M8. Transformer

Motivation

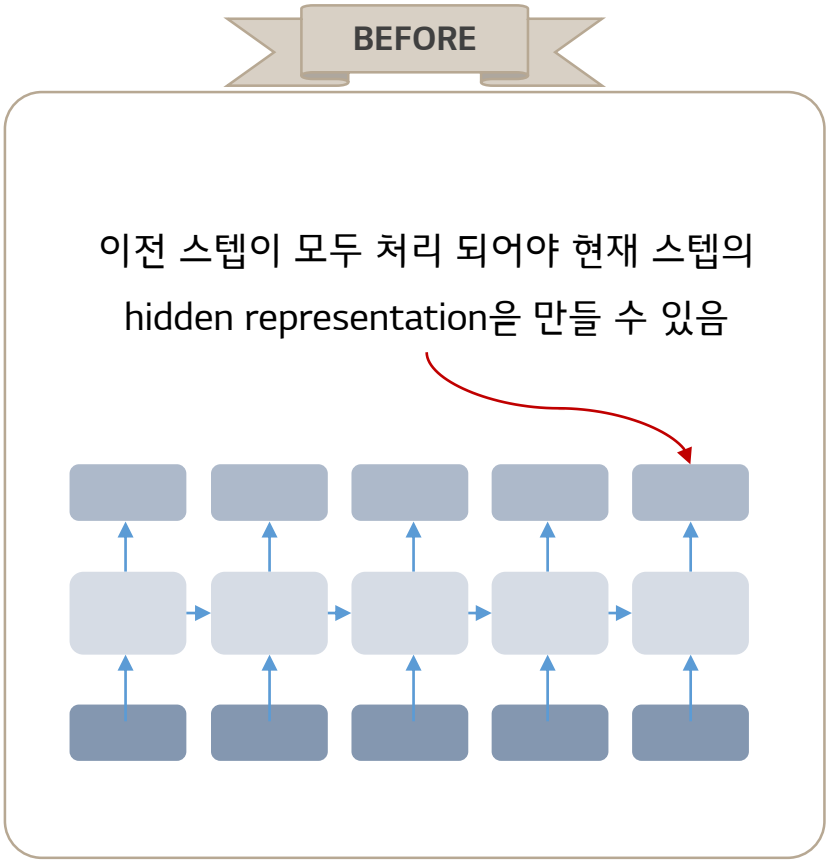
- GRU, LSTM 등의 등장에도 불구하고, RNN은 여전히 long-term dependency를 해결하기 위해 attention 이용해야 한다 → 먼 거리에 있는 hidden state과의 shortcut 필요

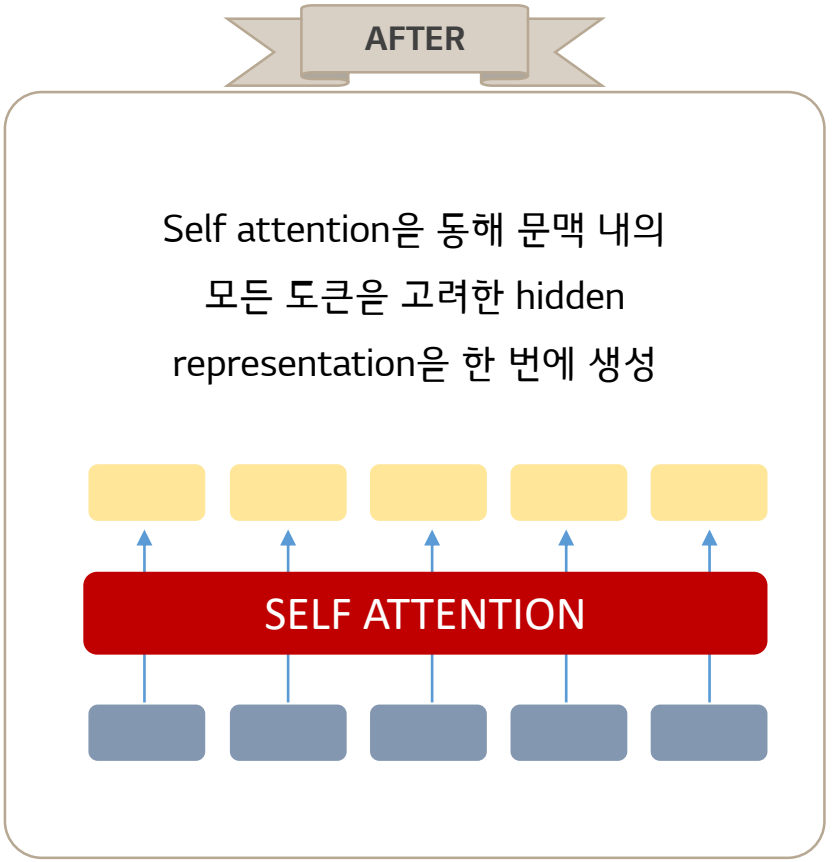
- 하지만 RNN에는 sequential한 입력 값이 주어지기 때문에 병렬 처리가 불가능하며, 모든 타임 스텝에서 hidden이 계산된 후에야 attention 진행 가능하다 → **느림**

- Seq2Seq 모델에 RNN이 필요 없다면...? → "Attention Mechanism"으로 연결하자!

Transformer: RNN이나 CNN 없이 attention만으로 인풋을 연결한 구조

- Transformer 구조를 제안한 “Attention is All you Need”는 2017년에 발표된 가장 흥미로운 논문 중 하나!
- Transformer에서는 Self attention을 사용해 Recurrent Unit 없이도 문장을 모델링 할 수 있다.

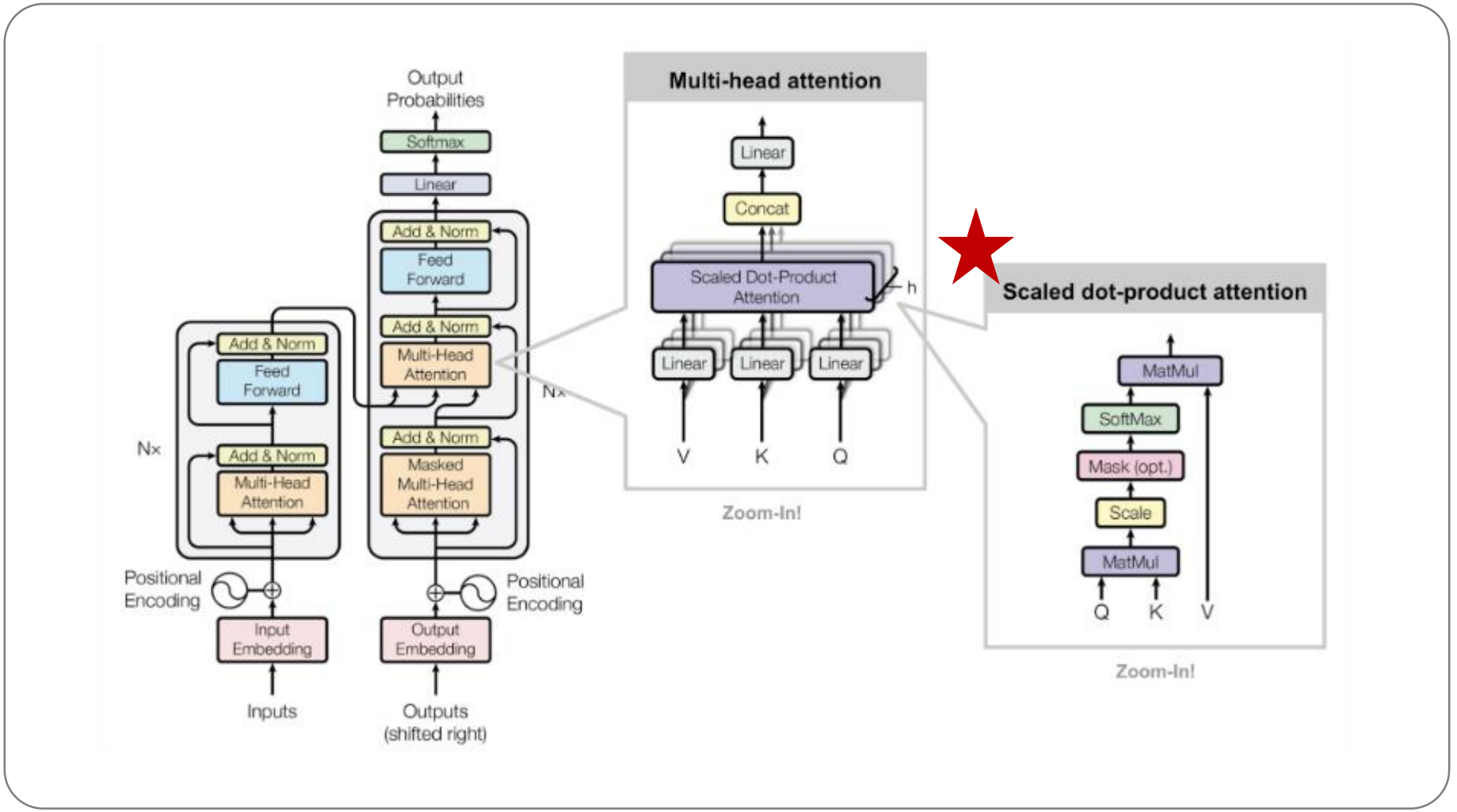




Paper : Attention is All You Need

Transformer: RNN이나 CNN 없이 attention만으로 인풋을 연결한 구조

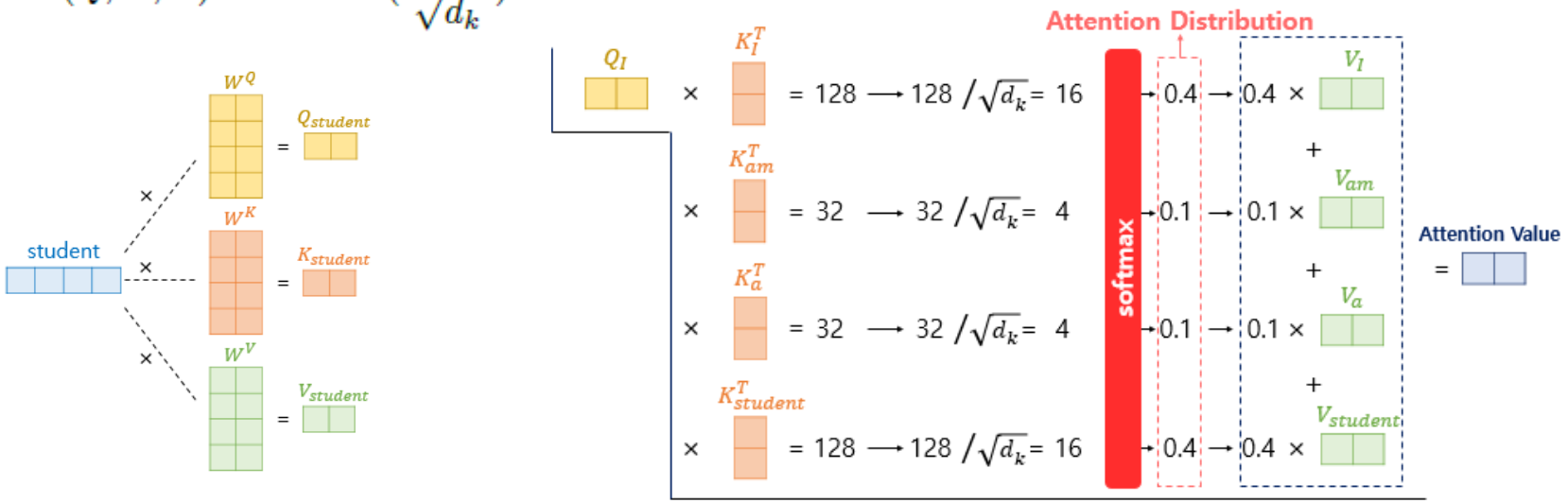
- 핵심은 multi-head self-attention에서 사용하는 scaled dot-product attention



Scaled dot-product attention

Input : I am a student

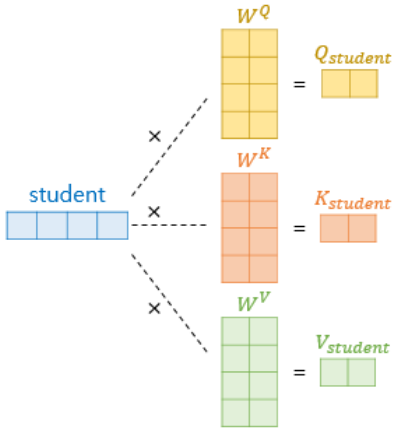
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Scaled dot-product attention

Input : I am a student

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Query : self attention의 주체.
'student'라는 단어에 대한 representation이 생성됨

Key : query와 문장의 각 도큰 사이의 attention score을 계산하기 위한 벡터.

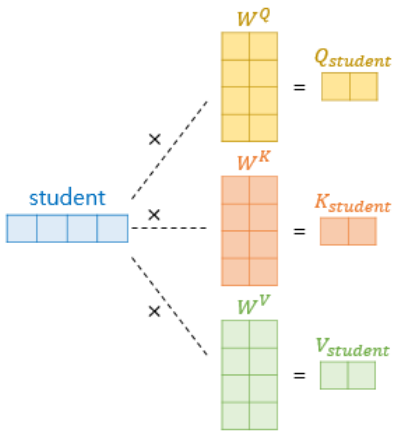
Value : 각 도큰에 대한 새도운 representation
Attention output은 value에 attention score을 곱하여 계산됨.

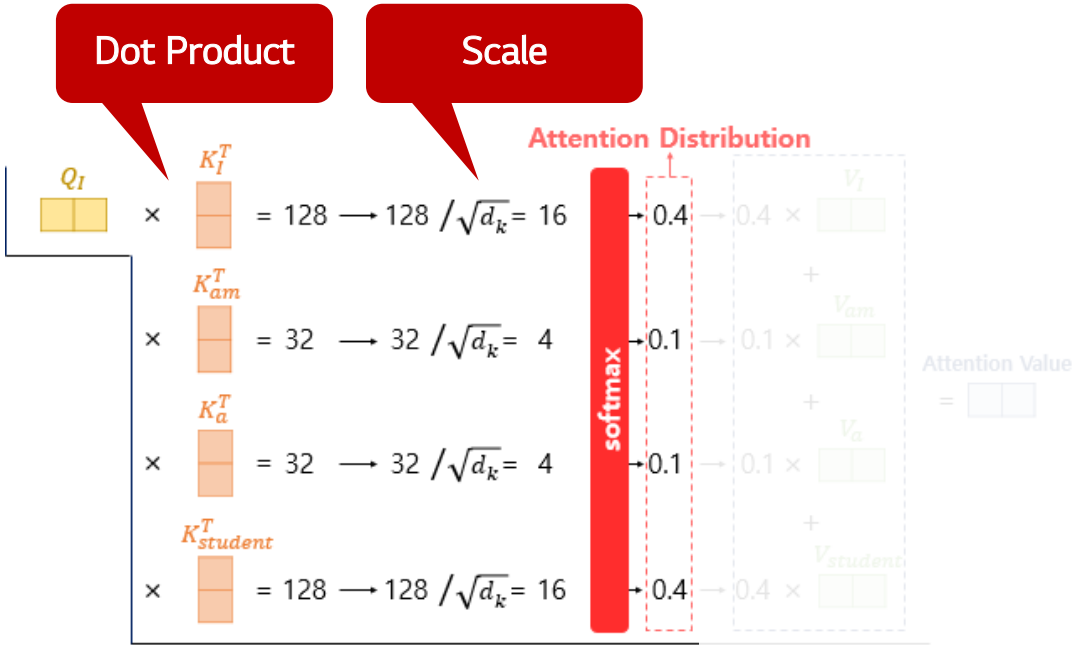
STEP 1.
Attention 대상이 되는 토큰들을
key와 value,
attention하는 토큰을 query로 변환
(행뎔 곱)

Scaled dot-product attention

Input : I am a student

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$





STEP 1.

Attention 대상이 되는 토큰들을 key와 value, attention하는 토큰을 query로 변환 (행렬 곱)

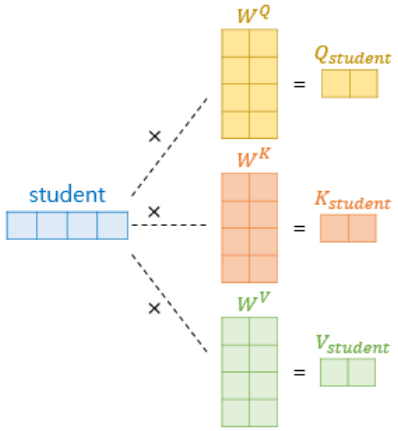
STEP 2.

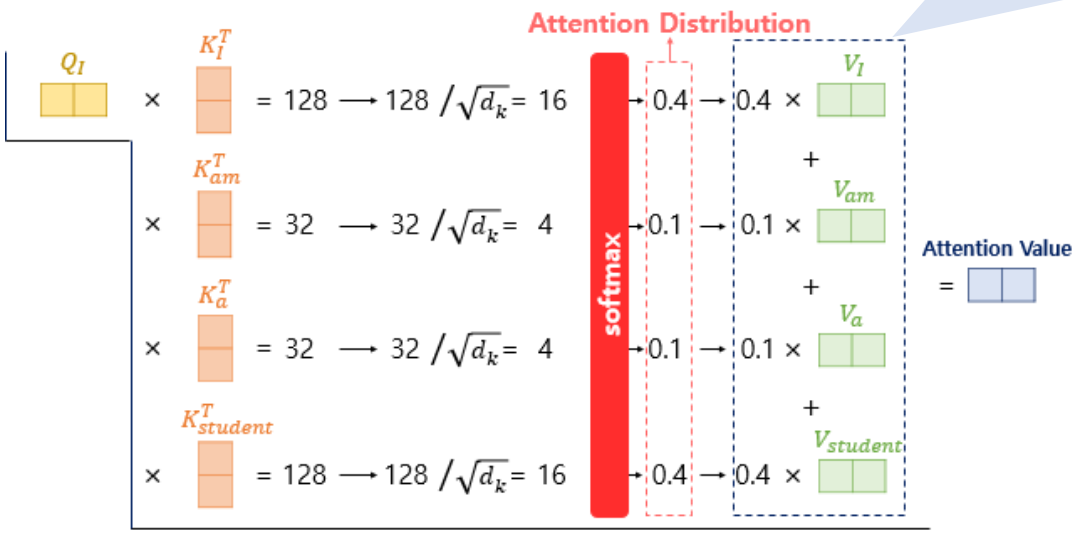
Query에 대해 각 key들과의 내적을 통해 attention 가중치 계산.
이 때 scale된 벡터 내적에 softmax를 취하는 방식으로 '확률 분포'와 같이 만듦.

Scaled dot-product attention

Input : I am a student

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$





STEP 1.
Attention 대상이 되는 토큰들을 key와 value, attention하는 토큰을 query로 변환 (행렬 곱)

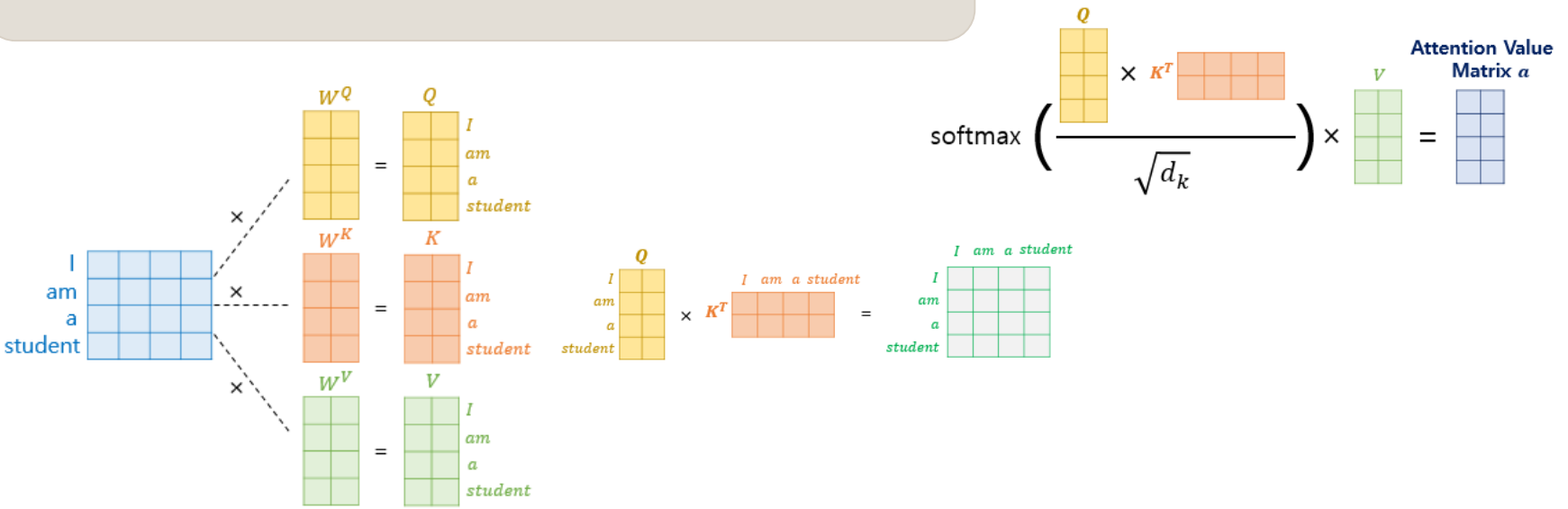
STEP 2.
Query에 대해 각 key들과의 내적을 통해 attention 가중치 계산.
이 때 scale된 벡터 내적에 softmax를 취하는 방식으로 '확률 분포'와 같이 만듦.

STEP 3.
가중치를 이용해 value를 가중 합 하여 query의 representation을 업데이트

Self attention의 의미

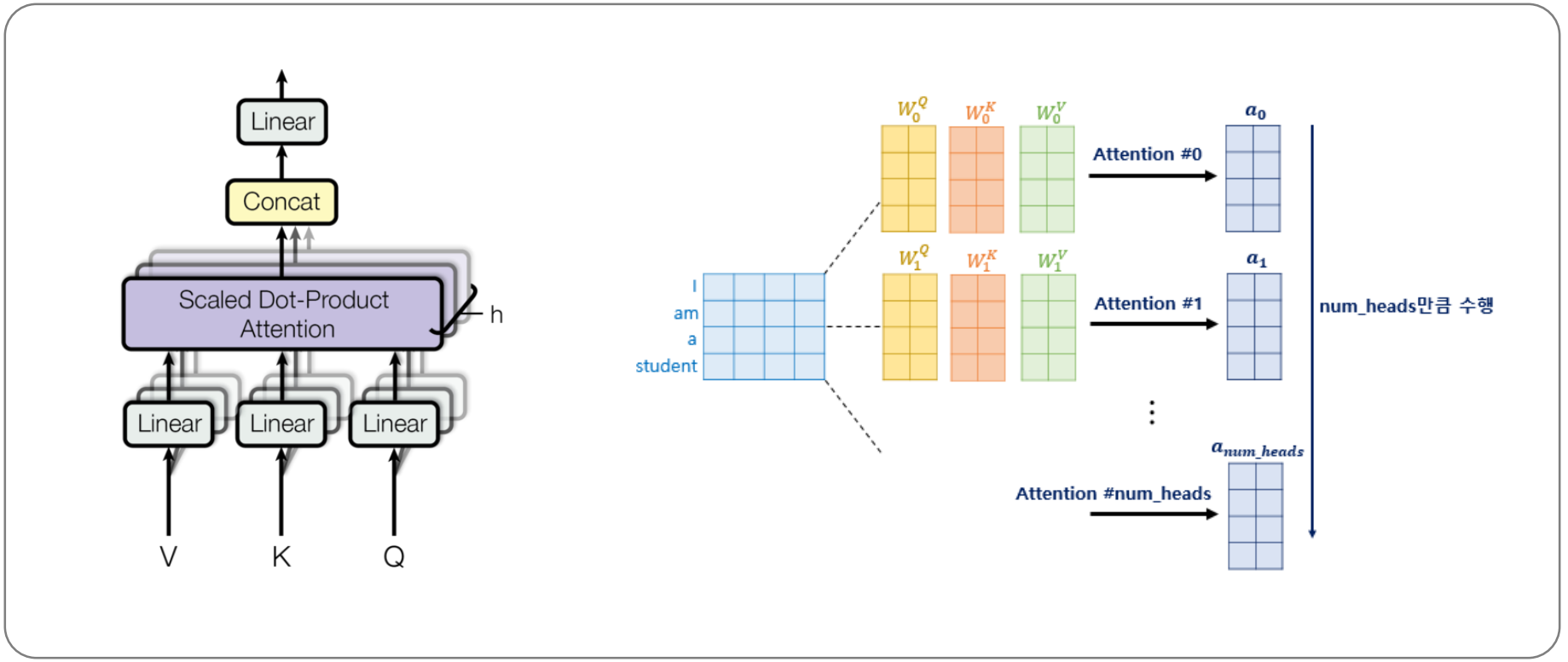
- Self attention은 인풋 시권스 전체에 대해 attention을 계산해 각 도큰의 representation을 만들어가는 과정으도, 업데이트된 representation은 **문맥 정보**를 가지고 있다.
- 예를 들어 “아이유는 1993년에 태어났다. 그녀는 최근에 드라마 호텔 델루나에 출연했다” 라는 인풋에 대해 self-attention을 적용하면 “그녀”에 해당하는 representation은 “아이유”에 대한 정보를 담게 된다.

Scaled dot-product attention은 matrix로 계산할 수 있기 때문에 RNN처럼 이전 토큰이 처리되길 기다릴 필요가 없음!



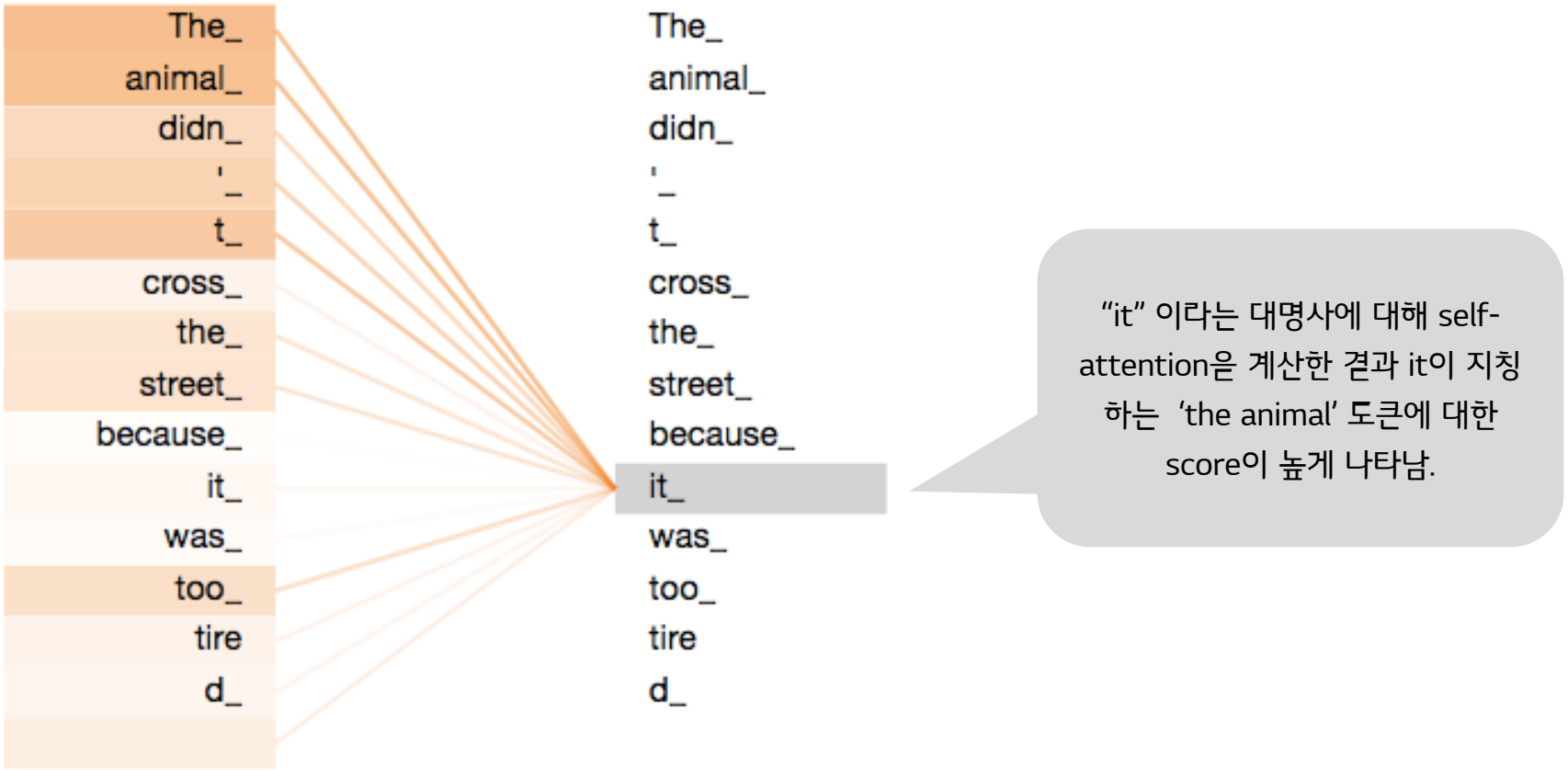
Multi head attention

- Scaled dot-product attention은 한 번에 계산하는 것이 아니라 여러 개의 head를 이용해 계산함.
- 즉, attention 계산 과정은 여러 weight를 사용해 반복하고 그 결과를 concat하여 최종 attention output 계산
- 이는 CNN filter을 여러 장 사용함으로써 이미지에 있는 다양한 특성을 포착하는 것처럼,
  도큰 사이의 다양한 관계를 포착하기 위함임.



Transformer Self-attention 예시

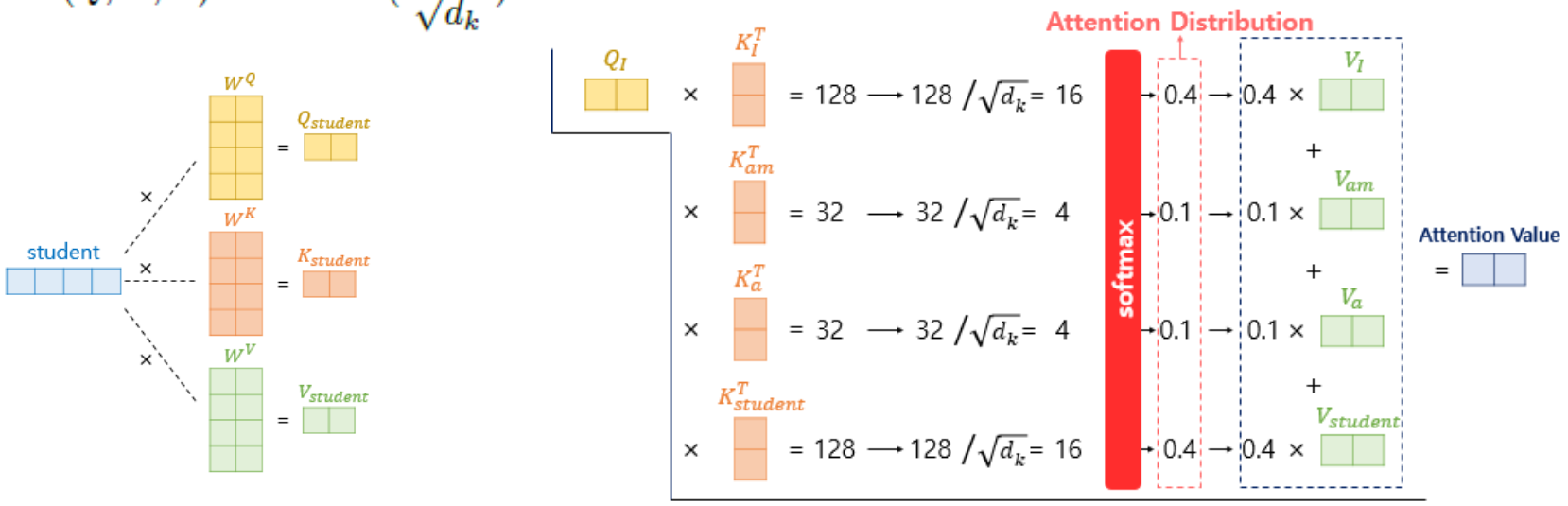
- “The animal didn’t cross the street because it was too tired”
- 라는 문장에 Transformer 구조를 이용해 self attention 적용



Positional Encoding

Input : I am a student

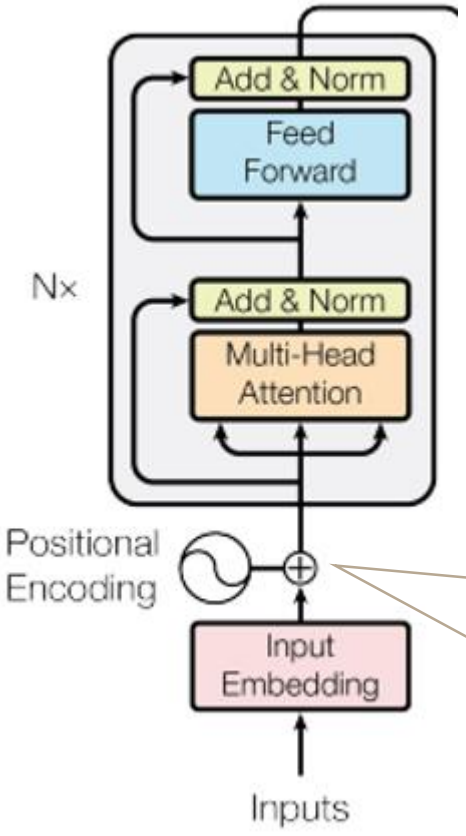
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



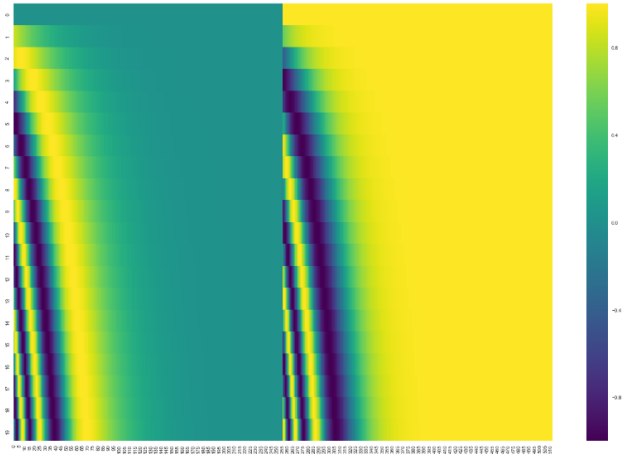
쿼리 토큰에 대해 key 토큰과의 attention score을 구해 attention value를 만드는 과정,
이상한 점이 보이시나요?

Positional Encoding

- 토큰 임베딩에 위치 정보를 나타내는 positional encoding을 만들어 더해준다



sin & cos 함수를 사용해
위치에 따른 인코딩을 만들어냄

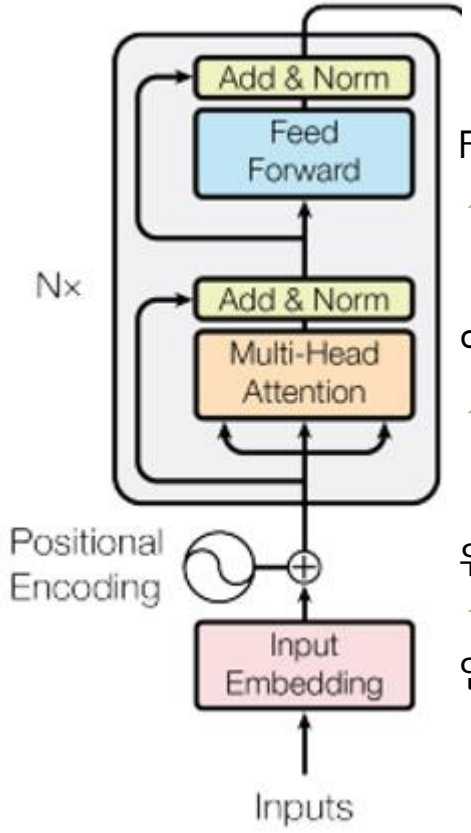


$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

토큰 위치

i번째 차원

Transformer 인코더 구조 summary



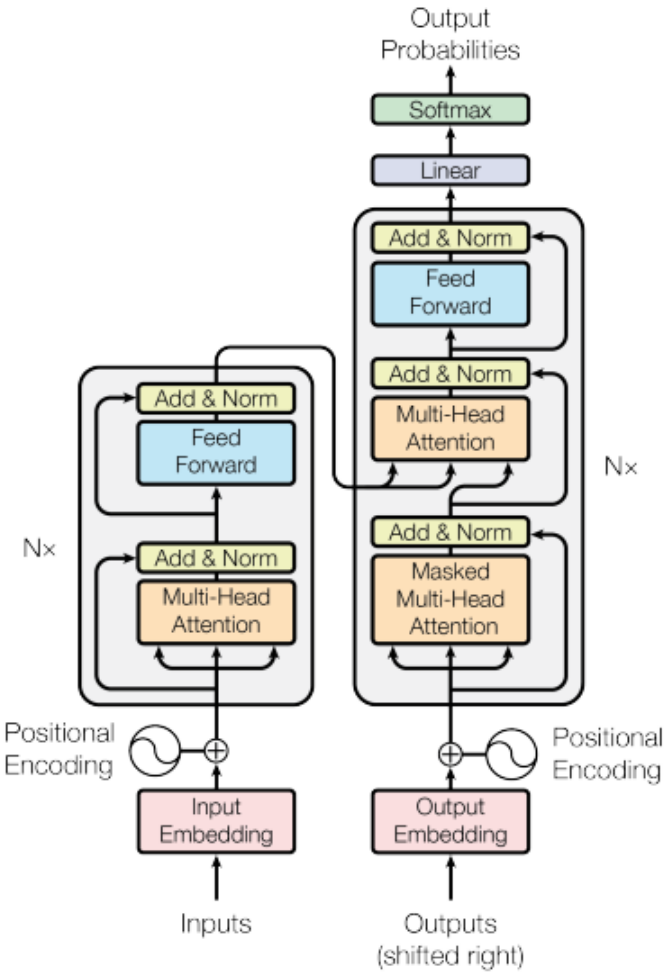
Fully-Connected Network를 이용해 정보를 변환

이에 대해 Multi-Head attention을 적용하고

위치 정보를 더해준 후

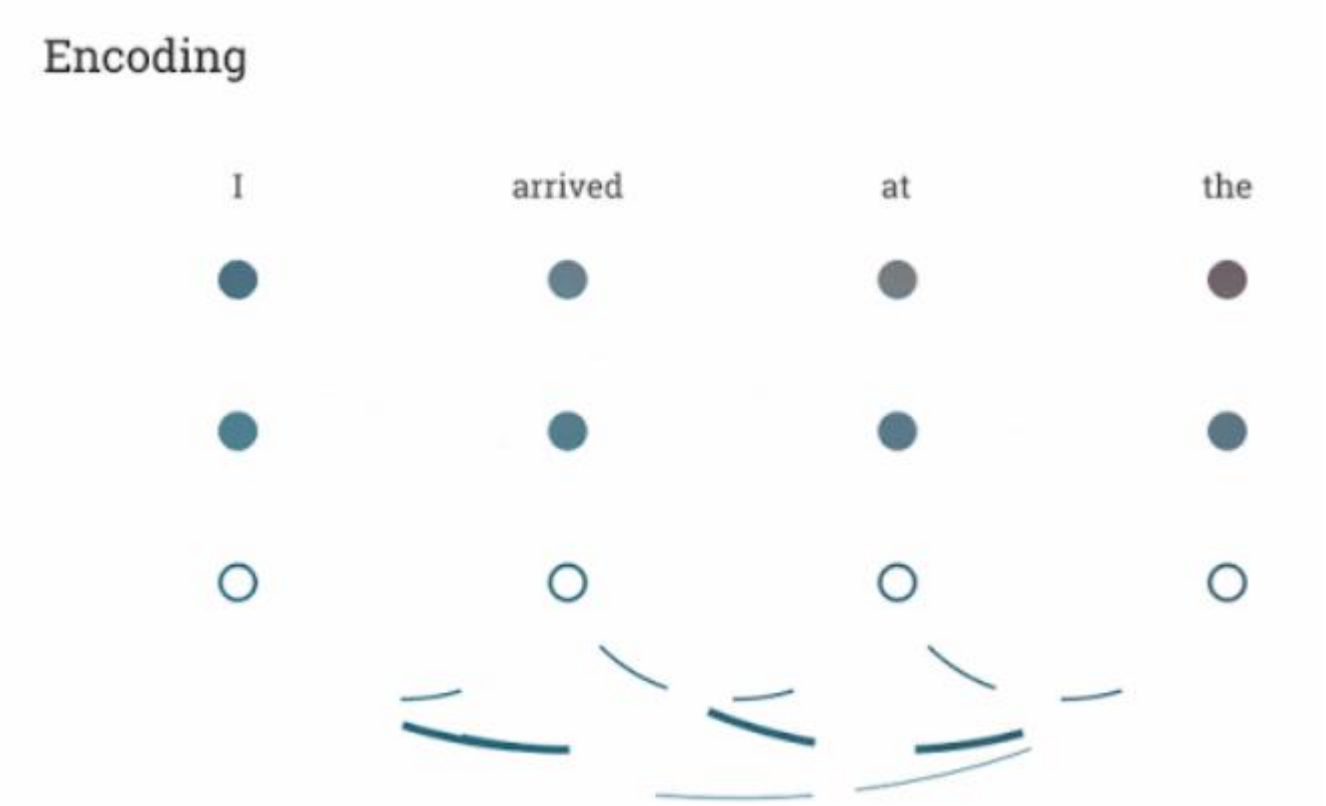
인풋 시권스를 임베딩하고

Seq2Seq를 Transformer도 모델링하기



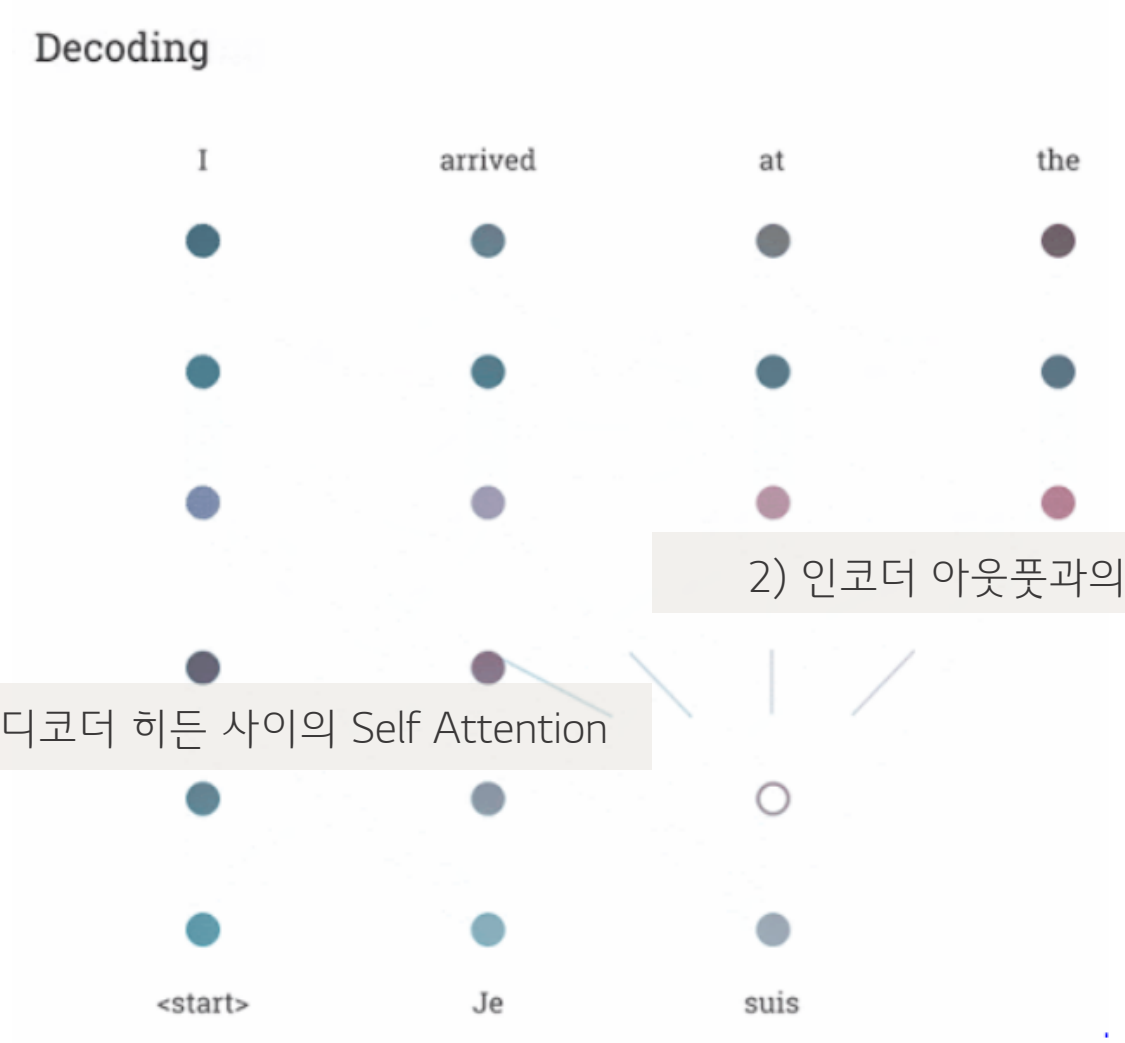
- 인코더에서는 인풋 문장 안에 있는 도큰들간의 관계를 고려하는 Self-attention 사용
- 디코더에서는 디코더 히든과 인코더 히든들간의 attention을 고려해 도큰을 예측함
- 애니메이션으도 이해하기 >>
- https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Seq2Seq를 Transformer도 모델링하기



- 인코딩 단계에서는 [Self-attention + Feed Forward] 레이어를 여러 층 쌓아 인풋 토큰들에 대한 representation 생성

Seq2Seq를 Transformer도 모델링하기



2) 인코더 아웃풋과의 attention

1) 디코더 히든 사이의 Self Attention

Transformer 특징 & 장점 summary

- RNN을 통해 각 타임스텝의 hidden state이 계산되기를 기다리지 않아도 된다.
- 즉, 문장에 있는 모든 단어의 representation들을 병렬적으로 한번에 만들 수 있다 → 빠르다!

- 기존의 RNN, CNN 방식을 과감히 포기하고 FNN과 Skip connection만을 이용했다는 점에서 획기적

- GRU, LSTM 같은 아키텍처 없이도 Long-term dependency를 해결한 새로운 방식
- **병렬화**와 **학습 시간 단축**에 기여

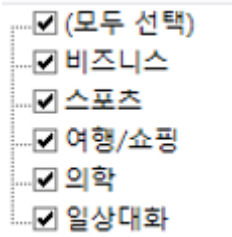
- 각종 기계번역 대회에서 세계 최고의 기록 보유 ! ! (WMT 2014 등)

Transformer Self Attention 이해하기

실습 6_Transformer.ipynb

- **데이터** : 대화체 데이터 100,000건

원문	번역문
이번 신제품 출시에 대한 시장의 반응은 어떤가요?	How is the market's reaction to the newly rele...
판매량이 지난번 제품보다 빠르게 늘고 있습니다.	The sales increase is faster than the previous...
그렇다면 공장에 연락해서 주문량을 더 늘려야겠네요.	Then, we'll have to call the manufacturer and ...
네, 제가 연락해서 주문량을 2배로 늘리겠습니다.	Sure, I'll make a call and double the volume o...
지난 회의 마지막에 논의했던 안건을 다시 볼까요?	Shall we take a look at the issues we discusse...



- **학습 목표** :
  - Transformer을 이용한 Seq2Seq 모델 구조를 이해한다.
  - TF Keras를 이용해 Transformer 인코더-디코더 구조로 번역 모델을 학습/ 추론한다.