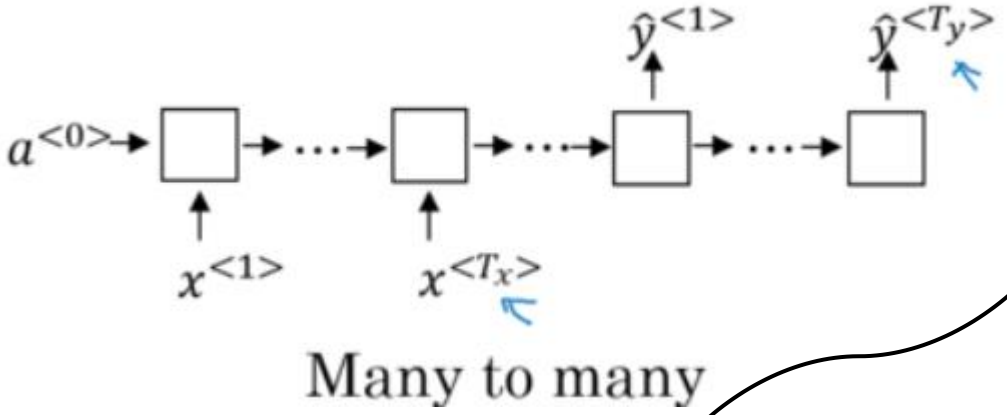
NLP Study Week3

Attention is all you need (Transformer)

Seq2seq Model

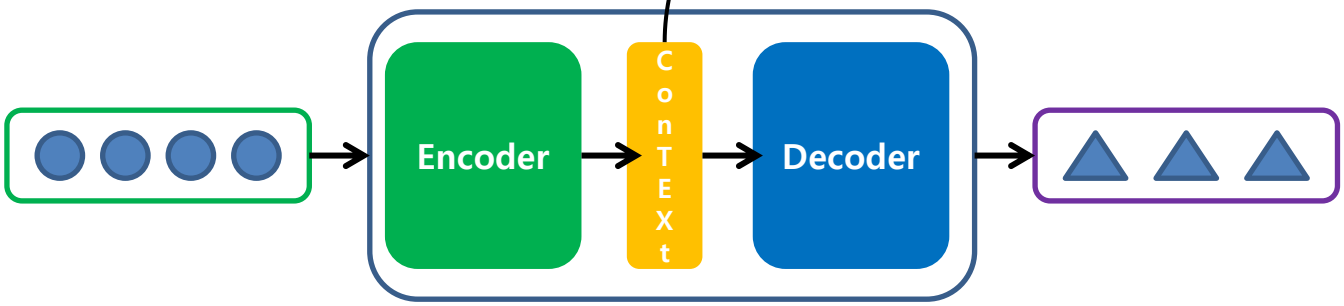
Sequence Models (Coursera)



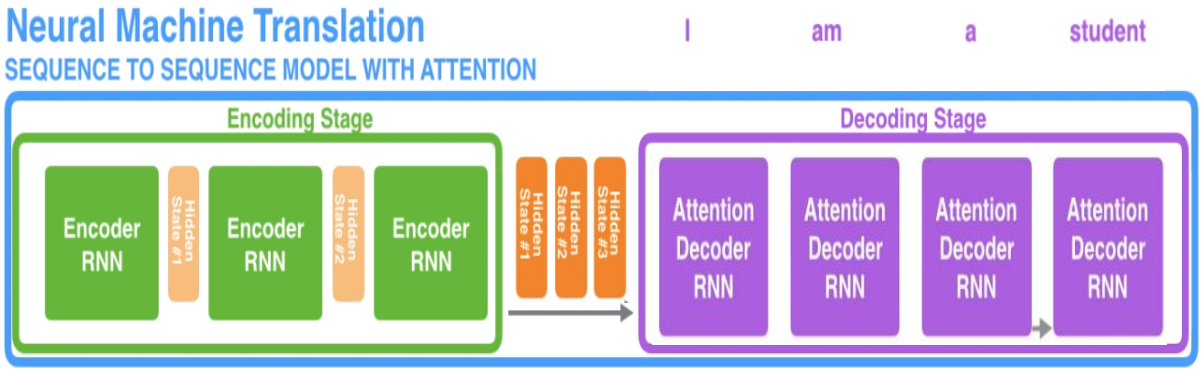
Context

: 크기가 정해진 벡터

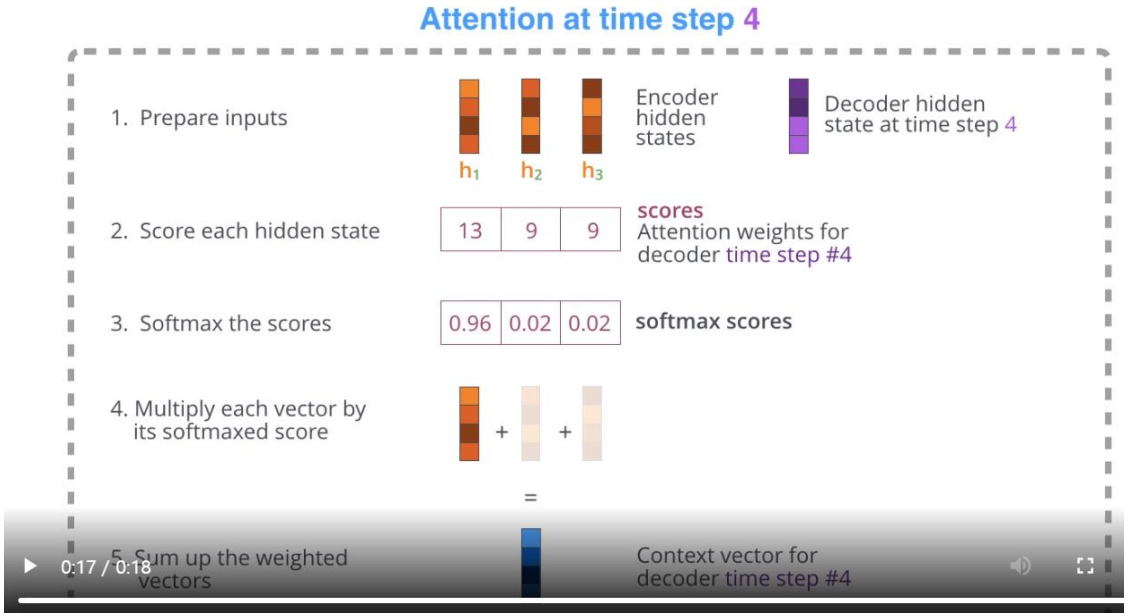
고정된 하나의 벡터로 전체 맥락을 나타내는 방법은 긴 문장을 처리하기 어렵다



Seq2seq With Attention



기존 seq2seq 모델에서는 마지막 hidden state 벡터만 넘겼던 반면 attention 모델에서는 모든 스텝의 hidden states를 decoder로

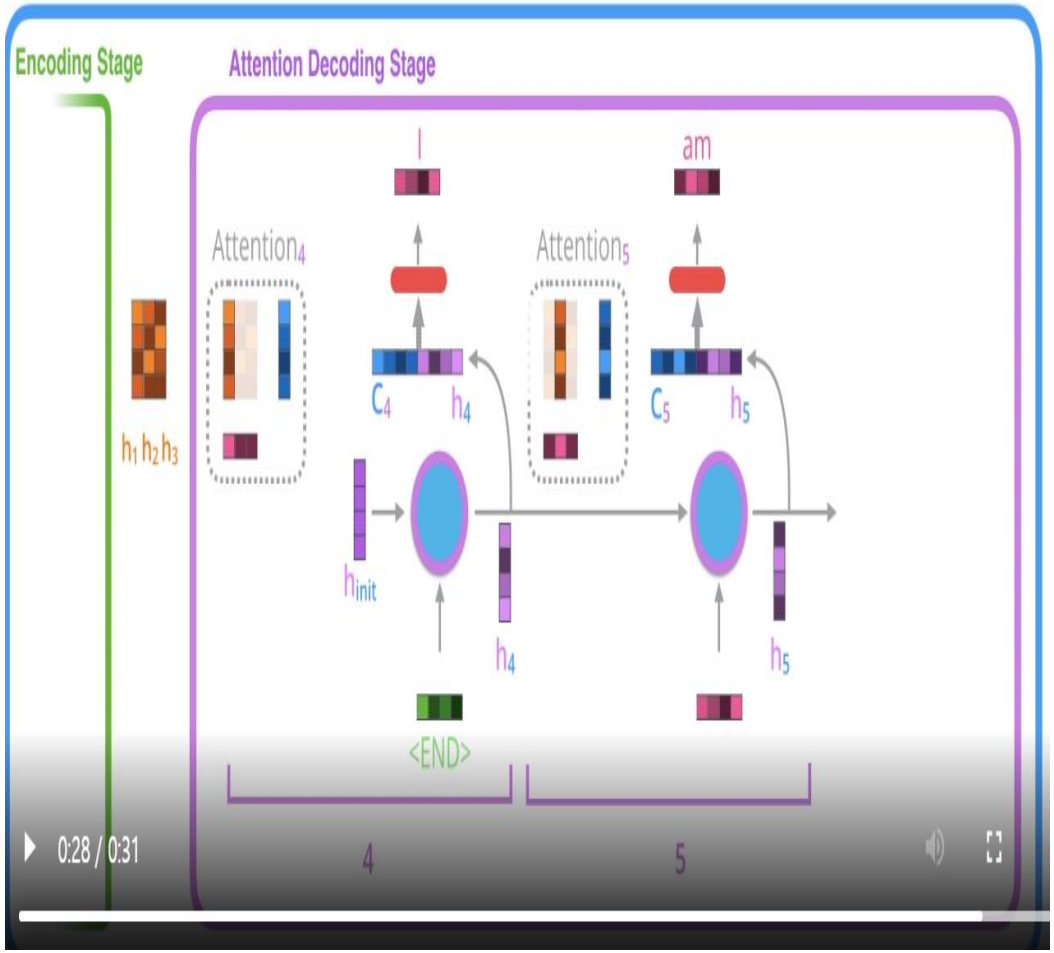


1. encoder 에서 받은 전체 hidden state 참고

2. 각 스텝의 hidden state마다 점수를 매긴다.

3. 매겨진 점수들에 softmax를 취하고 각 타임 스텝의 hidden states에 곱해서 더한다.

Seq2seq With Attention



1. attention 모델에서의 decoder 는 <END>과 추가로 initial decoder hidden state을 입력받는다.

2. decoder는 두 개의 입력을 가지고 새로운 hidden state벡터를 출력한다 (h4)

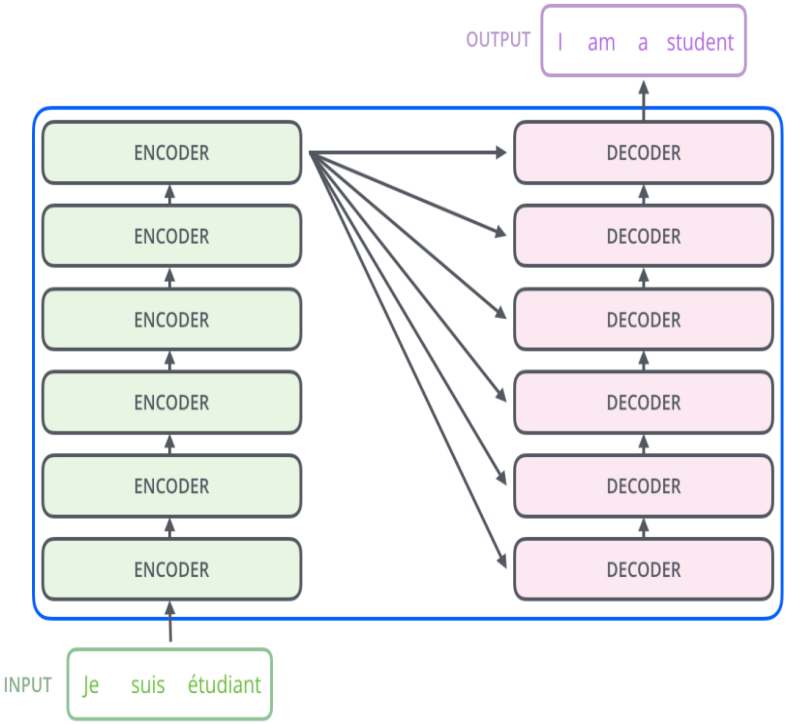
3. Attention 과정: encoder의 hidden state 들과 decoder 의 hidden state h4 벡터를 이용하여 그 스텝에 해당하는 context 벡터 (C4) 를 계산한 후, h4와 C4를 하나의 벡터로 concatenate 한다.

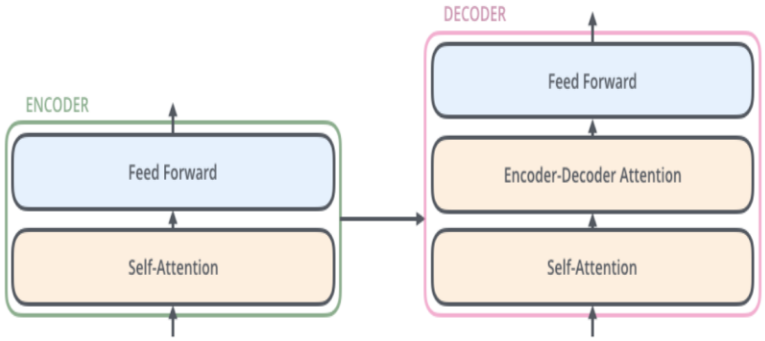
4. 이 벡터를 feedforward 신경망에 통과 시킨다.

5. feedforward 신경망에서 나오는 출력은 현재 타임 스텝의 출력 단어를 나타낸다.

6. 이 과정을 다음 타임 스텝에서도 반복한다.

Transformer



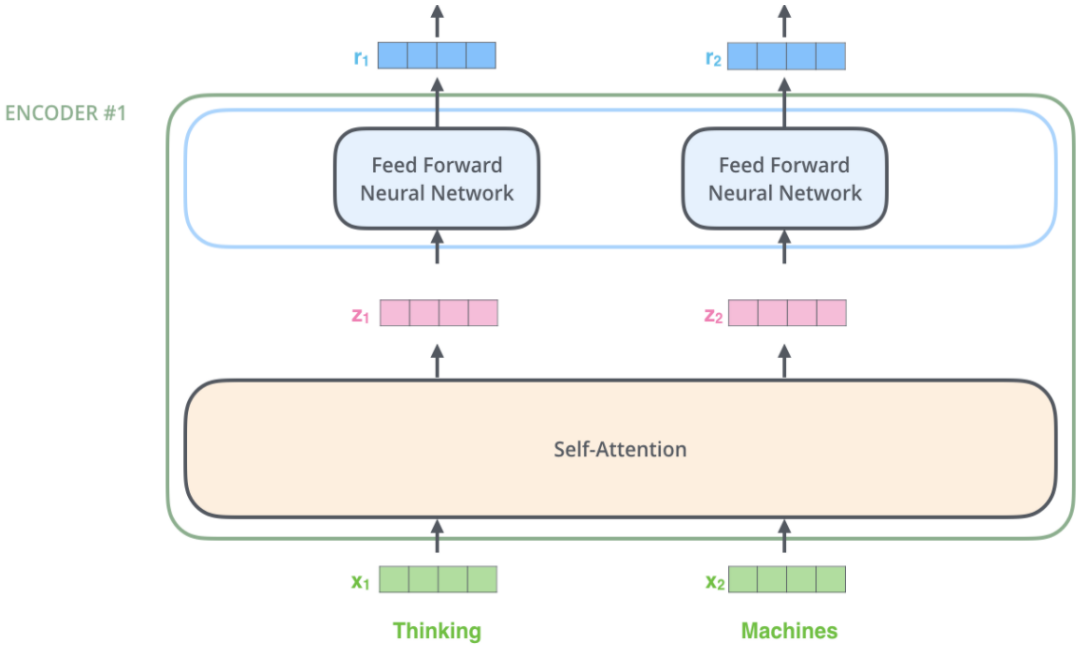


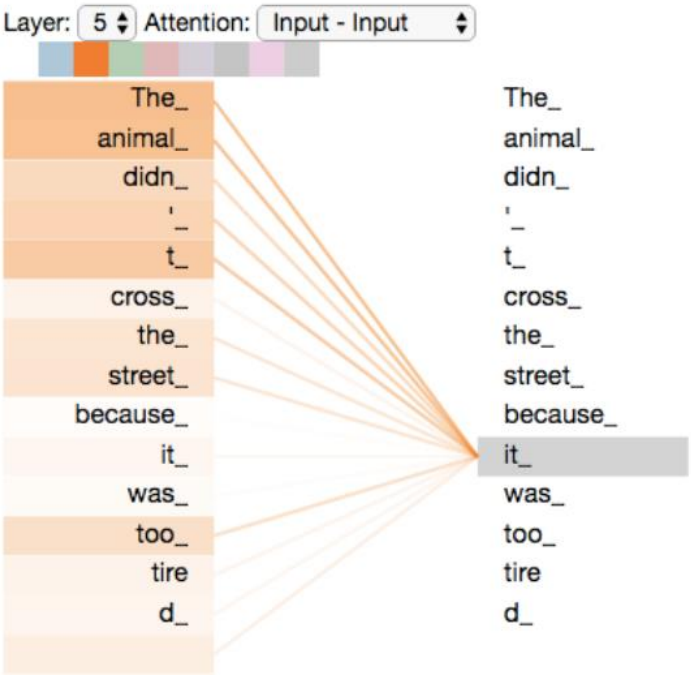
특징

- 기존의 seq2seq의 구조인 인코더-디코더를 따르면서도, Attention만으로 구현한 모델

- RNN을 사용하지 않고, 인코더-디코더 구조로 설계하였음에도 성능이 RNN을 사용했을 때보다 우수

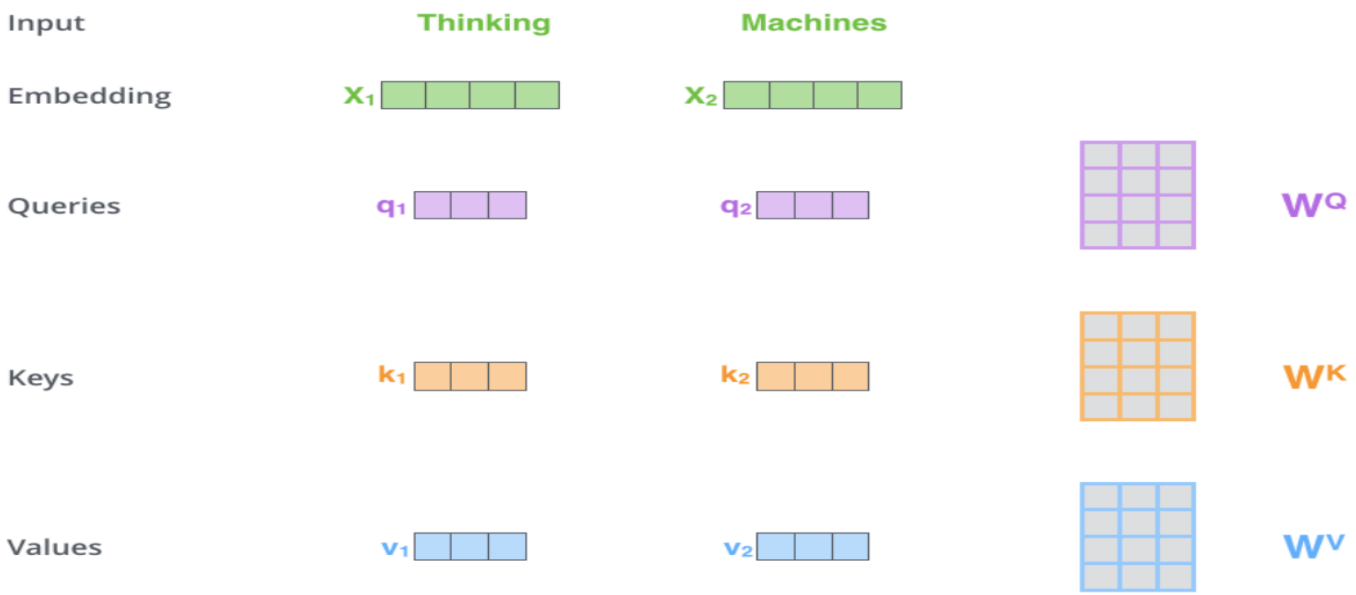
Transformer (Encoder)





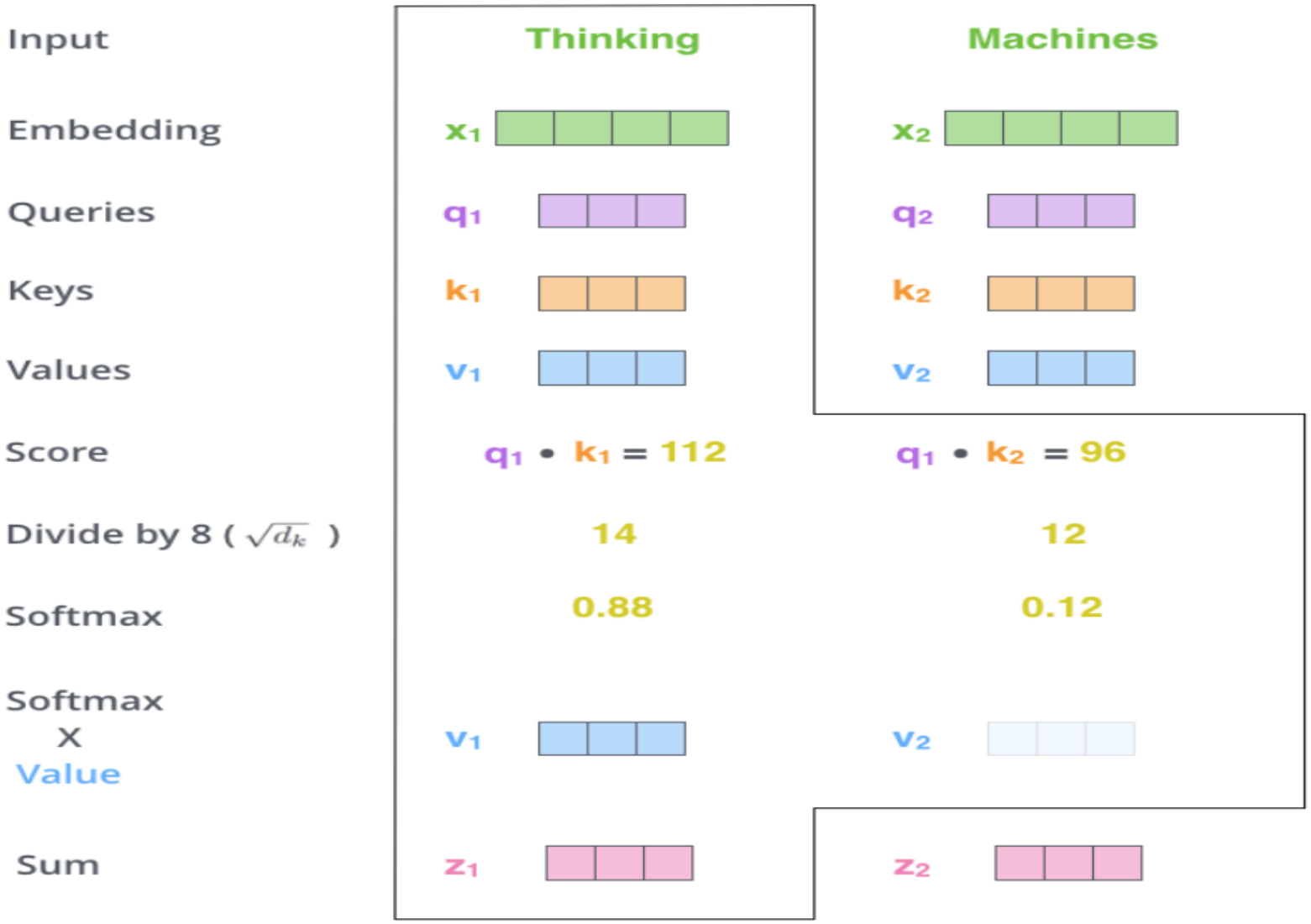
- 각 단어들은 512 차원의 벡터로 embeding (hyperparameter)
- Self-Attenttion 통과하며 dependency 포착
- Feed-Forward는 dependency 없기 때문에 병렬로 처리 가능

Transformer (Self-Attention)

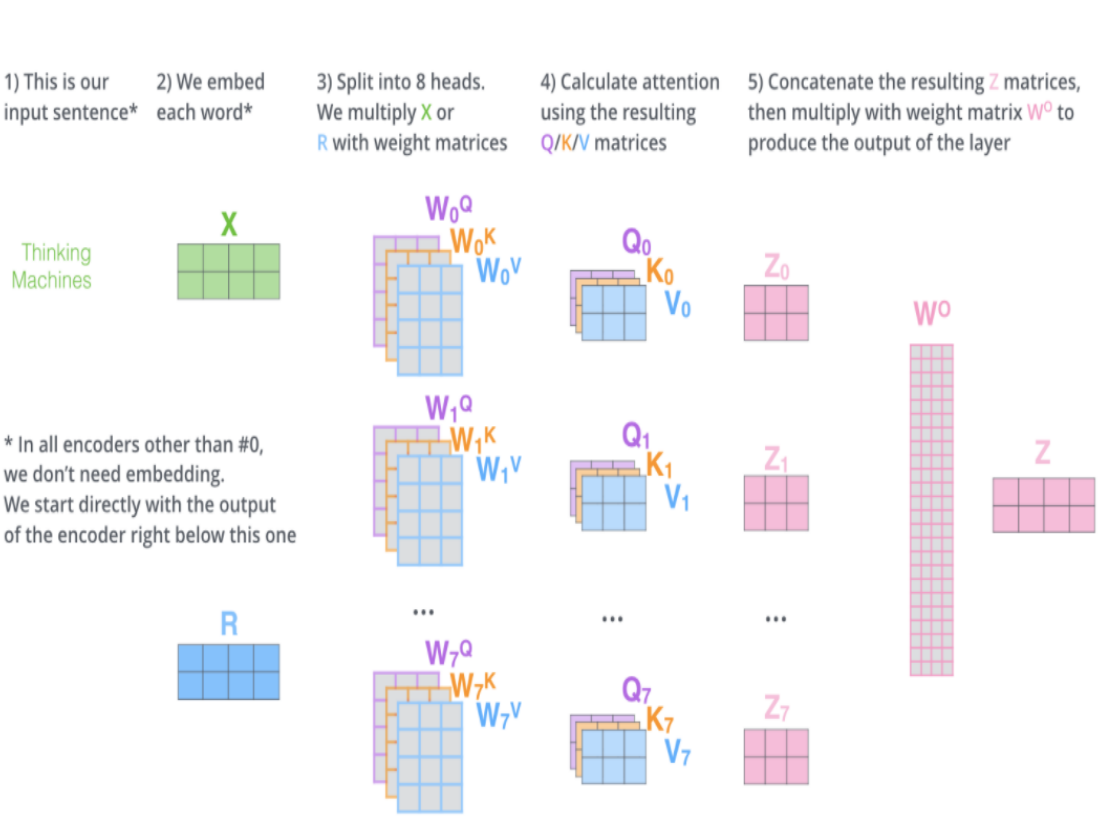


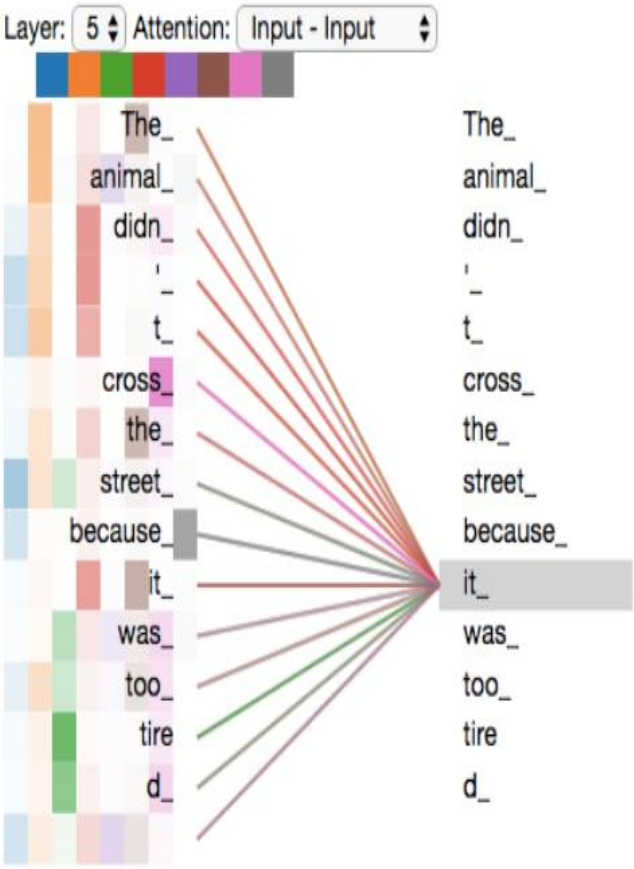
- 각 단어마다 세개의 벡터(Query, Key, Value) 생성
  (512 -> 64 / multi-head attention 계산 복잡도 일정하게 만들기 위해)
- Query, Key, Value는 attention 이해를 위한 추상적 개념
- W는 학습하며 update된다

Transformer (Self-Attention)



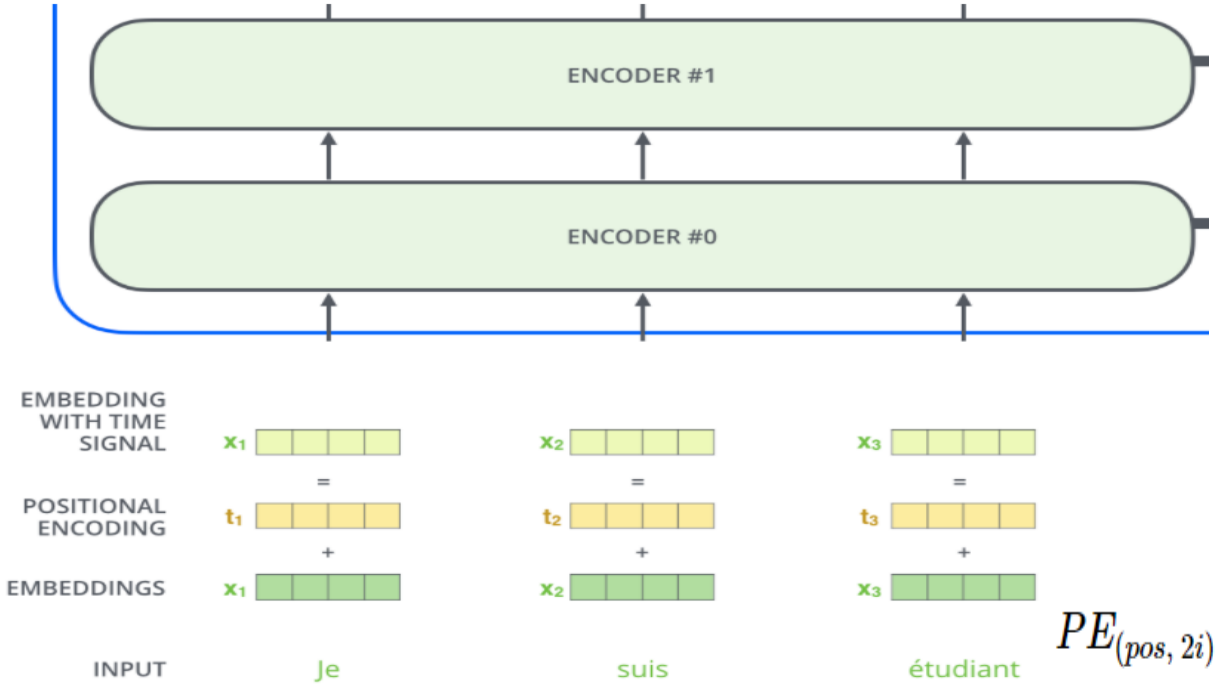
Transformer (Multi-Headed Attention)





- Dependency 포착 성능 향상
- Attention Layer가 여러 개의 representation 공간을 가지게 한다

Transformer (Positional Encoding)

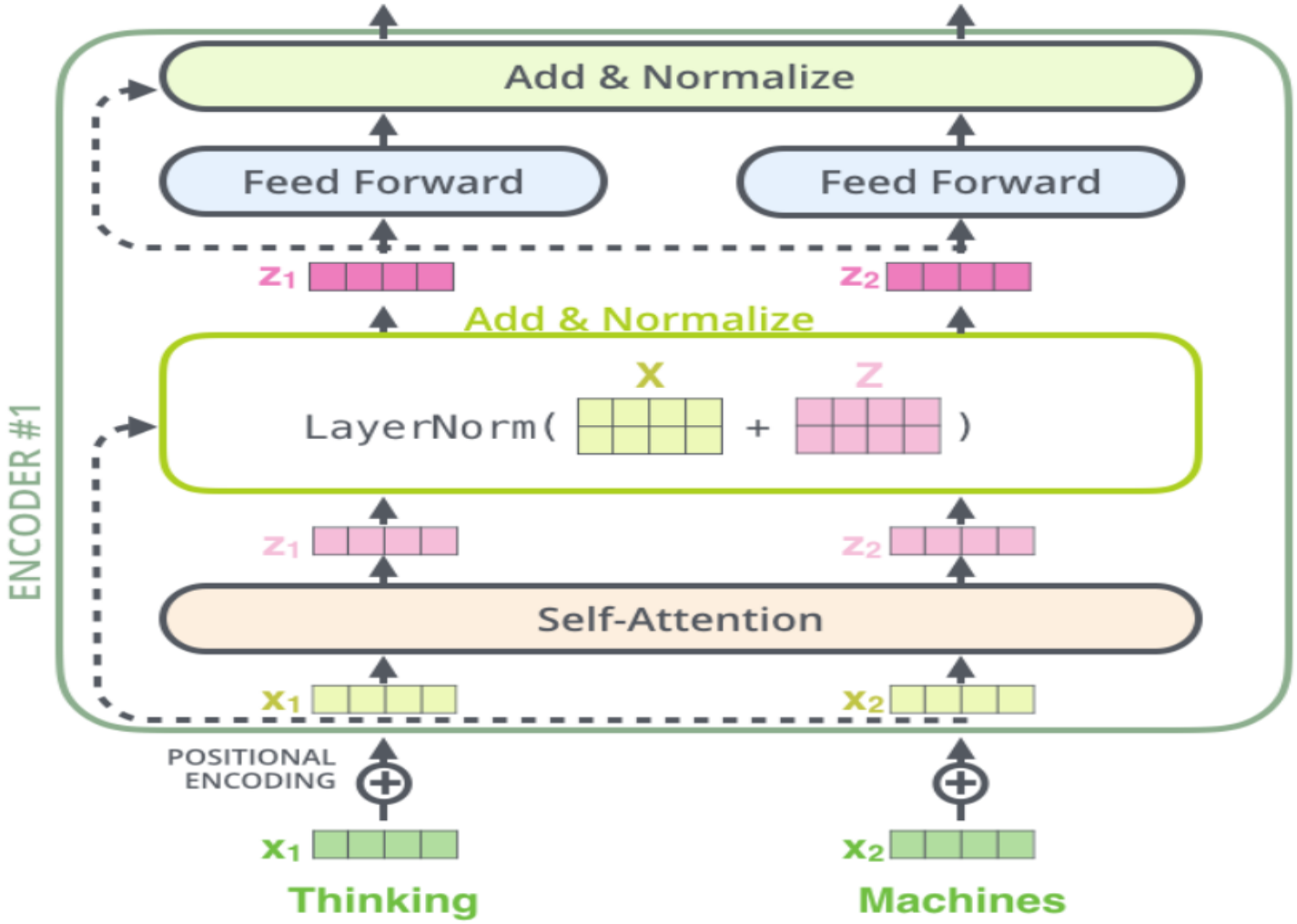


$$PE_{(pos,\,2i)} = sin(pos/10000^{2i/d_{model}})$$

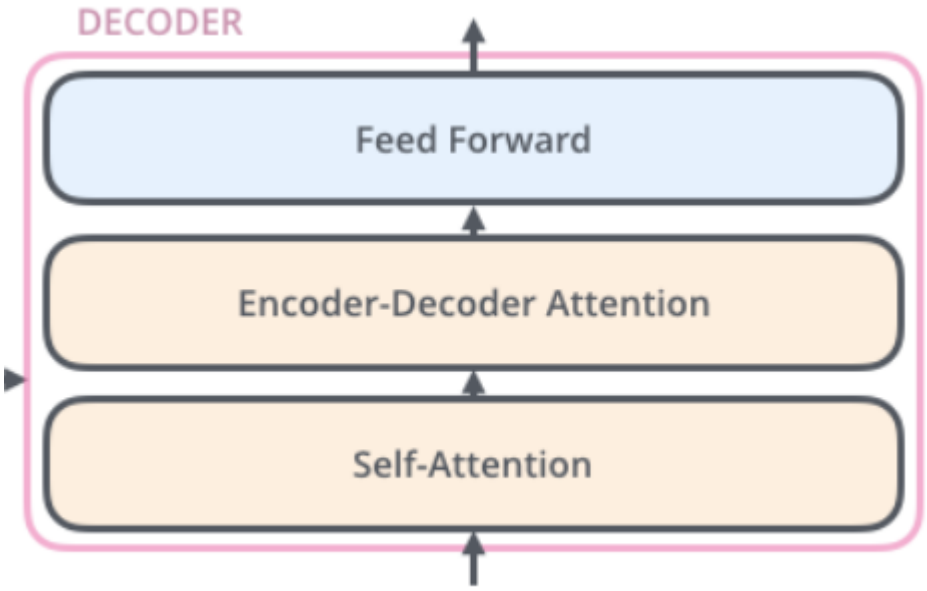
$$PE_{(pos,\,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

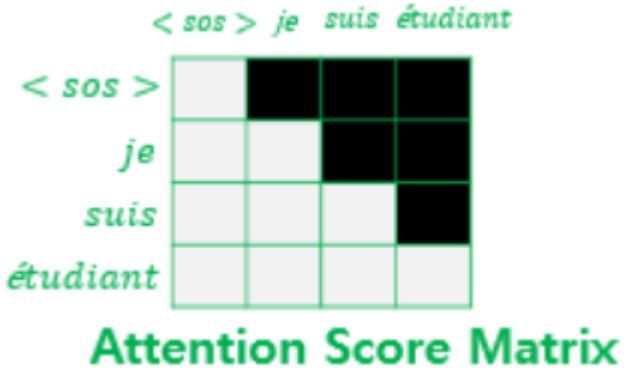
- 모델이 학습하는 특정한 패턴을 따르며 단어의 위치와 다른 단어와의 위치 차이 알 수 있다
- query/key/value 벡터로 투영되었을 때, 단어들 간의 거리를 늘릴 수 있다

Transformer (Normalizing)



Transformer (Decoder)

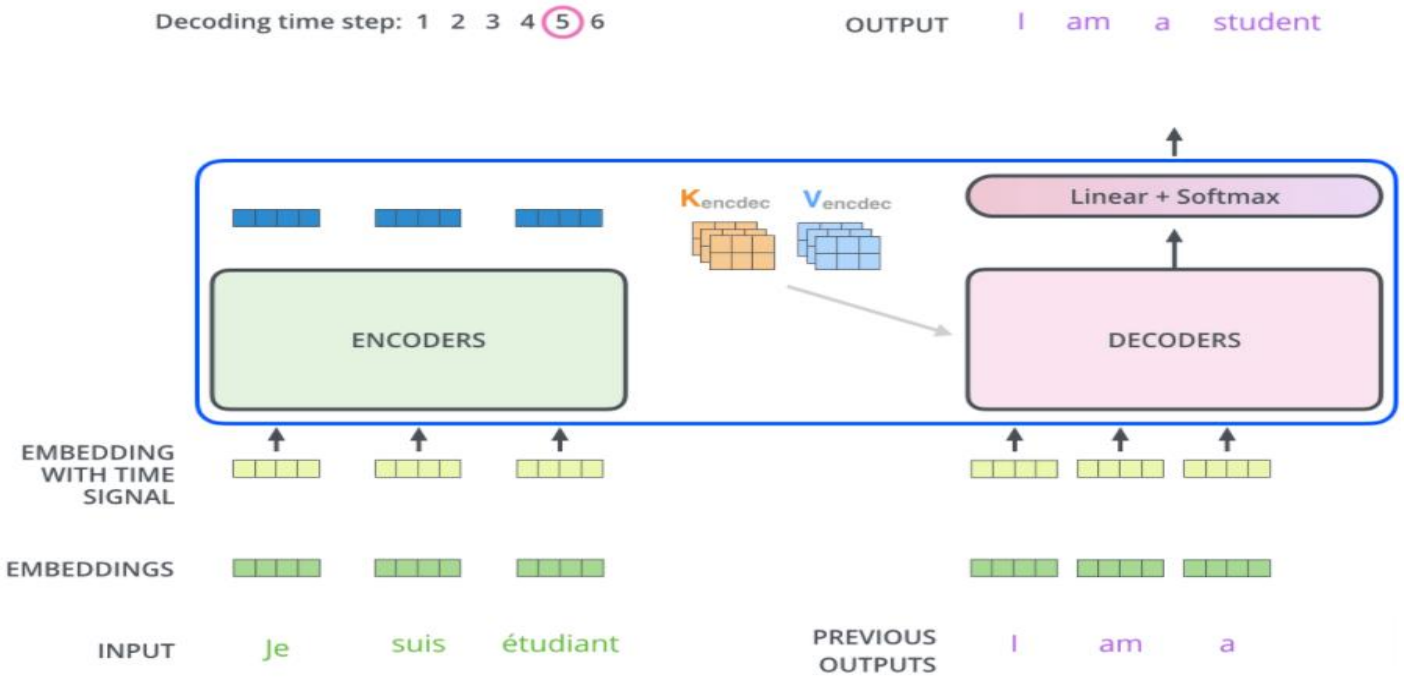




- self-attention 계산 과정에서 softmax를 취하기 전에 현재 스텝 이후의 위치에 대해서 masking(-1e9) 하여 현재 위치 이전 위치에 대해서만 attend

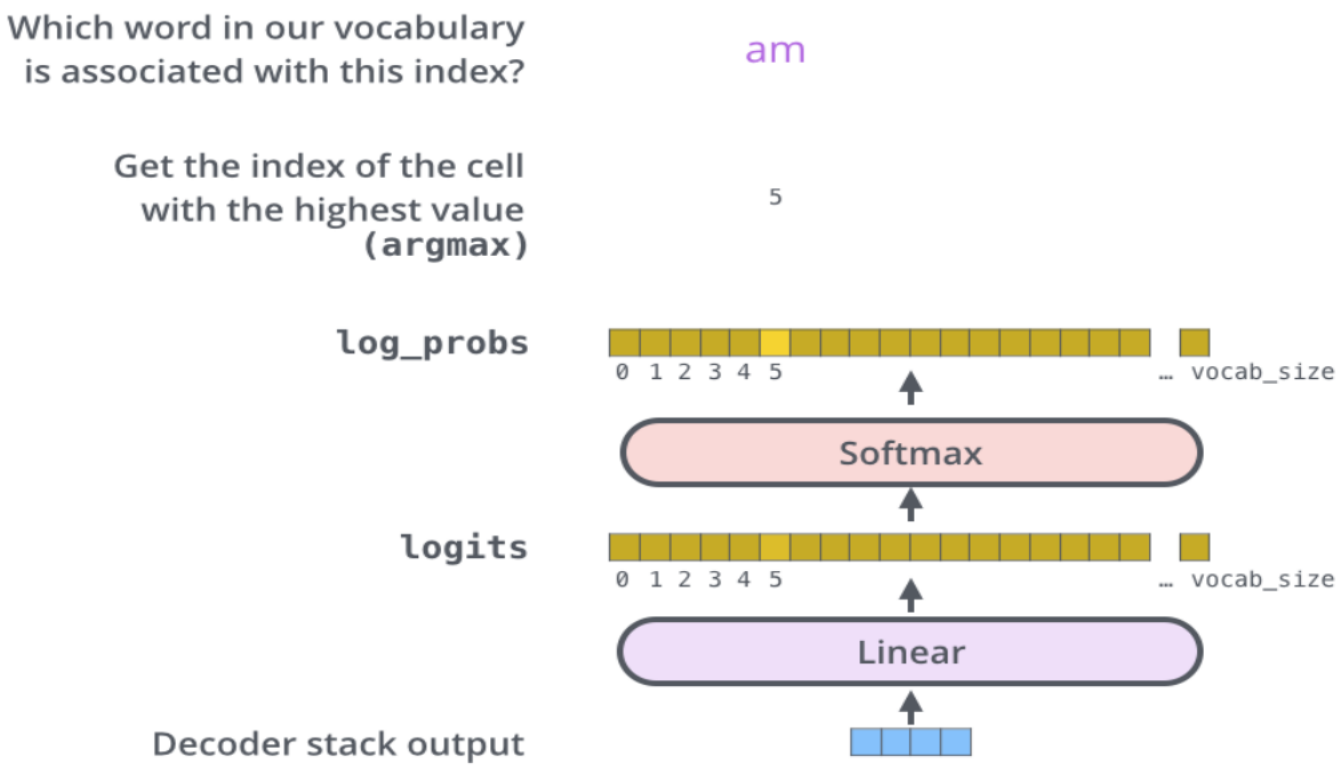
- Encoder-Decoder Attention layer는 multi-head self-attention와 비슷하게 작동하지만, Query 행렬을 그 밑의 layer에서 가져오고 Key, Value 행렬은 encoder의 출력에서 가져오는 것에서 차이를 보인다

Transformer (Decoder)



- Positional Encoding 추가
- Decoders 통과하며 한 단어씩 출력
- <EOS> 출력할 때까지 반복

Transformer (Decoder)

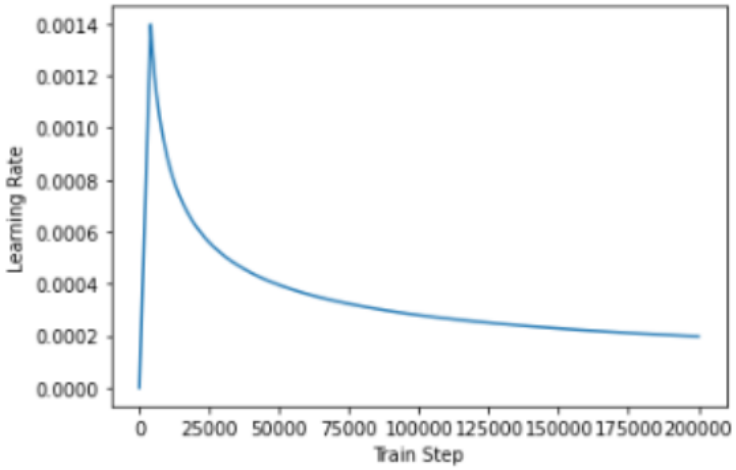


- Decoders의 output이 Fully Connected Layer 통과하여 logits 벡터로
- Softmax통과하여 학습한 vocabulary 출력

Transformer (Training)

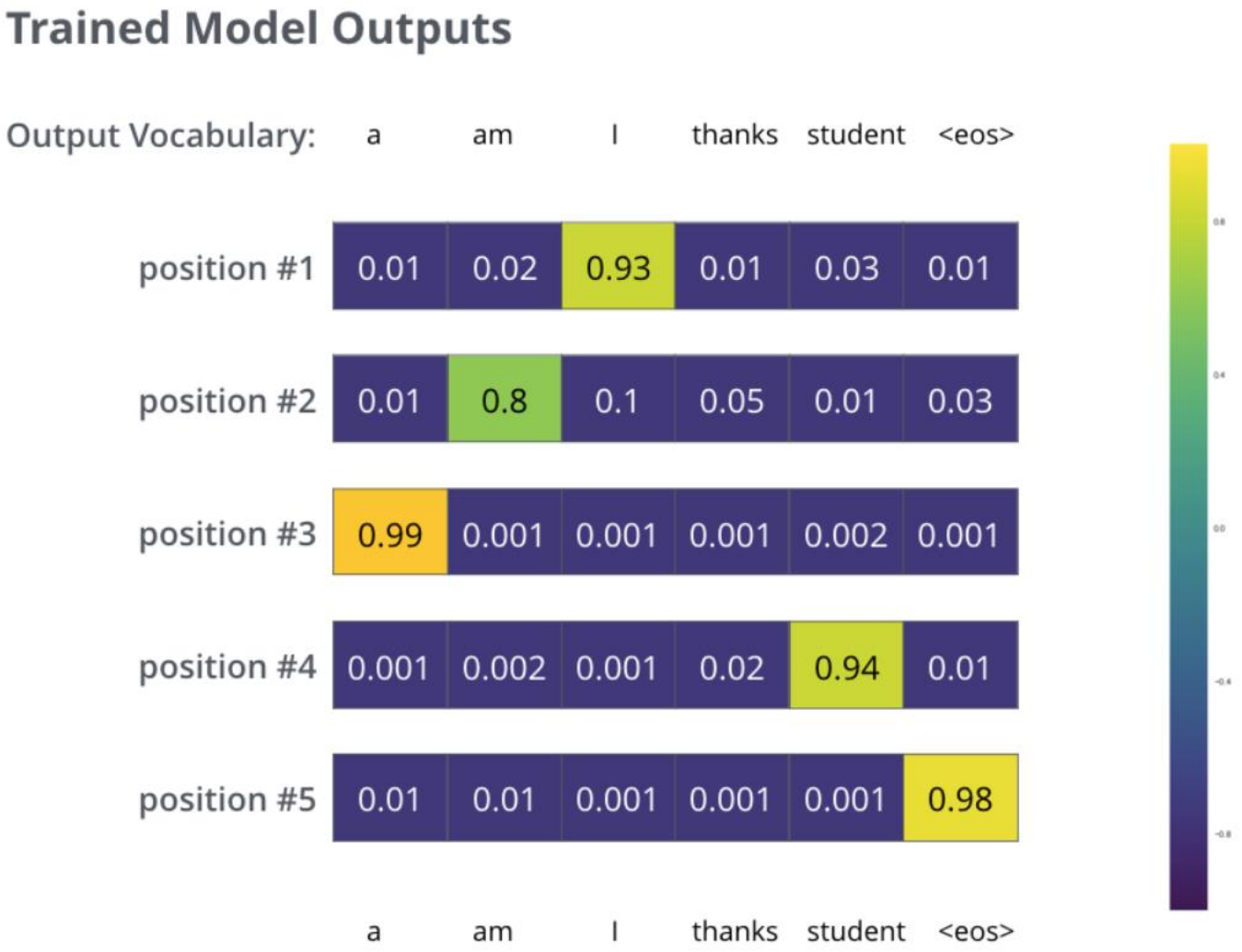
- Learning Rate

$$lrate = d_{model}^{-0.5} \times min(\text{step\_num}^{-0.5}, \text{step\_num} \times \text{warmup\_steps}^{-1.5})$$



- Loss Function : Cross-Entropy

Transformer (Output)



Reference

- Attention Is All You Need
  (https://arxiv.org/abs/1706.03762)

- 딥러닝을 이용한 자연어 처리 입문
  (https://wikidocs.net/31379)

- NLP in Korean
  (https://nlpinkorean.github.io/illustrated-transformer/)