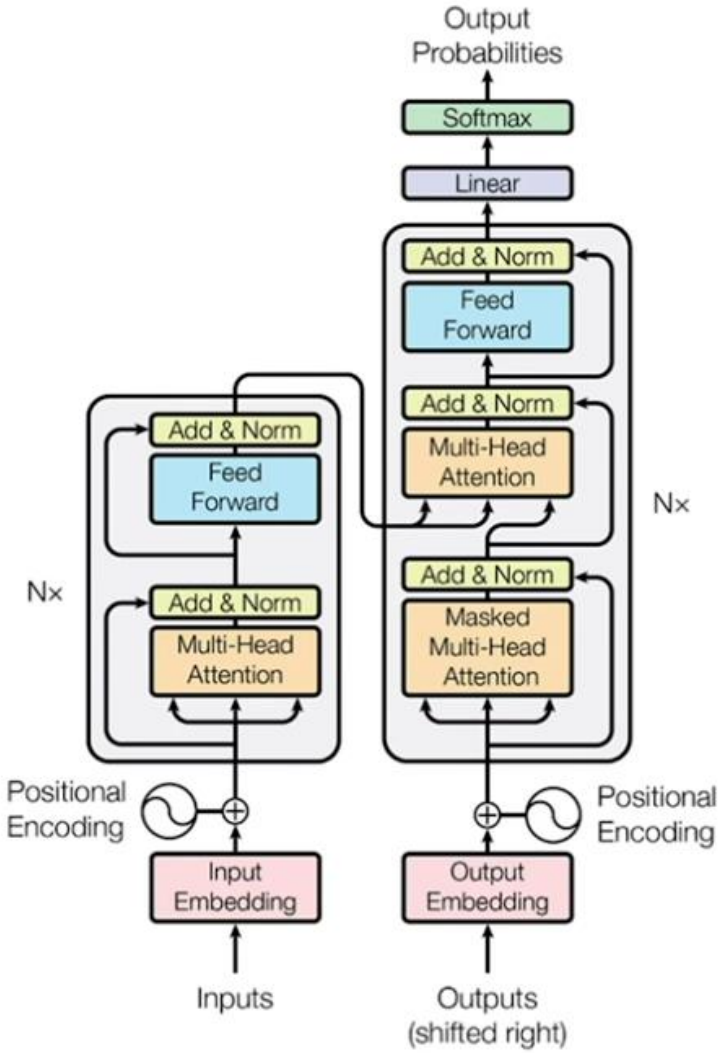
구글 BERT의 정석

트랜스포머

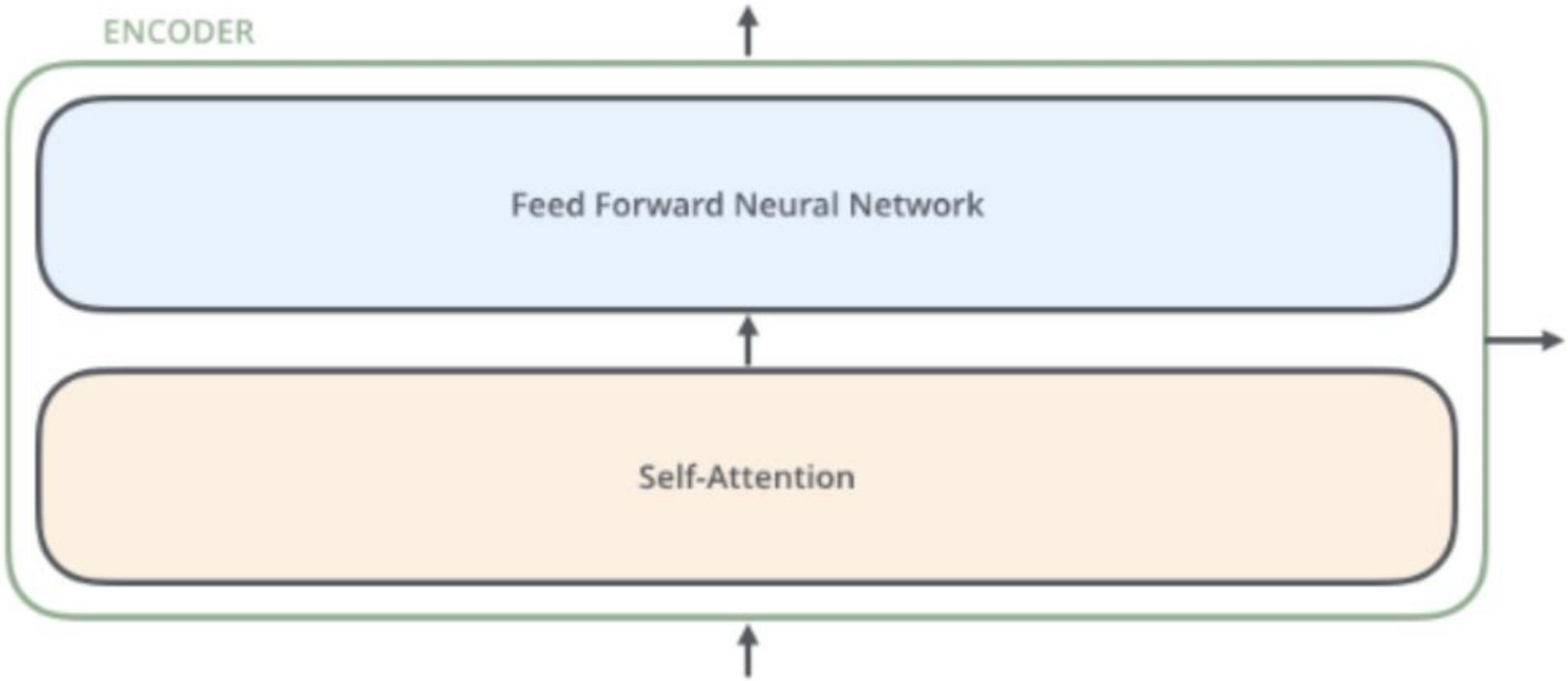
20180376 안제준

트랜스포머소개



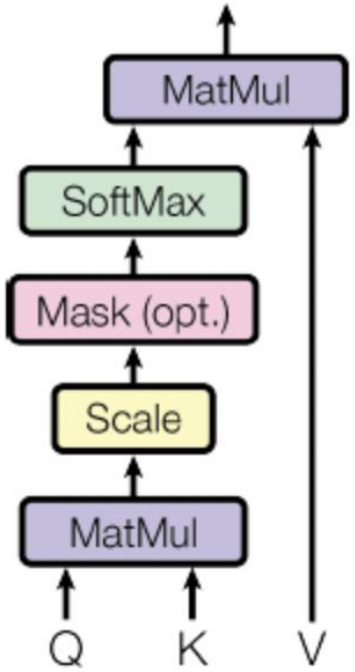
트랜스포머 -> 자연어 처리에서 주로 사용하는 딥러닝 아키텍쳐, 출현 이후 RNN과 LSTM을 대체했다.

트랜스포머의 인코더 이해하기



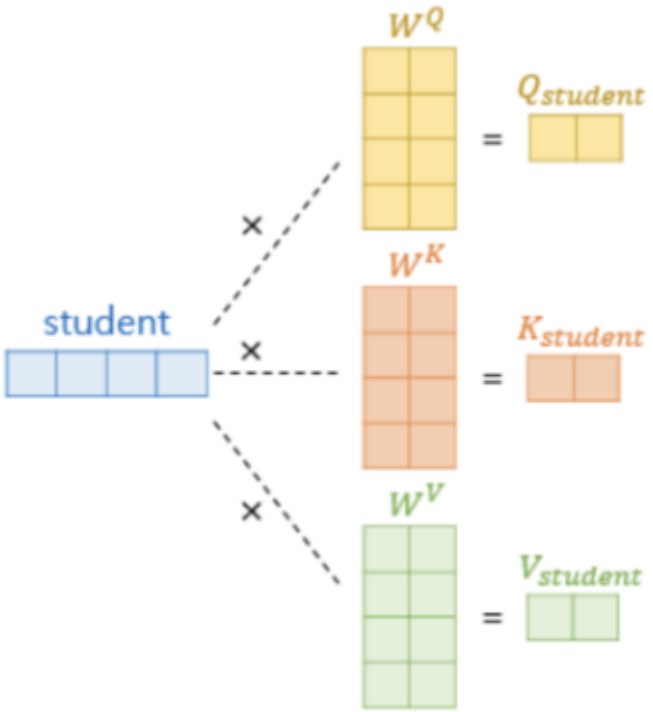
셀프 어텐션의 작동원리

Scaled Dot-Product Attention

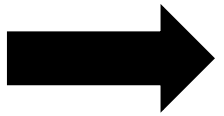


$$Attention(Q,\ K,\ V) = softmax(\frac{QK^T}{\sqrt{d_k}})\ V$$

셀프 어텐션의 내부 동작

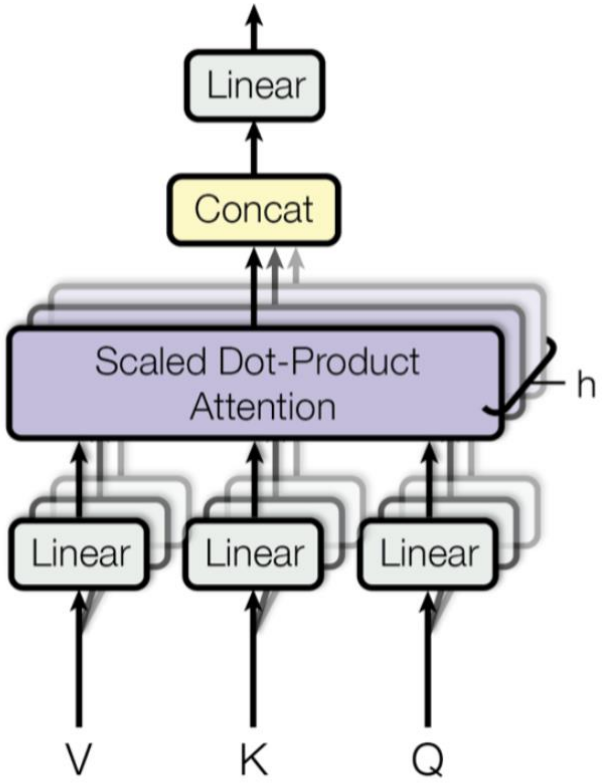


d_{model}/num_heads의 차원을 가지는 Q, K, V



$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V = Z$$

멀티 헤드 어텐션 원리



Multi Head Attention

$$\text{멀티헤드어텐션} = concatenate(Z_1, Z_2, ...Z_i, ...Z_s)W_0$$

Summary

- Previous method: attention in sequence to sequence
  - Query를 잘 만들어 key-value를 잘 matching 시키자

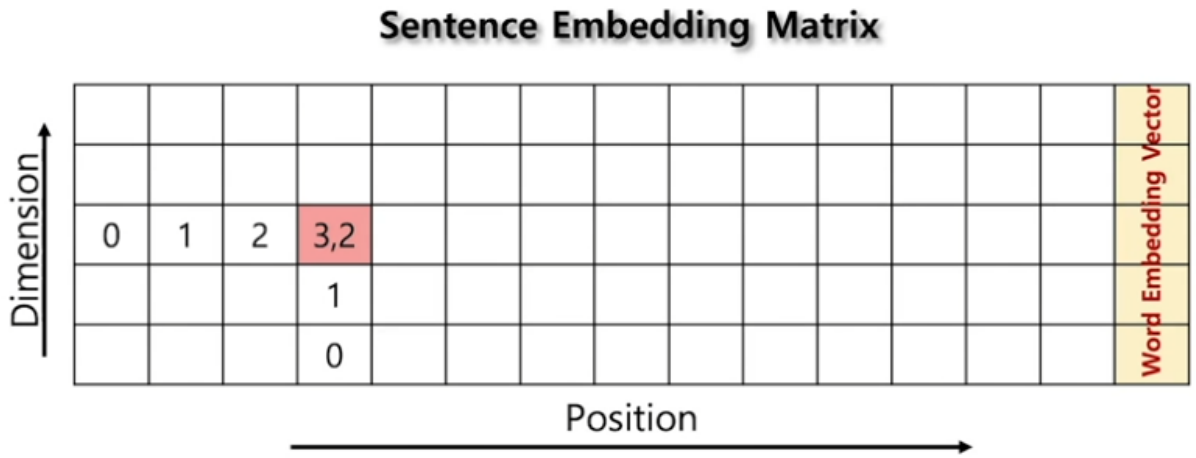
- Multi-head Attention
  - 여러 개의 query를 만들어 다양한 정보를 잘 얻어오자

- Attention 자체로도 정보의 encoding과 decoding이 가능함을 보여줌

위치 인코딩으로 위치 정보 학습

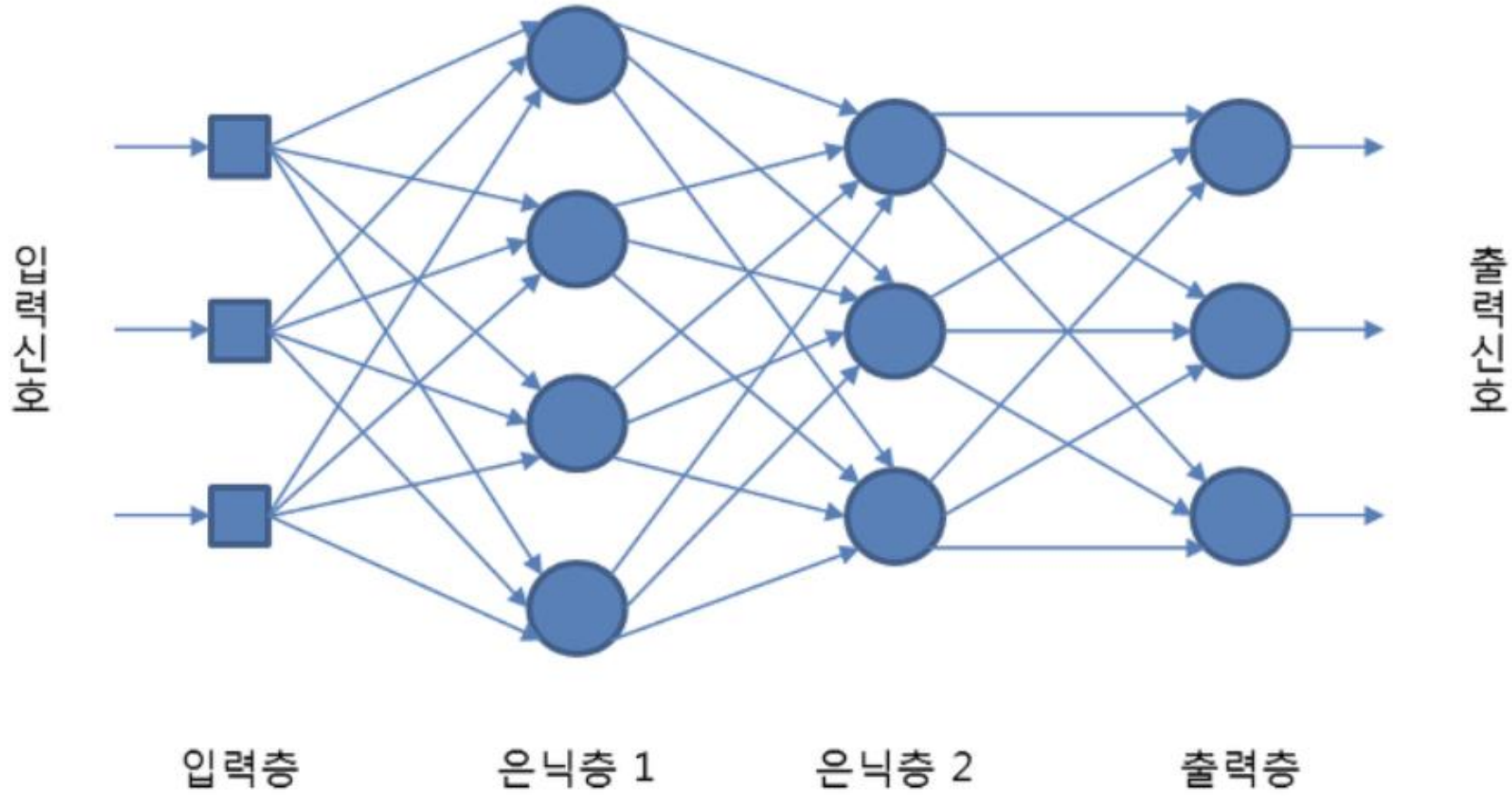
Positional Encoding

- 기존의 word embedding 값에 positional encoding 값을 더해줌

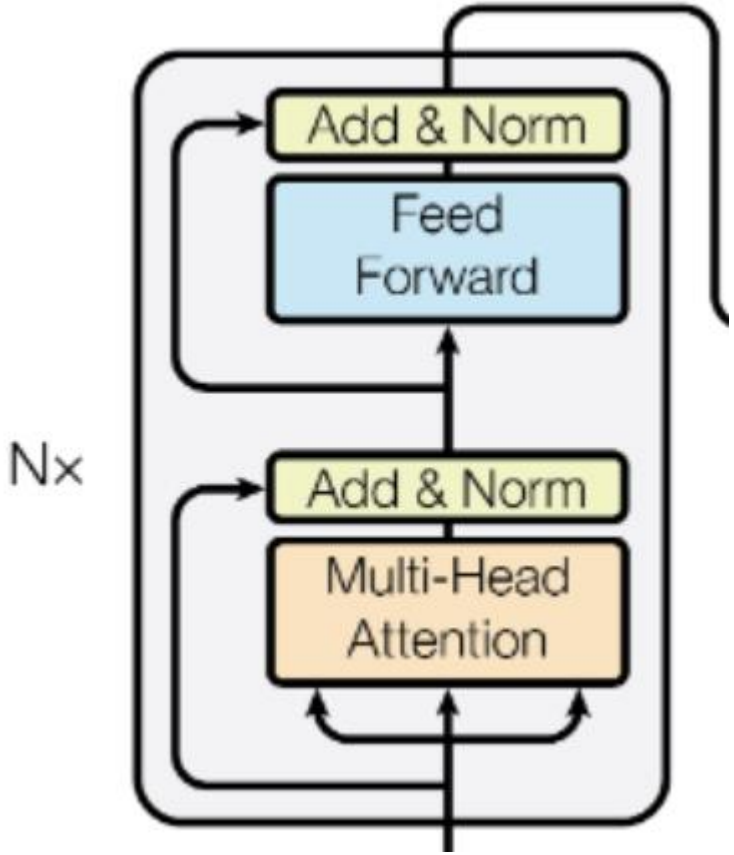


$$PE(pos = 3, dim\_idx = 2 = 2\times i)$$
$$= PE(pos = 3, i = 1)$$
$$= \sin\left(\frac{pos}{10^{4\times\frac{2\times i}{d}}}\right) = \sin\left(\frac{3}{10^{4\times\frac{2\times 1}{d}}}\right)$$

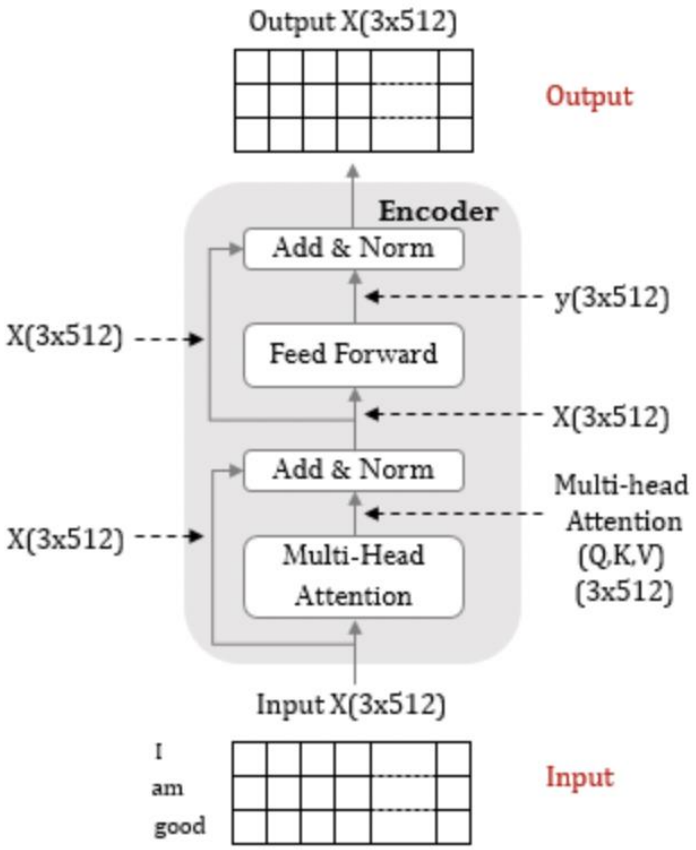
피드포워드 네트워크



add와 norm 요소



모든 인코더 구성 요소 통합



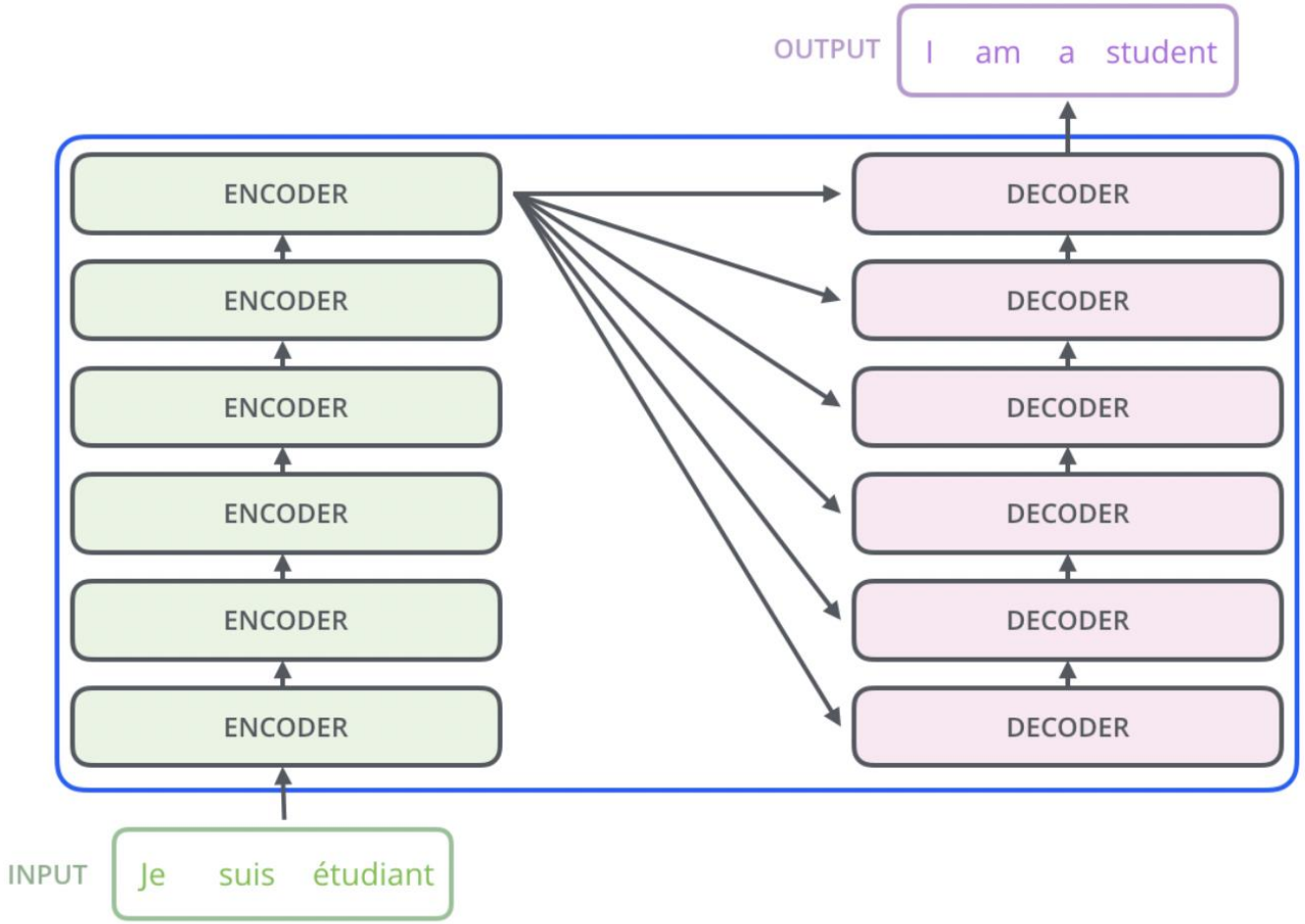
Summary

- Encoder는 self-attention으로 구성되어 있음
  - Q, K, V는 이전 레이어의 출력 값 – 즉, 같은 값

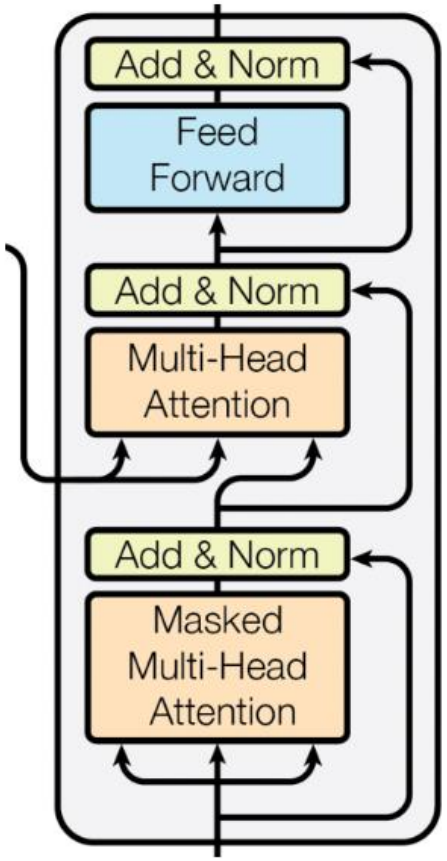
- Seq2seq의 attention과 달리, Q도 모든 time-step을 동시에 연산
  - 빠르지만 굉장히 메모리를 많이 먹게 됨

- Residual connection으로 인해 깊은 네트워크 구성 가능
  - Big LM의 토대 마련

트랜스포머 디코더 이해하기



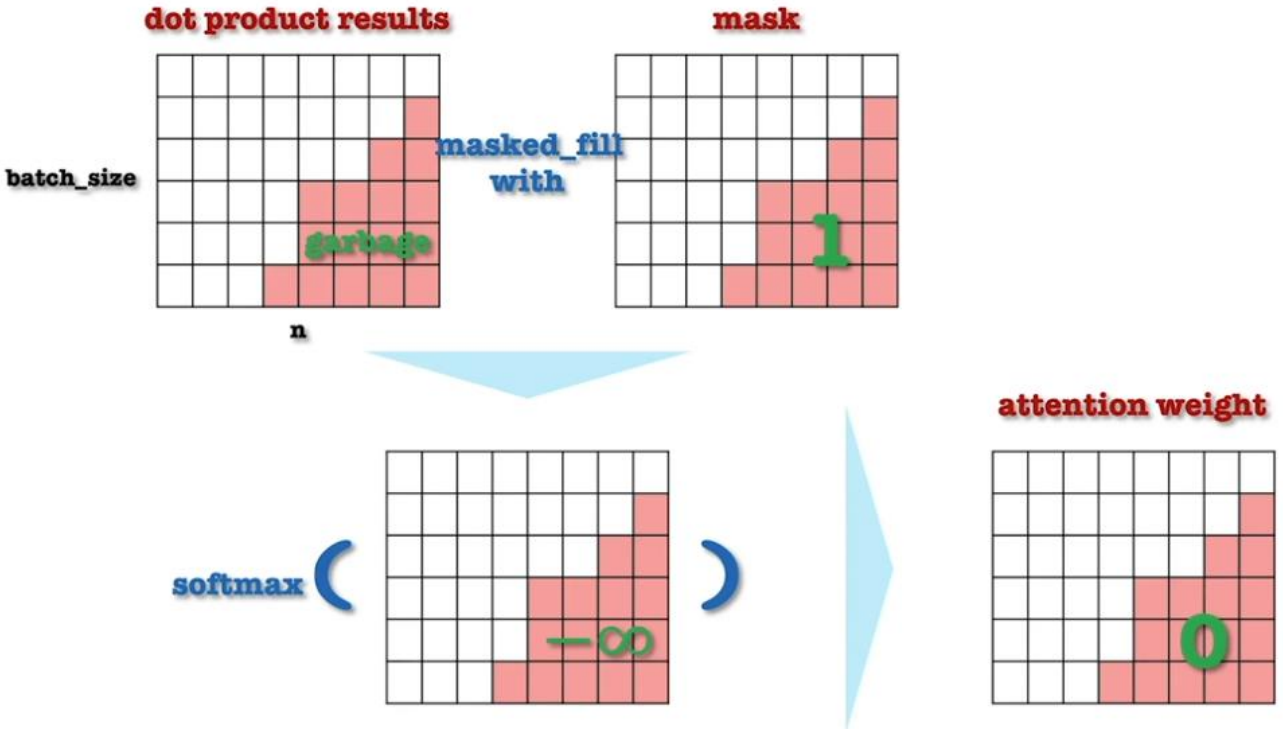
디코더의 내부 동작


Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention

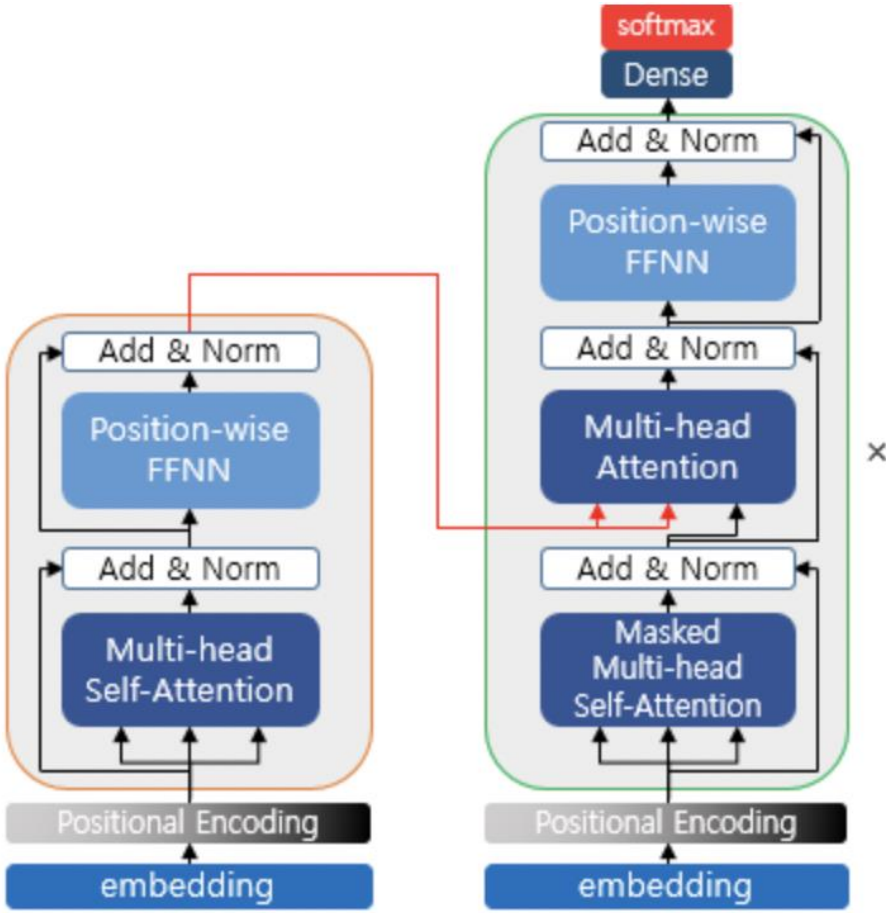
마스크된 멀티 헤드 어텐션

Before we start,

- Using mask, assign $-\infty$ to make 0s for softmax results.



멀티 헤드 어텐션


softmax
Dense
Add & Norm
Position-wise FFNN
Add & Norm
Multi-head Attention
Add & Norm
Masked Multi-head Self-Attention
Positional Encoding
embedding
Add & Norm
Position-wise FFNN
Add & Norm
Multi-head Self-Attention
Positional Encoding
embedding
×

Summary

- Decoder는 2가지의 attention으로 구성됨
  - Attention from encoder:
    - K와 V는 encoder의 최종 출력 값, Q는 이전 레이어의 출력 값
  - Self-attention with mask:
    - Q, K, V는 이전 레이어의 출력 값
    - Attention weight 계산 시, softmax 연산 이전에 masking을 통해 음의 무한대를 주어, 미래 time-step을 보는 것을 방지

- 추론 때에는 self-attention의 mask는 필요 없으나,
  모든 layer의 t 시점 이전의 모든 time-step($< t$)의 hidden state가 필요

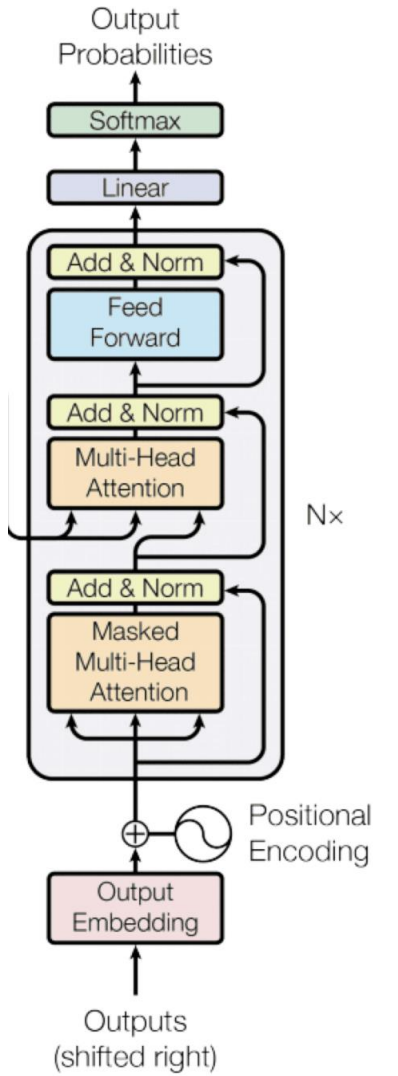
선형과 소프트맥스 레이어

- 디코더가 타깃 문장에 대한 표현을 학습시키면 최상위 디코더에서 얻은 출력값을 선형 및 소프트맥스 레이어에 전달한다.

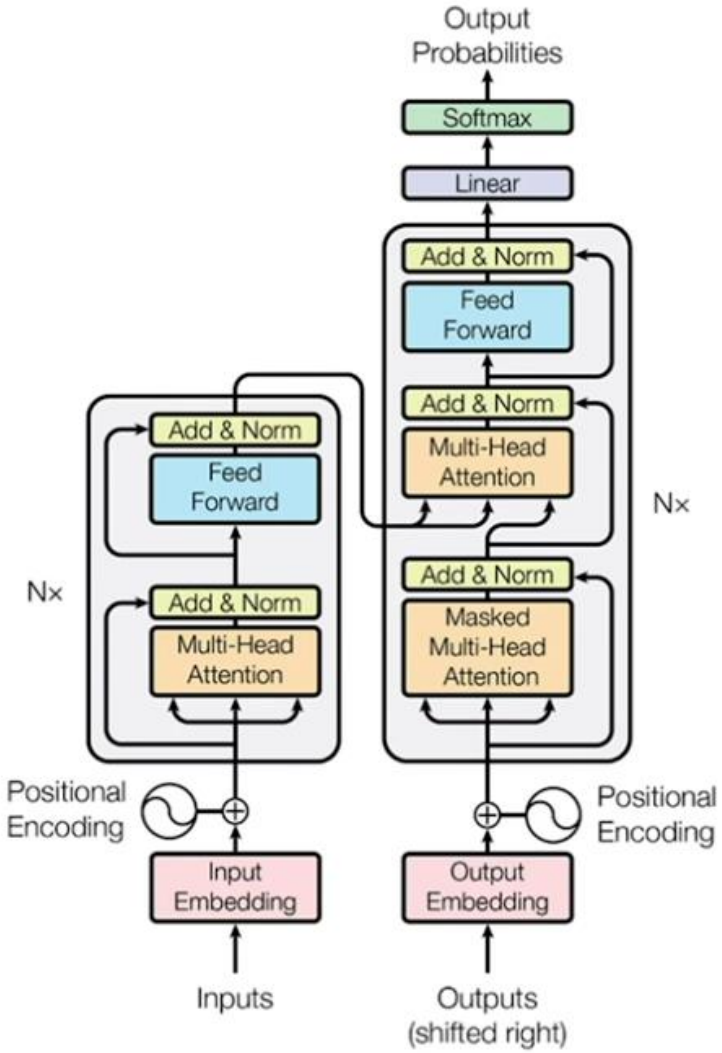
- 선형 레이어의 경우 그 크기가 vocab크기와 같은 logit 형태이다.

- ex) vocab = [bien, Je, vais]이면, 선형 레이어가 반환하는 로짓은 크기가 3인 벡터가 된다.

디코더 모든 구성 요소 연결하기



인코더와 디코더 결합


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

트랜스포머 학습

- 트랜스포머 학습
- 손실 함수를 최소화하는 방향으로 트랜스포머 네트워크를 학습 시킨다.
- → 디코더가 vocab에 대한 확률 분포를 예측하고 확률이 가장 큰 단어를 선택한다는것을 배웠다.
- →즉, 올바른 문장을 생성하려면 예측 확률 분포와 실제 확률 분포 사이의 차이를 최소화 해야 한다.
- → 그러려면 두 분포의 차이를 알아야 한다.
- → 교차 엔트로피 사용
- 손실함수 = 교차 엔트로피 손실
- 이때 옵티마이저는 아담
- 고려사항 : 과적합 방지 → 각 서브레이어의 출력에 드롭아웃을 적용, 임베딩 및 위치 인코딩의 합을 구할 때도 드롭아웃을 적용