Transformer Network

트랜스포머 모델은 트랜스포머 블록으로 구성된다.

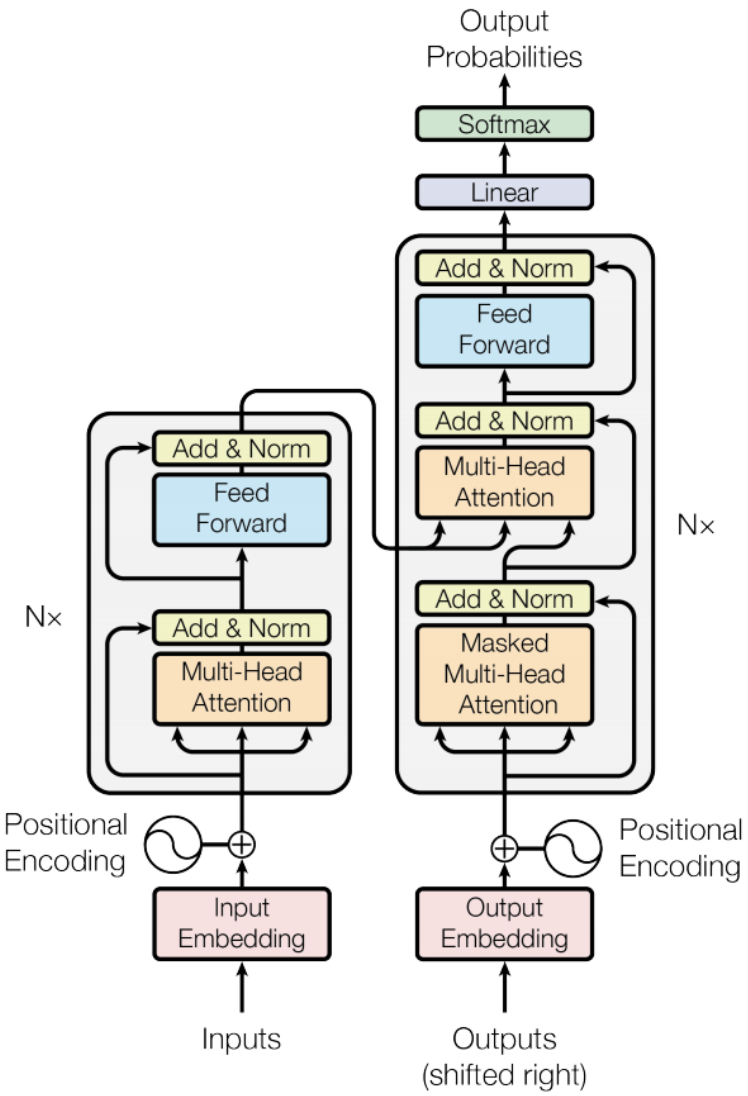
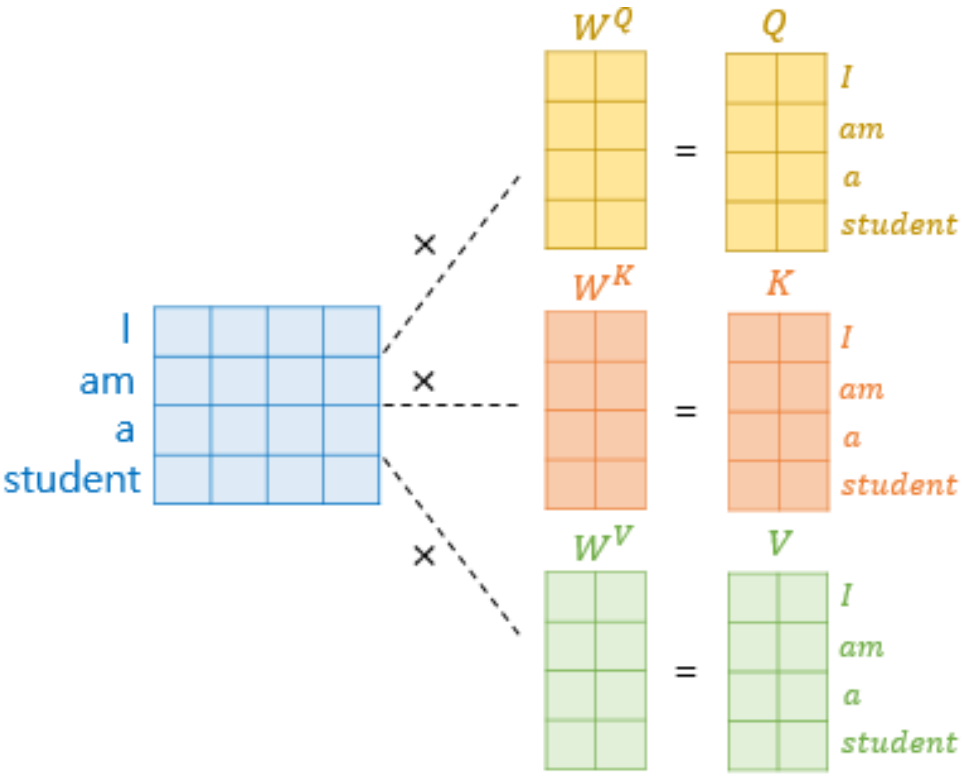


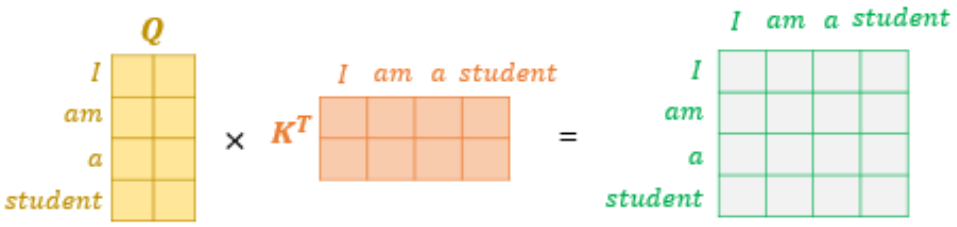
Figure 1: The Transformer - model architecture.

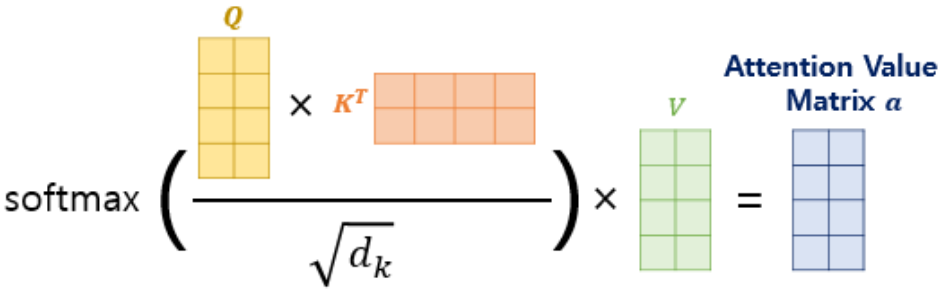
Scaled Dot-Product Attention

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

- Q : 입력행렬 X와 쿼리에 해당하는 행렬 W^Q을 곱해서 계산됨
- K : 입력행렬 X와 키에 해당하는 행렬 W^K을 곱해서 계산됨
- Q : 입력행렬 X와 값에 해당하는 행렬 W^V을 곱해서 계산됨
- $softmax(\frac{QK^T}{\sqrt{(d_k)}})$: 쿼리와 키가 얼마나 관련있는지를 확률로 나타냄. (d_k는 키벡터의 차원수를 의미하는데, d_k로 나누는 것은 scale 역활을 의미. 즉, 쿼리-키 내적 행렬의 분산을 줄여 개별 소프트맥스 값이 지나치게 작아지는 것을 방지 → 지나치게 작아지면 gradient vanishing 문제 초래)







위 그림에서 Scaled Dot-Product Attention 결과 새롭게 만들어진 I에 해당하는 벡터는 해당 문장 내 단어 쌍 간 관계가 모두 농축된 결과.

※ Self Attention의 장점

- CNN의 경우 사용자가 정한 특정 윈도우 내의 로컬 문맥만 살핌. 하지만 문장길이가 꽤 길고 맨 처음 단어와 마지막 단어 사이의 연관성 파악이 태스크 수행에 중요한 데이터라면 CNN은 사용하기에 적절하지 않음. → Self Attention은 이런 문제점이 없음.
- RNN은 시퀀스 길이가 길어질수록 그래디언트 문제가 발생할 염려가 있다. → Self Attention은 문장내 모든 단어 쌍 사이의 관계를 늘 전체적으로 파악할 수 있고 RNN보다 빠름

Multi-Head Attention

멀티헤드 어텐션은 Scaled Dot-Product Attention을 여러번 시행하는 것으로, 동일한 문장을 여러명의 독자가 동시에 분석해 최선의 결과를 내려고 하는 것으로 볼 수 있다.

$$MultiHead(Q,K,V) = Concat(head_1, head_2, ..., head_h)W^O$$
$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

W_O은 멀티헤드 어텐션 수행 결과 행렬의 크기를 Scaled Dot-Product Attention의 입력 행렬과 동이하게 맞추는 역활 및 학습 역활

Pointwise Feedforward Networks

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$$

그외 core idea

- warm up : 사용자가 정한 스텝 수에 이르기까지 학습율을 올렸다가 스텝수를 만족하면 조금씩 떨어뜨리는 방식으로 대규모 데이터, 큰 모델 학습에 적합.
- 레이어 정규화
- Positional Encoding : 위치정보를 삽입
- Residual Connection : Layer가 많이 쌓일수록 정보가 유실되는 현상을 방지. Input Vector가 Multi-Head Attention에 입력으로도 사용되고, Add&Norm 입력으로도 사용됨으로 정보 유실 최소화

참조 :

- 한국어 임베딩 책
- https://wikidocs.net/31379