Transformer & BERT

최 석 재

lingua@naver.com

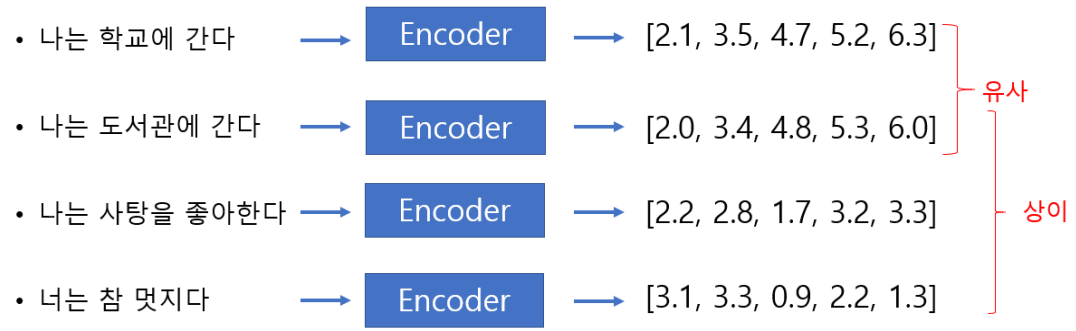
Attention

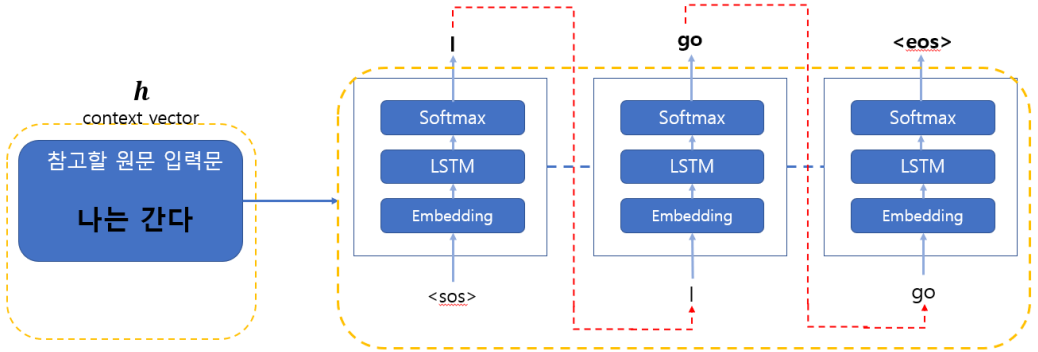
Seq2seq의 한계

- Seq2seq는 번역을 단어 대 단어가 아니라 유사한 문장을 찾아 자동 생성하는 방법을 취함으로써 기존 자동번역의 한계를 뛰어넘었다
  - 기존에는 번역할 원어의 문법적 구조를 파악하고, 각 문법 단위를 규칙 또는 통계로 가장 적절한 단어 또는 표현으로 대체하는 방법이었다

① 그러나 고정된 길이의 벡터를 사용하므로 문장이 길어질수록 문장의 차이를 구별하기가 어려워진다

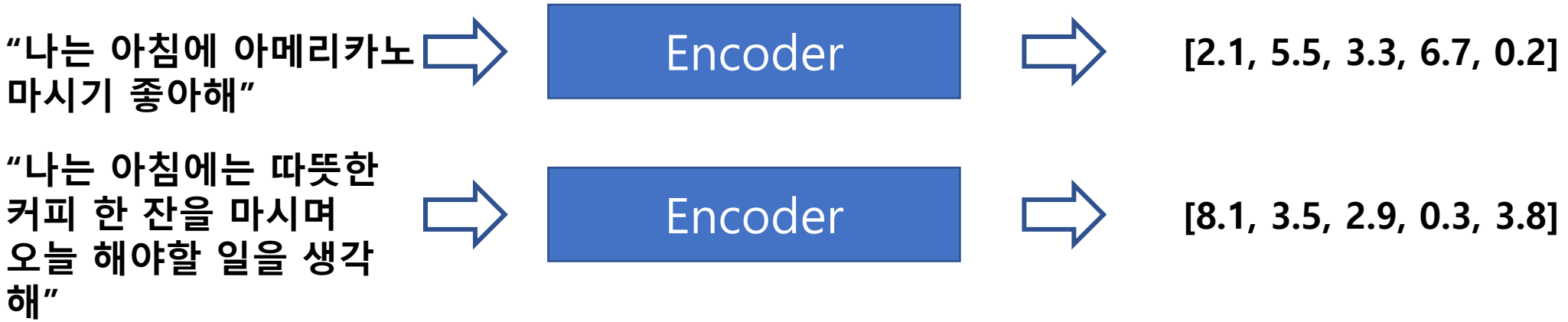
② 순환신경망은 문장이 길어질수록 그래디언트 소실 문제가 발생한다





Encoder 개선

- 첫 번째 개선 사항은 고정 길이 벡터를 벗어나는 것
- 기존에는 다음의 두 문장이 모두 동일한 길이(maxlen)을 가졌다

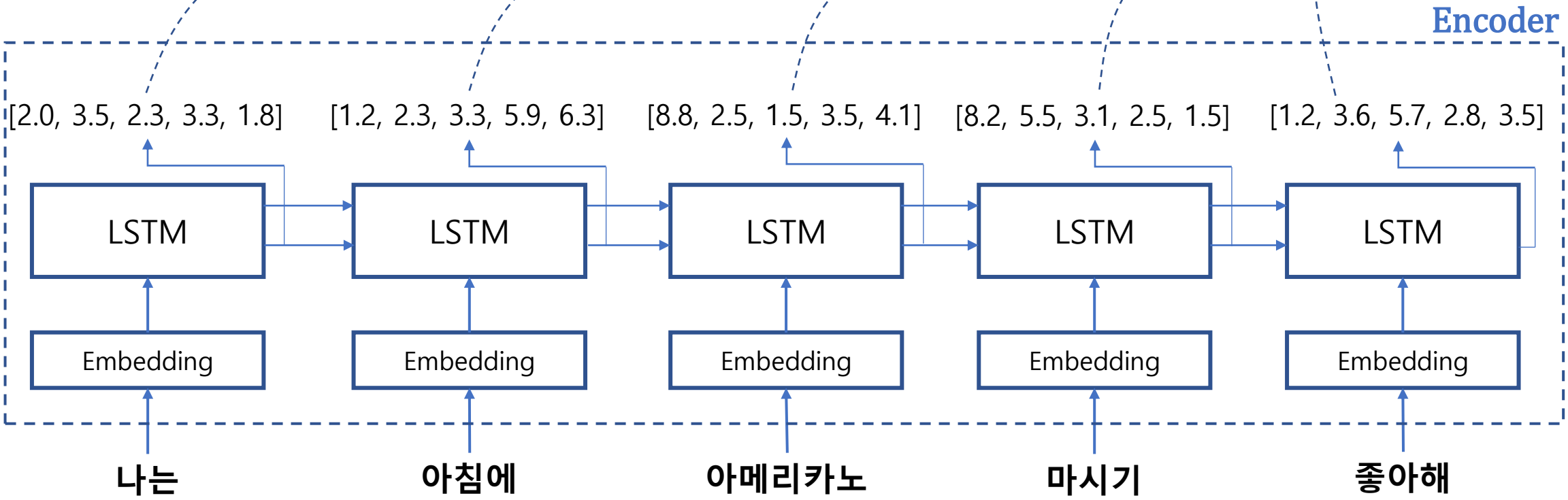


Encoder 개선

- 전체 문장에 대한 고정 길이 벡터를 갖는 것이 아니라,
- 단어별 벡터(Time Step별 벡터)를 생성하고 이를 적층하는 것
- 이로써 Encoder는 '하나의 고정 길이 벡터'라는 제약을 넘어선다
- 또한 텍스트 길이에 따른 순환신경망의 기울기 소실 문제도 해결된다

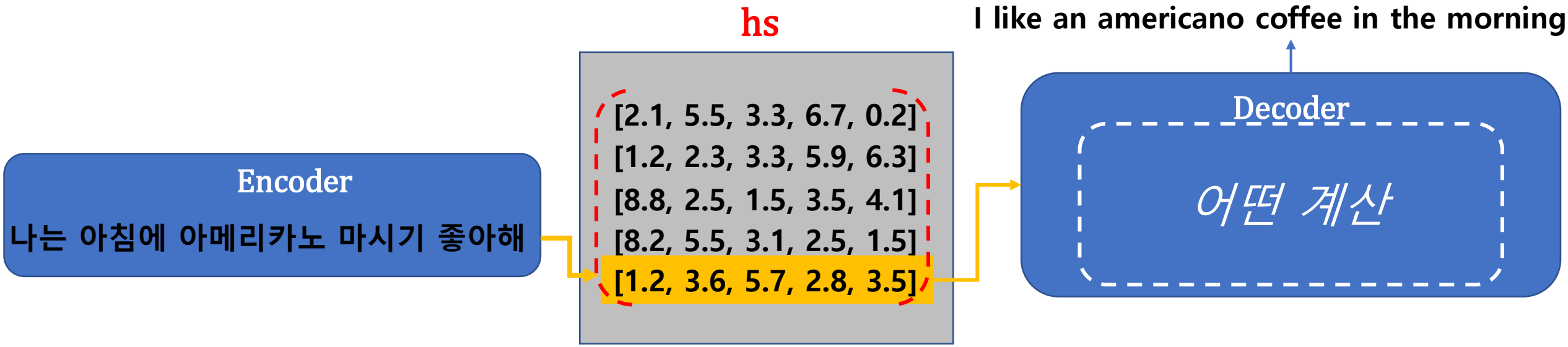
① [2.1, 5.5, 3.3, 6.7, 0.2]
② [1.2, 2.3, 3.3, 5.9, 6.3]
③ [8.8, 2.5, 1.5, 3.5, 4.1]
④ [8.2, 5.5, 3.1, 2.5, 1.5]
⑤ [1.2, 3.6, 5.7, 2.8, 3.5]

hs



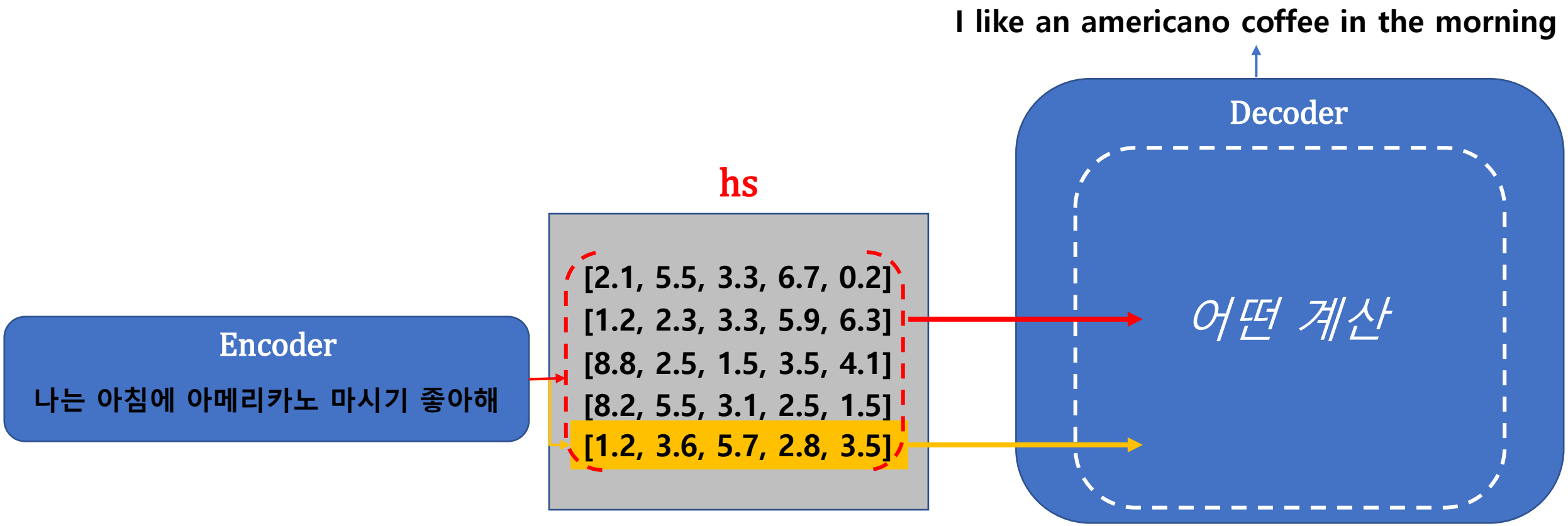
Seq2Seq와의 비교

- 일반 Seq2Seq에 비하여 context vector가 hs 행렬로 바뀐 것으로 볼 수 있다
- Attention은 각 Time Step의 누적 연산 결과인 마지막 은닉 상태만을 이용하는 것이 아닌,
- 각 Time Step의 모든 결과를 이용한다
- 즉, Attention 관점에서는 Seq2Seq는 hs 행렬의 마지막만 이용한 것으로 볼 수 있다
  - 그러나 이 방식은 "나는 = I", "아메리카노 = americano coffee", "좋아해 = like"의 관계를 파악하지 못한다

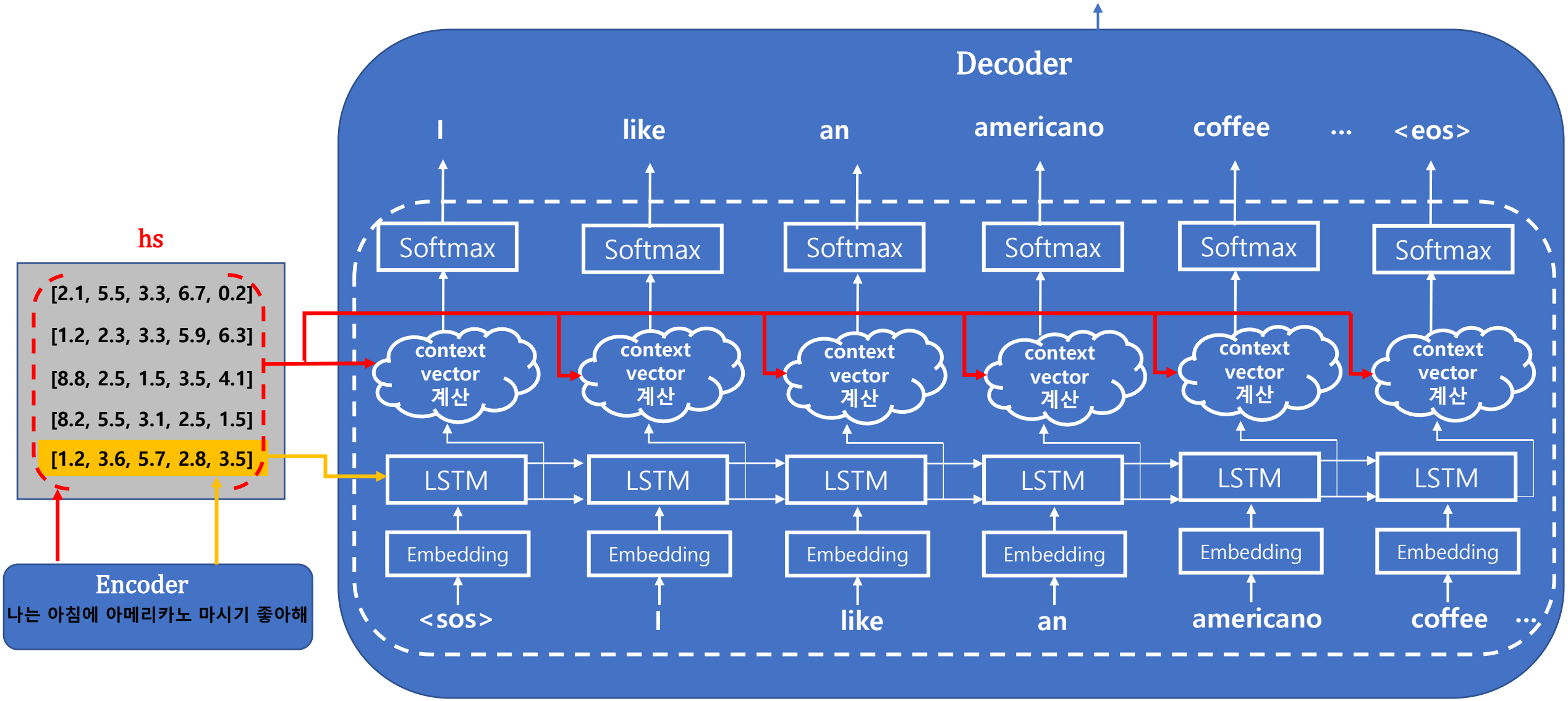


Decoder 개선

- Attention은 행렬의 마지막은 물론, 전부를 이용한다

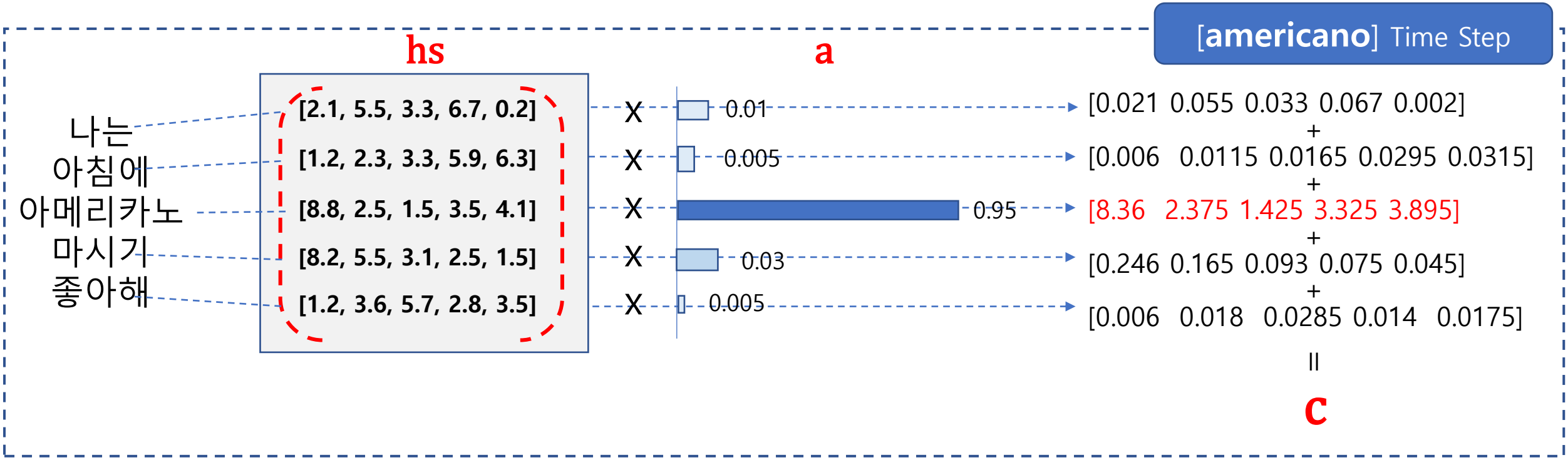


Decoder 기본 구조

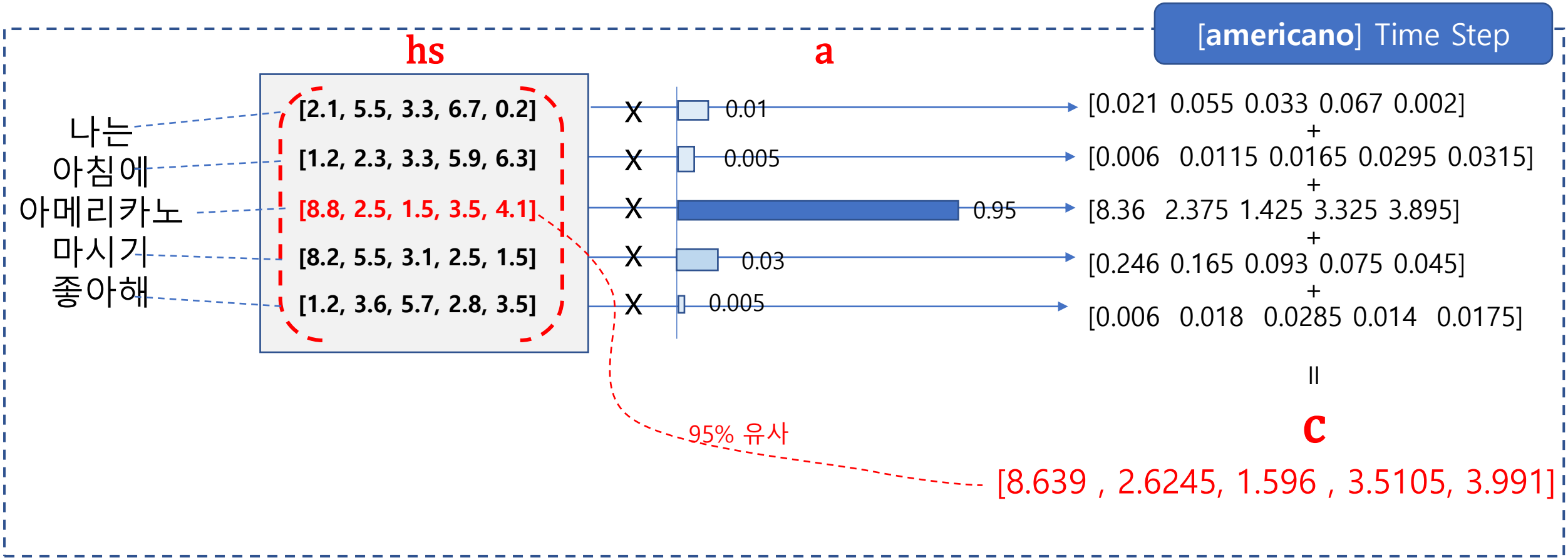


주목할 벡터 찾기 – americano 타임스텝

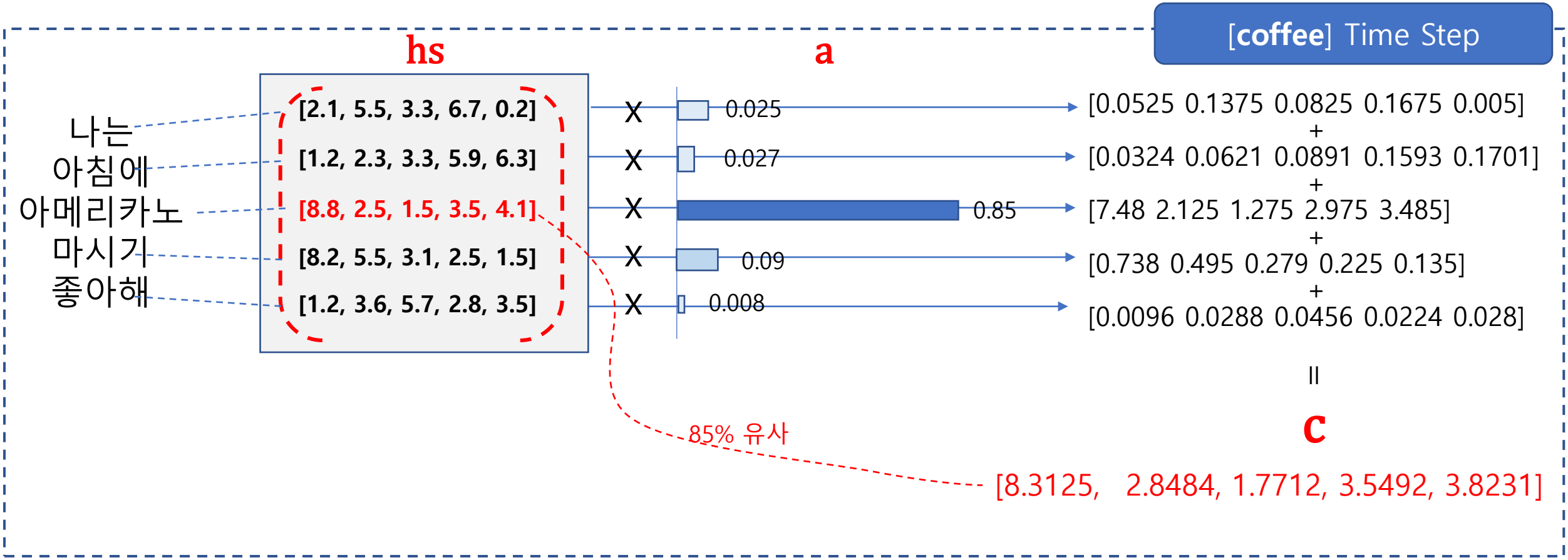
- Attention은 필요한 정보에만 주목하여 시계열 변환을 수행한다
- *hs* 행렬에서 각 Time Step의 단어와 관련된 벡터를 찾음
- 즉, "아메리카노 = americano coffee"의 관계를 찾겠다는 것
- *hs* 행렬과 가중치 벡터 *a*의 가중합으로 context vector *c*를 얻음



주목할 벡터 찾기 – americano 타임스텝



주목할 벡터 찾기 - coffee 타임스텝

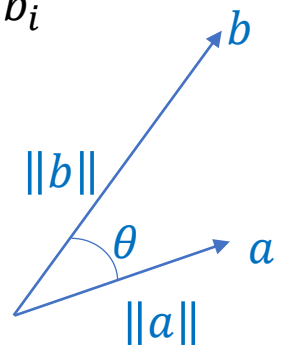


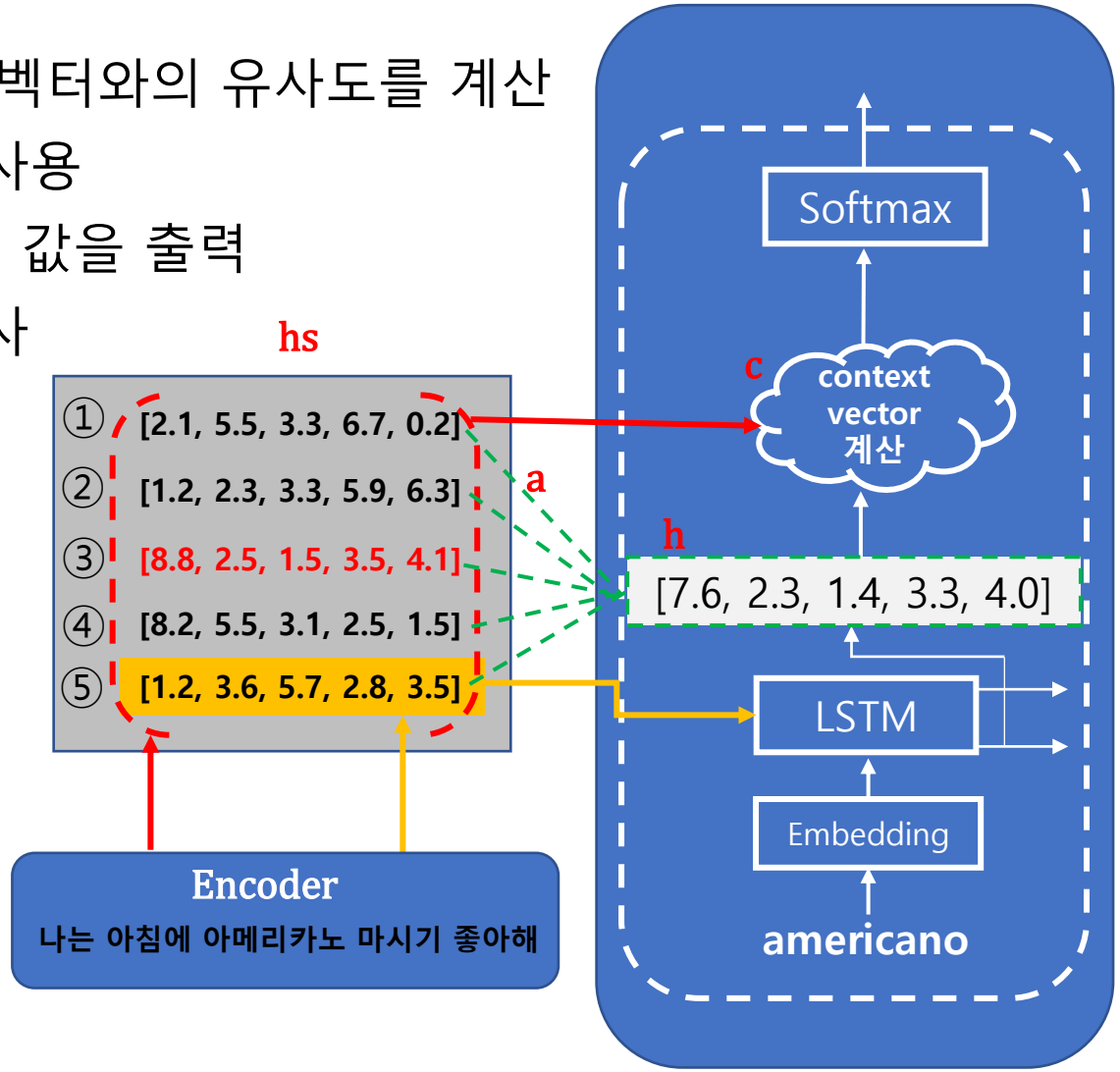
가중치 구하기

- 가중치 벡터 *a*를 구하기 위하여 *h*와 *hs* 행렬 각 벡터와의 유사도를 계산
- 유사도를 계산하는 데는 내적(dot product) 연산을 사용
- 내적 연산은 두 벡터가 같은 방향을 향하면 높은 값을 출력
- '아메리카노'와 'americano'는 벡터의 방향이 유사

$$\mathrm{a} \cdot \mathrm{b} = a_1 b_1 + a_2 b_2 + \cdots = \sum_{i=1}^{n} a_i b_i$$

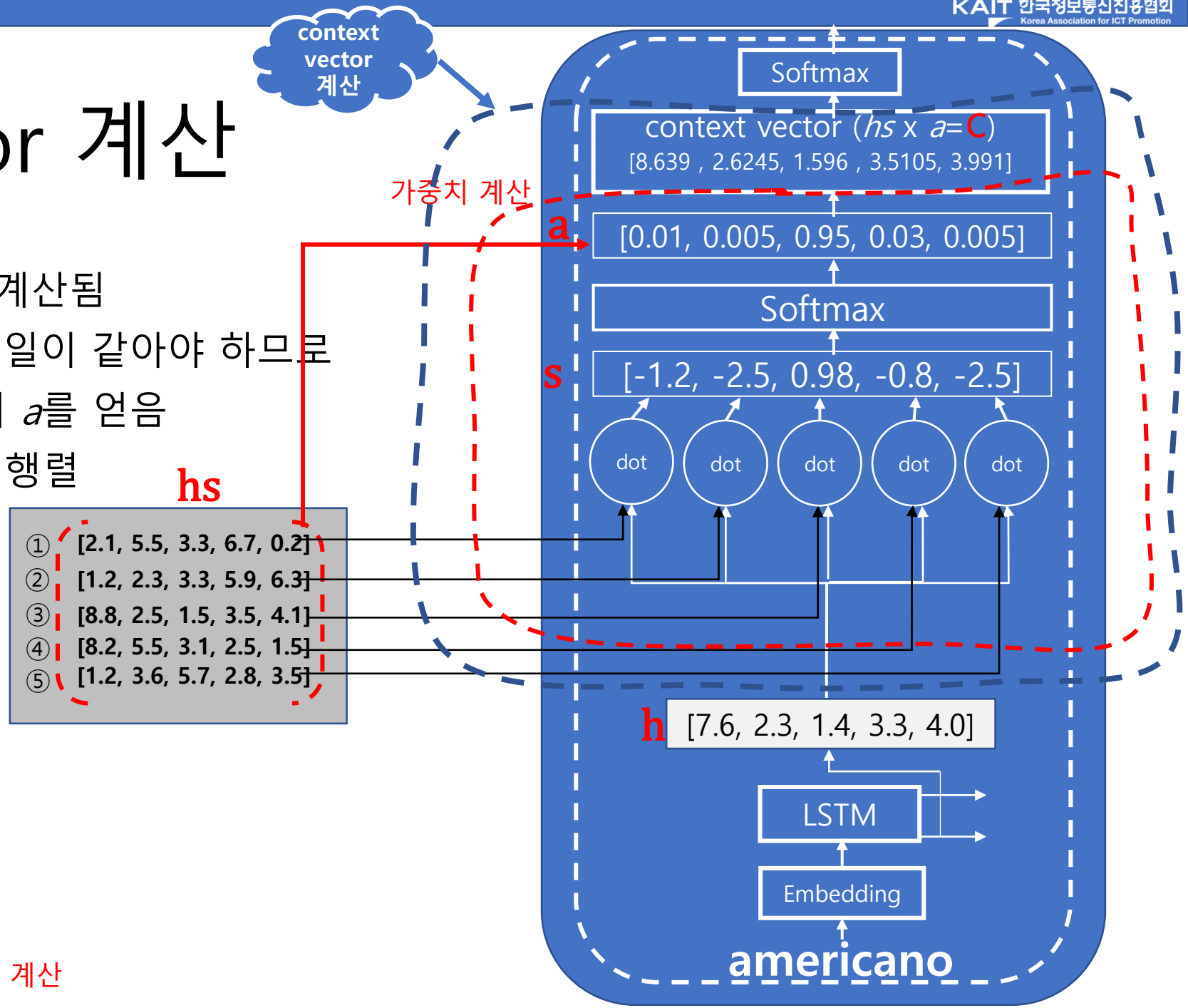
$$\mathrm{a} \cdot \mathrm{b} = \|a\| \|b\| \cos\theta$$





context vector 계산

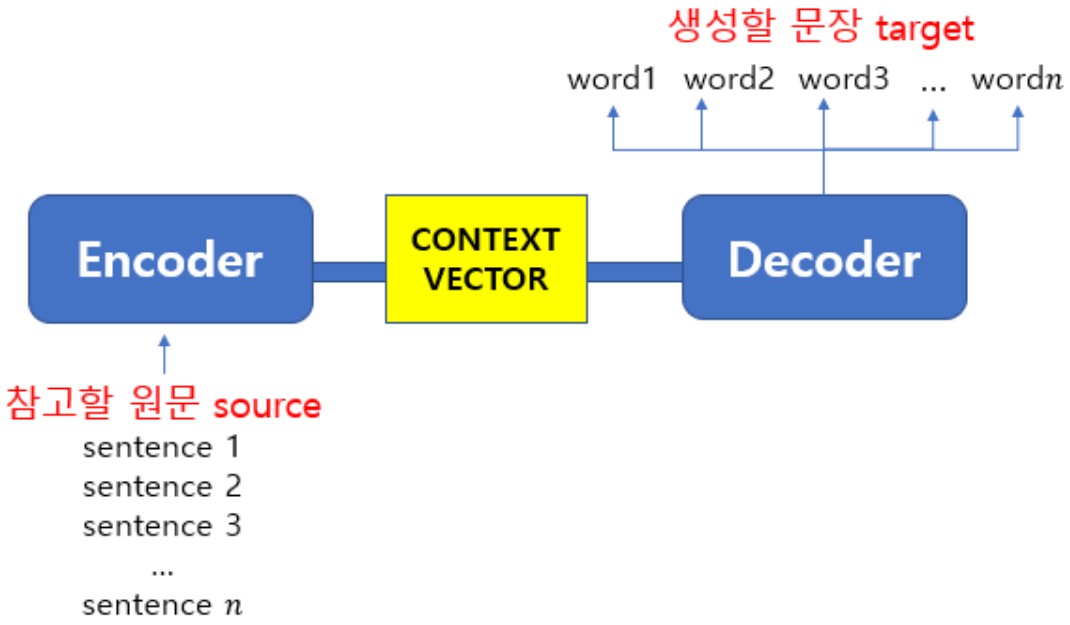
- 내적(dot) 결과로 *s* 벡터가 계산됨
- Time Step마다 가중치 스케일이 같아야 하므로
- Softmax 함수로 정규화하여 *a*를 얻음
- 계산된 가중치 *a*는 다시 *hs* 행렬과 곱해져 context vector *c*를 계산한다

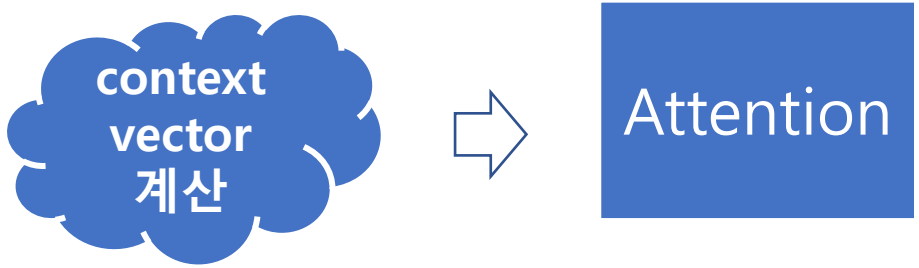


※ Softmax의 결과는 0~1 사이이다
※ 붉은 점선으로 표시된 부분이 가중치 계산

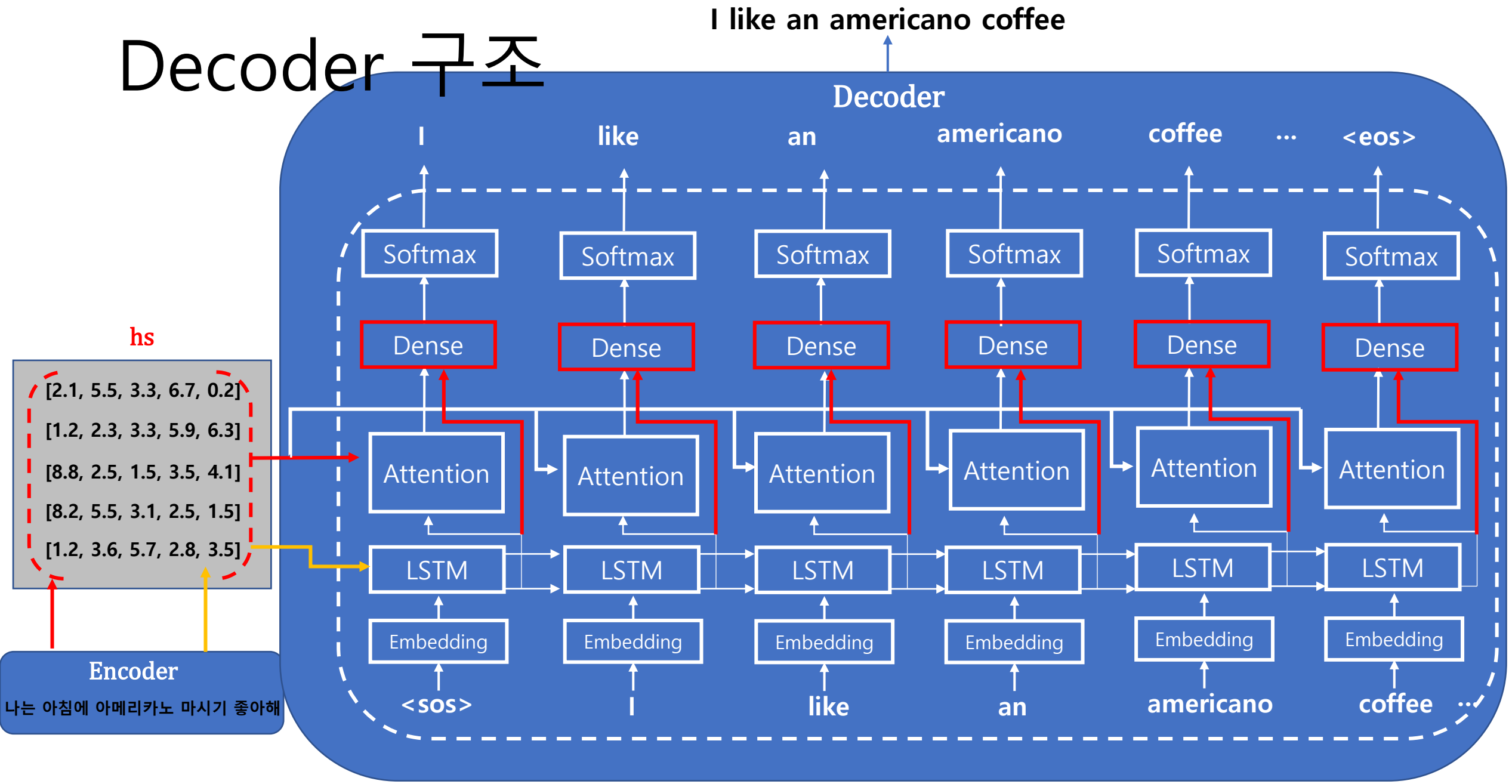
번역 모델

- 결국 seq2seq에서와 같이 번역을 위한 context vector를 만든 것
- 다만 context vector 생성 과정에 Attention이 사용된 것
- 이제까지의 과정을 Attention 층으로 단순화하여 표현한다





Decoder 구조


I like an americano coffee
Decoder
I
like
an
americano
coffee
…
<eos>
Softmax
Dense
Attention
LSTM
Embedding
<sos>
I
like
an
americano
coffee
…
hs
[2.1, 5.5, 3.3, 6.7, 0.2]
[1.2, 2.3, 3.3, 5.9, 6.3]
[8.8, 2.5, 1.5, 3.5, 4.1]
[8.2, 5.5, 3.1, 2.5, 1.5]
[1.2, 3.6, 5.7, 2.8, 3.5]
Encoder
나는 아침에 아메리카노 마시기 좋아해

Transformer

Transformer

- 트랜스포머는 2017년 구글에서 발표한 “Attention is all you need”라는 논문에서 알려짐
- 논문 제목처럼 Attention이 핵심이지만, 기존과는 그 방법이 다름
- 또한 인코더-디코더의 구조는 가지고 있지만, 순환신경망을 사용하지 않는 등 기존의 seq2seq에 비하여 많은 부분이 다름

Attention Is All You Need

Ashish Vaswani[*] Google Brain avaswani@google.com
Noam Shazeer[*] Google Brain noam@google.com
Niki Parmar[*] Google Research nikip@google.com
Jakob Uszkoreit[*] Google Research usz@google.com
Llion Jones[*] Google Research llion@google.com
Aidan N. Gomez[*] [†] University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser[*] Google Brain lukaszkaiser@google.com
Illia Polosukhin[*] [‡] illia.polosukhin@gmail.com

Abstract

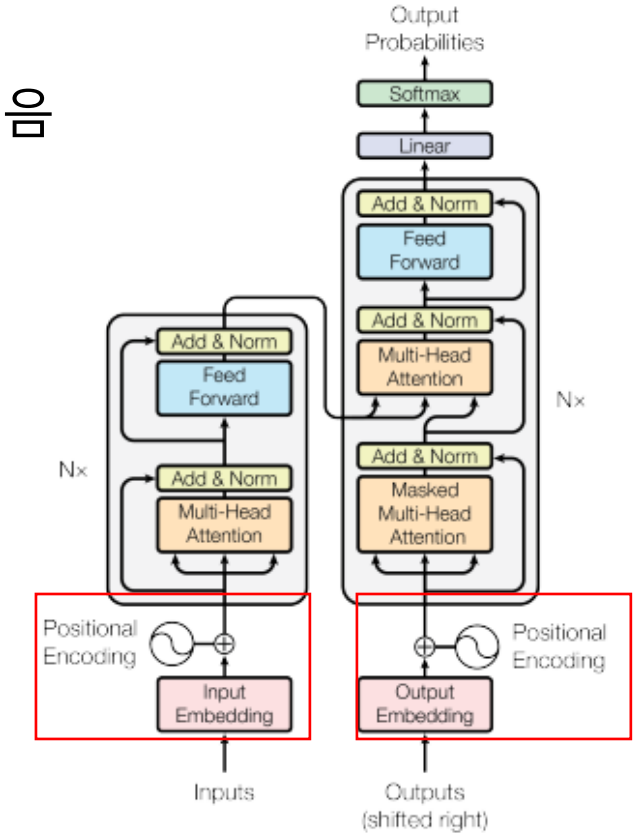
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

순환신경망 제거

- 이제까지 RNN은 다양한 상황에서 사용되어 왔음
- 특히 문장과 같이 가변 길이 시계열성 데이터를 다루는 데 효과가 있었음

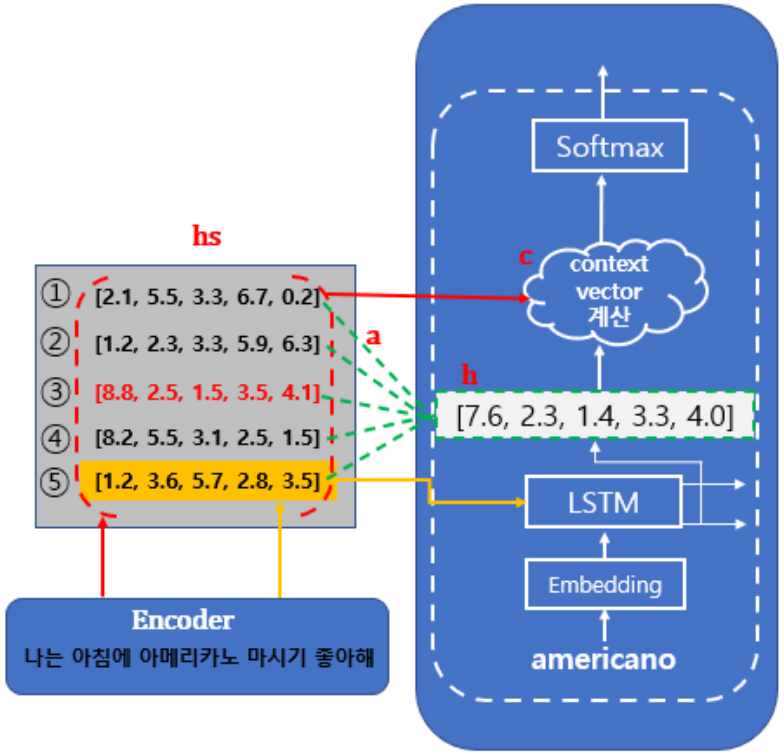
- 그러나 RNN은 앞 time step을 계산해야 뒤 time step을 계산할 수 있음
- GPU를 이용한 병렬 연산이 활발한 딥러닝에서 이는 큰 단점
- 또한 그래디언트 소실 문제도 고질적인 문제로 남아 있음

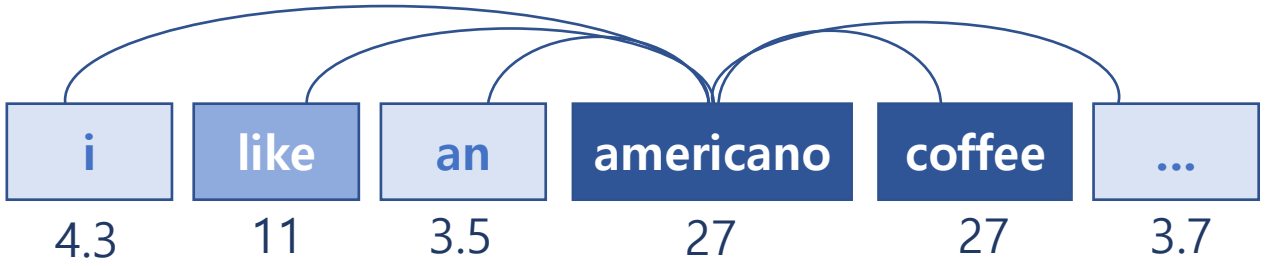
- 따라서 RNN을 사용하지 않는 방법이 고안되고 있으며,
- 트랜스포머는 RNN 없이 Embedding된 결과에 Attention을 이용한다
- 대신 현재 어휘가 몇 번째인지 위치 정보를 추가한다



Self-Attention 개요

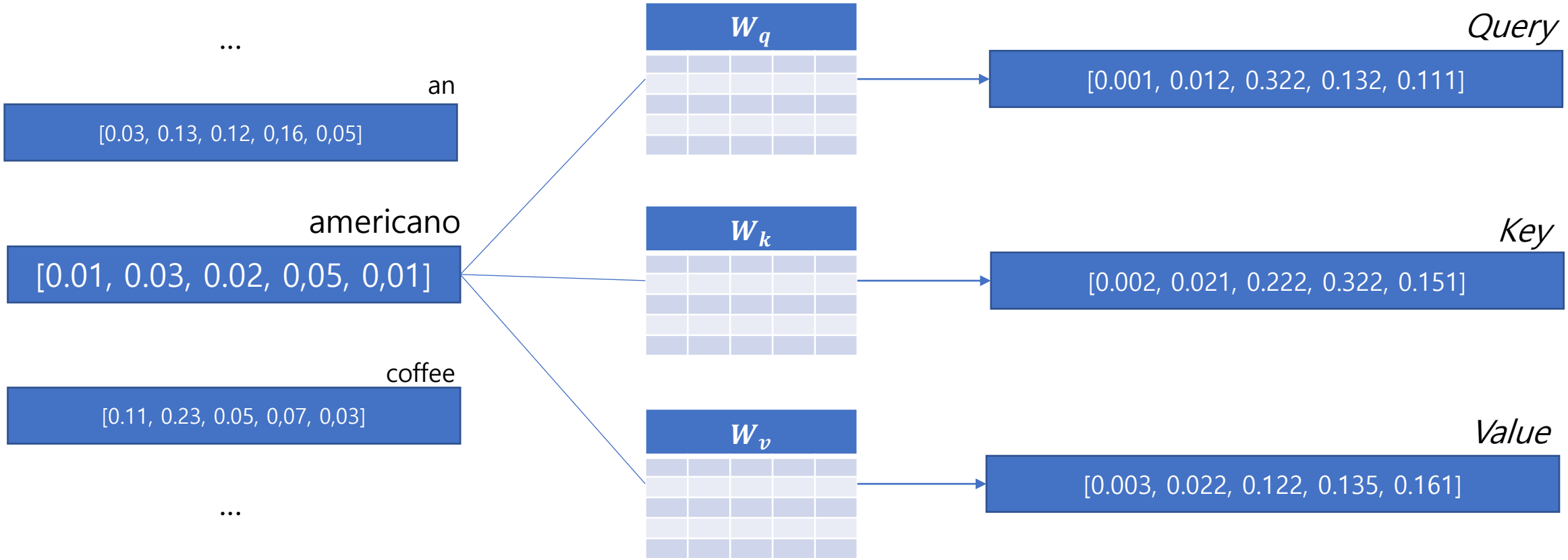
- 일반 Attention에서는 순환신경망을 거친 벡터 *h*와 *hs* 행렬 각 벡터와의 유사도를 가중치 *a*를 이용하여 계산한 뒤, 다시 *hs* 행렬과 곱하여 context vector *c*를 계산하였다
- Self-Attention에서는 입력된 문장 사이의 관계만을 이용하여 context vector를 만든다





Scaled Dot-Product Attention

① 임베딩 값의 단어별 벡터(X)로 되어 있는 입력문에 세 개의 서로 다른 가중치 W_q, W_k, W_v를 행렬곱하여 *Query, Key, Value* 를 만든다

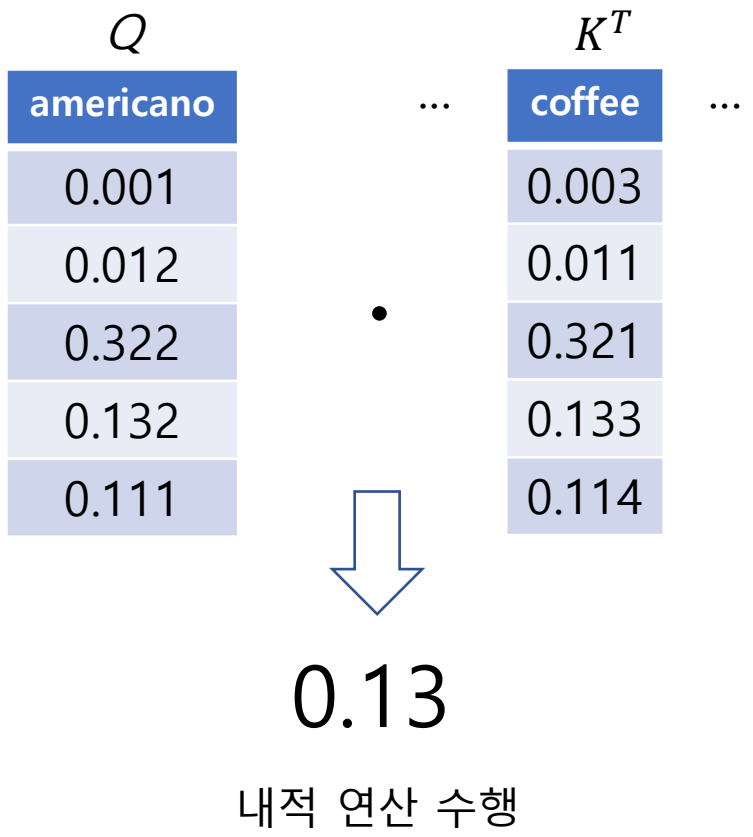


② 모든 단어가 모이면 Q, K, V 는 행렬이 된다

Q

i	0.001	0.032	0.002	0.121	0.322
like	0.201	0.081	0.329	0.238	0.314
an	0.101	0.012	0.105	0.131	0.192
americano	**0.001**	**0.012**	**0.322**	**0.132**	**0.111**
coffee	0.011	0.013	0.325	0.153	0.134

③ Q 행렬의 특정 단어와 K 행렬을 내적연산 한다



④

i	like	an	americano	coffee
0.08	0.1	0.08	0.13	0.13

⑤ Softmax

0.05	0.2	0.05	0.35	0.35

⑥

X	X	X	X	X
i	like	an	americano	coffee
0.101	0.301	0.201	0.021	0.022
0.122	0.071	0.031	0.302	0.213
0.005	0.219	0.126	0.112	0.251
0.152	0.135	0.321	0.152	0.125
0.121	0.213	0.251	0.121	0.213

⑦

i		like		an		americano		coffee	=	americano
0.00505	+	0.01505	+	0.01005	+	0.00105	+	0.0011	=	0.032
0.0061		0.00355		0.00155		0.0151		0.01065		0.037
0.00025		0.01095		0.0063		0.0056		0.01255		0.036
0.0076		0.00675		0.01605		0.0076		0.00625		0.044
0.00605		0.01065		0.01255		0.00605		0.01065		0.046

'americano'에 대한 Attention Score

$$(Q \cdot K^T)$$

Key 벡터의 차원수의 제곱근으로 나눠 크기를 줄인다
그리고 Softmax 연산을 하여 확률 정보로 만든다

$$softmax(\frac{Q \cdot K^T}{\sqrt{d_k}})$$

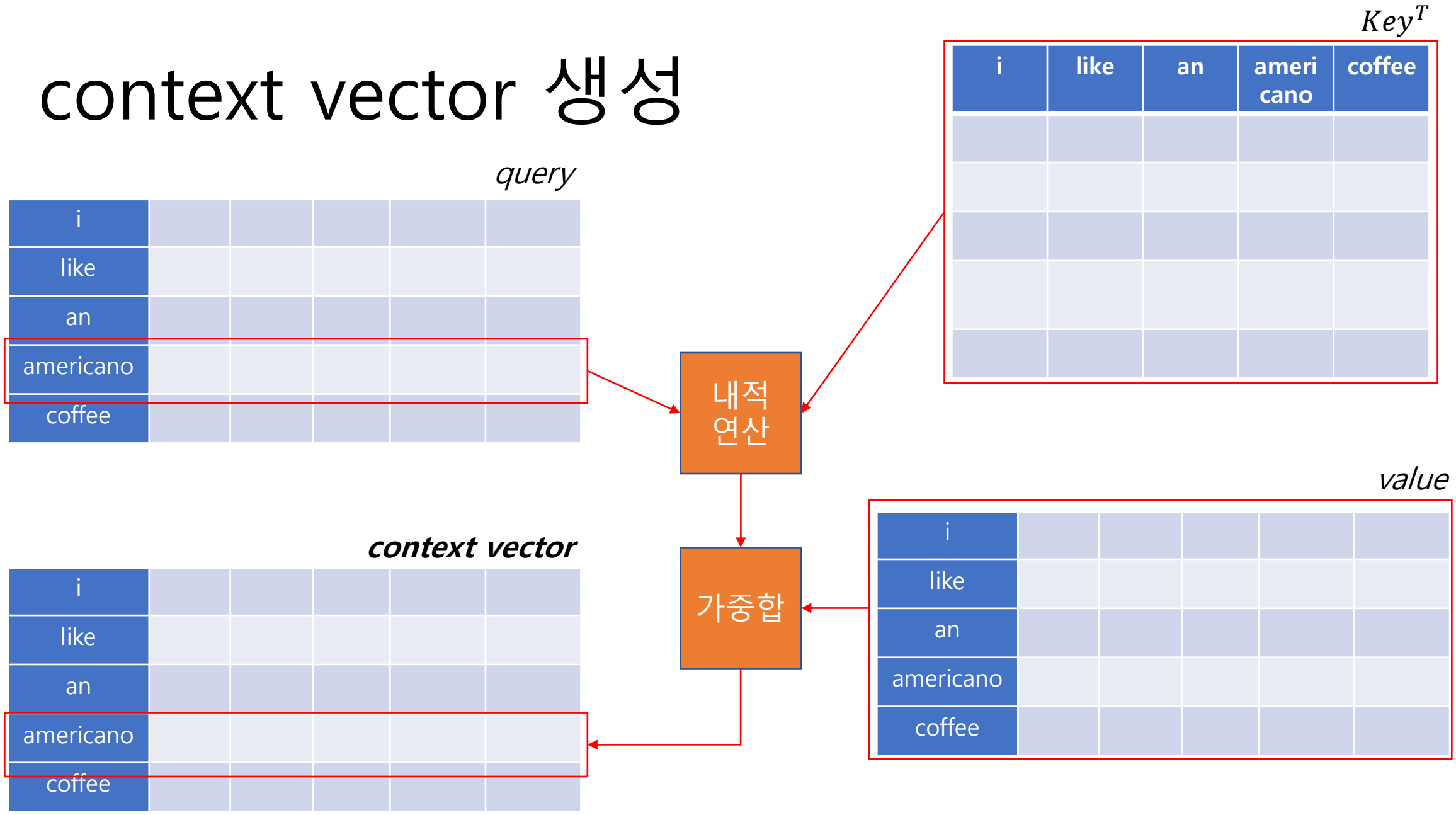
확률값 Attention Score에 *Value* 벡터와의 가중합을 수행한다
각 단어마다의 *Value* 벡터를 곱하고, 그 결과를 모두 더한다

$$softmax\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V$$

'americano'에 대한 context vector 가 만들어짐

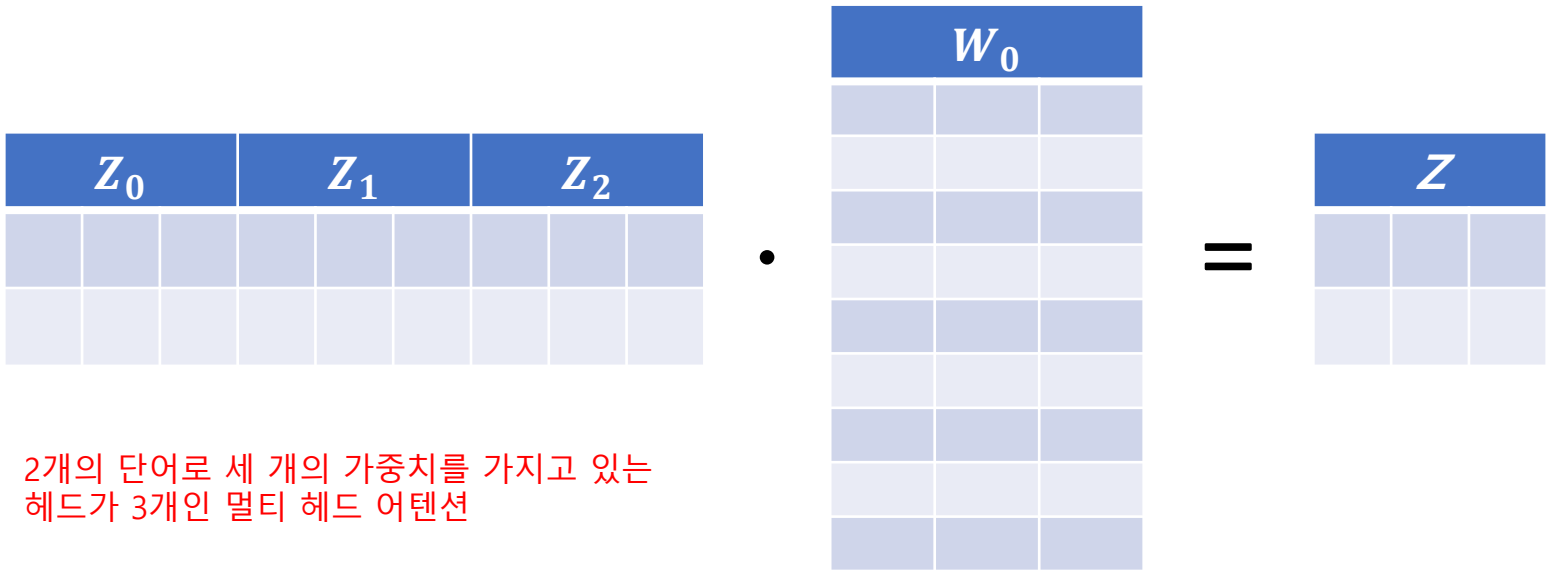
이러한 방식으로 다른 단어에 대하여도 계산을 진행하여 전체 단어에 대한 context vector를 만든다

context vector 생성

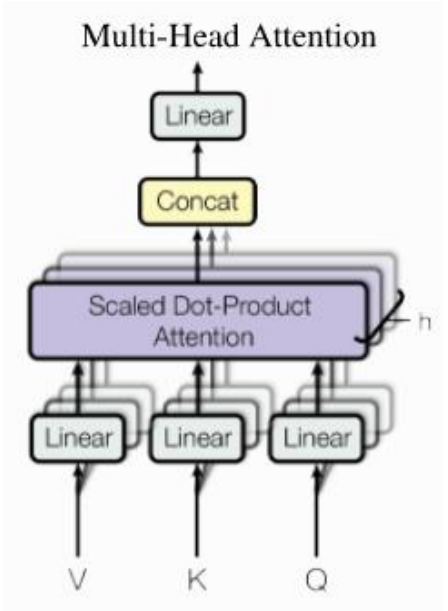


Multi Head Attention

- 멀티 헤드 어텐션은 셀프 어텐션을 동시에 여러 번 수행하는 것
- 어텐션마다 랜덤 초기값으로 시작하는 W_q, W_k, W_v 세 개 가중치를 훈련
- 어텐션을 여럿 수행하면 같은 연산 내용에 대하여 다른 초기값으로 진행
- 결국 같은 목표를 이루기 위한 조금씩 다른 여러 개의 W_q, W_k, W_v 생성
- 이들을 이어 붙여 다양한 상황에 잘 대응할 수 있게 함



2개의 단어로 세 개의 가중치를 가지고 있는
헤드가 3개인 멀티 헤드 어텐션



Positional Encoding

- 순환신경망과 달리 트랜스포머는 단어를 순차적이 아닌, 한번에 넣는다
- 따라서 입력문에 대한 순서 정보를 추가로 주입한다
- 즉, 트랜스포머의 순서 정보가 반영되지 않는 문제를 해결하는 방법

- 짝수 위치에는 사인 함수를, 홀수 위치에는 코사인 함수를 사용해 구분

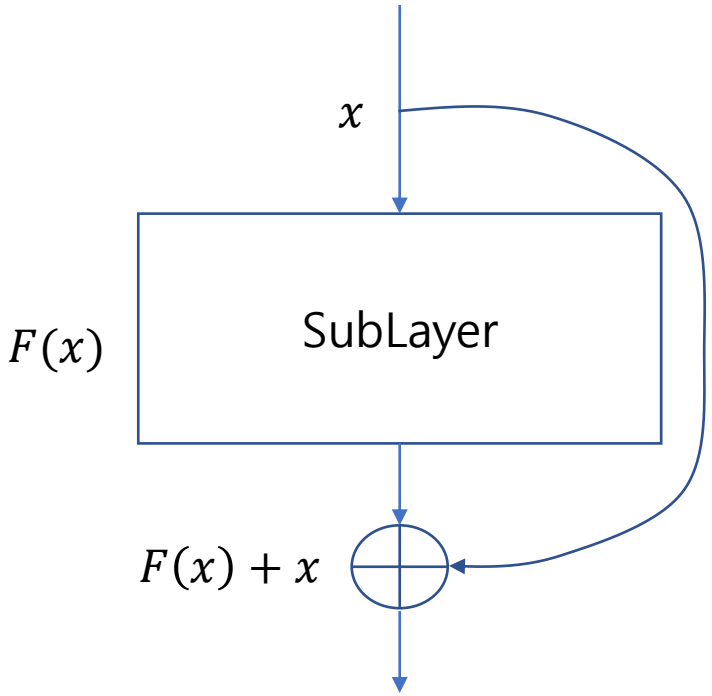
- $PE_{(pos,2i)} = \sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}})$
- $PE_{(pos,2i+1)} = \cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}})$

Feed Forward Network

- FFN은 가장 기본적인 신경망 구조로서, 입력층, 은닉층, 출력층으로 구성
- 한 문장에 있는 단어 벡터 각각에 대해 연산하는 네트워크
- 논문에서는 Position-wise Feed Forward Network라고 표현한다

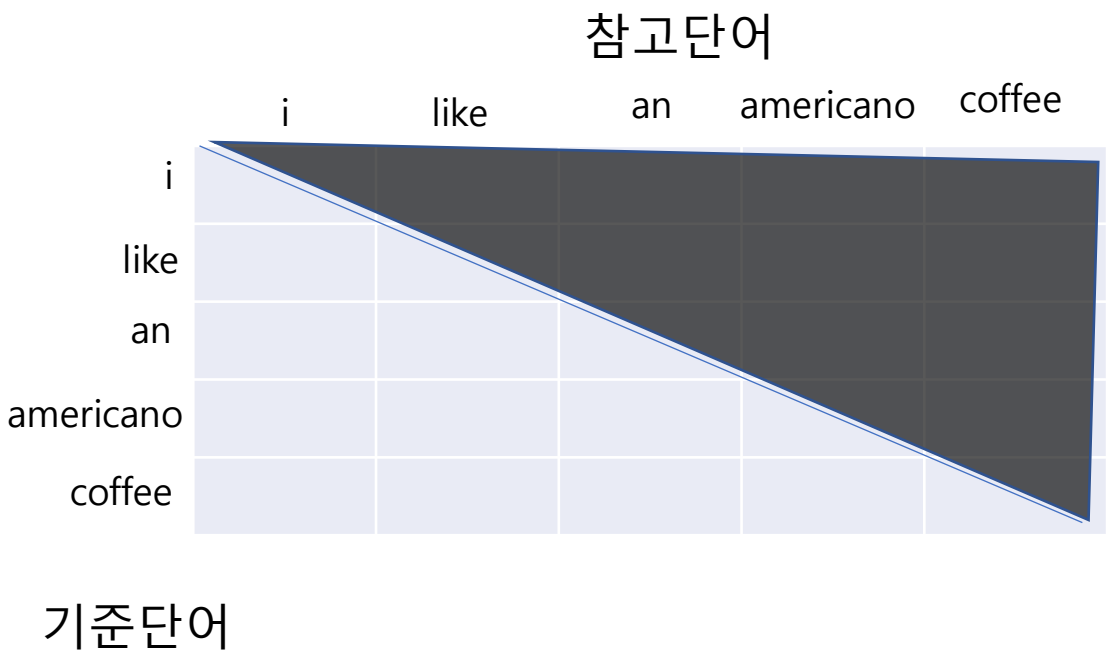
Residual Connection

- 잔차 연결은 특정 레이어를 거친 정보를 후에 다시 더하는 과정이다
- 레이어를 거치기 전의 정보를 보존하려는 목적이다



Subsequent Masked Attention

- 순방향 마스크 어텐션
- 자신보다 뒤에 있는 단어를 참고하지 않게 하는 마스킹 기법



기준단어 i는 그 뒤의 like, an, americano, coffee를,
기준단어 like는 그 뒤의 an, americano, coffee를
참고하지 않도록 하는 기법

앞에 i가 나왔다면 like가 나올 가능성이 있지만,
like는 뒤에 무엇이 올지 모르므로 뒤의 단어를
참고할 수는 없다
이처럼 이미 앞에 나온 단어만 참고하게 하는 기법이
순방향 마스크 기법이다

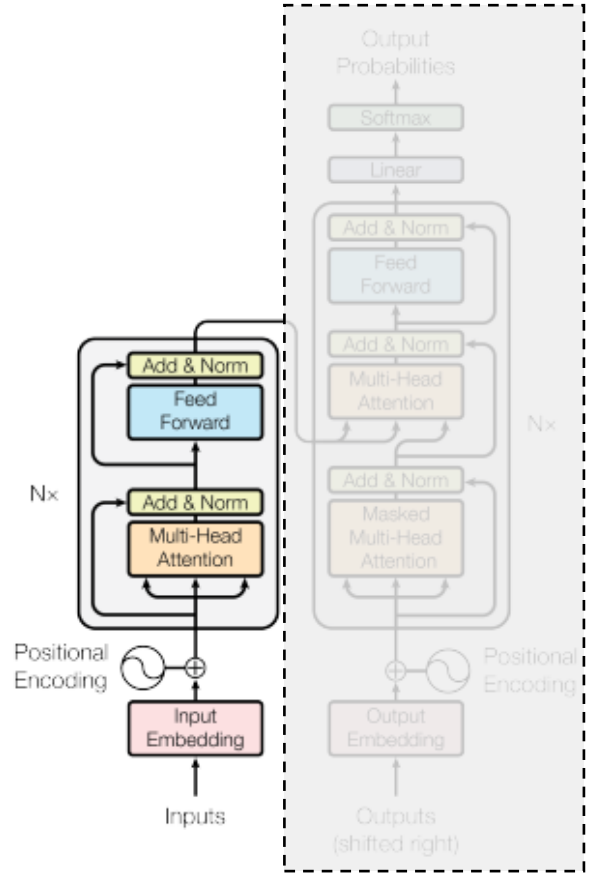
BERT

Bidirectional Encoder Representations from Transformers

BERT

- BERT는 Transformer의 인코더만 가지고 있는 구조이다
- Transformer의 인코더를 통하여 만들어진 가중치는 언어에 대한 정보를 가지고 있다

- BERT는 언어에 대한 이해를 높이기 위하여
  사전 학습 과정에서 다음 2개의 문제를 학습한다
  학습 결과로 나온 가중치가 언어에 대한 전반적인 모습을 담게 된다
  ① 마스크 언어 모델
  ② 다음 문장 예측



마스크 언어 모델

- 마스크 언어 모델이란 특정 위치에 어떤 단어가 올지를 예측하는 모델
- 문장 내에서 단어 사이의 연관성을 학습하게 된다

- 아래와 같이 특정 단어를 마스킹하고 이 위치에 '아메리카노', '물', '주스' 등이 올 수 있음을 예측하게 한다
- [MASK] 단어 예측에는 문장 안의 모든 단어들이 양방향으로 사용된다

  - 나는 아침에 *[MASK]* 마시기 좋아해

다음 문장 예측

- 또한 두 문장이 이어진 문장인지 아닌지도 예측할 수 있도록 학습된다
- 이를 위하여 입력 내용의 첫 부분에 [CLS]를, 입력 내용 중 문장이 구분되는 곳과 마지막에 [SEP]라는 스페셜 토큰을 넣는다
- 이를 통하여 두 문장 간의 관계를 예측하는 문제에 도움을 준다

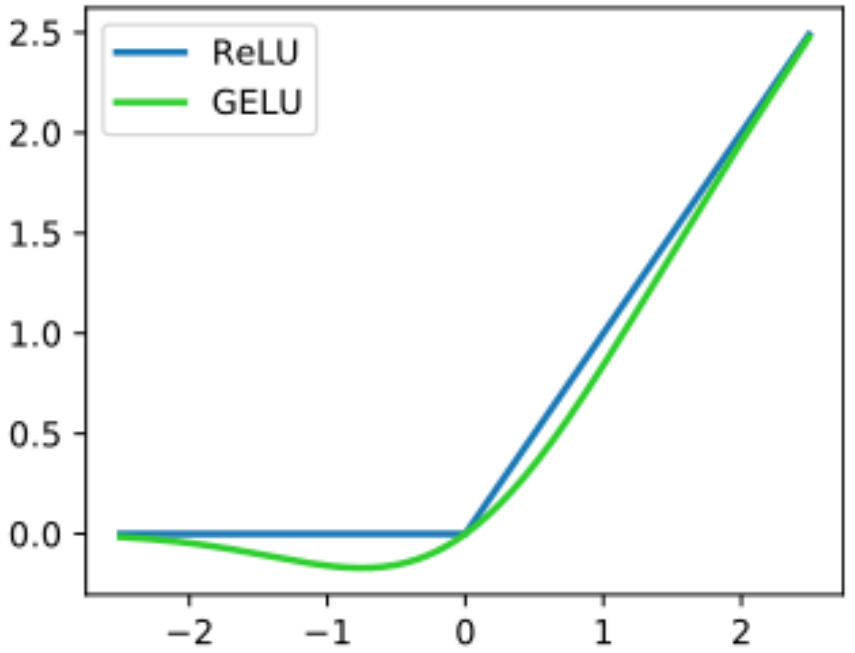
- BERT는 마스크 언어 모델과 다음 문장 예측 두 가지를 동시에 사전 학습하며, 따라서 문장은 다음과 같이 입력된다

  - *[CLS]* 나는 아침에 *[MASK]* 마시기 좋아해 *[SEP]* 자연어 *[MASK]*는 즐거워 *[SEP]*

→ [MASK]에 들어갈 단어를 예측
→ 두 문장이 이어진 문장이 아님을 예측

GELU Function

- Position-Wise Feed Forward Network에 사용되는 활성화 함수로
- ReLU 함수가 아니라 GELU 함수를 사용한다
- GELU 함수는 0 주위에서 부드럽게 변화해 학습 성능을 높인다



Subword Tokenizer

- BERT는 형태소 분석기를 사용하지 않고 자체적으로 문자를 토큰화한다
- 사람이 분리된 결과를 해석하는 빈도분석이 아니라면,
- 분류 결과만 좋다면 되지 굳이 형태소/단어 단위로 구분할 필요는 없다

- 형태소 분석기를 사용하지 않는 토큰화 방법으로 subword 단위 토큰화가 있다
- 서브워드 단위 토큰화는 단어와 문자의 중간에 있는 형태이다
- BERT가 사용하는 서브워드 단위 토큰화 기법은 wordpiece이다

  - 아침에 아메리카노 마시기 좋아해
    - 형태소 분석: 아침, 에, 아메리카노, 마시, 기, 좋, 아해
    - 서브워드 단위 예: 아, 침에, 아메리, 카노, 마시기, 좋아, 해

WORDPIECE

- 워드피스는 같이 나타날 확률이 높은 문자열을 토큰으로 삼는 기법
- 다음의 값을 높이는 문자 a, b를 묶어서 하나의 토큰으로 삼는다
- 서브워드에 해당하는 문자열이 없으면 미등록 단어로 취급한다

$$\frac{\frac{\#ab}{n}}{\frac{\#a}{n} \times \frac{\#b}{n}}$$

기호	의미
#a	a 문자열의 빈도수
#b	b 문자열의 빈도수
#ab	ab라는 문자열의 빈도수
n	전체 글자수

사전 학습 모델

- 이와 같은 특징을 가지고 학습된 BERT 모델은 사전 학습 모델로 이용된다
- 사전 학습 모델은 모델의 가중치를 처음부터 찾지 않아도 되게 한다
- 사전 학습 모델의 가중치를 먼저 활용해 언어의 특성을 파악하고, 이어 레이어를 추가하여 원하는 형태로 모델을 만든다
- 이를 미세 조정(fine-tuning)이라 한다