Attention Is All You Need

Attention, Transformer

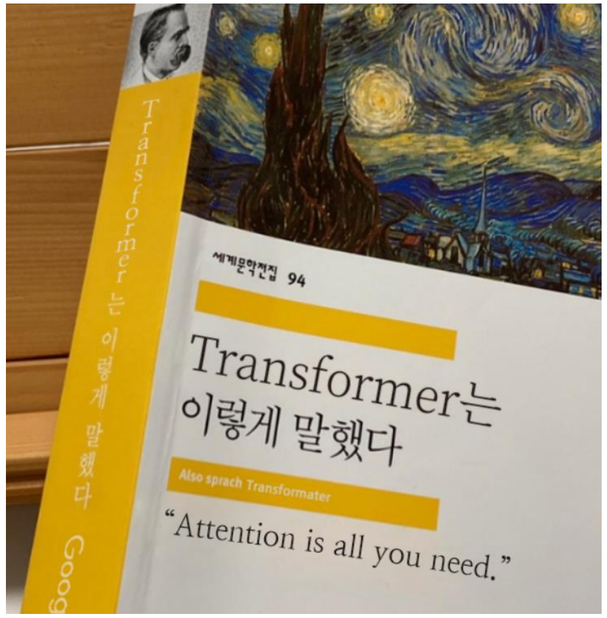
- Google Brain <2017.06.12>
- Repeat Count: 90000+

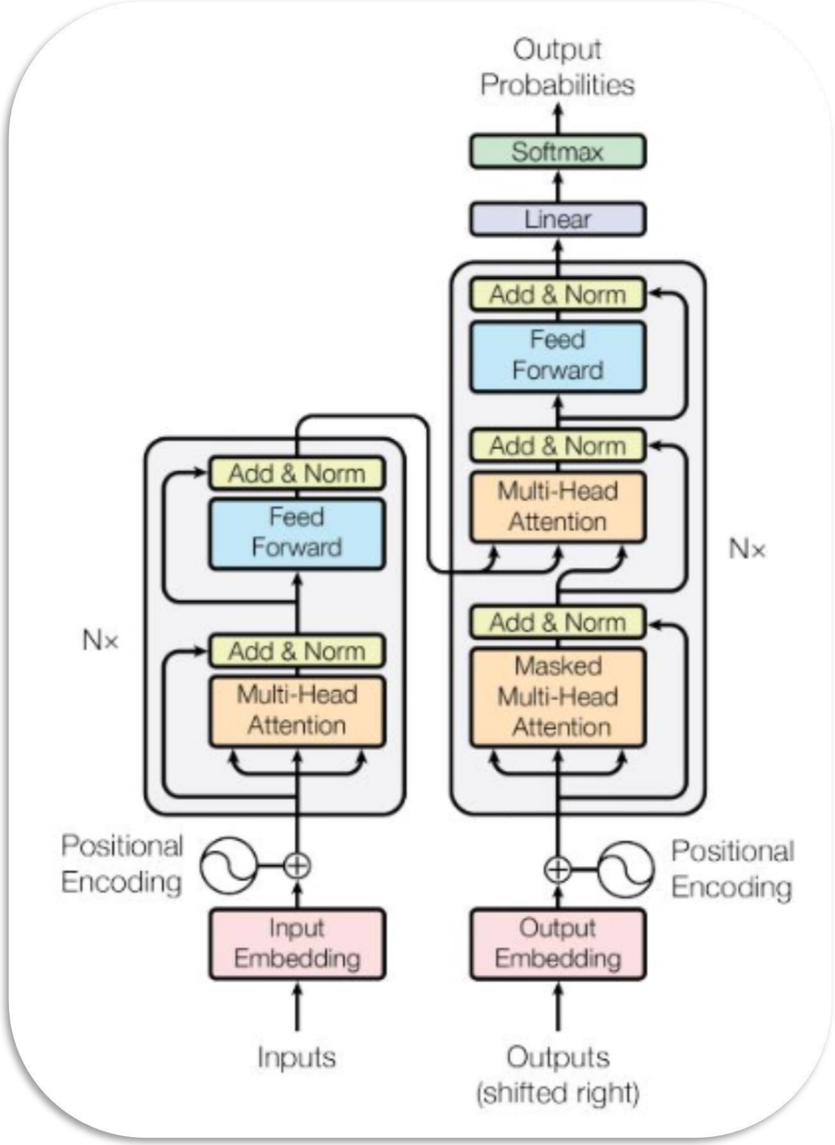
백 대 환

PaperReview

OVERVIEW

- Transformer는 이렇게 말했다, “Attention is all you need”
- Attention ?
- Transformer ?
- BERT, ChatGPT
- NLLB-200





OVERVIEW

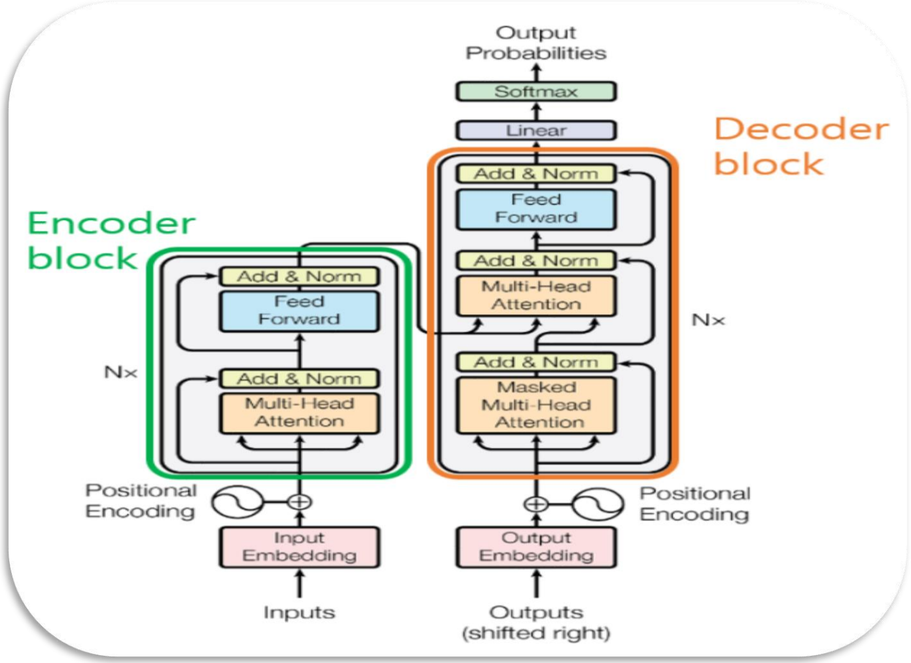
- Attention,

- 입력데이터의 특정 부분에 집중하여 해당 부분이 더 중요하게 반영되도록 하는 방법

- Transformer,

- 기존의 seq2seq의 구조인 인코더-디코더를 따르면서도, 논문의 이름처럼 어텐션(Attention)만으로 구현한 모델

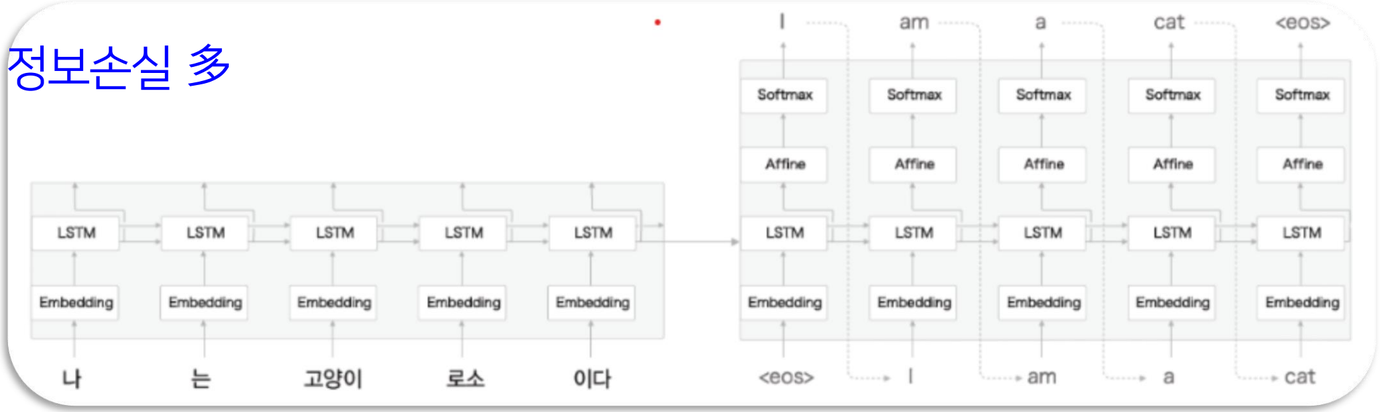
기존 Seq2seq 모델의 한계



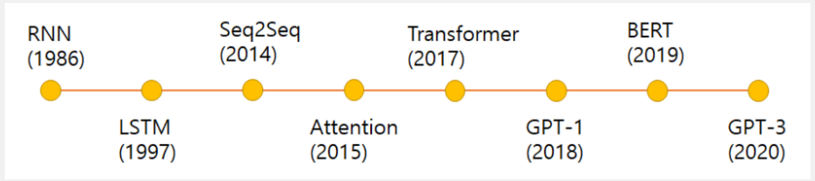
- 인코더-디코더 구조
- 인코더: 입력 시퀀스를 하나의 벡터(고정길이)로 압축
- 디코더: 인코더의 벡터 표현을 통해서 출력 시퀀스로 만듦

➔ 인코더가 하나의 벡터로 압축하는 과정에서 정보손실 多

➔ 이를 보정하기 위해 Attention 사용



0. Abstract



최고의 성능 모델은 Attention 메커니즘의 인코더와 디코더를 연결하는 것

Atttention 메커니즘 기반의 Transformer를 제안

어텐션 기법으로 행렬곱을 이용해서 병렬적으로 시퀀스 데이터를 처리
-> 전보다 훨씬 더 빠른 처리가 가능함

WMT 2014 data set을 이용해서 번역하는 작업에서
훨씬 개선된 성능을 보여줌

1,2. Introduction & Background

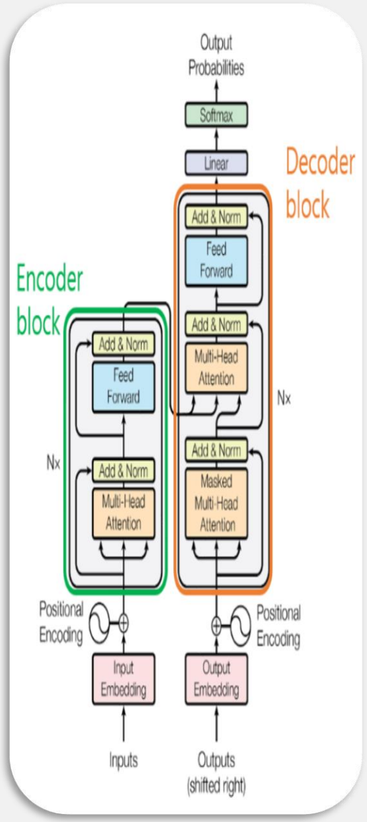
RNN, LSTM, GRM -> 병렬연산 불가능
Transformer -> 병렬연산 가능

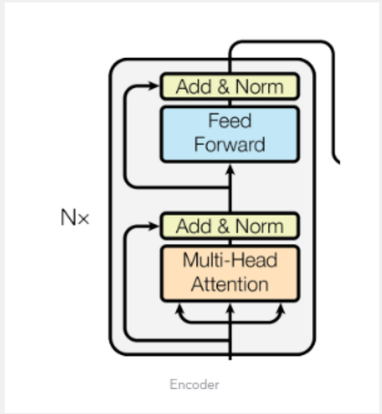
Ex) GPU 8개 연결하여 분석

RNN과 CNN을 사용하지 않고, self-Attention에 의존하는
최초의 변환 모델

병렬연산이란?
큰 문제를 여러 개의 작은 문제로 분할하여 풂.
= 작업 분할

3-0 Model Architecture

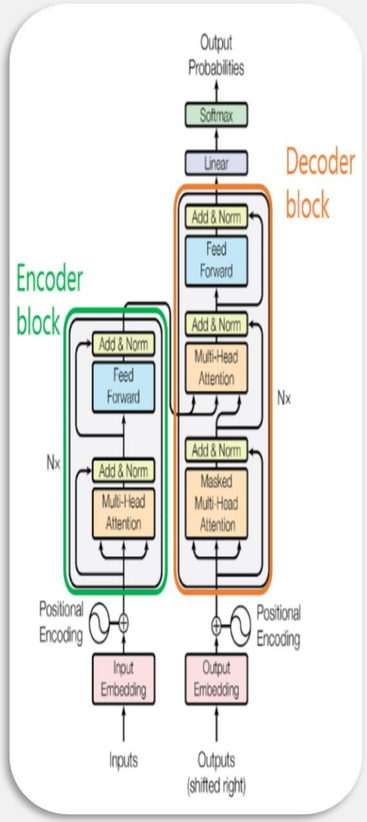


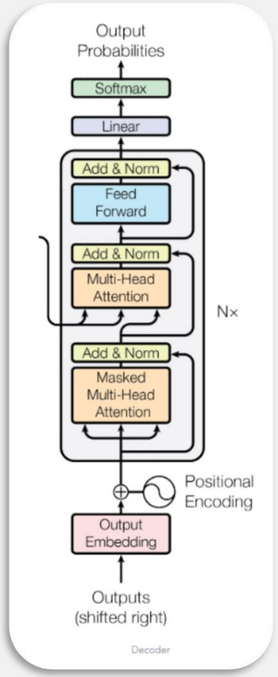


Encoder

- 6개의 동일한 레이어 스택(Nx 부분이 6번 반복) -> 하이퍼파라미터
- 각 레이어에는 Multi-Head Attention 두 개의 하위 계층
- Multi-Head Attention은 Input text 내에서 문맥 정보를 파악
  (encoder self-attention 이라고도 함.)

3-0 Model Architecture

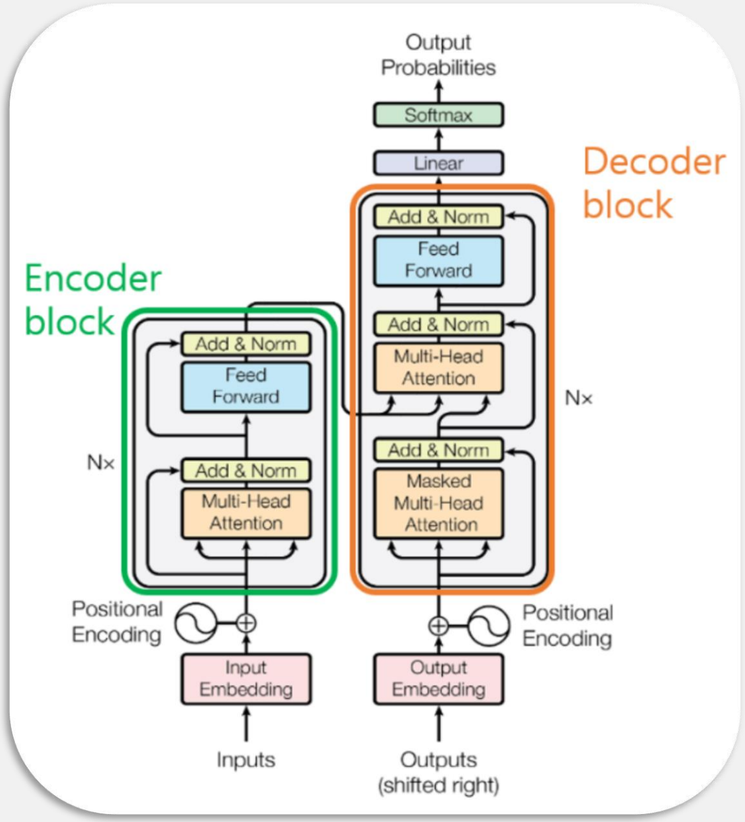




Decoder

- 6개의 동일한 레이어 스택(Nx 부분이 6번 반복) -> 하이퍼파라미터
- 각 레이어에는 Multi-Head Attention 세 개의 하위 계층
- Multi-Head Attention은 Input text 내에서 문맥 정보를 파악
  (encoder self-attention 이라고도 함.)
- Encoder 로 부터 직접적인 residual connection 을 입력받고 있다. 해당 입력은 당연히 인코더의 마지막 레이어로부터 입력받음

3-1 Model Architecture



Positional Encoding

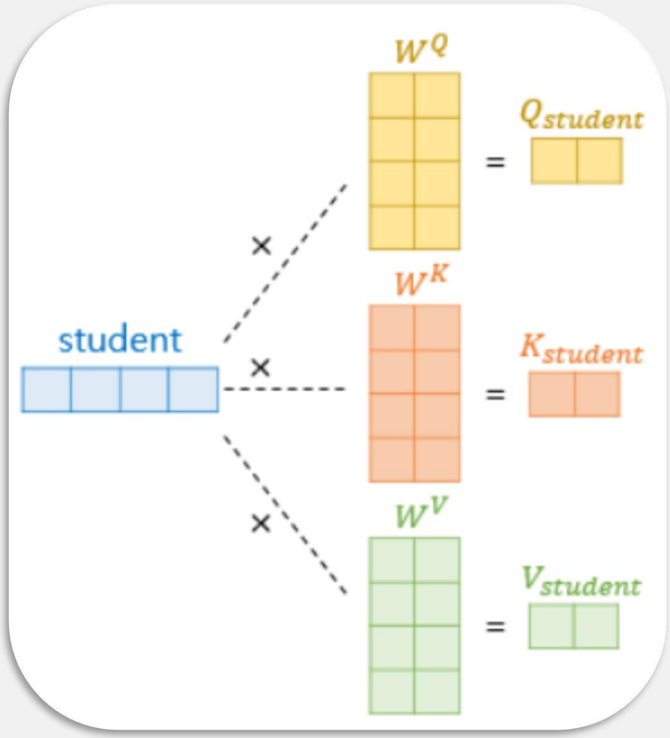
▶ RNN 계열의 모델들 - 단어별 순서가 자연스럽게 기록 -> 분석

-> Transformer - 순서를 주입 후 분석

- n : 자연어 처리에서의 '단어의 개수'
- d : 표현 차원
- k : Convolution Layer 의 View Size
- r : 말 그대로 self-attention 의 거리제한.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

3-1-1 Self-Attention



1. Self-Attention

▶ Query? 모든 시점의 디코더 셀에서의 은닉 상태들

-> 무언가를 물어보는 주체, 특정 단어가 다른 단어들과 어떤 연관성을 가지는지

▶ Key? 모든 시점의 인코더 셀의 은닉 상태들

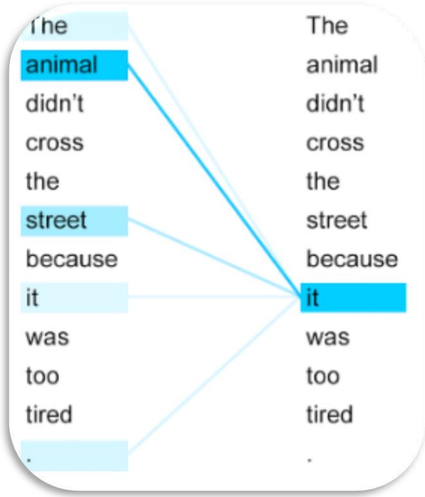
-> 비교될 단어 벡터(타겟)

▶ Value? 모든 시점의 인코더 셀의 은닉 상태들

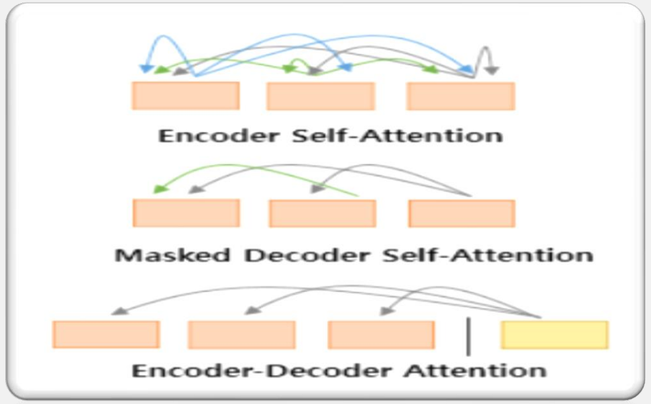
-> Key 값에 대해 학습된 가중치 벡터

d(model)의 차원에서 벡터를 가짐

(Q = K = V)



3-1-2 Scaled Dot-Product Attention



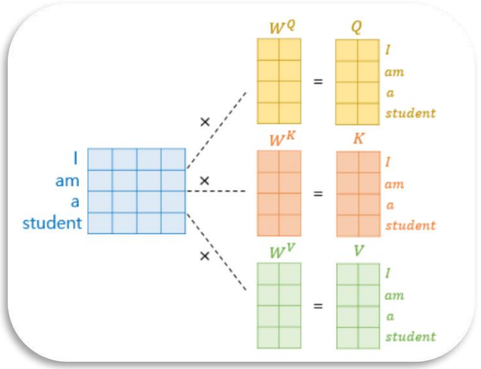
2. Scaled Dot-Product Attention

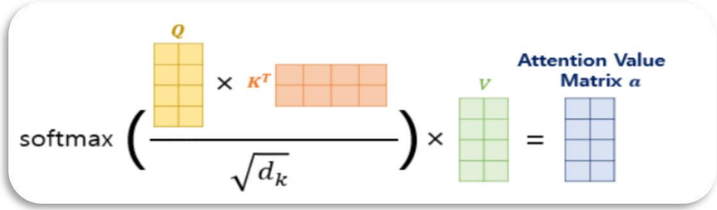
함수 score(q, k) = q · k (내적)

이 아닌 score(q, k) = $q \cdot k / \sqrt{n}$ <- 스케일링

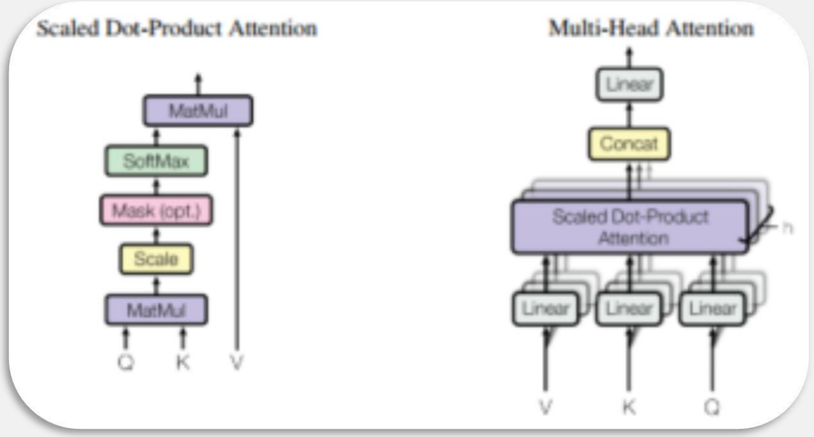
But, 일일이 연산한다는 문제점.

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



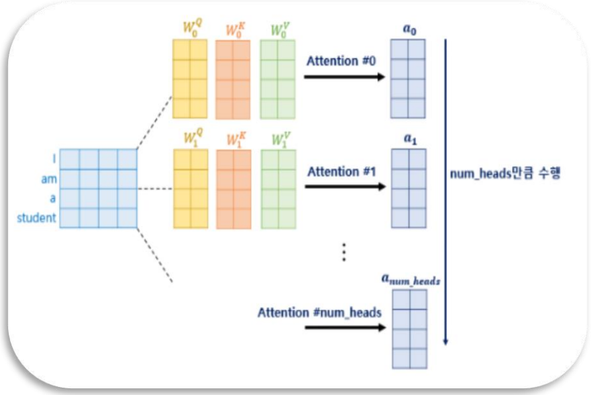


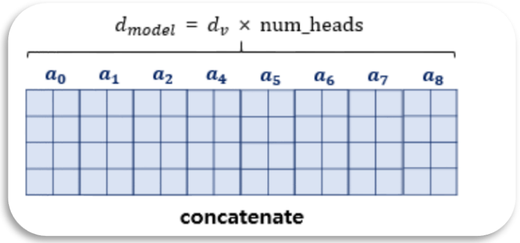
3-1-3 Multi-head Attention

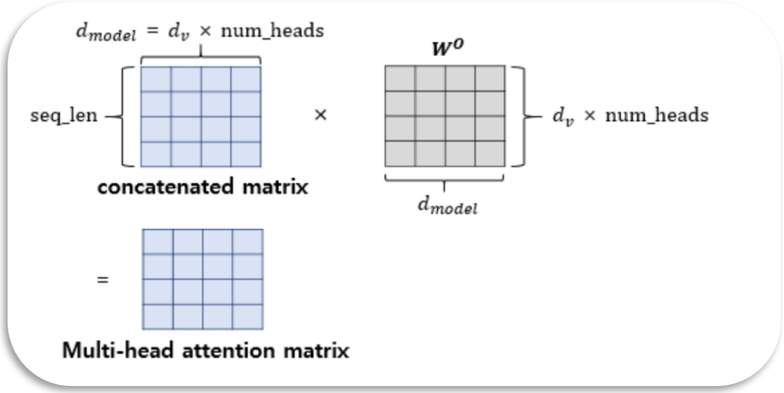


3. Multi-head Attention

하나의 attention function을 사용하는 것보다, queries와 keys, values를 linear projection을 통해 중간에 매핑해줘서 각 다른 값들을 입력으로 함.

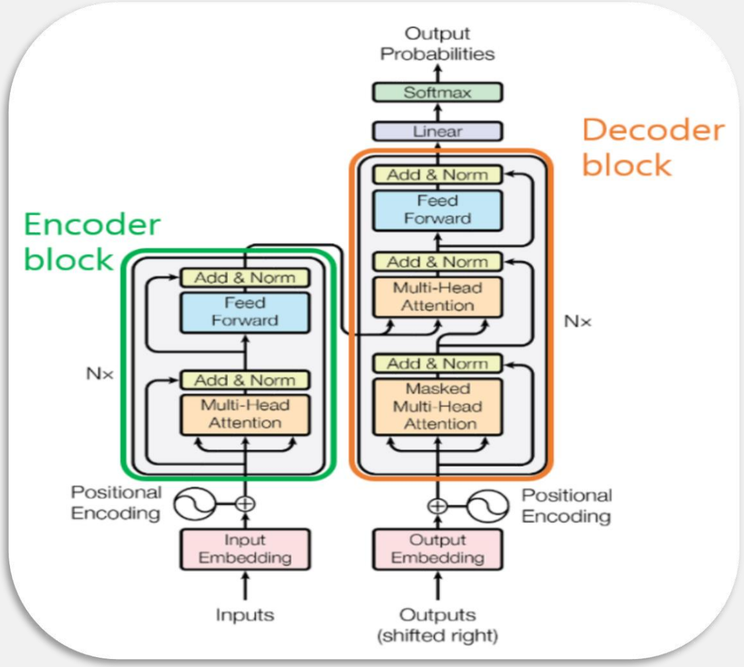






3-2

Feed Forward

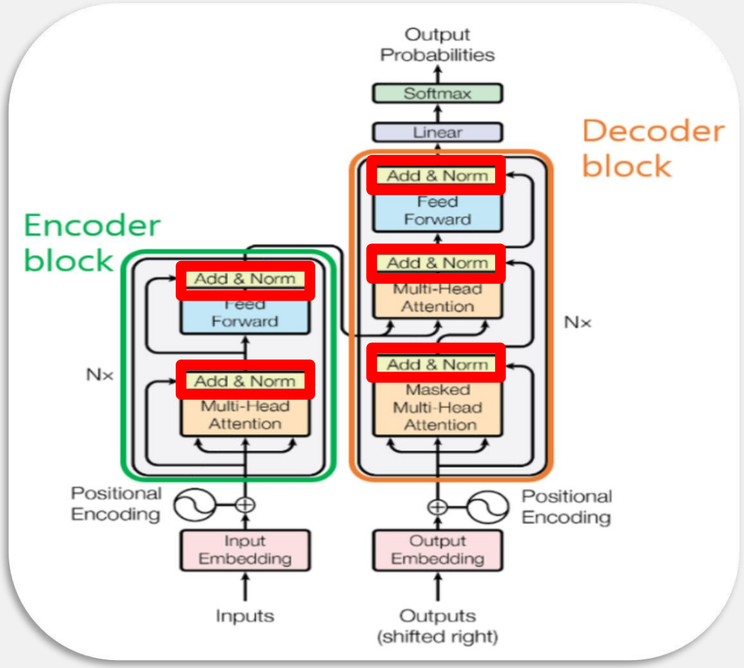


3-2. Feed Forward

- attention의 결과를 취합하여 전달하는 역할을 수행한다.

- feed-forward는 layer가 2개인 MLP
- multi-head attention으로 나온 다양한 정보들을 합쳐주는 역할

3-3
Add & Norm

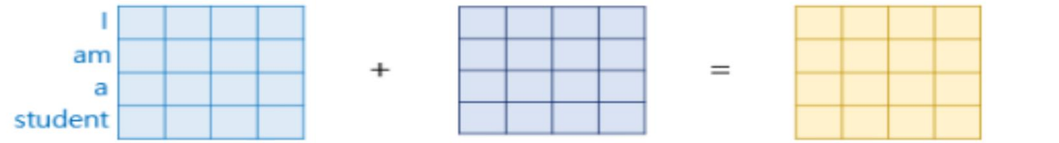


3-3. Add & Norm

잔차 연결(residual connection)과 층 정규화(layer normalization)

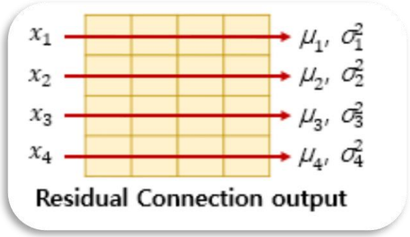
residual connection: CV 분야에서 주로 사용되는 모델의 학습을 돕는 기법

$$H(x) = x + Multi - head\ Attention(x)$$



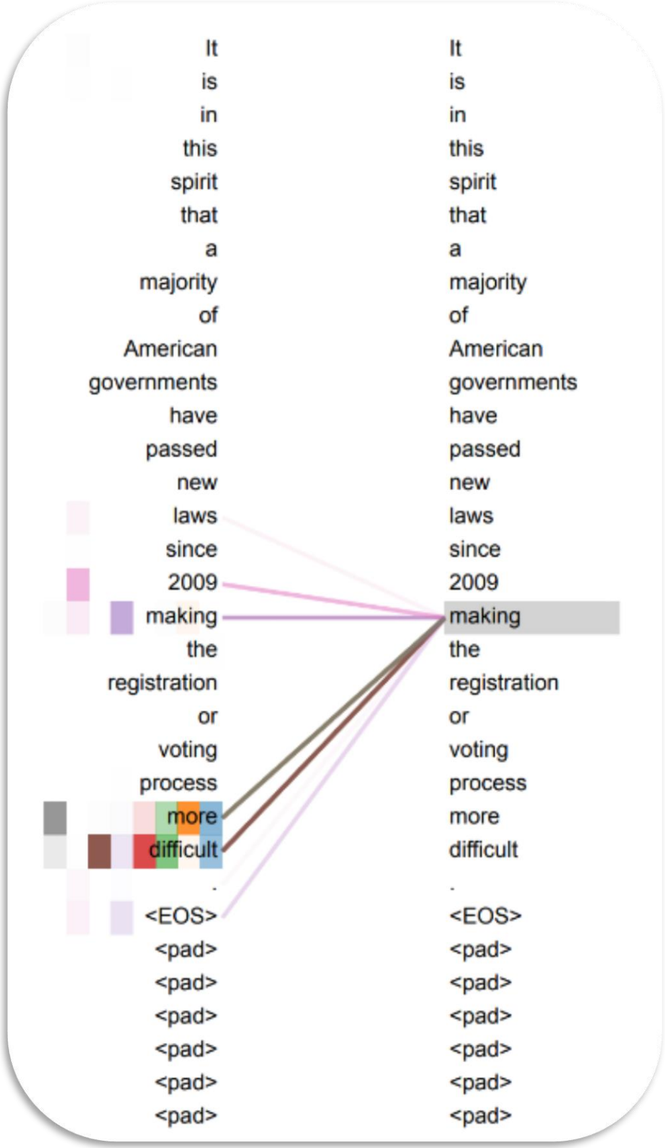
layer normalization: 텐서의 마지막 차원에 대해서 평균과 분산을 구하고,

이를 가지고 어떤 수식을 통해 값을 정규화하여 학습을 도움



$$ln_i = \gamma\hat{x}_i + \beta = LayerNorm(x_i)$$

Why Self-Attention?



Layer 당 전체 계산 복잡도

순차작업을 최소화하고,
병렬화 할 수 있는 계산량

네트워크 내부의 장거리 종속성 간의
경로 길이

Transformer