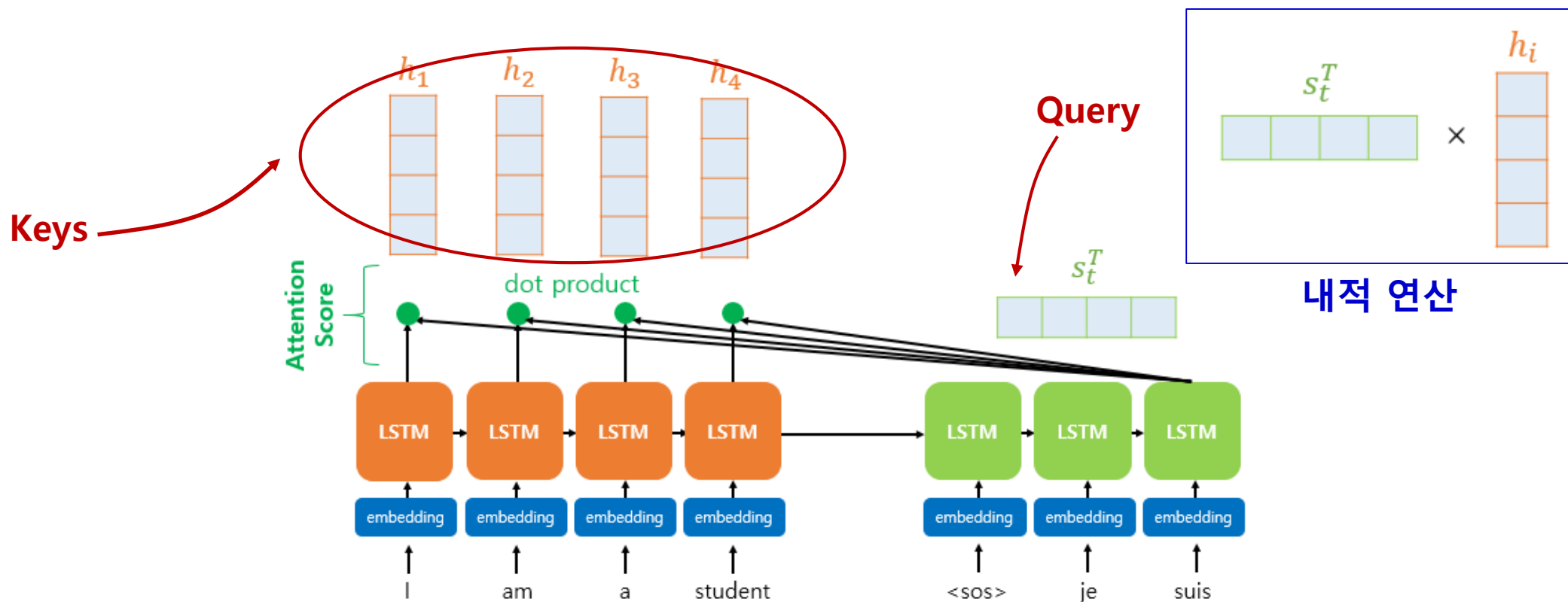
Dot product Attention

- 현재 상황은 t 번째 시점에서 단어를 예측하고자 하는 상태.
- 디코더의 t 번째 은닉 상태를 인코더의 모든 시점의 은닉 상태와 내적(Dot product)을 한다.
- 디코더의 t 번째 은닉 상태를 인코더의 모든 시점의 은닉 상태와 유사도를 구하는 것이다.



Dot product Attention

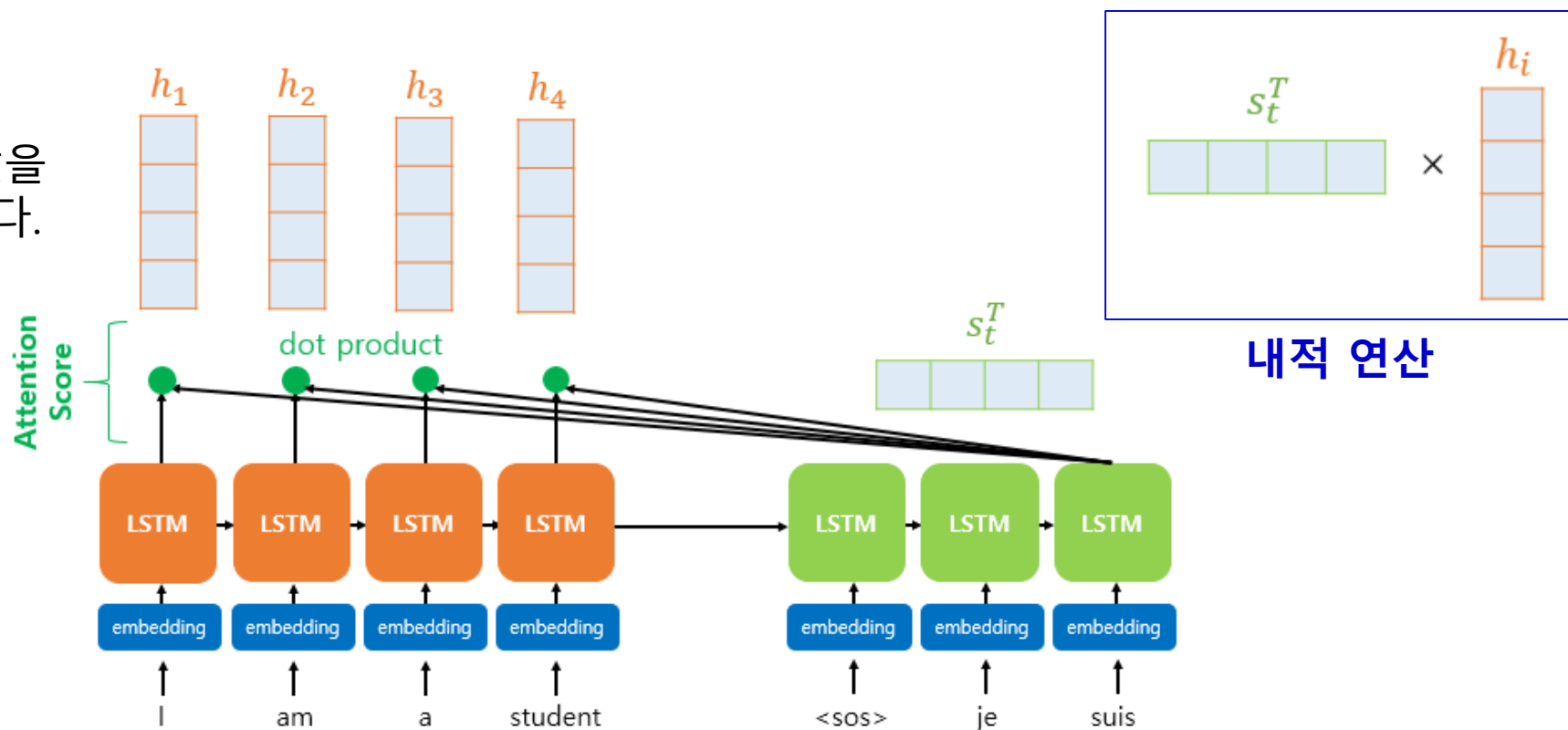
- 현재 상황은 t 번째 시점에서 단어를 예측하고자 하는 상태.
- 디코더의 t 번째 은닉 상태를 인코더의 모든 시점의 은닉 상태와 내적(Dot product)을 한다.
- 디코더의 t 번째 은닉 상태를 인코더의 모든 시점의 은닉 상태와 유사도를 구하는 것이다.



Dot product Attention

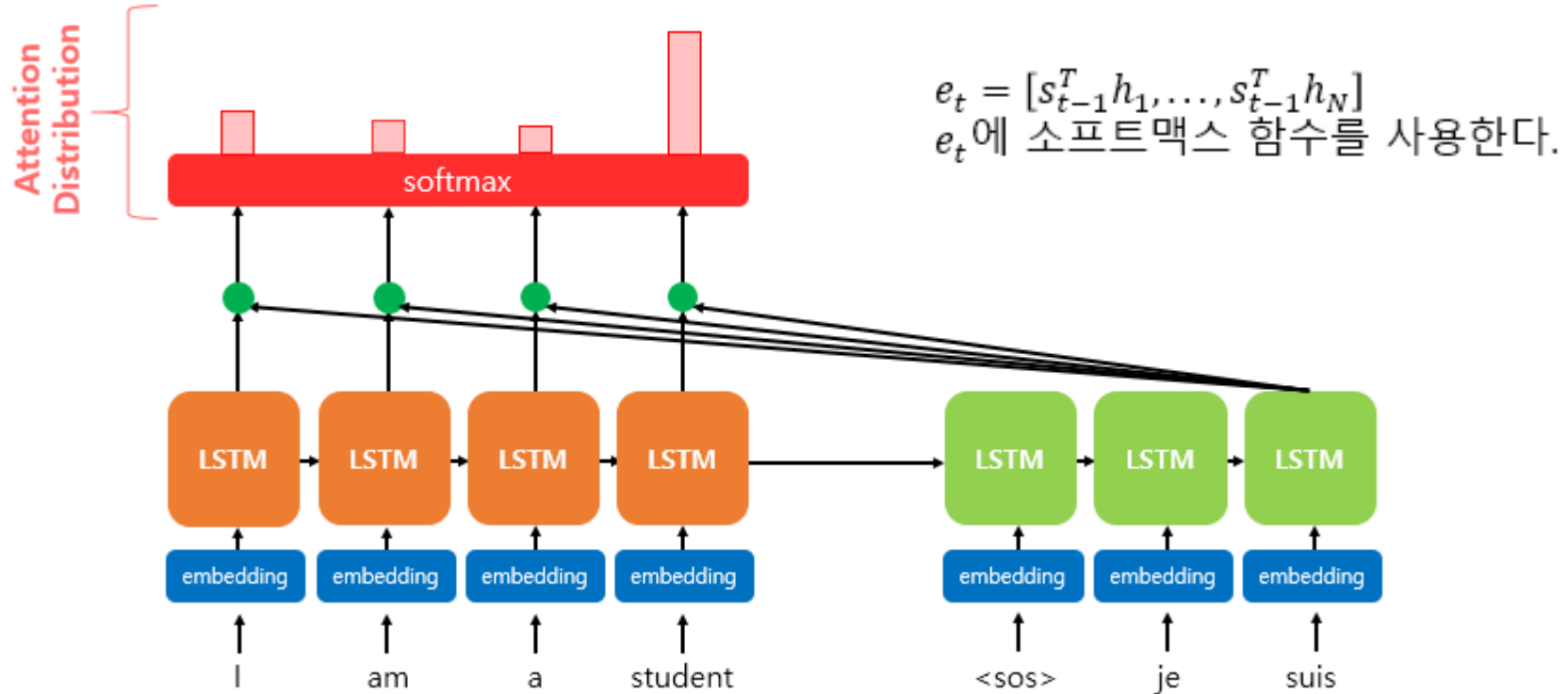
- 현재 상황은 t 번째 시점에서 단어를 예측하고자 하는 상태.
- 디코더의 t 번째 은닉 상태를 인코더의 모든 시점의 은닉 상태와 내적(Dot product)을 한다.
- 디코더의 t 번째 은닉 상태를 인코더의 모든 시점의 은닉 상태와 유사도를 구하는 것이다.

이렇게 구한 유사도 값을
어텐션 스코어라고 한다.



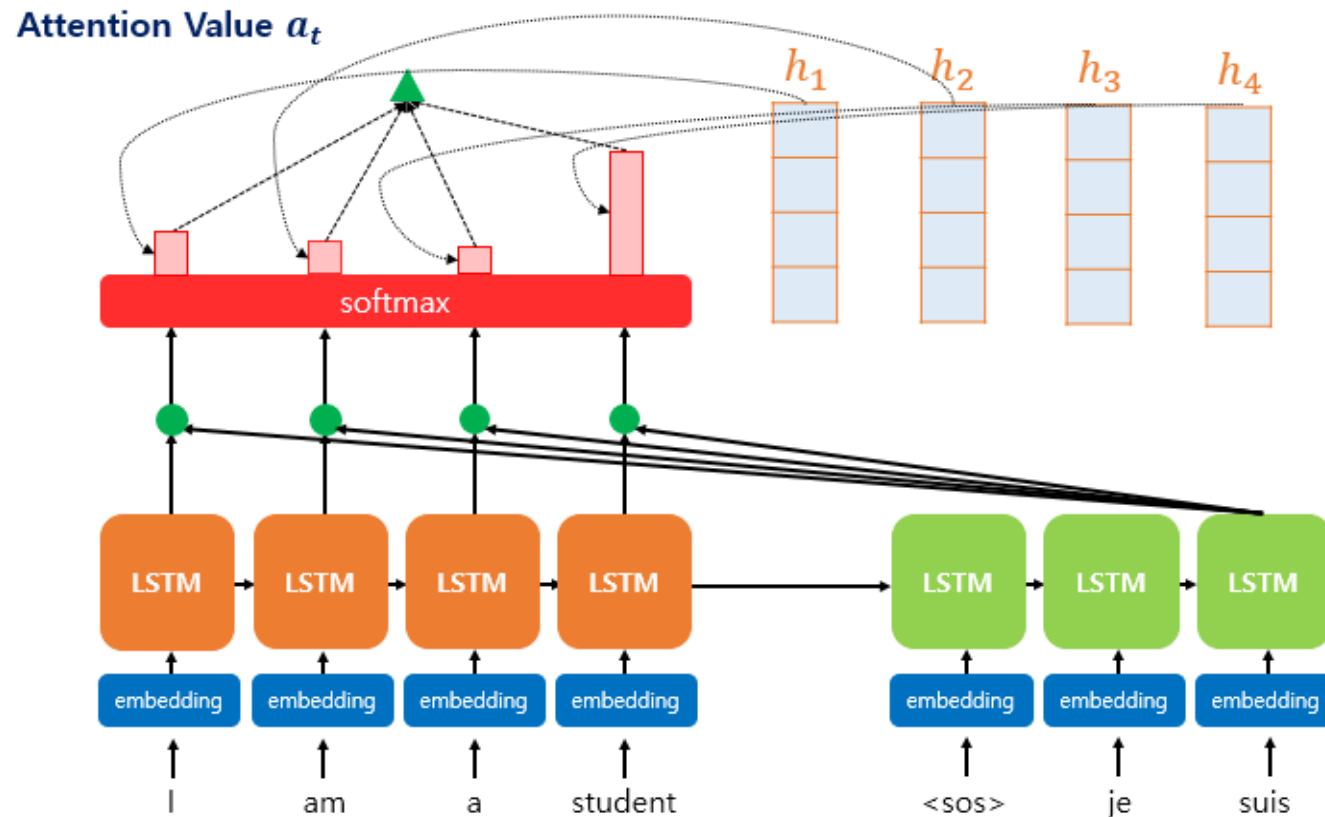
Dot product Attention

- 내적을 통해 구한 유사도를 나열한 벡터가 있다고 해보자. **어텐션 스코어 벡터**.
- 이 벡터에 소프트맥스 함수를 지나면 각 유사도는 0과 1사이의 값으로 변환된다.
- 소프트맥스 함수의 특성으로 이 유사도 값의 총 합은 1이다.



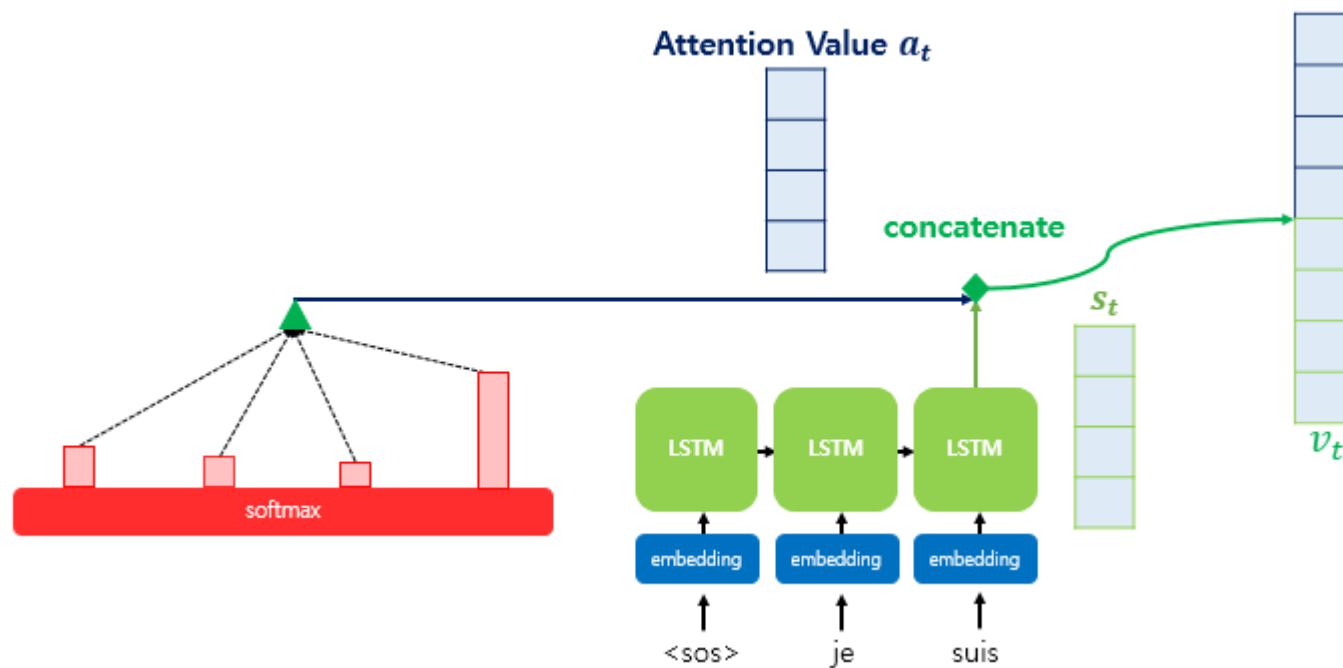
Dot product Attention

- 어텐션 분포. 즉, 소프트맥스 함수로 정규화 된 유사도값을 인코더의 각 은닉 상태에 곱해준다.
- 그리고 이를 모두 더해준다. 이를 **어텐션 값(Attention Value)**이라고 한다.
- Seq2Seq + Attention에서는 이 값을 **컨텍스트 벡터(Context Vector)**라고도 부른다.



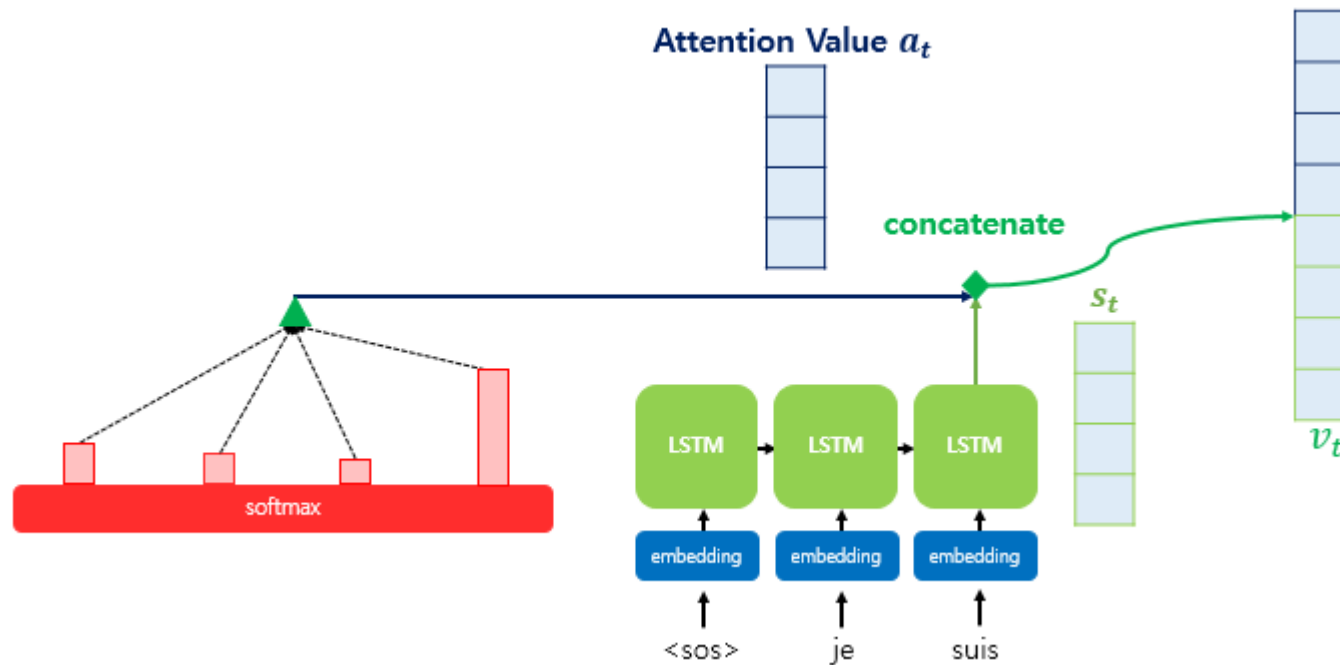
Dot product Attention

- 앞서 구한 **컨텍스트 벡터(Context Vector)**는 디코더의 t시점의 은닉 상태와 연결된다.
- 이 벡터를 v_t 라 해보자. 이 벡터로 현재 시점의 예측 단어인 y_t 를 예측하게 된다.



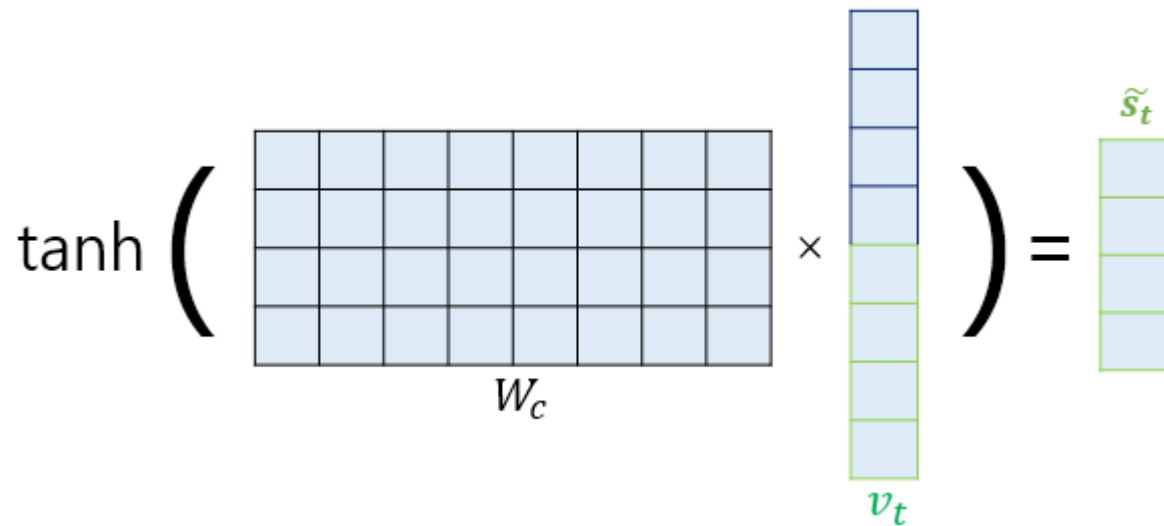
Dot product Attention

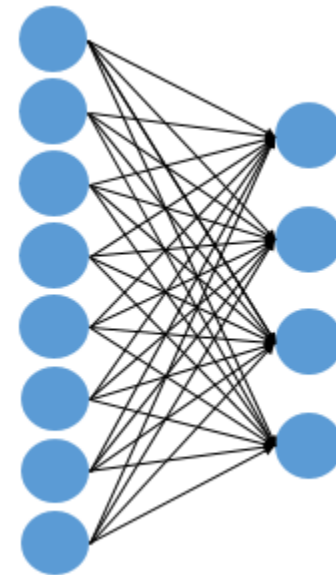
- 앞서 구한 **컨텍스트 벡터(Context Vector)**는 디코더의 t시점의 은닉 상태와 연결된다.
- 이 벡터를 v_t 라 해보자. 이 벡터로 현재 시점의 예측 단어인 y_t 를 예측하게 된다.
- 물론, 아직 구체적인 추가 연산이 더 남아있다.



Dot product Attention

- 벡터 v_t 를 하이퍼볼릭탄젠트 함수를 사용하는 연산을 거쳐 \tilde{s}_t 를 만들어준다.
- 사실 이 연산은 일종의 인공 신경망 연산으로도 볼 수 있다.

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




Dot product Attention

- 벡터 v_t 를 하이퍼볼릭탄젠트 함수를 사용하는 연산을 거쳐 \tilde{s}_t 를 만들어준다.
- 사실 이 연산은 일종의 인공 신경망 연산으로도 볼 수 있다.
- 그 후 \tilde{s}_t 를 출력층의 입력으로 사용하여 y_t 를 예측하게 된다.

$$y_t = \text{softmax}(W_y \tilde{s}_t + b_y)$$

어텐션 메커니즘의 종류

어텐션 메커니즘은 다음 순서로 제안되었다. 이들의 큰 차이는 어텐션 스코어 함수가 다르다는 점이다.

이게 처음에 제안됨.

1. 바다나우 어텐션 (Neural Machine Translation by Jointly Learning to Align and Translate)

- **콘캬트 어텐션(Concat Attention)** : $score(s_t, h_i) = W_a^T \tanh(W_b[s_t; h_i])$

2. 루옹 어텐션 (Effective Approaches to Attention-based Neural Machine Translation)

- **닷 프로덕트 어텐션(Dot-Product Attention)** : $score(s_t, h_i) = s_t^T h_i$
- **제네럴 어텐션(General Attention)** : $score(s_t, h_i) = s_t^T W_a h_i$

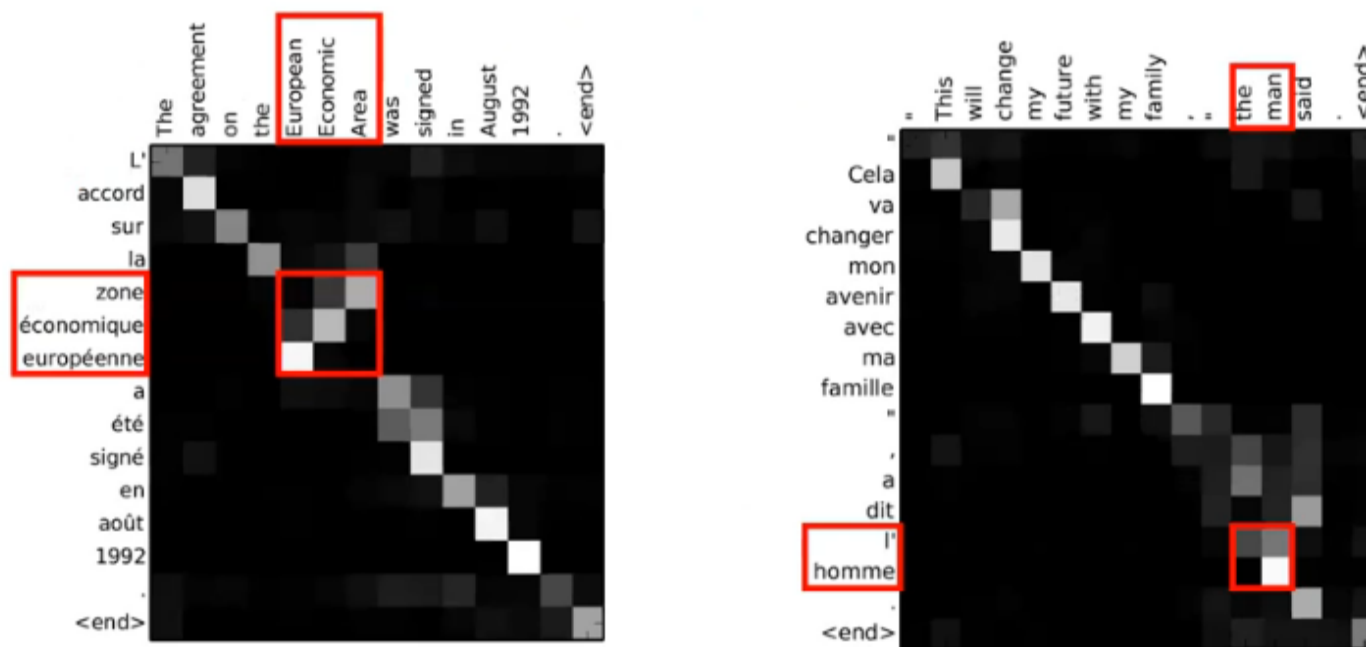
3. 스케일드 닷 프로덕트 어텐션(Attention is All you need)

- **스케일드 닷 프로덕트 어텐션(Scaled Dot-Product Attention)** : $score(s_t, h_i) = \frac{s_t^T h_i}{\sqrt{n}}$

Bahdanau Attention

- 컨텍스트 벡터가 Bottleneck이 되어 번역기의 성능이 저하되는 현상을 보완하기 위해서 제안.
- 어텐션 메커니즘을 사용하지 않은 seq2seq 모델보다 성능이 앞서는 것을 증명.
- 어텐션 메커니즘을 사용하지 않은 seq2seq는 문장의 길이가 길어질수록 성능이 급격히 저하.

어텐션 스코어를 시각화 한 결과



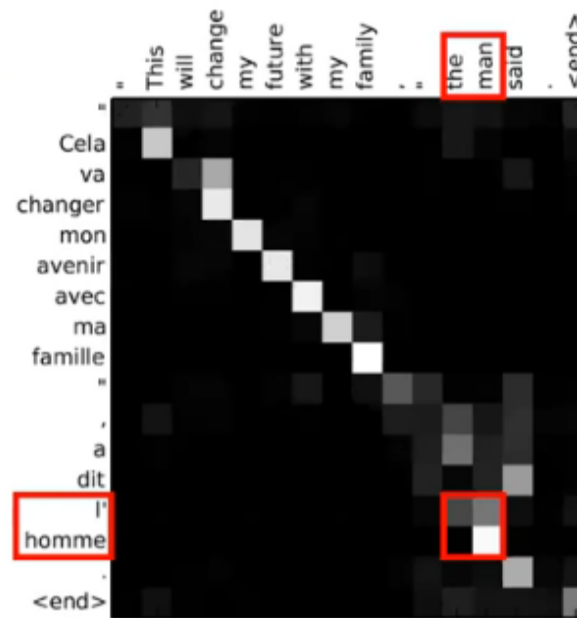
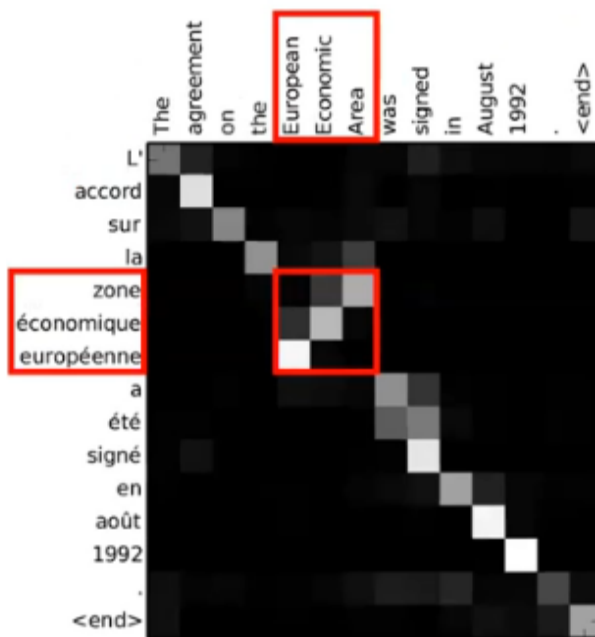
English-French: largely monotonic with a number of non-trivial, non-monotonic alignments

Bahdanau Attention

- 컨텍스트 벡터가 Bottleneck이 되어 번역기의 성능이 저하되는 현상을 보완하기 위해서 제안.
- 어텐션 메커니즘을 사용하지 않은 seq2seq 모델보다 성능이 앞서는 것을 증명.
- 어텐션 메커니즘을 사용하지 않은 seq2seq는 문장의 길이가 길어질수록 성능이 급격히 저하.

어텐션 스코어를 시각화 한 결과

어순이 달라지는
구간을 캐치



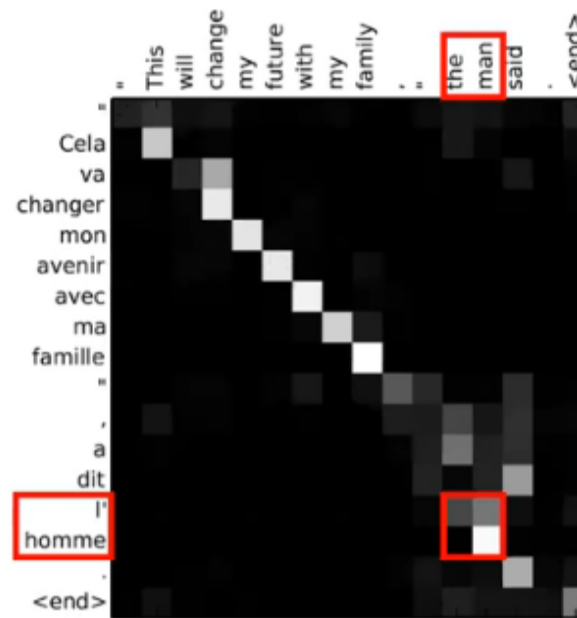
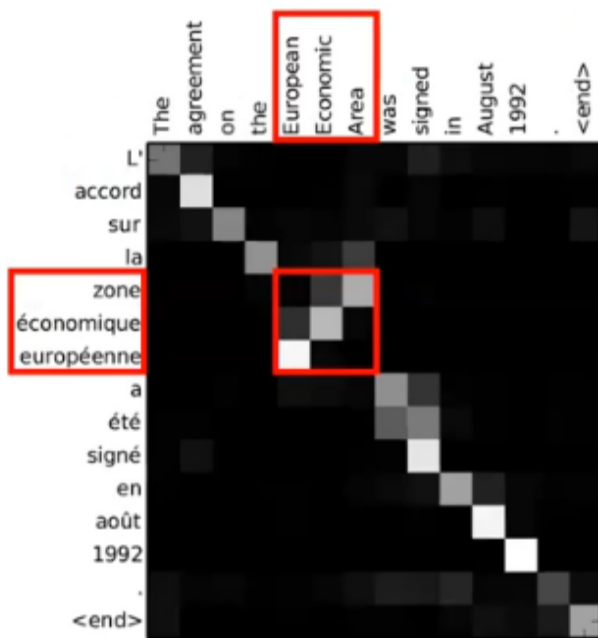
English-French: largely monotonic with a number of non-trivial, non-monotonic alignments

Bahdanau Attention

- 컨텍스트 벡터가 Bottleneck이 되어 번역기의 성능이 저하되는 현상을 보완하기 위해서 제안.
- 어텐션 메커니즘을 사용하지 않은 seq2seq 모델보다 성능이 앞서는 것을 증명.
- 어텐션 메커니즘을 사용하지 않은 seq2seq는 문장의 길이가 길어질수록 성능이 급격히 저하.

어텐션 스코어를 시각화 한 결과

어순이 달라지는
구간을 캐치

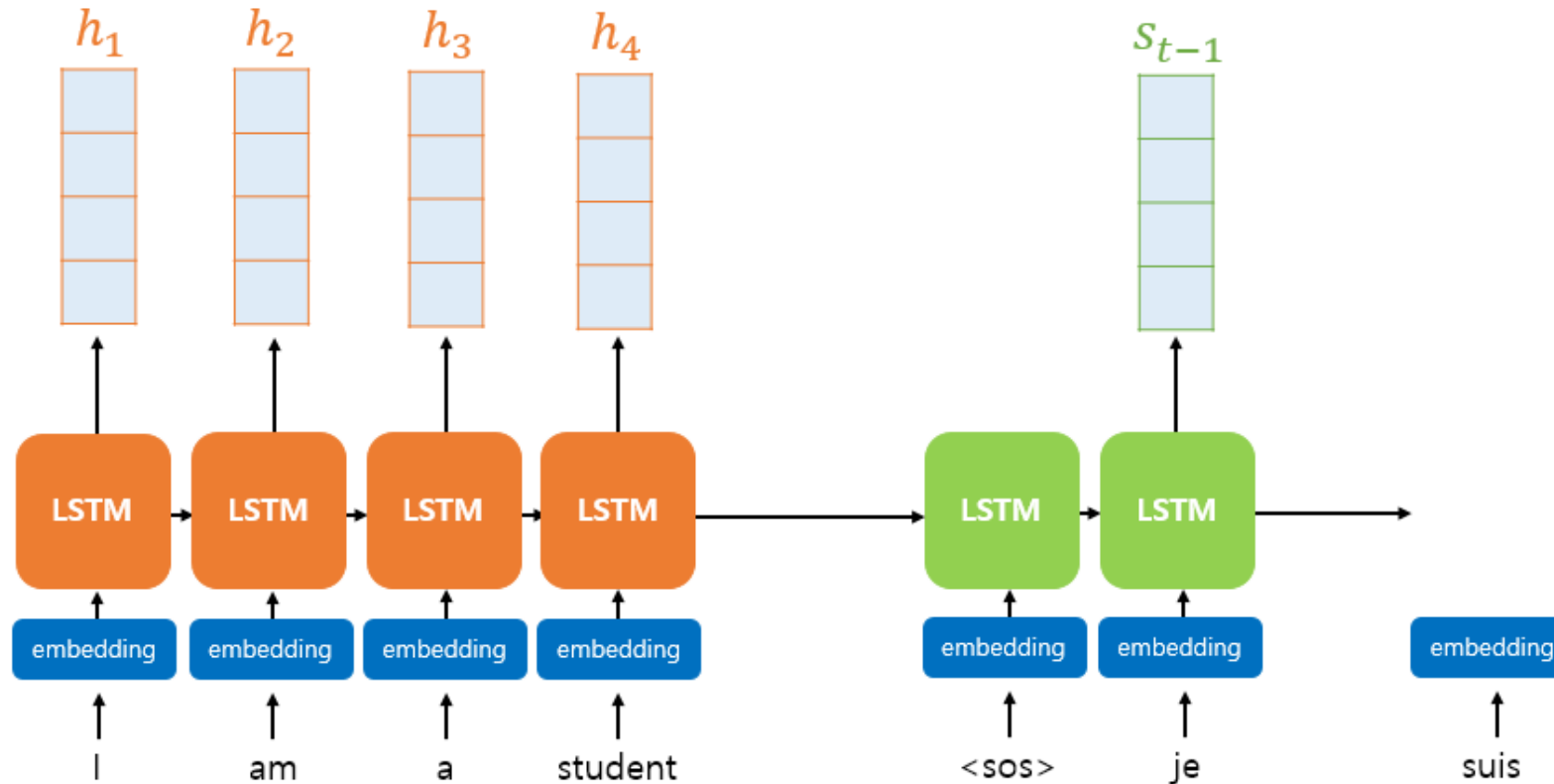


불어의 경우 정관사
종류가 1개가 아님에도
적절한 정관사를 찾아냄.

English-French: largely monotonic with a number of non-trivial, non-monotonic alignments

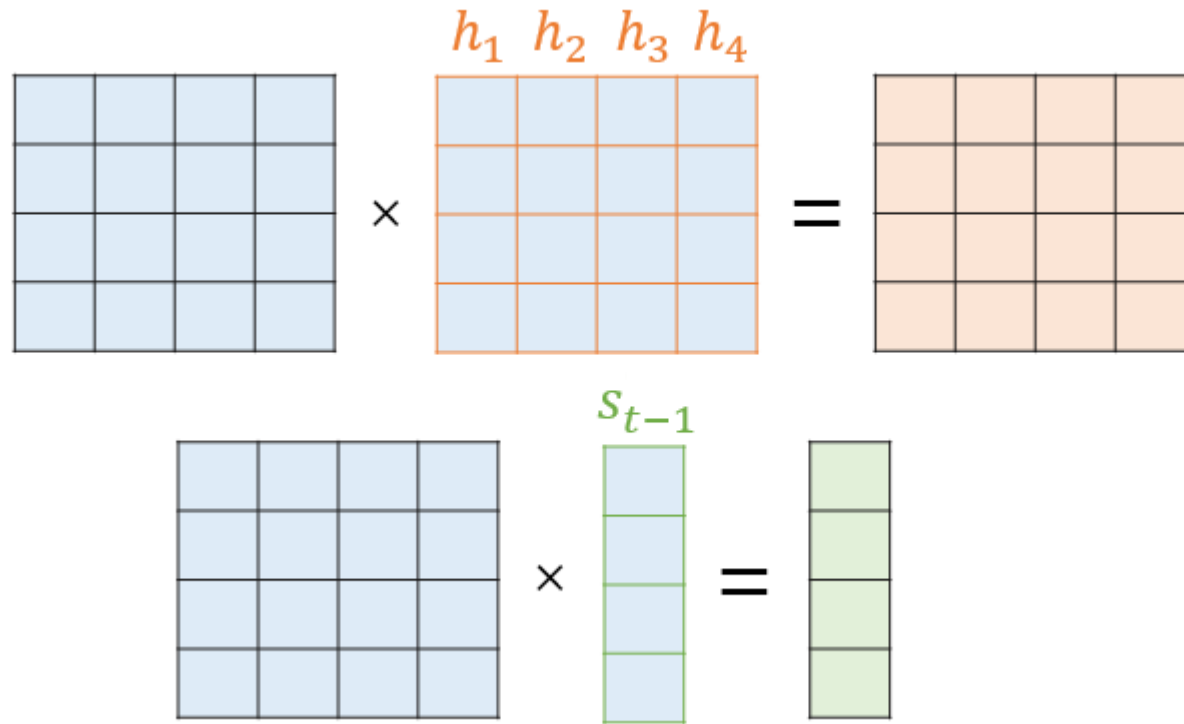
Bahdanau Attention

- 바다나우 어텐션에서는 루옹 어텐션과는 달리 t 시점의 예측을 위해서 $t-1$ 시점의 은닉 상태를 어텐션에 사용.
- 그림에서는 디코더의 세 번째 시점의 예측을 위해 디코더의 두 번째 시점의 은닉 상태를 사용.
- 바다나우 어텐션의 어텐션 스코어 함수 : $score(query, key) = V^T \tanh(W_1 key + W_2 query)$



Bahdanau Attention

- 바다나우 어텐션에서는 루옹 어텐션과는 달리 t 시점의 예측을 위해서 $t-1$ 시점의 은닉 상태를 어텐션에 사용.
- 그림에서는 디코더의 세 번째 시점의 예측을 위해 디코더의 두 번째 시점의 은닉 상태를 사용.
- 바다나우 어텐션의 어텐션 스코어 함수 : $score(query, key) = V^T \tanh(W_1 key + W_2 query)$



Bahdanau Attention

- 바다나우 어텐션에서는 루옹 어텐션과는 달리 t시점의 예측을 위해서 t-1시점의 은닉 상태를 어텐션에 사용.
- 그림에서는 디코더의 세 번째 시점의 예측을 위해 디코더의 두 번째 시점의 은닉 상태를 사용.
- 바다나우 어텐션의 어텐션 스코어 함수 : $score(query, key) = V^T \tanh(W_1 key + W_2 query)$

$$\tanh \left(\begin{matrix} s_{t-1} \\ \vdots \end{matrix} + \begin{matrix} h_1 & h_2 & h_3 & h_4 \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{matrix} \right) = \begin{matrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{matrix}$$

$$\begin{matrix} \square & \square & \square & \square \end{matrix} \times \begin{matrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{matrix} = \begin{matrix} \text{Attention Score} \\ \square & \square & \square & \square \\ h_1 & h_2 & h_3 & h_4 \end{matrix}$$

Bahdanau Attention

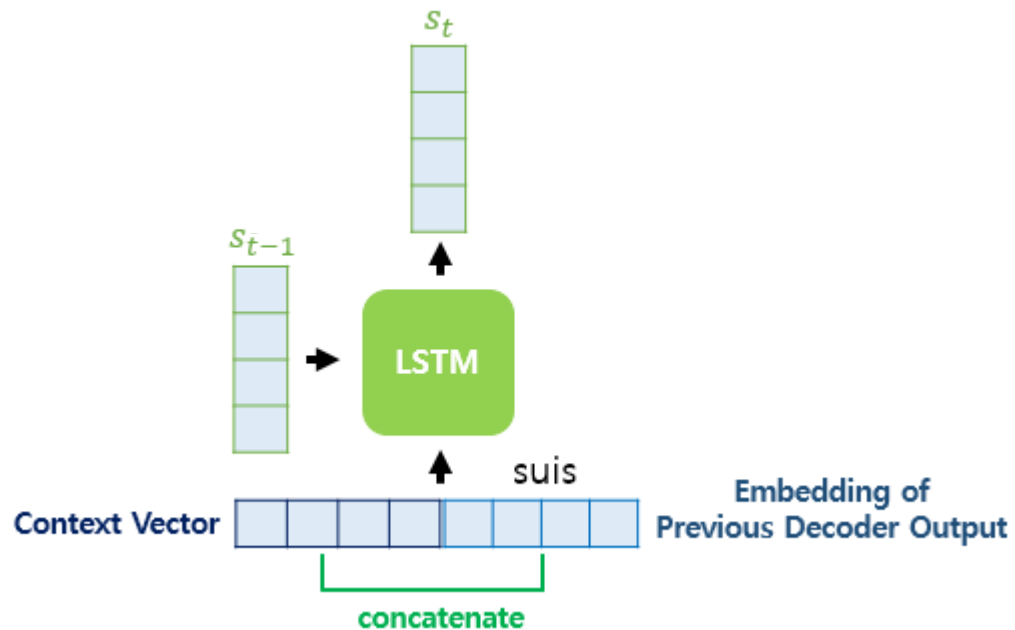
- 바다나우 어텐션의 어텐션 스코어 함수 : $score(query, key) = V^T \tanh(W_1 key + W_2 query)$
- 어텐션 스코어로부터 어텐션 분포를 구한 후, 이를 인코더의 은닉 상태와 가중합하여 컨텍스트 벡터를 구한다.

$$\text{softmax} \left(\begin{array}{c} \text{Attention Score} \\ \begin{array}{|c|c|c|c|} \hline \text{ } & \text{ } & \text{ } & \text{ } \\ \hline \end{array} \\ \begin{array}{c} h_1 \quad h_2 \quad h_3 \quad h_4 \end{array} \end{array} \right) = \begin{array}{c} \text{Attention} \\ \text{Distribution} \\ \begin{array}{|c|c|c|c|} \hline \text{ } & \text{ } & \text{ } & \text{ } \\ \hline \end{array} \end{array}$$

$$\begin{array}{c} h_1 \quad h_2 \quad h_3 \quad h_4 \\ \begin{array}{|c|c|c|c|} \hline \text{ } & \text{ } & \text{ } & \text{ } \\ \hline \text{ } & \text{ } & \text{ } & \text{ } \\ \hline \text{ } & \text{ } & \text{ } & \text{ } \\ \hline \text{ } & \text{ } & \text{ } & \text{ } \\ \hline \end{array} \end{array} \times \begin{array}{|c|} \hline \text{ } \\ \hline \text{ } \\ \hline \text{ } \\ \hline \text{ } \\ \hline \end{array} = \begin{array}{c} \text{Context Vector} \\ \begin{array}{|c|c|c|c|} \hline \text{ } & \text{ } & \text{ } & \text{ } \\ \hline \end{array} \end{array}$$

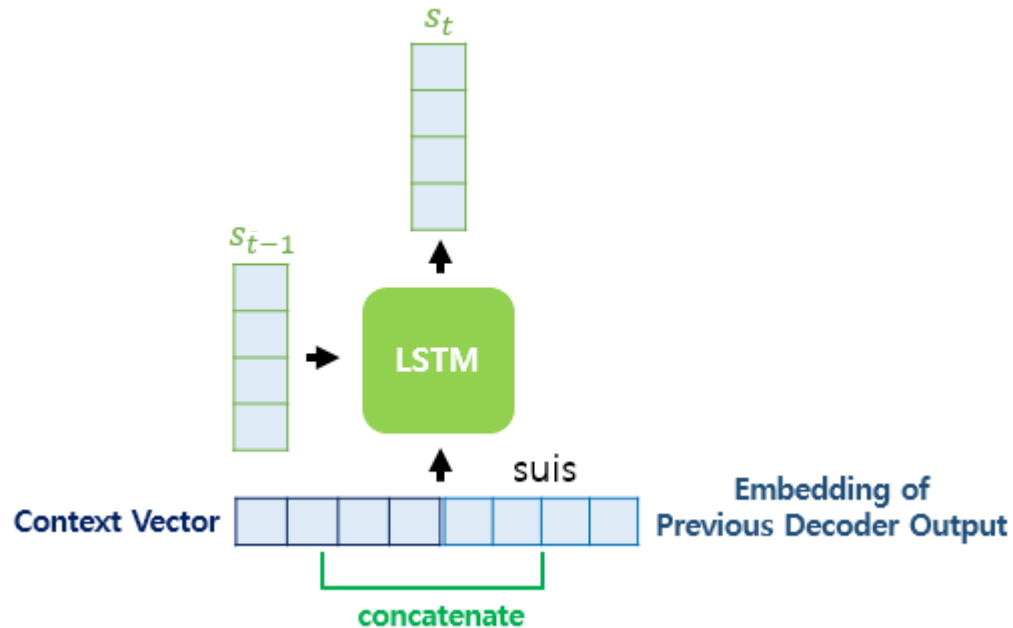
Bahdanau Attention

- 컨텍스트 벡터와 이전 시점의 예측된 단어의 임베딩 벡터를 연결하여 현재 시점의 디코더 셀의 입력으로 사용.
- 이전 시점의 은닉 상태와 현재 시점의 입력을 통해서 현재 시점의 은닉 상태를 구한다.



Bahdanau Attention

- 컨텍스트 벡터와 이전 시점의 예측된 단어의 임베딩 벡터를 연결하여 현재 시점의 디코더 셀의 입력으로 사용.
- 이전 시점의 은닉 상태와 현재 시점의 입력을 통해서 현재 시점의 은닉 상태를 구한다.
- 그 후 s_t 를 출력층의 입력으로 사용하여 y_t 를 예측하게 된다.



$$y_t = \text{softmax}(W_y s_t + b_y)$$

Bahdanau Attention

짧은 영상을 통해서 정말로 다 이해했는지 복습해봅시다.
보시다가 이해가 안 되는 부분이 있다면 바로 질문주세요.

바다나우 어텐션을 설명하는 영상(한국어) :

<https://www.youtube.com/watch?v=WsQLdu2JMgl>

어텐션 메커니즘의 종류

어텐션 메커니즘은 다음 순서로 제안되었다. 이들의 큰 차이는 어텐션 스코어 함수가 다르다는 점이다.

1. 바다나우 어텐션 (Neural Machine Translation by Jointly Learning to Align and Translate)

- 콘캣 어텐션(Concat Attention) : $score(s_t, h_i) = W_a^T \tanh(W_b[s_t; h_i])$

2. 루옹 어텐션 (Effective Approaches to Attention-based Neural Machine Translation) **루옹 어텐션에서 내적대신 이 식을 사용하면 제네럴 어텐션.**

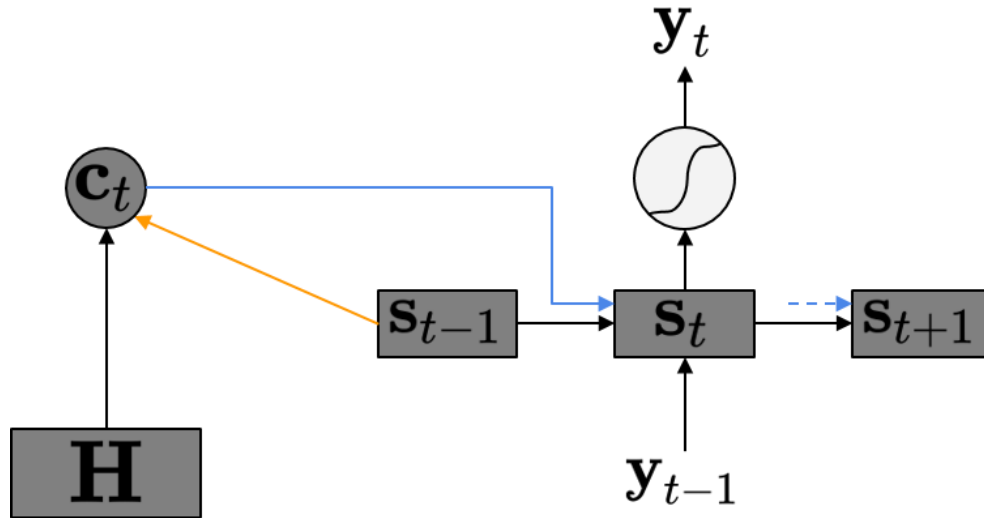
- ~~닷 프로덕트 어텐션(Dot-Product Attention)~~ : $score(s_t, h_i) = s_t^T h_i$
- **제네럴 어텐션(General Attention)** : $score(s_t, h_i) = s_t^T W_a h_i$

3. 스케일드 닷 프로덕트 어텐션(Attention is All you need)

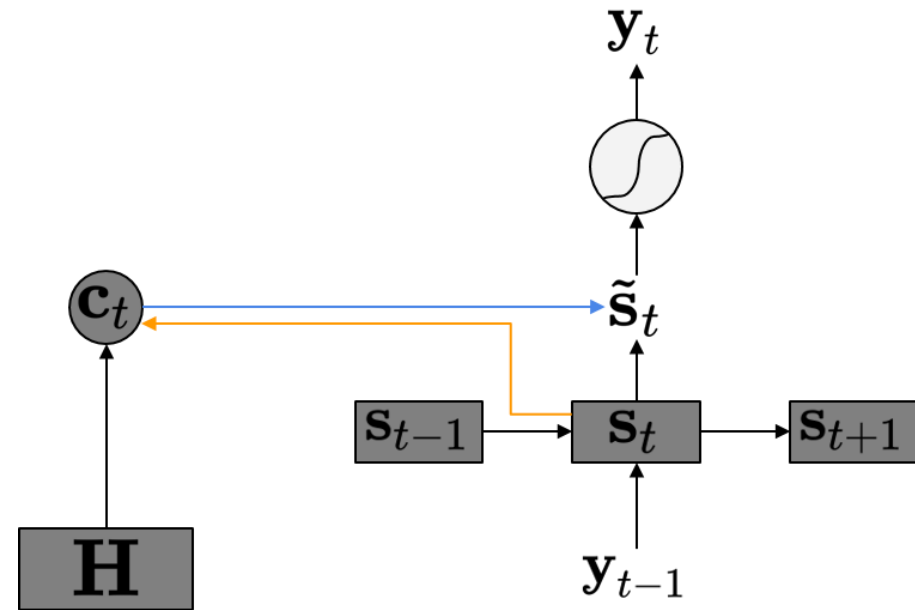
- 스케일드 닷 프로덕트 어텐션(Scaled Dot-Product Attention) : $score(s_t, h_i) = \frac{s_t^T h_i}{\sqrt{n}}$

Bahdanau Attention Vs. Luong Attention

두 어텐션 메커니즘의 연산 순서를 그림으로 표현하면 다음과 같다.
어텐션 스코어 함수와 상관없이 연산 순서를 표현한 그린 그림이다.



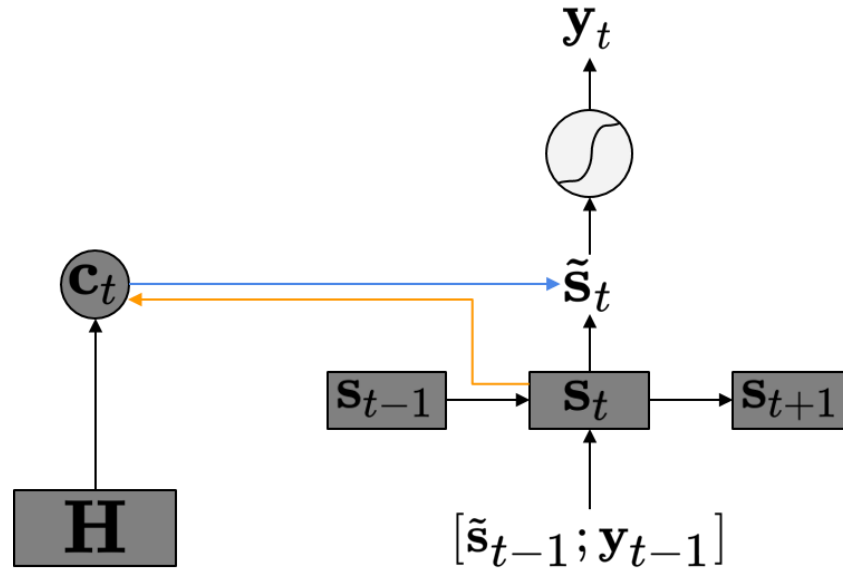
바다나우 어텐션



루옹 어텐션

Luong Attention (input feeding ver)

이전 타임 스텝의 출력을 넣을 때, 이전 타임 스텝에서 구한 \tilde{s}_{t-1} 를 같이 넣어주는 것은 어떨까라는 아이디어.

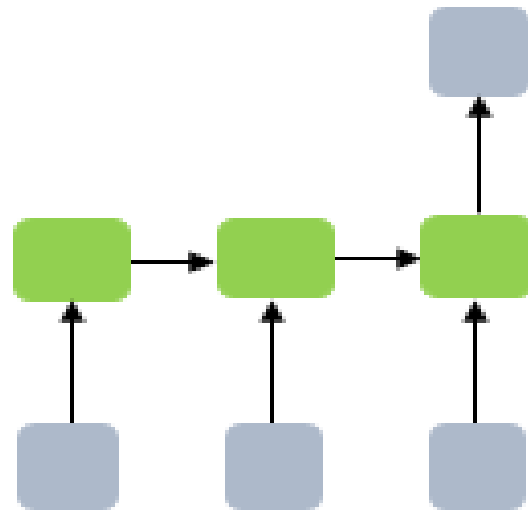


루옹 어텐션

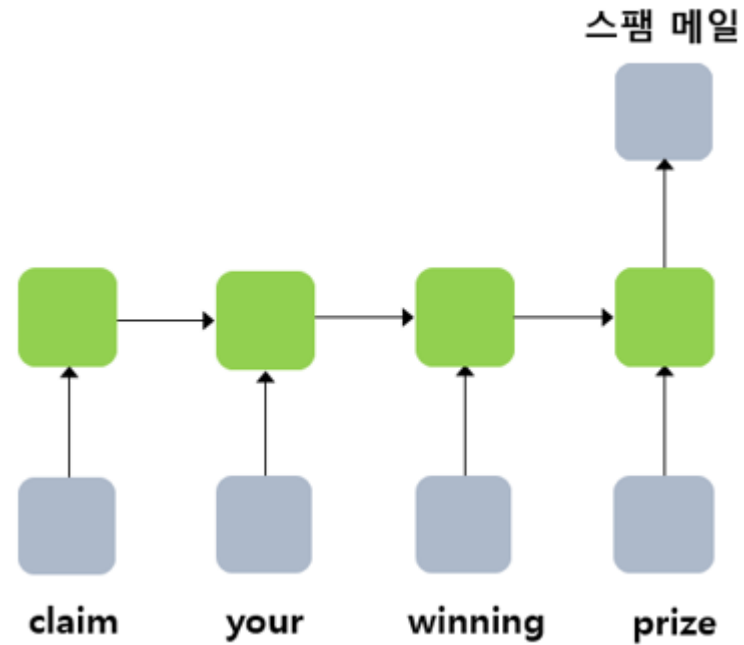
Text Classification Using Attention

Text Classification using LSTM

- LSTM을 이용한 텍스트 분류.
- many-to-one LSTM을 사용.
- 매 시점마다 다음 단어가 입력되어 최종 시점에서 텍스트를 분류.



many-to-one



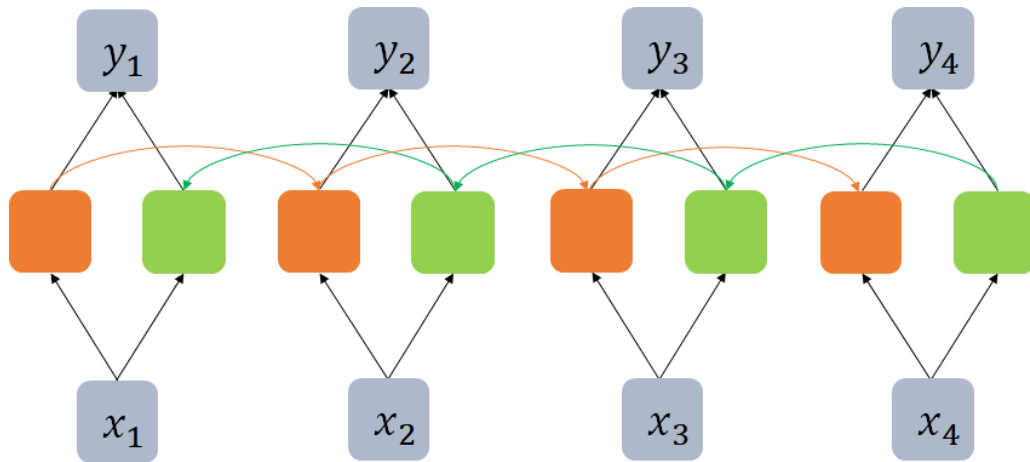
Text Classification using LSTM

- LSTM을 이용한 텍스트 분류.
- many-to-one LSTM을 사용.
- 매 시점마다 다음 단어가 입력되어 최종 시점에서 텍스트를 분류.



Text Classification using BiLSTM

- BiLSTM을 이용한 텍스트 분류.
- 텍스트 분류를 위해서는 결국 many-to-one을 사용해야 할텐데 그 구조는?

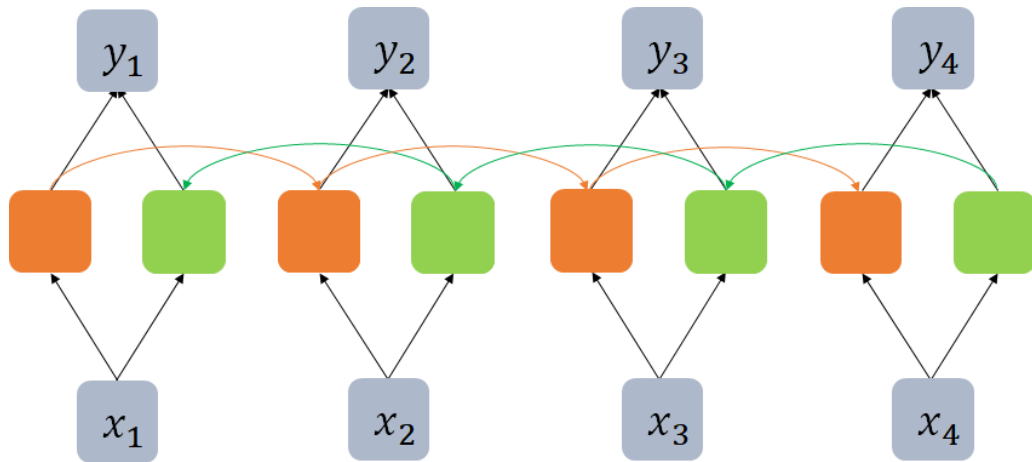


BiLSTM

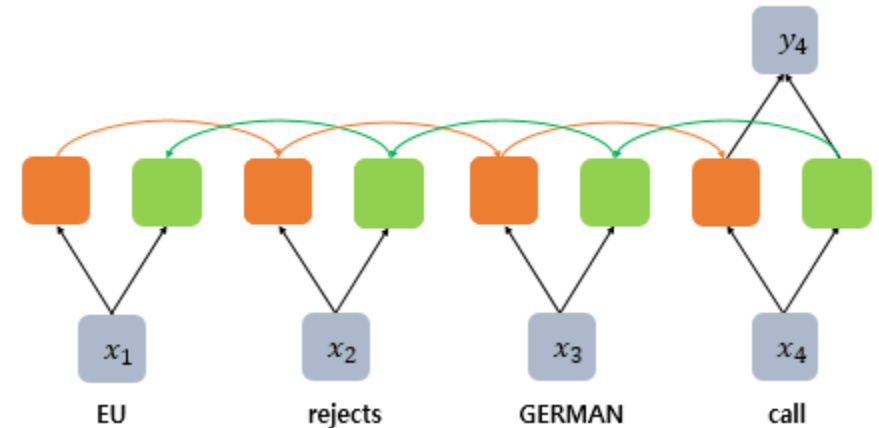
BiLSTM for Text Classification

Text Classification using BiLSTM

- BiLSTM을 이용한 텍스트 분류.
- 텍스트 분류를 위해서는 결국 many-to-one을 사용해야 할텐데 그 구조는?



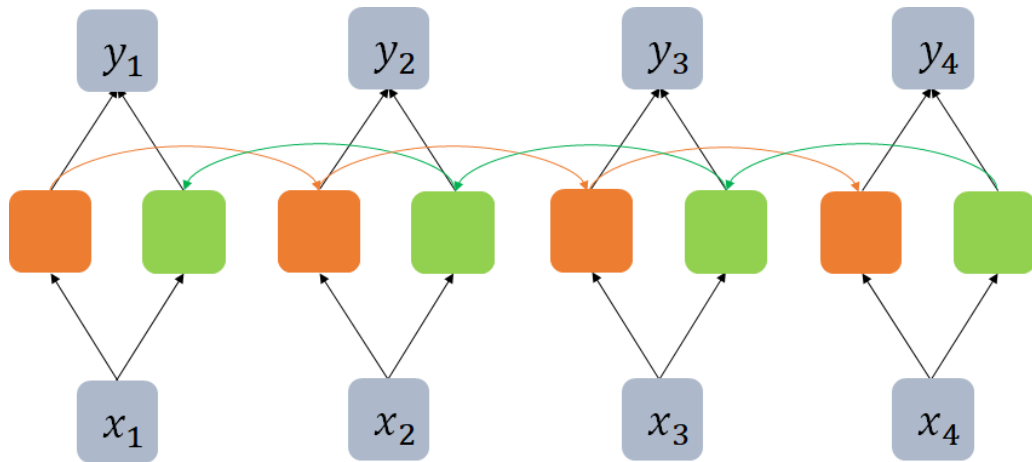
BiLSTM



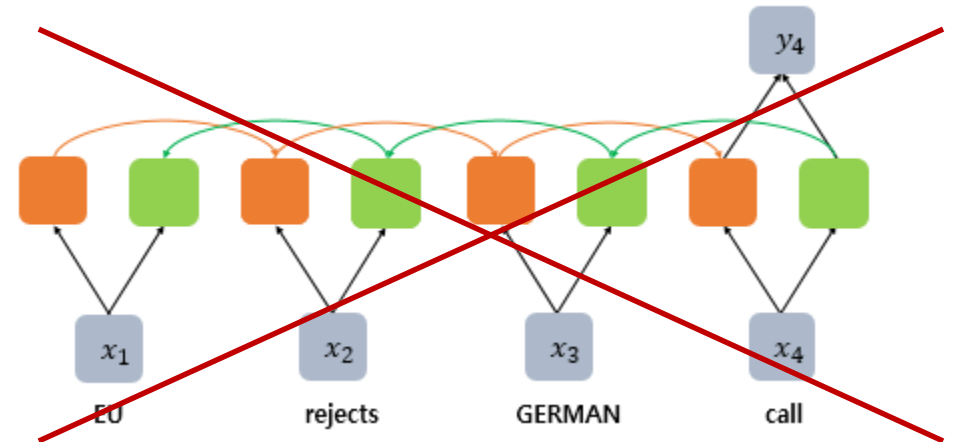
BiLSTM for Text Classification

Text Classification using BiLSTM

- BiLSTM을 이용한 텍스트 분류.
- 텍스트 분류를 위해서는 결국 many-to-one을 사용해야 할텐데 그 구조는?



BiLSTM

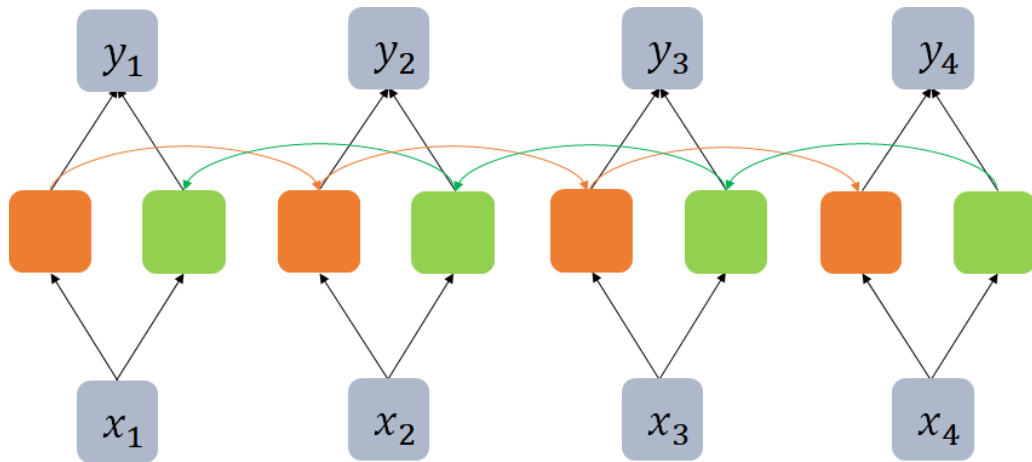


BiLSTM for Text Classification

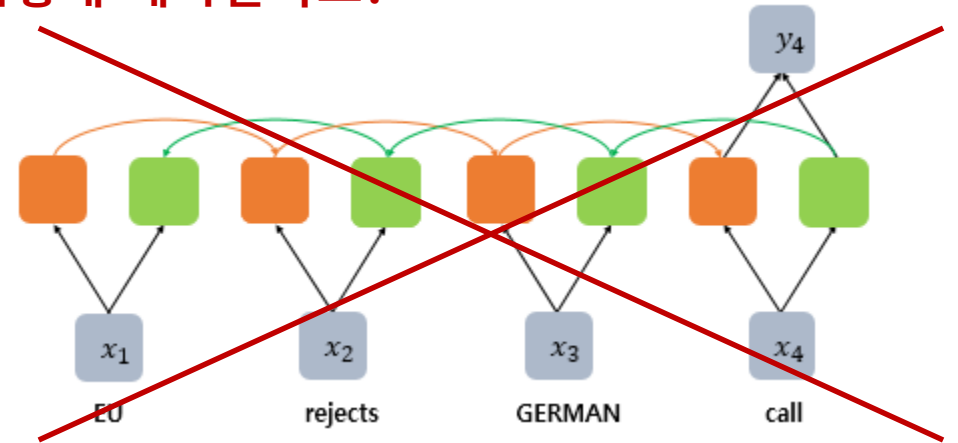
Text Classification using BiLSTM

- BiLSTM을 이용한 텍스트 분류.
- 텍스트 분류를 위해서는 결국 many-to-one을 사용해야 할텐데 그 구조는?

이렇게 하면 안 되는 이유가 뭡까요?
그리고 다른 방법이 있다면 어떻게 해야할까요?

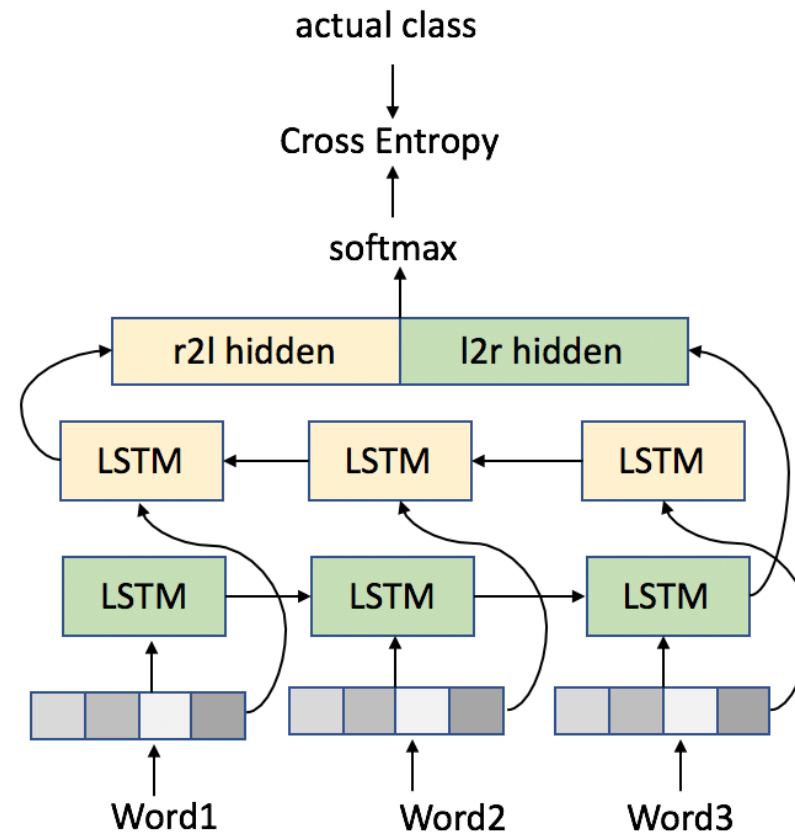


BiLSTM



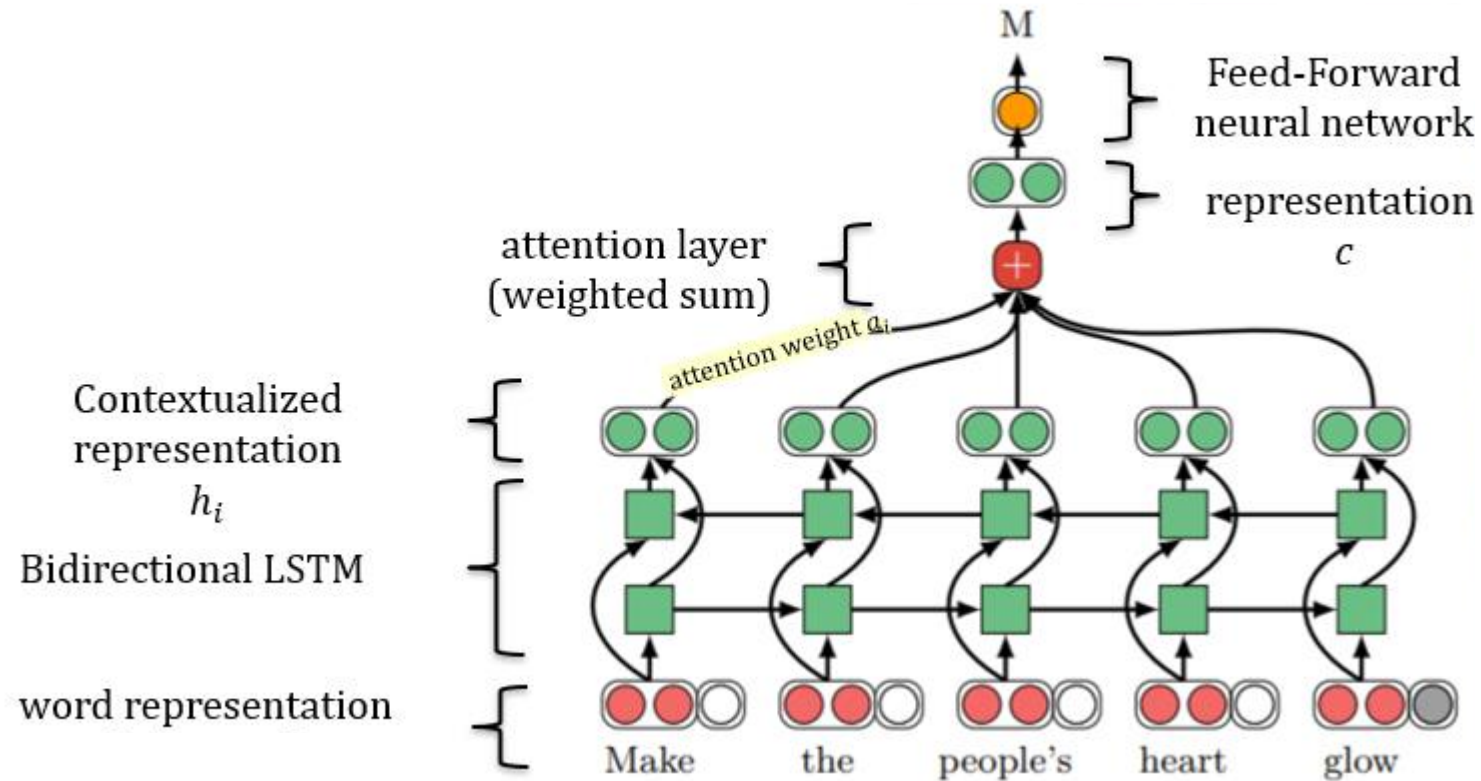
BiLSTM for Text Classification

Text Classification using BiLSTM



BiLSTM for Text Classification

Text Classification using Attention

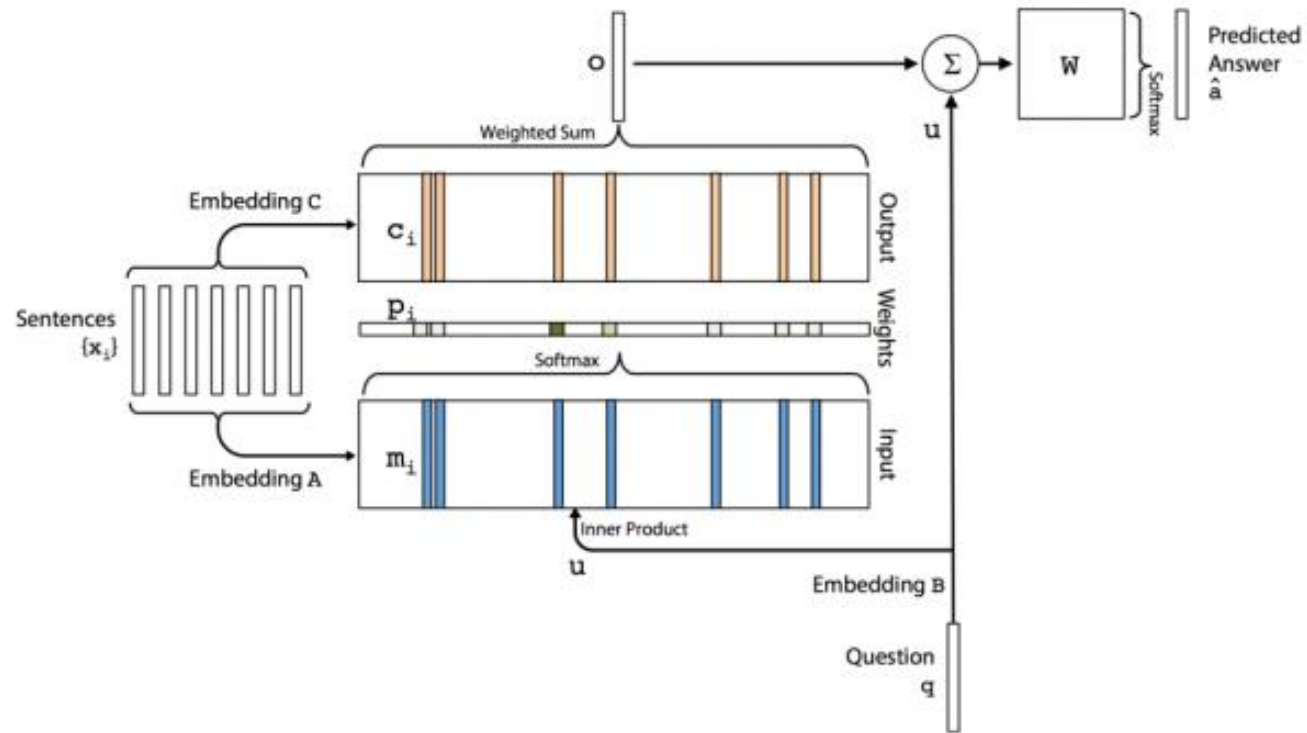


BiLSTM-Attention for Text Classification

Memory Network

Memory Network

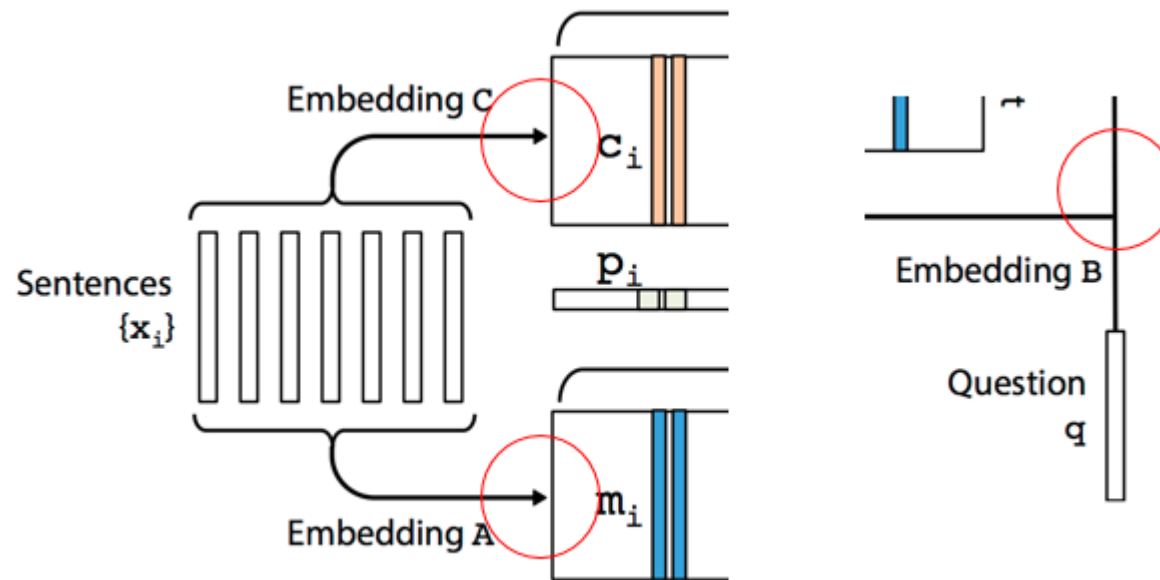
- 어텐션 메커니즘을 QA 문제에 사용하도록 제안.
- 질문 문장과 스토리 문장에 대해서 어텐션 메커니즘을 수행하여 스토리에서 답안을 도출.



링크 : <http://solarisailab.com/archives/934>

Memory Network

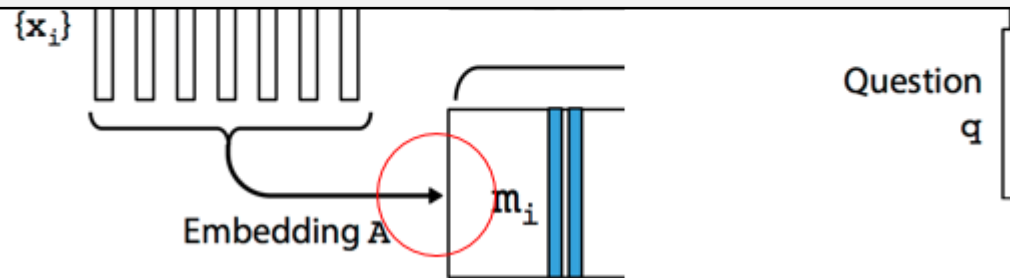
- 입력은 총 3개의 임베딩을 사용한다.
- 맨 좌측의 Sentences는 스토리에 해당된다. 스토리는 Embedding A와 Embedding C 두 번에 걸쳐서 임베딩이 진행된다. 이 두 임베딩은 별도의 다른 임베딩이다.
- 우측 하단은 Question 질문에 해당된다. 이는 Embedding B라는 별도로 임베딩된다.



Memory Network

- 입력은 총 3개의 임베딩을 사용한다.
- 맨 좌측의 Sentences는 스토리에 해당된다. 스토리는 Embedding A와 Embedding C 두 번에 걸쳐서 임베딩이 진행된다. 이 두 임베딩은 별도의 다른 임베딩이다.
- 우측 하단은 Question 질문에 해당된다. 이는 Embedding B라는 별도로 임베딩된다.

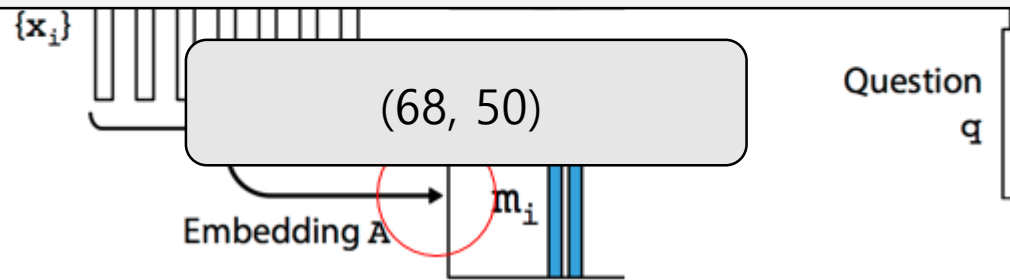
Q. 만약 스토리 문장의 길이가 68이라고 하고, 임베딩 벡터의 차원을 50이라고 하자. 68의 길이를 가지는 스토리 문장이 Embedding A에 입력되어 결과로 얻는 행렬의 크기는?
(단, 배치 크기는 고려하지 않는다.)



Memory Network

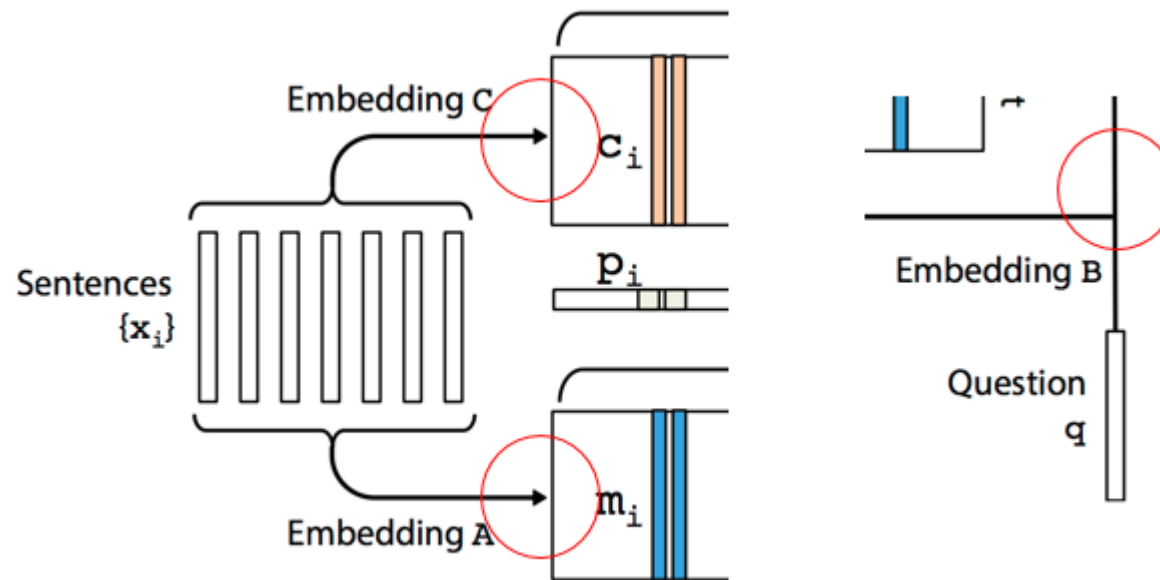
- 입력은 총 3개의 임베딩을 사용한다.
- 맨 좌측의 Sentences는 스토리에 해당된다. 스토리는 Embedding A와 Embedding C 두 번에 걸쳐서 임베딩이 진행된다. 이 두 임베딩은 별도의 다른 임베딩이다.
- 우측 하단은 Question 질문에 해당된다. 이는 Embedding B라는 별도로 임베딩된다.

Q. 만약 스토리 문장의 길이가 68이라고 하고, 임베딩 벡터의 차원을 50이라고 하자. 68의 길이를 가지는 스토리 문장이 Embedding A에 입력되어 결과로 얻는 행렬의 크기는?
(단, 배치 크기는 고려하지 않는다.)



Memory Network

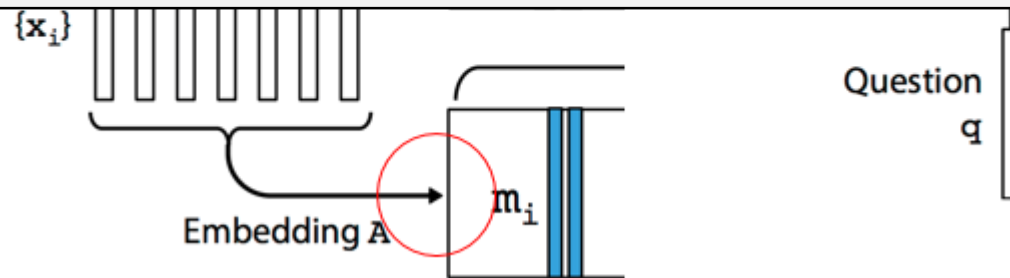
- 입력은 총 3개의 임베딩을 사용한다.
- 맨 좌측의 Sentences는 스토리에 해당된다. 스토리는 Embedding A와 Embedding C 두 번에 걸쳐서 임베딩이 진행된다. 이 두 임베딩은 별도의 다른 임베딩이다.
- 우측 하단은 Question 질문에 해당된다. 이는 Embedding B라는 별도로 임베딩된다.



Memory Network

- 입력은 총 3개의 임베딩을 사용한다.
- 맨 좌측의 Sentences는 스토리에 해당된다. 스토리는 Embedding A와 Embedding C 두 번에 걸쳐서 임베딩이 진행된다. 이 두 임베딩은 별도의 다른 임베딩이다.
- 우측 하단은 Question 질문에 해당된다. 이는 Embedding B라는 별도로 임베딩된다.

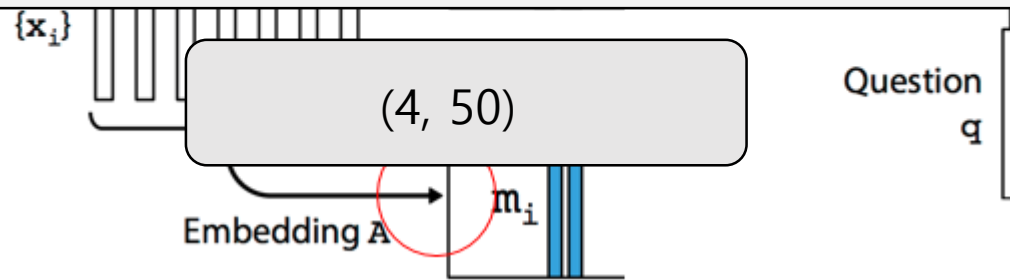
Q. 만약 질문 문장의 길이가 4라고 하고, 임베딩 벡터의 차원을 50이라고 하자. 4의 길이를 가지는 질문 문장이 Embedding B에 입력되어 결과로 얻는 행렬의 크기는?
(단, 배치 크기는 고려하지 않는다.)



Memory Network

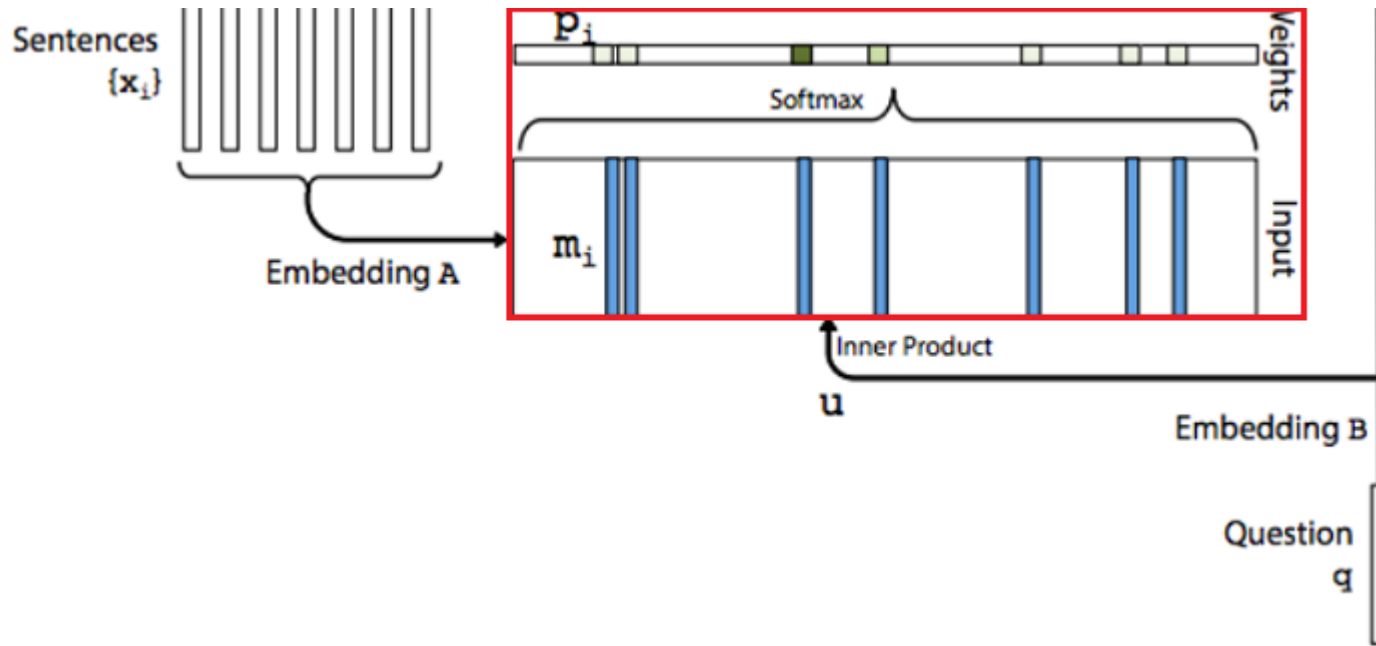
- 입력은 총 3개의 임베딩을 사용한다.
- 맨 좌측의 Sentences는 스토리에 해당된다. 스토리는 Embedding A와 Embedding C 두 번에 걸쳐서 임베딩이 진행된다. 이 두 임베딩은 별도의 다른 임베딩이다.
- 우측 하단은 Question 질문에 해당된다. 이는 Embedding B라는 별도로 임베딩된다.

Q. 만약 질문 문장의 길이가 4라고 하고, 임베딩 벡터의 차원을 50이라고 하자. 4의 길이를 가지는 질문 문장이 Embedding B에 입력되어 결과로 얻는 행렬의 크기는?
(단, 배치 크기는 고려하지 않는다.)



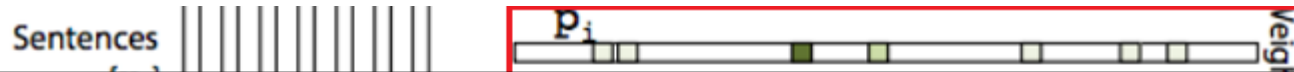
Memory Network

- 스토리 문장과 질문 문장의 매칭 유사도를 계산하는 과정은 내적을 통해 수행된다.
- 스토리 문장을 Embedding A를 거쳐서 얻은 텐서를 m (68, 50)
- 질문 문장을 Embedding B를 거쳐서 얻은 텐서를 u (4, 50)



Memory Network

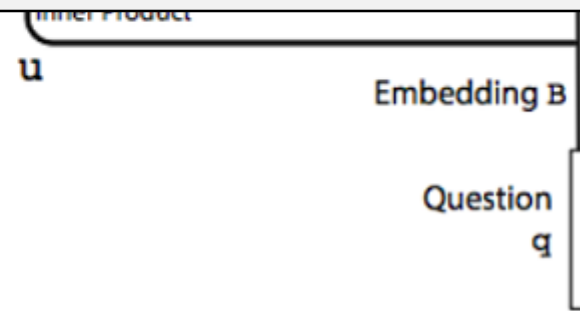
- 스토리 문장과 질문 문장의 매칭 유사도를 계산하는 과정은 내적을 통해 수행된다.
- 스토리 문장을 Embedding A를 거쳐서 얻은 텐서를 m (68, 50)
- 질문 문장을 Embedding B를 거쳐서 얻은 텐서를 u (4, 50)



Q. 텐서 m 과 텐서 u 를 내적한 후 소프트맥스 함수를 통과시킨다.

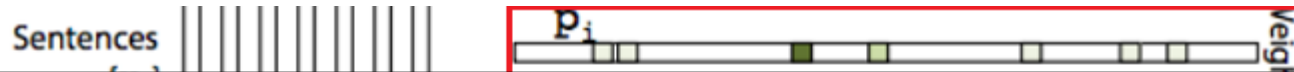
$$p = \text{Softmax}(\text{dot}(m, u))$$

이 결과 행렬의 크기와 의미하는 바는?



Memory Network

- 스토리 문장과 질문 문장의 매칭 유사도를 계산하는 과정은 내적을 통해 수행된다.
- 스토리 문장을 Embedding A를 거쳐서 얻은 텐서를 m (68, 50)
- 질문 문장을 Embedding B를 거쳐서 얻은 텐서를 u (4, 50)



Q. 텐서 m 과 텐서 u 를 내적한 후 소프트맥스 함수를 통과시킨다.

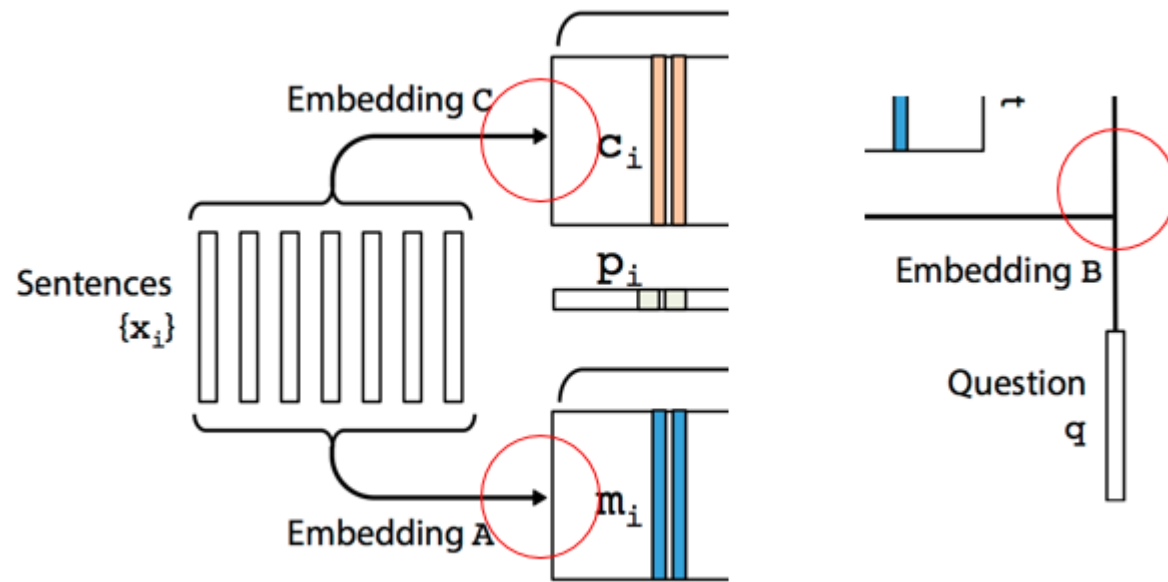
$$p = \text{Softmax}(\text{dot}(m, u))$$

이 결과 행렬의 크기와 의미하는 바는?

(68, 4),
스토리에 있는 각 68개의 단어와 질문에 있는 각
4개의 단어에 대한 유사도를 의미한다.
어텐션 분포 행렬이라고 볼 수 있다.

Memory Network

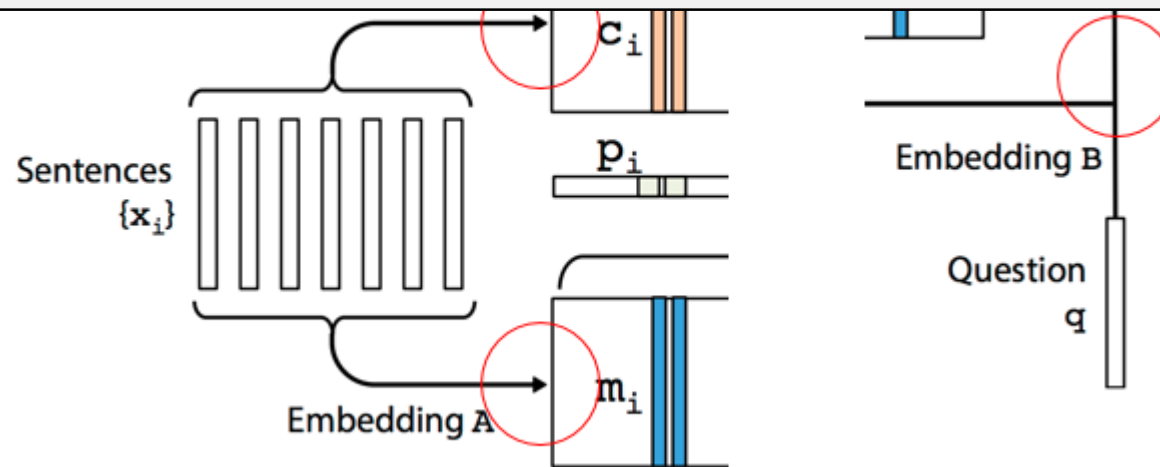
- 현재 유사도가 반영된 어텐션 분포 행렬 p 의 크기는 $(68, 4)$ 이다.
- 그런데 이 어텐션 분포 행렬을 어텐션 메커니즘 상 Value에 반영해주기 위해서는
- Value 행렬의 크기가 어텐션 분포 행렬의 크기와 동일해야 한다.
- 이에 따라 Embedding C의 결과로는 $(68, 4)$ 가 나오도록 한다.



Memory Network

- 현재 유사도가 반영된 어텐션 분포 행렬 p 의 크기는 $(68, 4)$ 이다.
- 그런데 이 어텐션 분포 행렬을 어텐션 메커니즘 상 Value에 반영해주기 위해서는
- Value 행렬의 크기가 어텐션 분포 행렬의 크기와 동일해야 한다.
- 이에 따라 Embedding C의 결과로는 $(68, 4)$ 가 나오도록 한다.

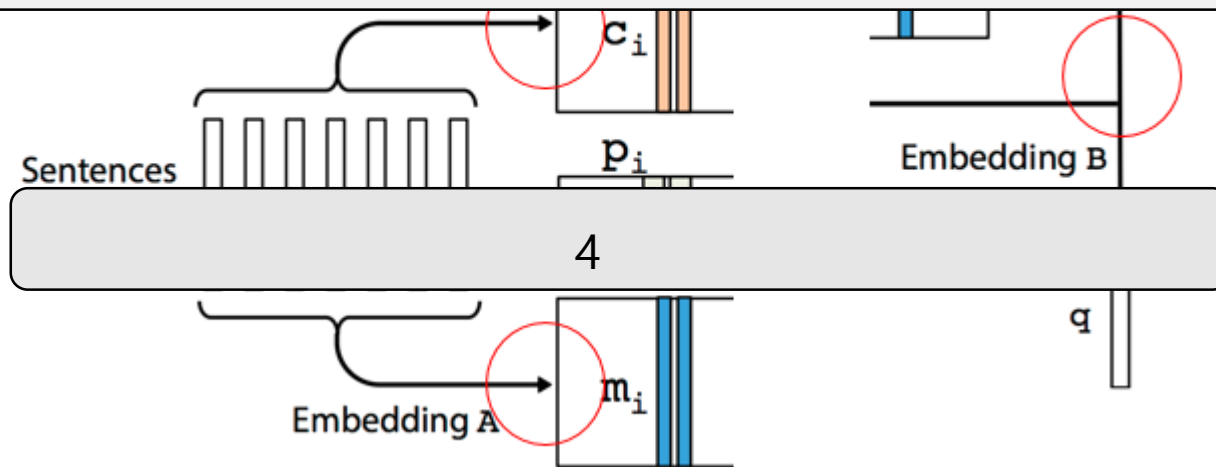
Q. Embedding C의 하이퍼파라미터인 임베딩 벡터의 차원은?



Memory Network

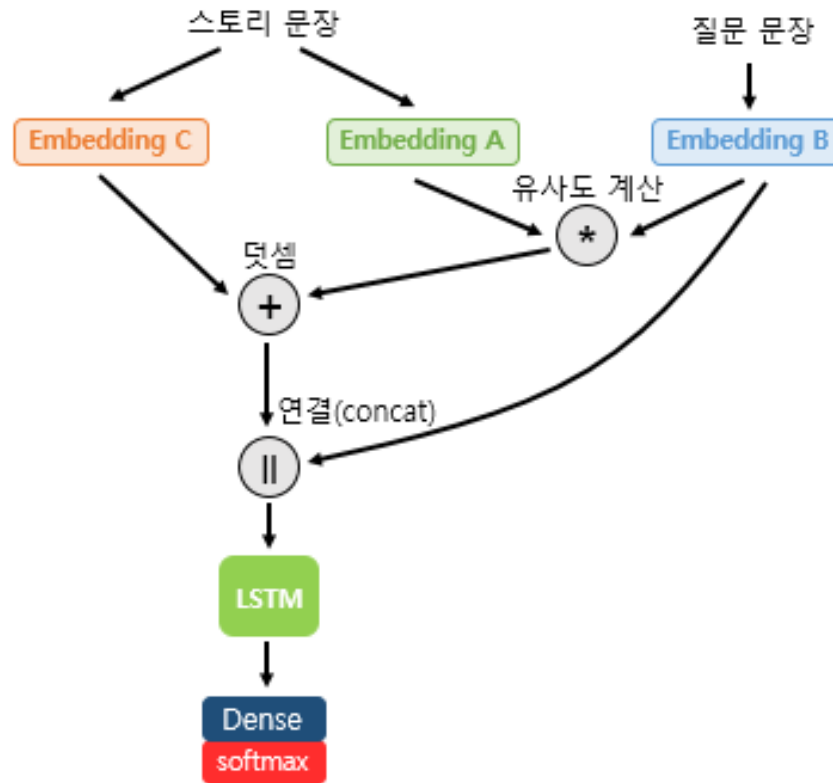
- 현재 유사도가 반영된 어텐션 분포 행렬 p 의 크기는 $(68, 4)$ 이다.
- 그런데 이 어텐션 분포 행렬을 어텐션 메커니즘 상 Value에 반영해주기 위해서는
- Value 행렬의 크기가 어텐션 분포 행렬의 크기와 동일해야 한다.
- 이에 따라 Embedding C의 결과로는 $(68, 4)$ 가 나오도록 한다.

Q. Embedding C의 하이퍼파라미터인 임베딩 벡터의 차원은?



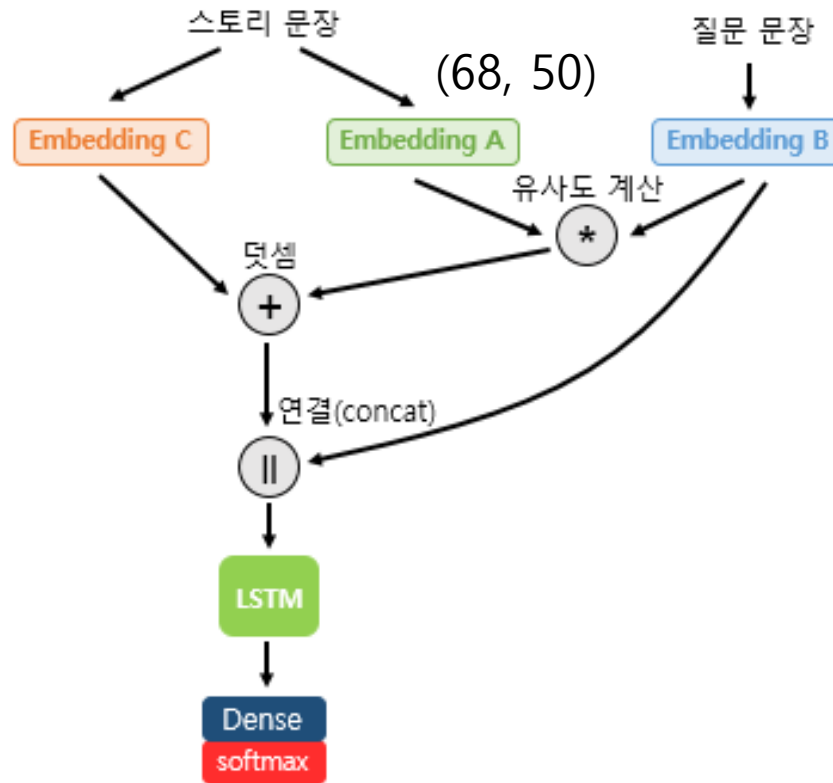
Memory Network

- 어텐션 메커니즘을 QA 문제에 사용하도록 제안.
- 질문 문장과 스토리 문장에 대해서 어텐션 메커니즘을 수행하여 스토리에서 답안을 도출.



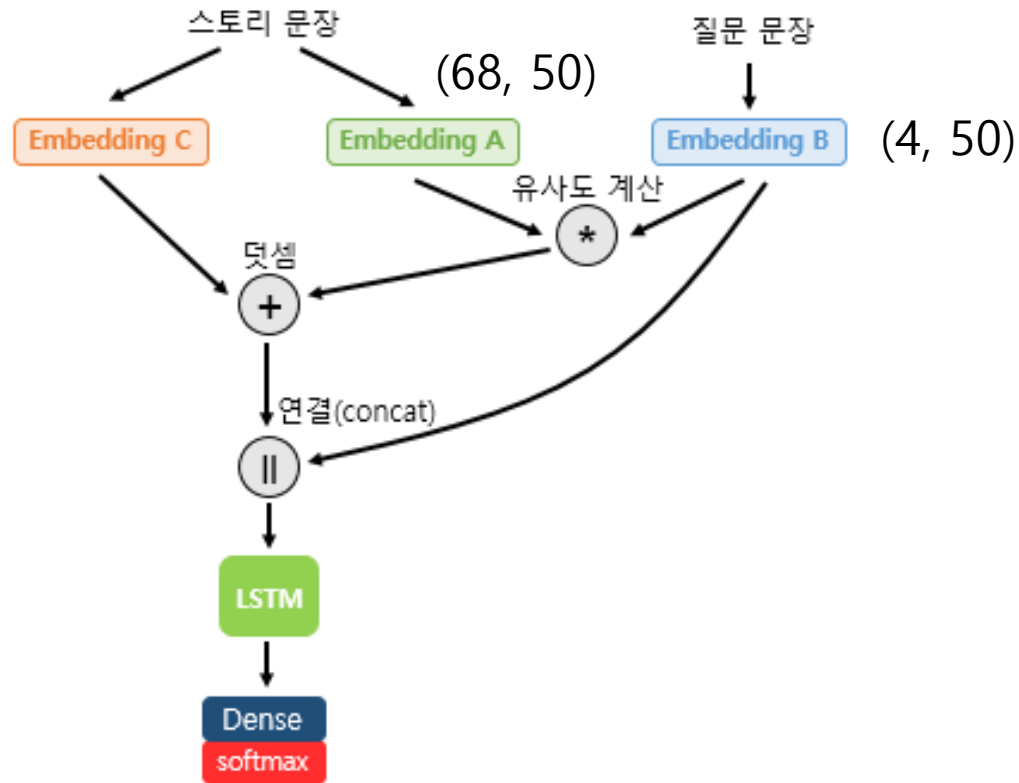
Memory Network

- 어텐션 메커니즘을 QA 문제에 사용하도록 제안.
- 질문 문장과 스토리 문장에 대해서 어텐션 메커니즘을 수행하여 스토리에서 답안을 도출.



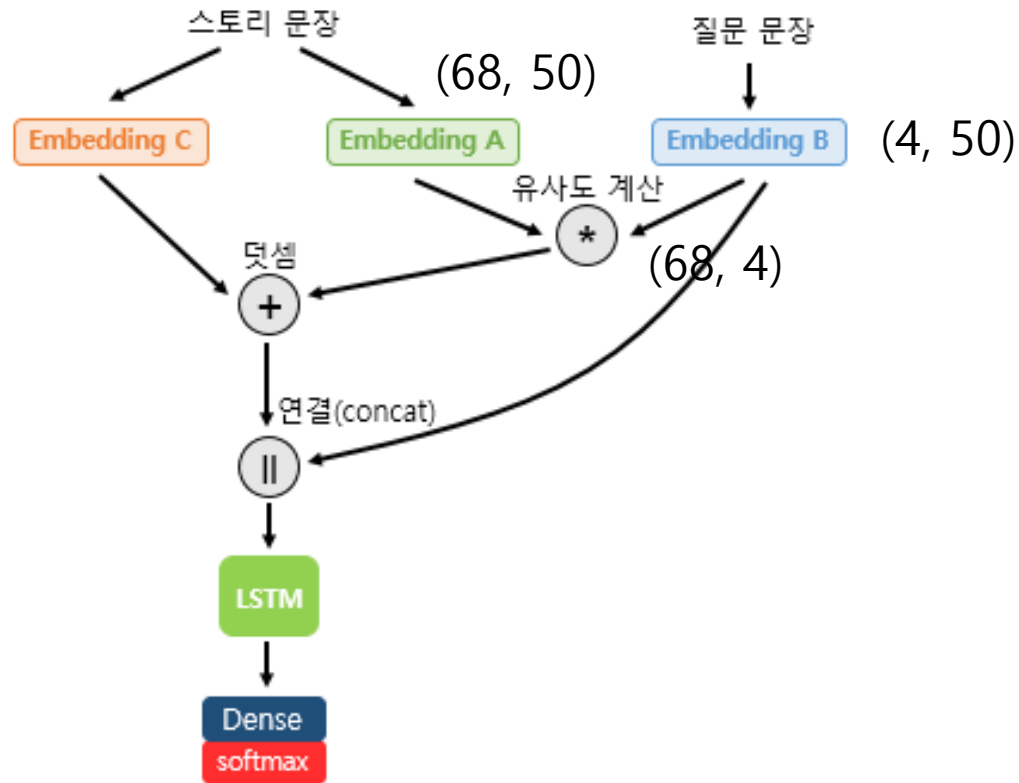
Memory Network

- 어텐션 메커니즘을 QA 문제에 사용하도록 제안.
- 질문 문장과 스토리 문장에 대해서 어텐션 메커니즘을 수행하여 스토리에서 답안을 도출.



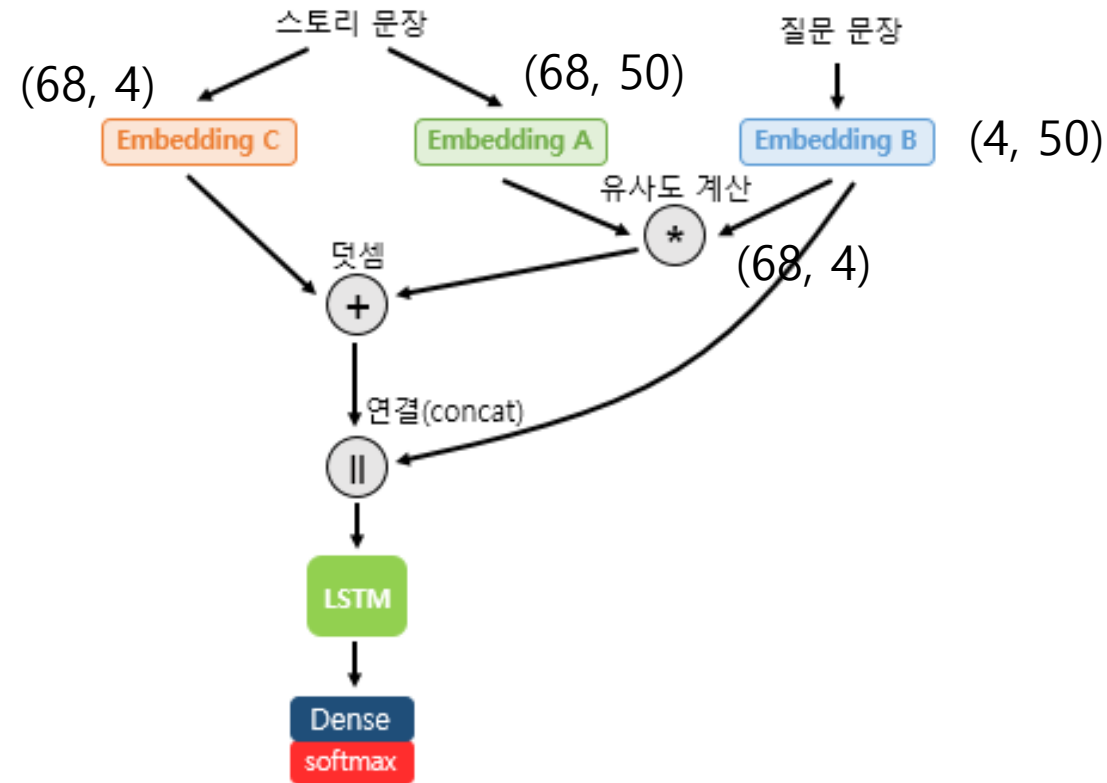
Memory Network

- 어텐션 메커니즘을 QA 문제에 사용하도록 제안.
- 질문 문장과 스토리 문장에 대해서 어텐션 메커니즘을 수행하여 스토리에서 답안을 도출.



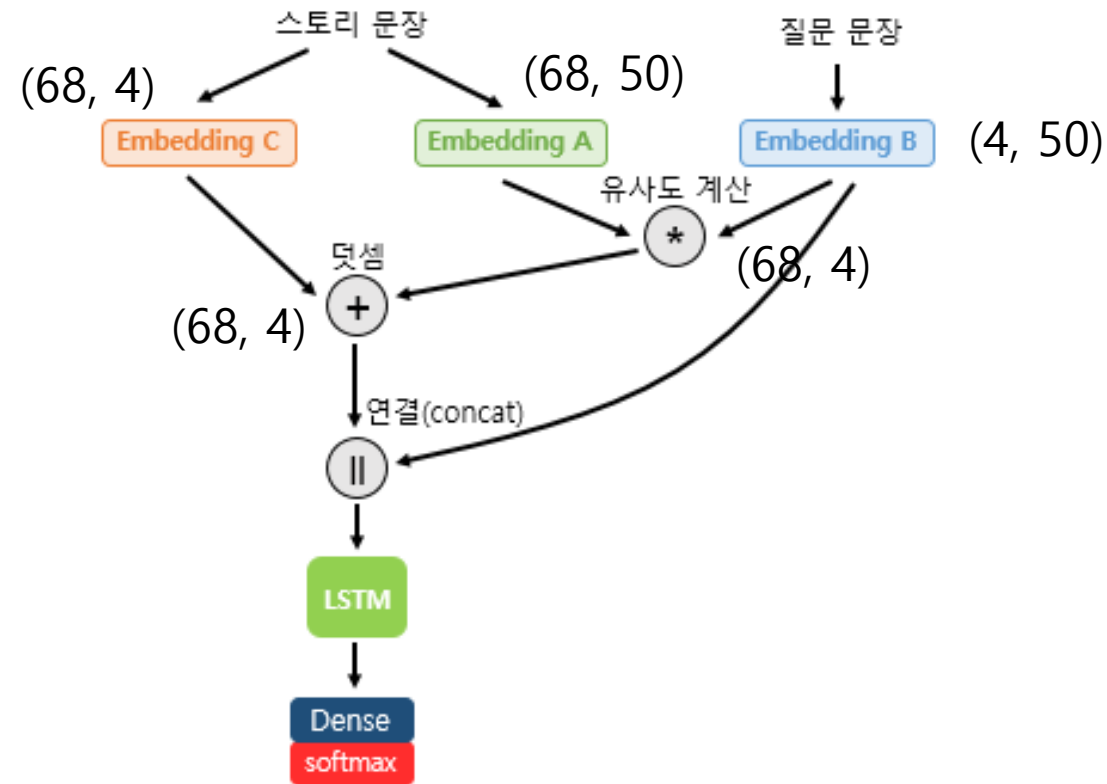
Memory Network

- 어텐션 메커니즘을 QA 문제에 사용하도록 제안.
- 질문 문장과 스토리 문장에 대해서 어텐션 메커니즘을 수행하여 스토리에서 답안을 도출.



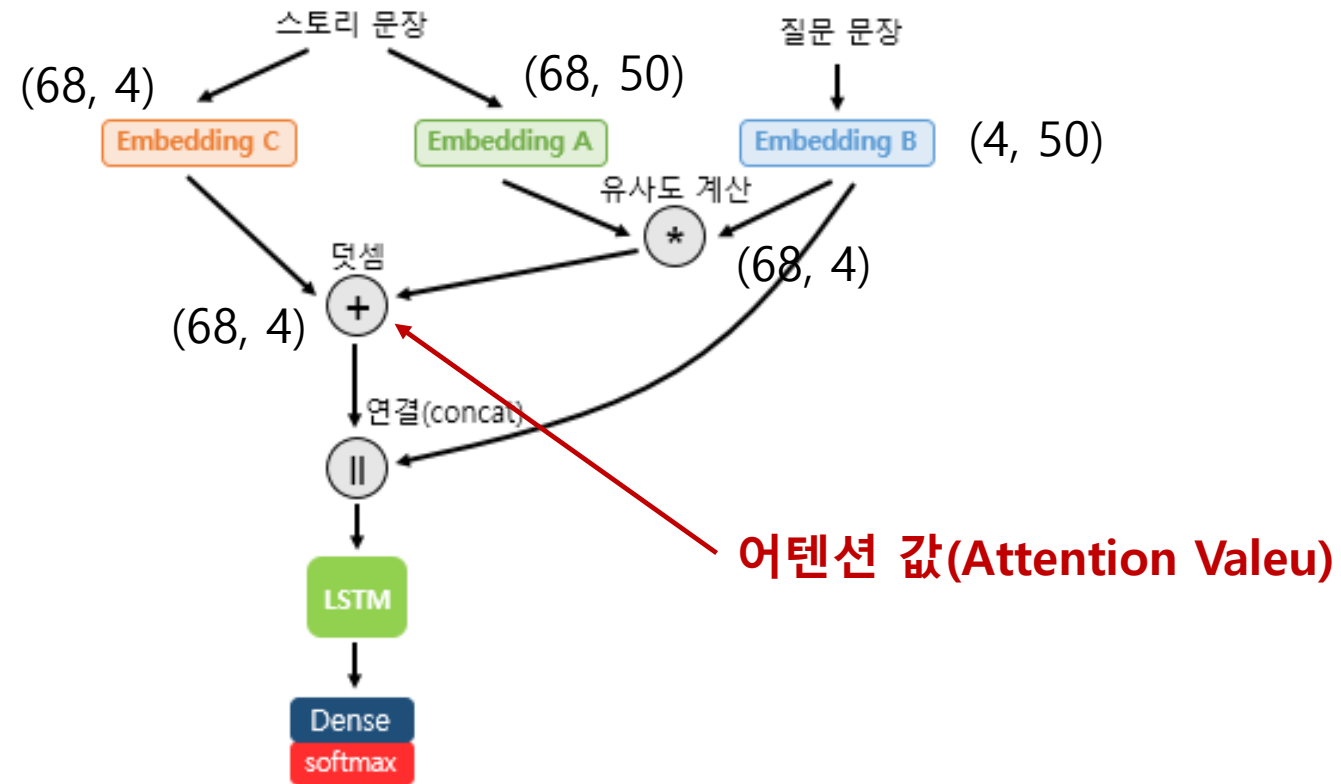
Memory Network

- 어텐션 메커니즘을 QA 문제에 사용하도록 제안.
- 질문 문장과 스토리 문장에 대해서 어텐션 메커니즘을 수행하여 스토리에서 답안을 도출.



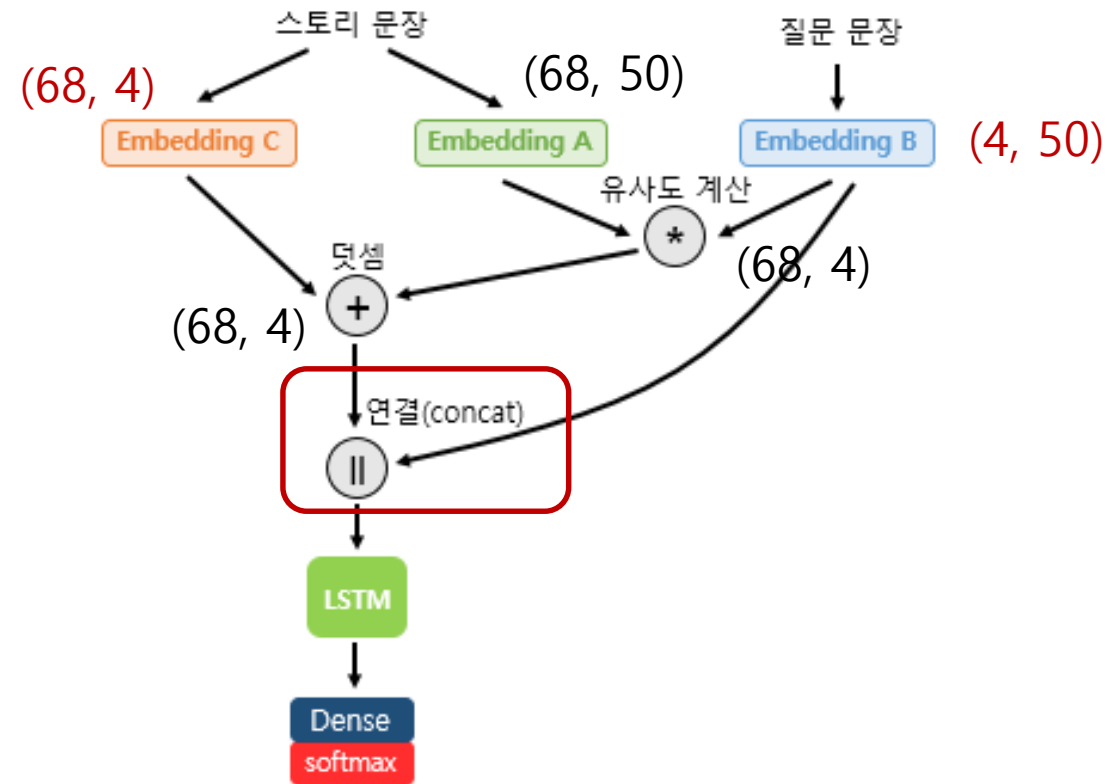
Memory Network

- 어텐션 메커니즘을 QA 문제에 사용하도록 제안.
- 질문 문장과 스토리 문장에 대해서 어텐션 메커니즘을 수행하여 스토리에서 답안을 도출.



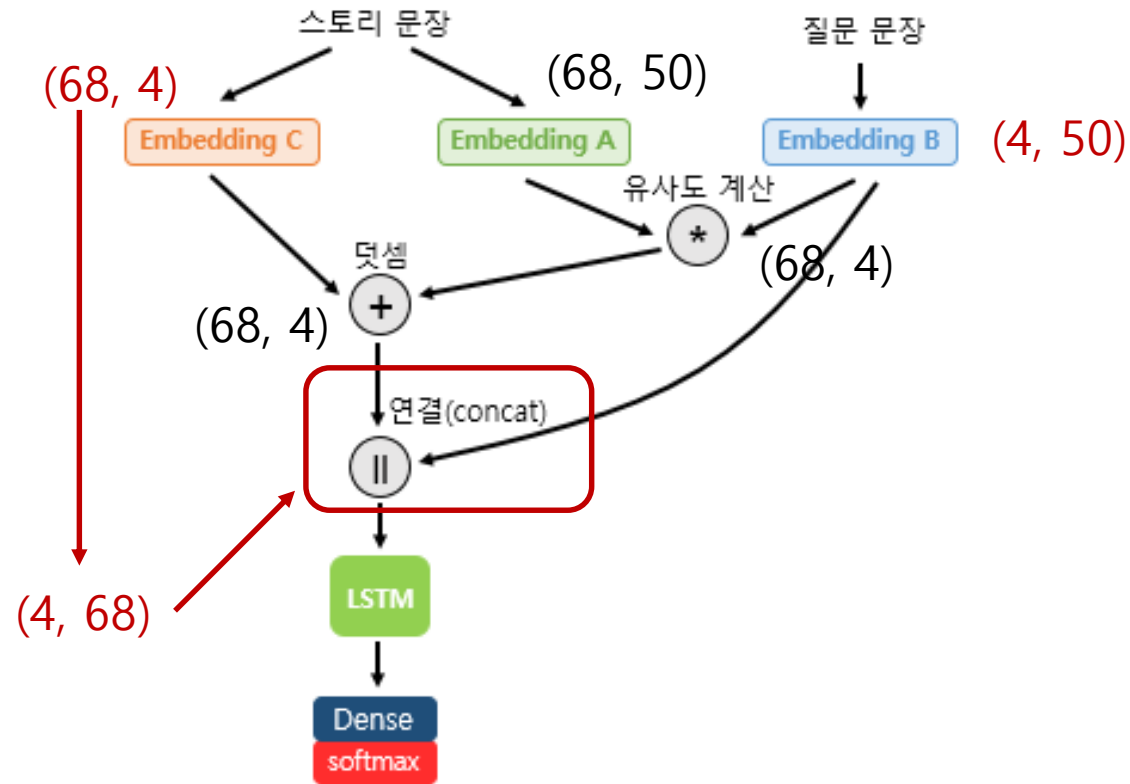
Memory Network

- 어텐션 메커니즘을 QA 문제에 사용하도록 제안.
- 질문 문장과 스토리 문장에 대해서 어텐션 메커니즘을 수행하여 스토리에서 답안을 도출.



Memory Network

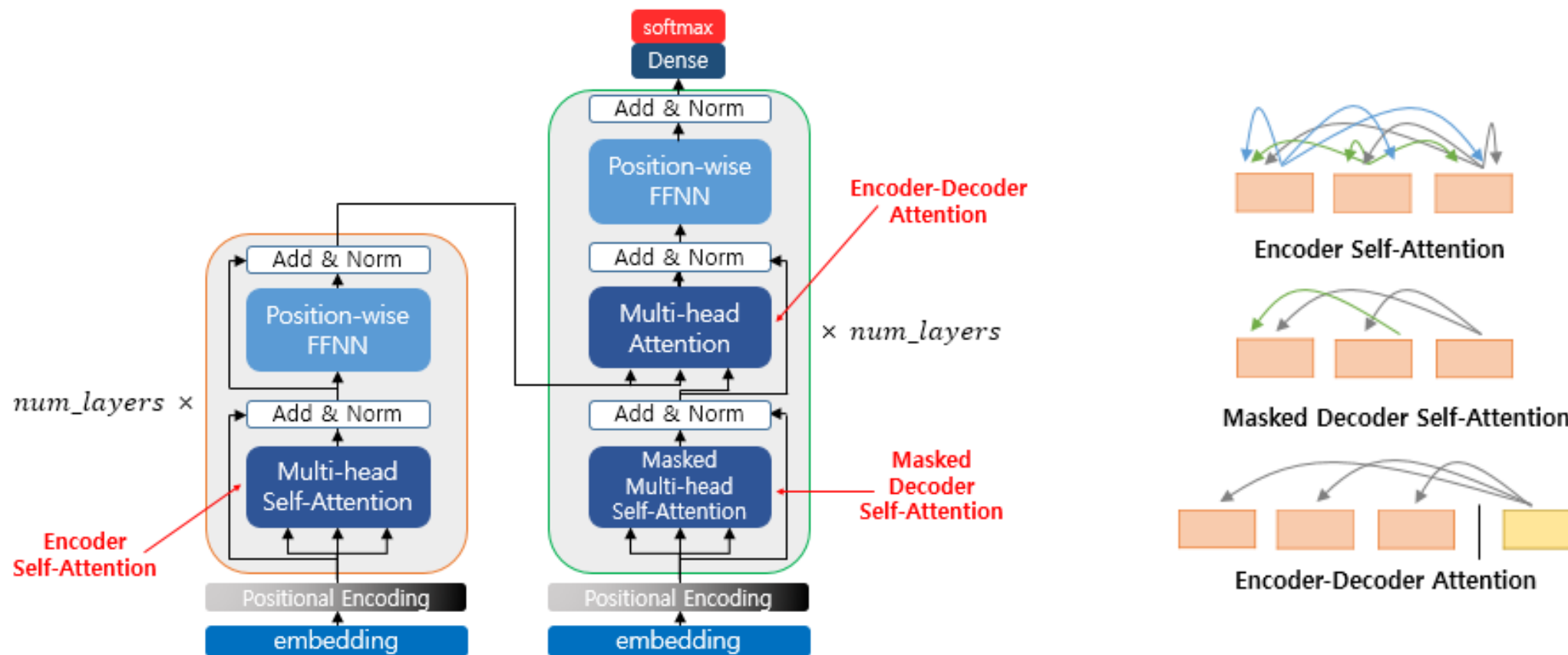
- 어텐션 메커니즘을 QA 문제에 사용하도록 제안.
- 질문 문장과 스토리 문장에 대해서 어텐션 메커니즘을 수행하여 스토리에서 답안을 도출.



Self Attention

Transformer Encoder

- 트랜스포머는 총 세 종류의 어텐션이 존재. 그 중 인코더는 셀프 어텐션을 수행.



Transformer Encoder

- seq2seq + 어텐션의 경우, Query, Key, Value는 다음과 같음

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

Transformer Encoder

- seq2seq + 어텐션의 경우,
디코더의 t시점이라는 것은 계속 변화하면서 반복되므로 다음과 같이 일반화가 가능.

Q = Querys : 모든 시점의 디코더 셀에서의 은닉 상태들
K = Keys : 모든 시점의 인코더 셀의 은닉 상태들
V = Values : 모든 시점의 인코더 셀의 은닉 상태들

Transformer Encoder

- seq2seq + 어텐션의 경우,
디코더의 t시점이라는 것은 계속 변화하면서 반복되므로 다음과 같이 일반화가 가능.

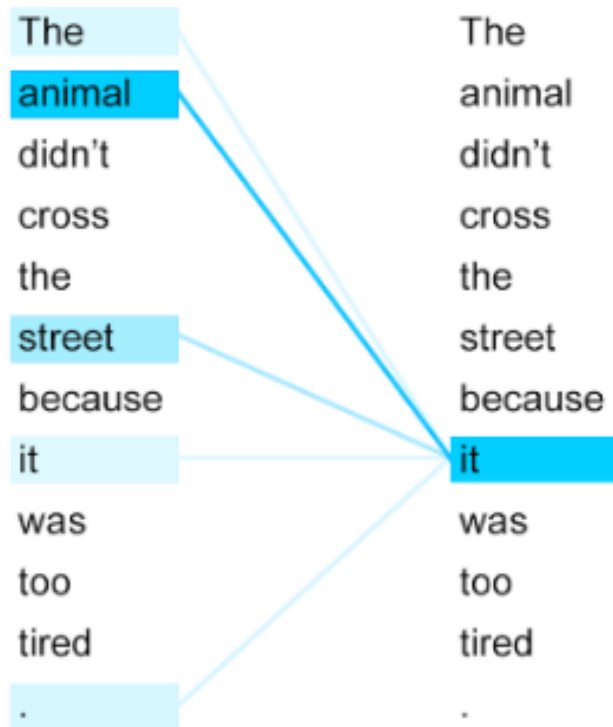
Q = Querys : 모든 시점의 디코더 셀에서의 은닉 상태들
K = Keys : 모든 시점의 인코더 셀의 은닉 상태들
V = Values : 모든 시점의 인코더 셀의 은닉 상태들

- 셀프 어텐션의 경우 Querys = Keys = Values가 모두 동일.

Q : 입력 문장의 모든 단어 벡터들
K : 입력 문장의 모든 단어 벡터들
V : 입력 문장의 모든 단어 벡터들

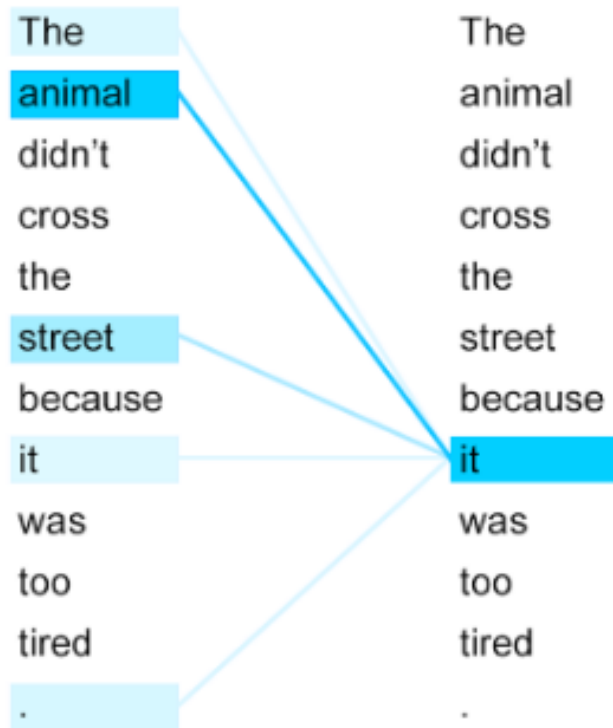
Self-Attention

- '그 동물은 길을 건너지 않았다. 왜냐하면 그것은 너무 피곤하였기 때문이다.' 라는 의미
- 여기서 그것(it)에 해당하는 것은 과연 길(street)일까, 동물(animal)일까?



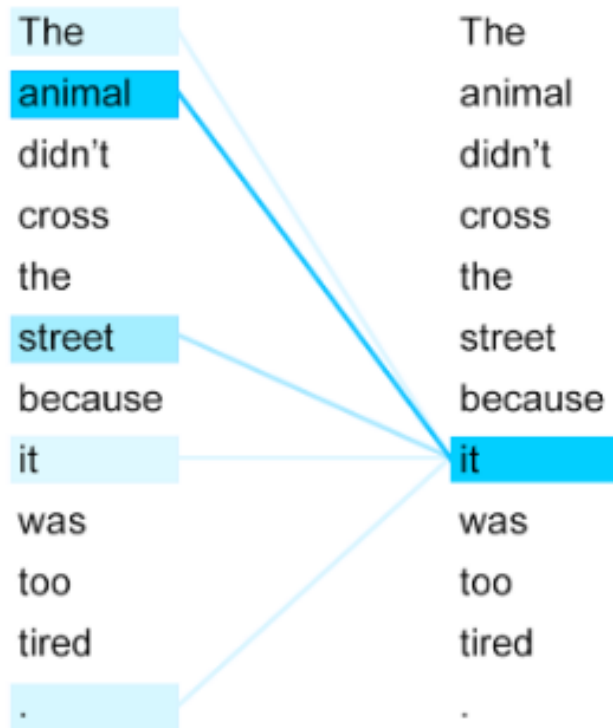
Self-Attention

- '그 동물은 길을 건너지 않았다. 왜냐하면 그것은 너무 피곤하였기 때문이다.' 라는 의미
- 여기서 그것(it)에 해당하는 것은 과연 길(street)일까, 동물(animal)일까?
- 우리에게서는 굉장히 쉬운 문제지만 기계에게는 그렇지 않음.



Self-Attention

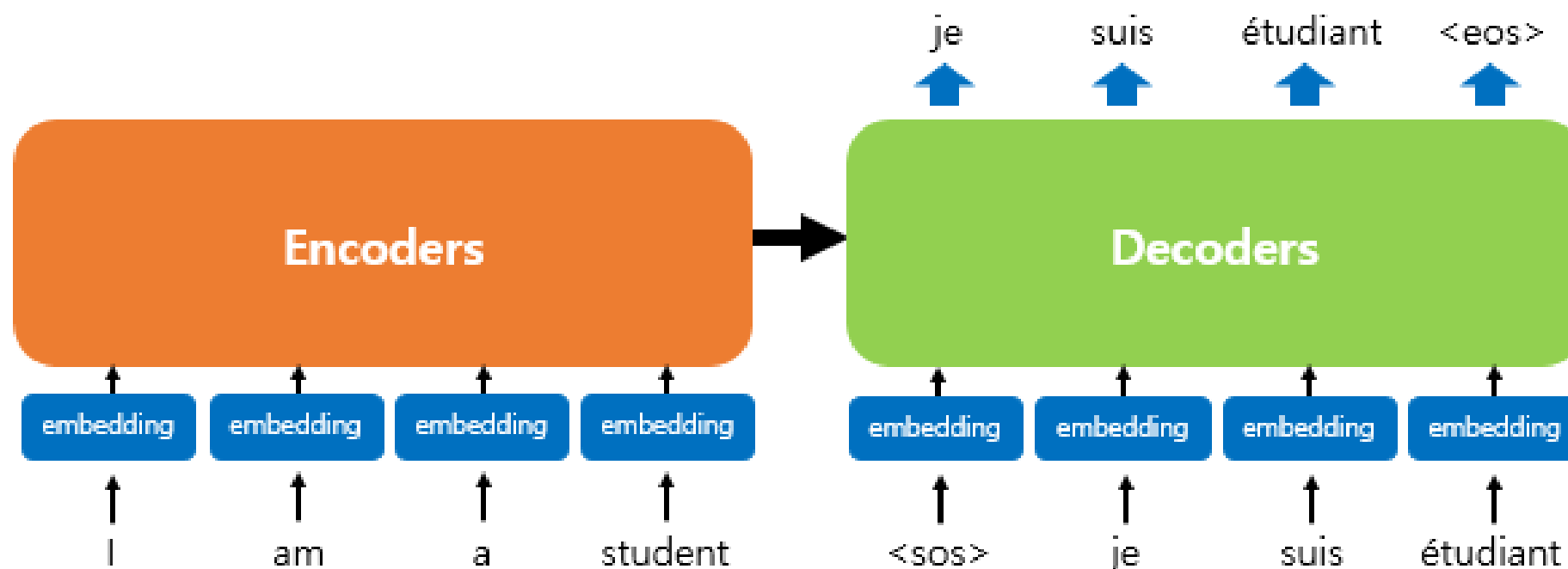
- '그 동물은 길을 건너지 않았다. 왜냐하면 그것은 너무 피곤하였기 때문이다.' 라는 의미
- 여기서 그것(it)에 해당하는 것은 과연 길(street)일까, 동물(animal)일까?
- 우리에게는 굉장히 쉬운 문제지만 기계에게는 그렇지 않음.



셀프 어텐션은 입력 문장 내의 단어들끼리 유사도를 구하므로서 그것(it)이 동물(animal)과 연관되었을 확률이 높다는 것을 찾아낸다.

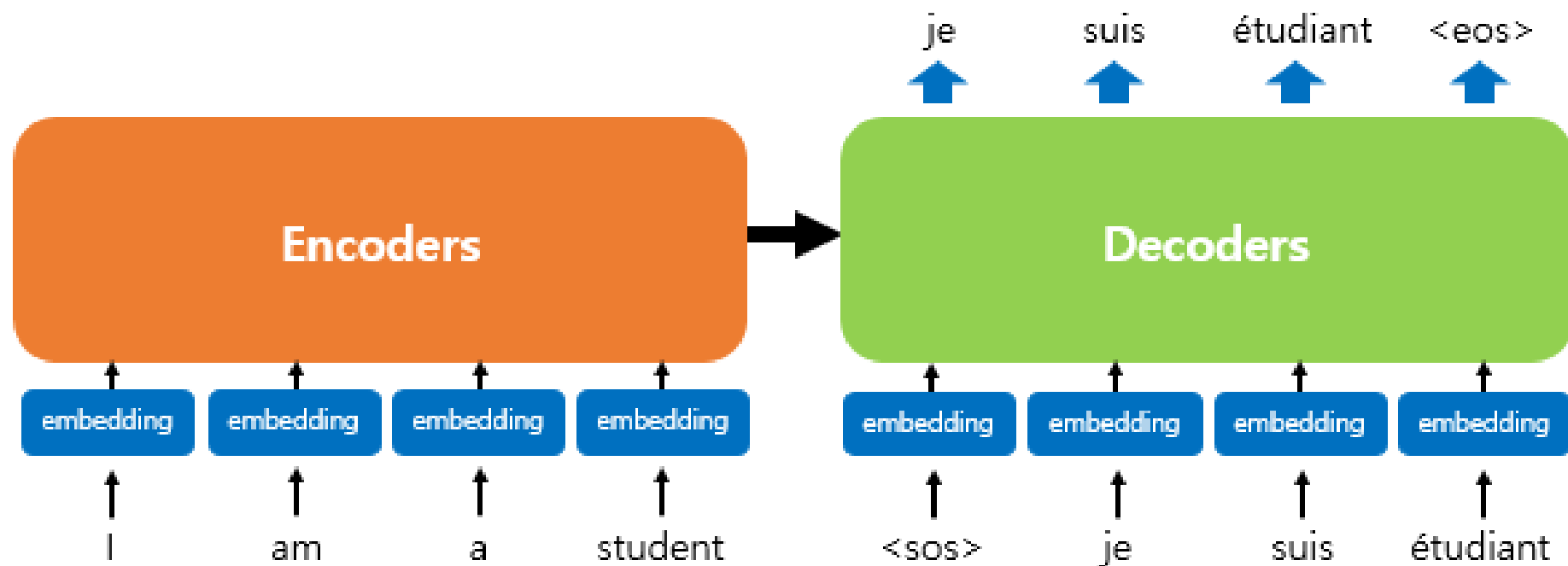
Self-Attention

- 트랜스포머는 기본적으로 인코더-디코더 구조를 가지는 모델.



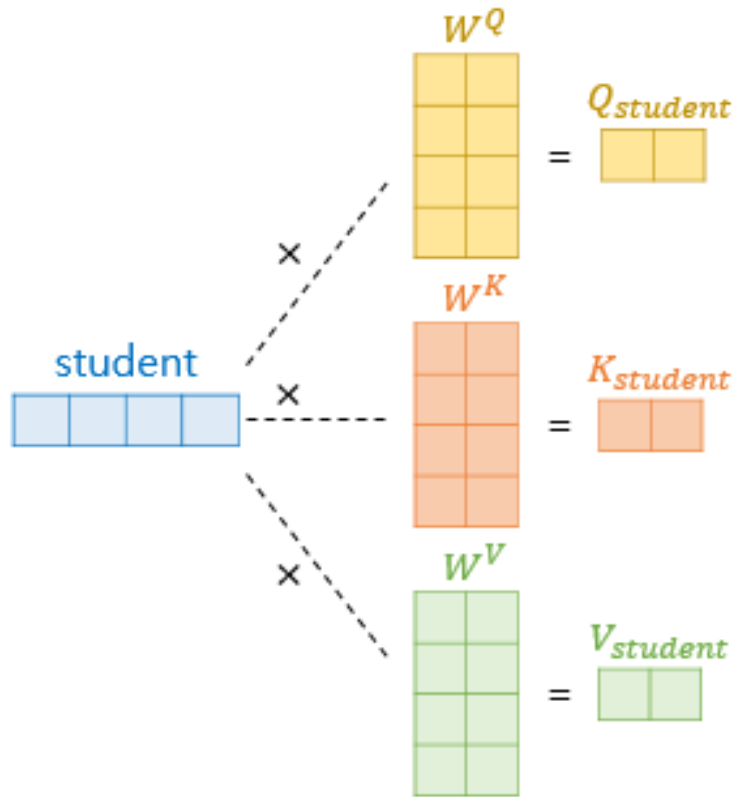
Self-Attention

- 트랜스포머는 기본적으로 인코더-디코더 구조를 가지는 모델.
- 인코더의 입력으로 'I', 'am', 'a', 'student'가 들어왔다고 가정.



Self-Attention

- 셀프 어텐션은 입력 벡터에서 가중치 행렬 곱으로 Q, K, V 벡터를 얻고 수행.
- 스케일드 닷 프로덕트 어텐션을 통해 각각의 Q벡터가 각각의 K벡터에 대해서 스코어 연산.



Scaled dot product Attention : $score\ function(q, k) = q \cdot k / \sqrt{n}$

The diagram shows the calculation of attention scores for a specific Q vector, Q_I , against four different K vectors. The results are shown as a 2x2 grid of scores, which are then summed to get the final attention score.

Q Vector	K Vector	Dot Product	Scaled Dot Product
Q_I	K_I^T	128	$128 / \sqrt{d_k} = 16$
Q_I	K_{am}^T	32	$32 / \sqrt{d_k} = 4$
Q_I	K_a^T	32	$32 / \sqrt{d_k} = 4$
Q_I	$K_{student}^T$	128	$128 / \sqrt{d_k} = 16$

The final attention score is the sum of the scaled dot products: $16 + 4 + 4 + 16 = 40$.

어텐션 메커니즘의 종류

어텐션 메커니즘은 다음 순서로 제안되었다. 이들의 큰 차이는 어텐션 스코어 함수가 다르다는 점이다.

1. 바다나우 어텐션 (Neural Machine Translation by Jointly Learning to Align and Translate)

- 콘캣 어텐션(Concat Attention) : $score(s_t, h_i) = W_a^T \tanh(W_b[s_t; h_i])$

2. 루옹 어텐션 (Effective Approaches to Attention-based Neural Machine Translation)

- 닷 프로덕트 어텐션(Dot-Product Attention) : $score(s_t, h_i) = s_t^T h_i$
- 제네럴 어텐션(General Attention) : $score(s_t, h_i) = s_t^T W_a h_i$

3. 스케일드 닷 프로덕트 어텐션(Attention is All you need)

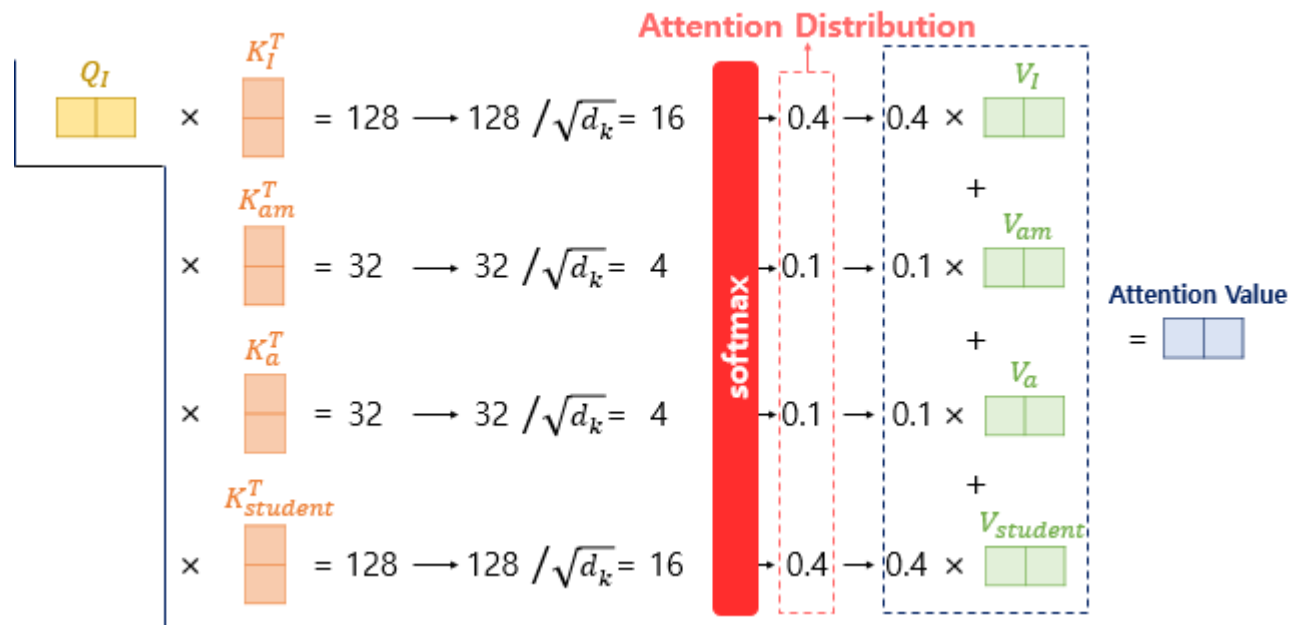
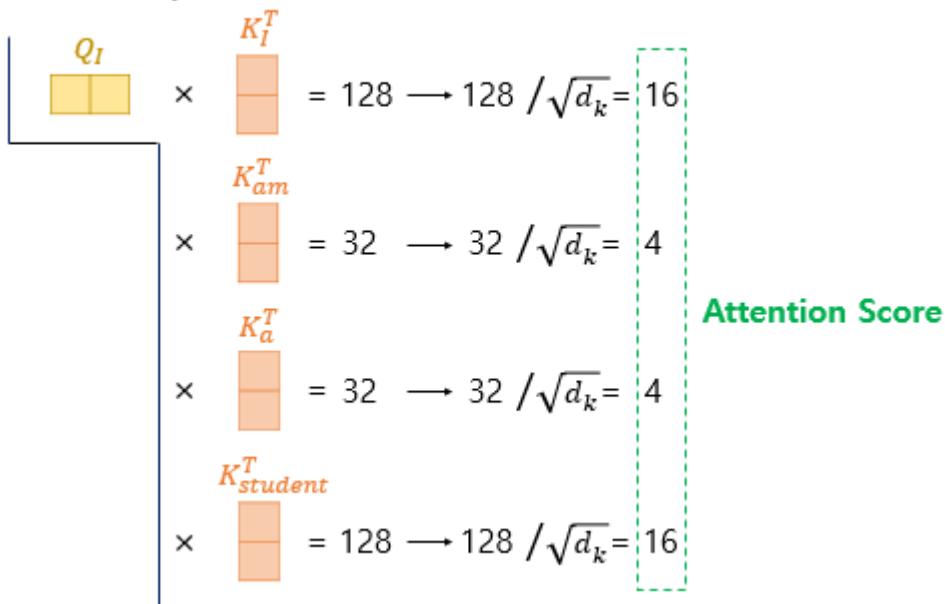
- **스케일드 닷 프로덕트 어텐션(Scaled Dot-Product Attention)** : $score(s_t, h_i) = \frac{s_t^T h_i}{\sqrt{n}}$

트랜스포머 논문에서 등장

Self-Attention

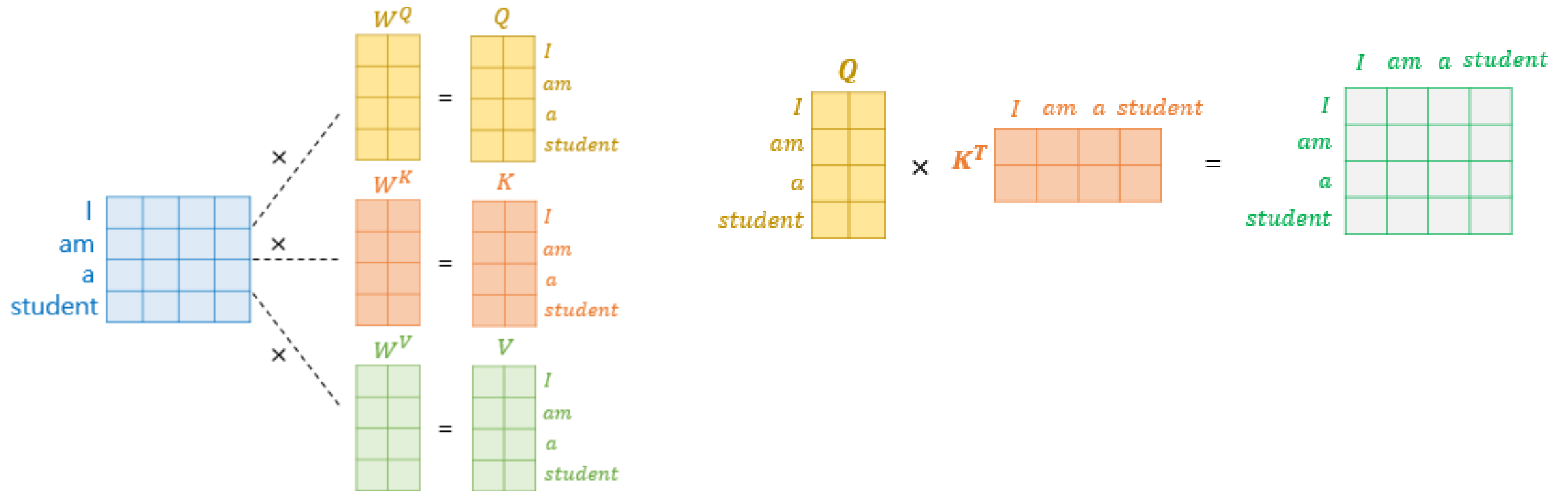
- 셀프 어텐션은 입력 벡터에서 가중치 행렬 곱으로 Q, K, V 벡터를 얻고 수행.
- 스케일드 닷 프로덕트 어텐션을 통해 각각의 Q벡터가 각각의 K벡터에 대해서 스코어 연산.

Scaled dot product Attention : $score\ function(q, k) = q \cdot k / \sqrt{n}$



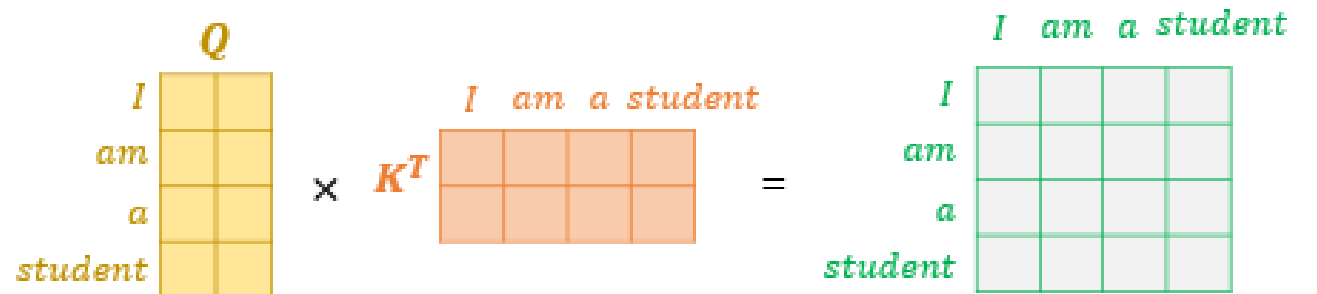
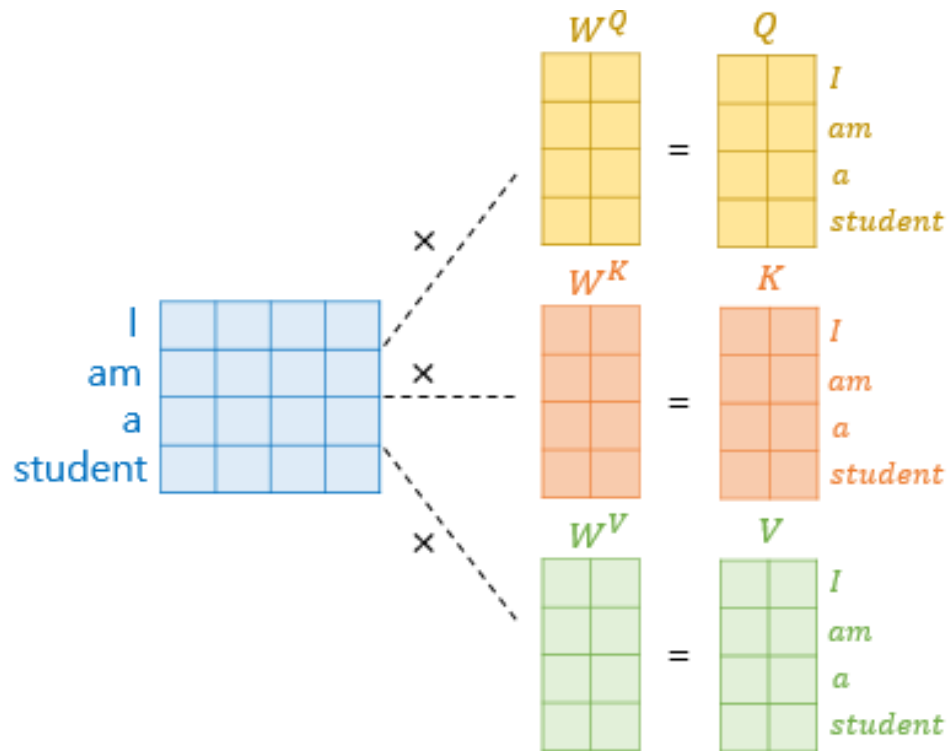
행렬 연산으로 이해하기

- 실제로 이 연산은 벡터 간 연산이 아니라 행렬 연산으로 이루어진다.



행렬 연산으로 이해하기

- 실제로 이 연산은 벡터 간 연산이 아니라 행렬 연산으로 이루어진다.



위 행렬에 특정 값을 나누어주면 어텐션 스코어 행렬.

행렬 연산으로 이해하기

- 실제로 이 연산은 벡터 간 연산이 아니라 행렬 연산으로 이루어진다.

