Paper Review :

[Attention Is All You Need]

Ashish Vaswani et al.(2017)

Winter Vacation Capstone Study

TEAM Kai.Lib

발표자 : 배세영

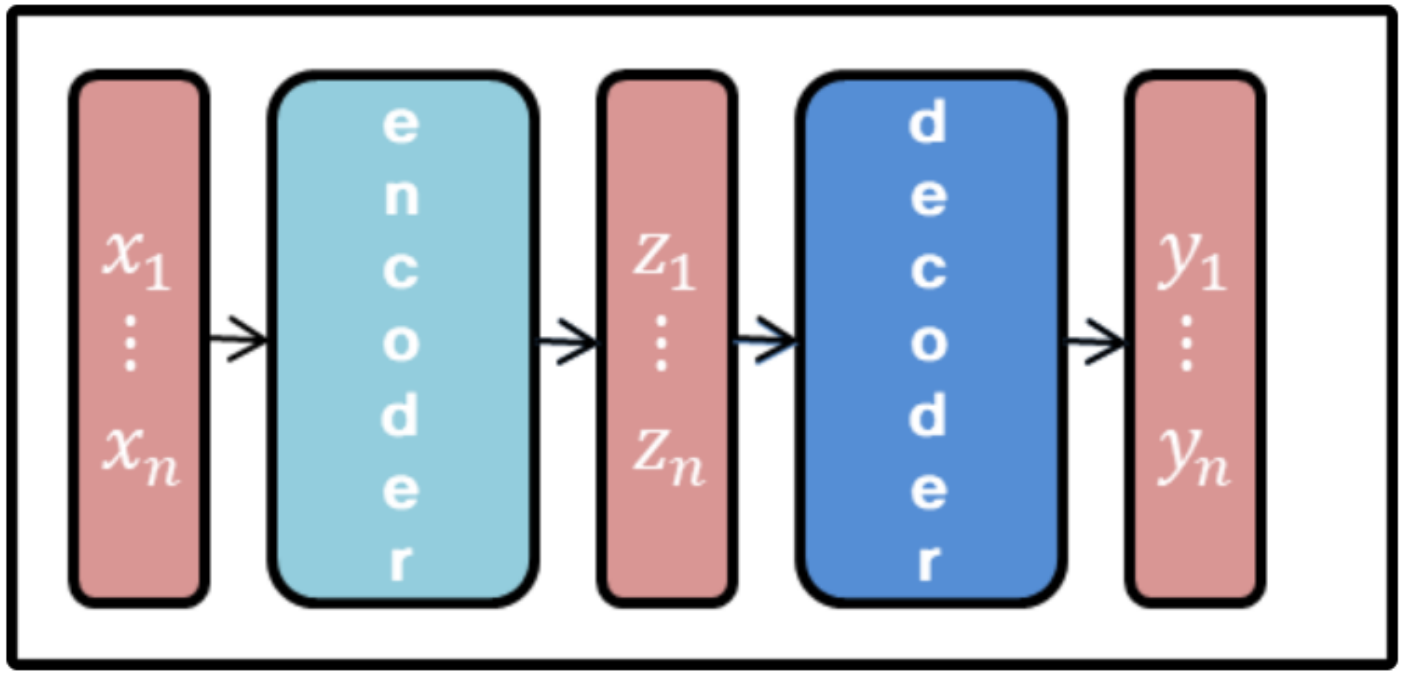
2020.02.03 (MON)

Why Transformer?

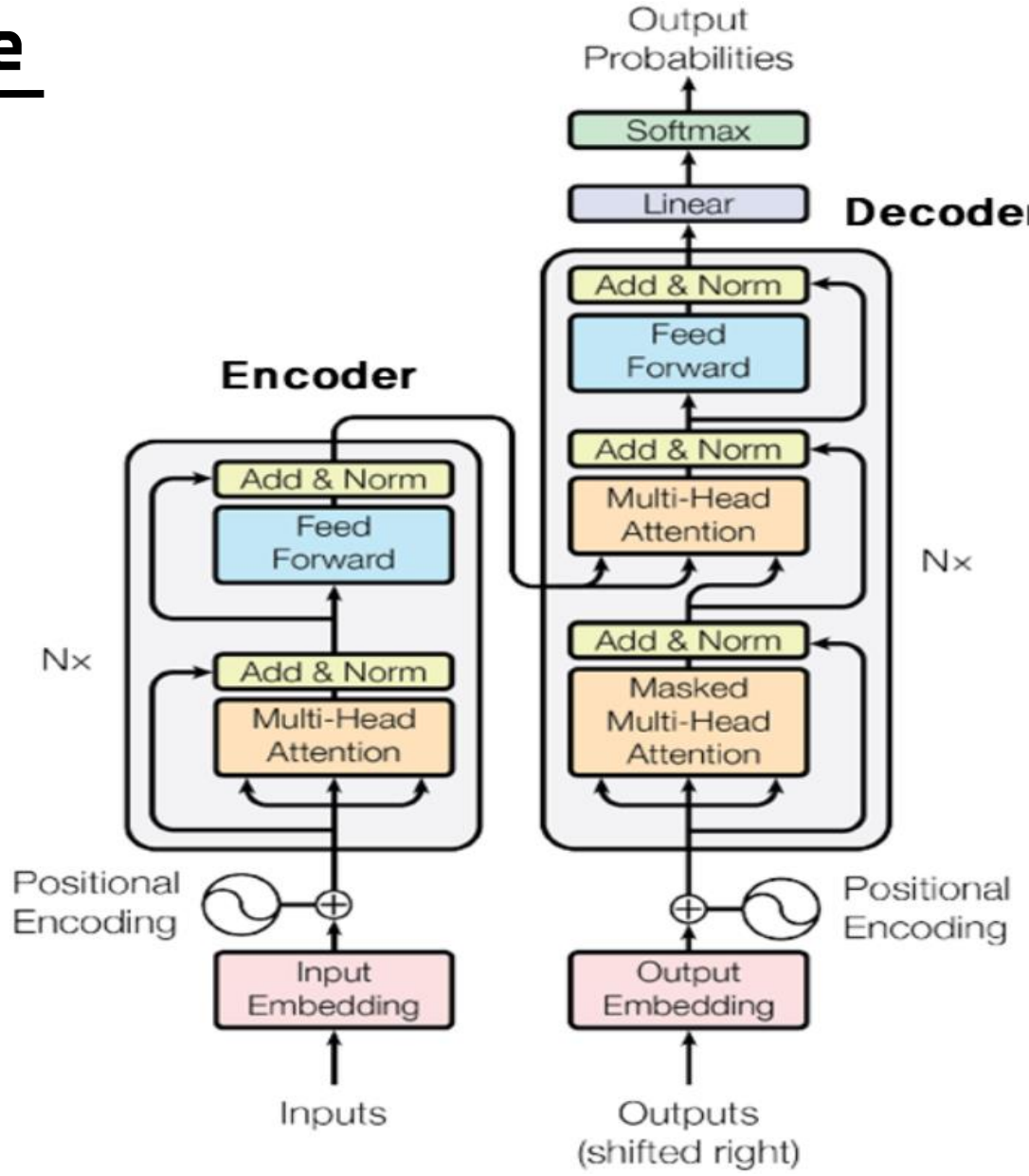
- **Recurrent Model**
  - Sequencial Data 처리를 위하여 이전까지 많이 쓰이던 모델
  - t번째 output 생성을 위하여 t번째 input과 t-1번째 hidden state를 사용
  - 때문에 자연스럽게 문장의 순차적인 특징이 유지되나, 필연적으로 병렬처리가 불가능하고 Long-Term Dependency Problem이 존재
- Long-Term Dependency Problem
  - 문장 내에서 어떤 단어와 단어 사이의 거리가 멀 때, 정보가 전파되는 도중 0에 가깝도록 소실되어 해당 정보를 이용하지 못하게 되는 문제
- **Transformer Model**
  - Recurrency를 사용하지 않고 Attention Mechanism만으로 input과 output 사이의 Dependency를 포착하고 유지할 수 있음
  - Encoder에서는 각각의 Position에 대하여 Attention 해주기만 하고, Decoder에서는 Masking 기법을 통하여 병렬처리가 가능해짐

Model Architecture

- Encoder & Decoder Structure
  - Encoder는 input sequence (x1, x2, ...)에 대해 다른 representation인 z = (z1, z2, z3, ...) 생성
  - Decoder는 z를 받아 output sequence (y1, y2, y3, ...)를 하나씩 생성
  - 각각의 step에서 다음 symbol을 만들 때 이전에 만들어진 output(symbol)을 사용 (Auto-Regressive)

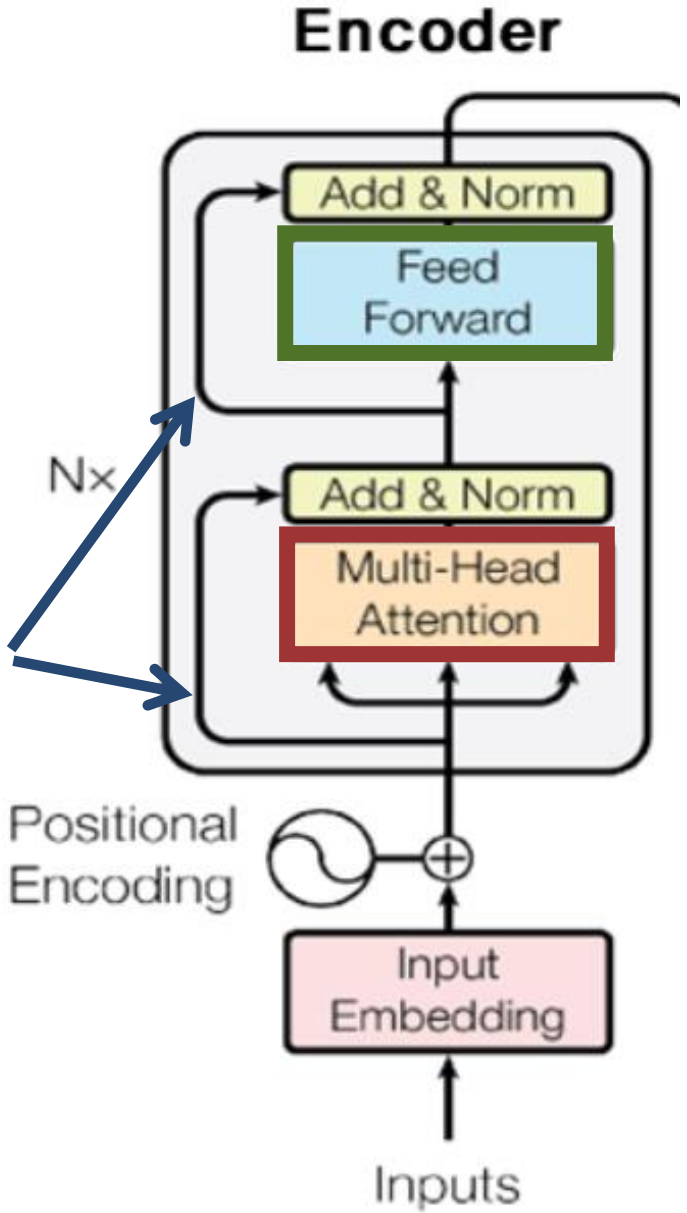


Model Architecture



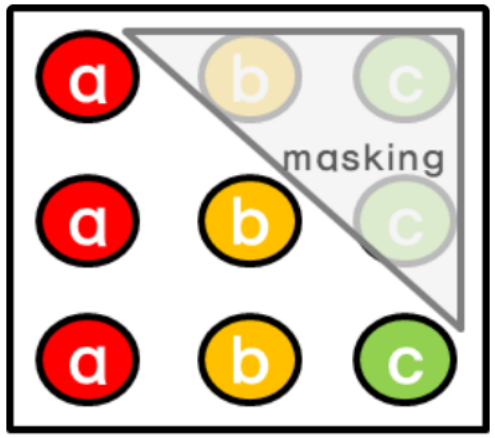
Model Architecture

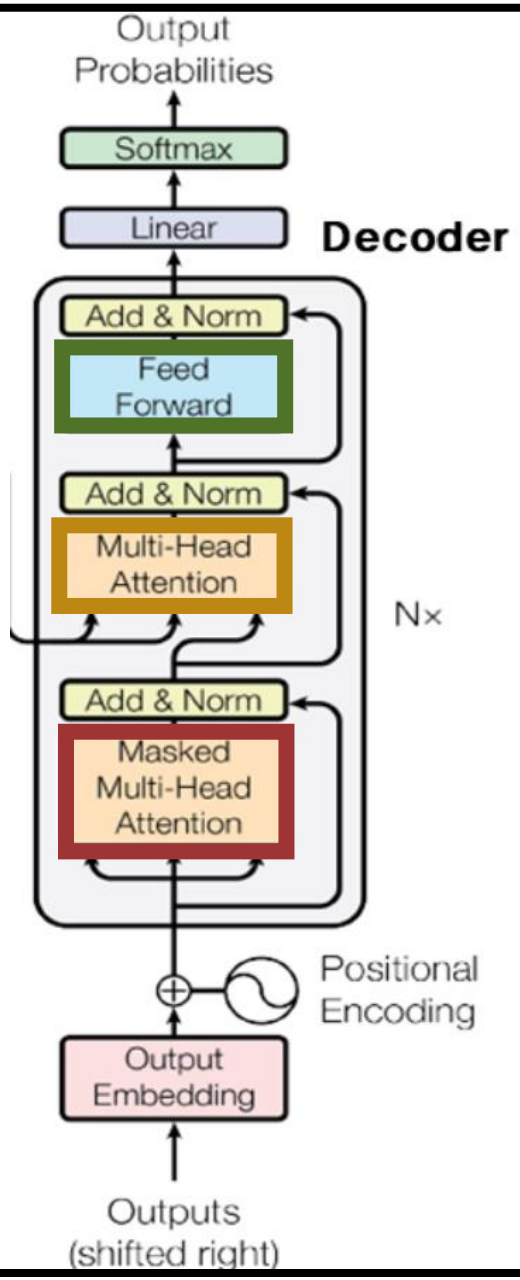
- Encoder Part
  - N개의 layer로 구성
  - 각각의 layer는 multi-head self-attention mechanism과 position-wise fully connected feed-forward network를 가지고 있음
  - 위 두 개의 sub layer에 residual connetion을 사용
  - 이후 layer normalization 수행



Model Architecture

- Decoder Part
  - N개의 layer로 구성
  - 각각의 layer는 multi-head self-attention mechanism과 position-wise fully connected feed-forward network, encoder output multi-head attention mechanism을 포함
  - 위 두 개의 sub layer에 residual connetion을 사용
  - 이후 layer normalization 수행
  - self attention 수행 시 masking을 통하여 순차적 특성 유지





Scaled Dot Product Attention

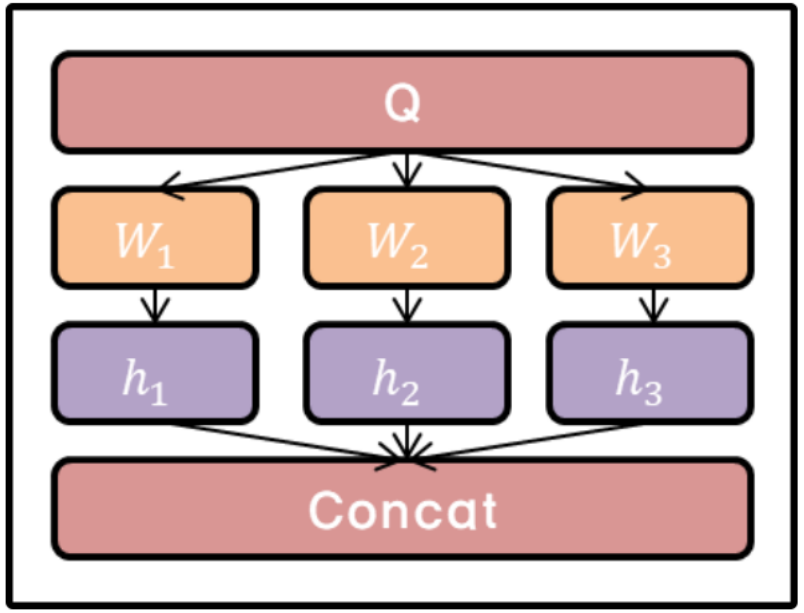
$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

- input은 dk dimension의 query와 key들, dv dimension의 value들로 구성
- 이때 모든 query와 key에 대한 dot-product를 구하고 각각을 (dk)**0.5로 나눈 후, 이 값에 softmax를 적용하여 value들에 대한 weight를 계산
- key, value는 attention이 이루어지는 위치에 관계없이 같은 값을 가짐
- 이 때 query와 key에 대한 dot-product를 구하면 각각의 query와 key 사이의 유사도를 구할 수 있음 (cosine similarity)
- (dk)**0.5로 scaling하는 이유는 dot-product의 값이 클수록 softmax 함수의 기울기 변화가 작은 구간으로 가기 때문
- softmax를 거친 값을 value에 곱해 준다면 query와 유사한 value일수록 (더 중요한 value일수록) 더 높은 값을 가지게 됨

Scaled Dot Product Attention

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W^O$$
$$\text{where } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

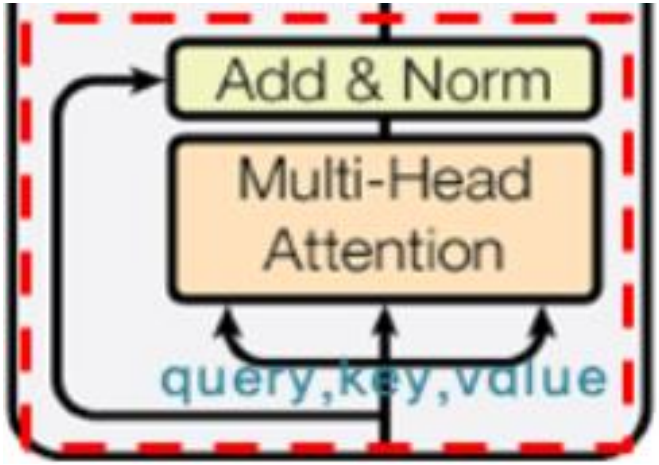
- key, value, query로 하나의 attention을 수행하는 것 보다 K, V, Q에 각각 다른 weight matrix를 곱해주는 것이 더 좋다고 함
- 이렇게 weight가 곱해진 key, value, query는 병렬적으로 attention function을 거쳐 dimensional output으로 나오게 됨
- 그 다음 여러 개의 head를 concatenate하고 다시 weight 값을 곱하여 최종 output을 얻는다



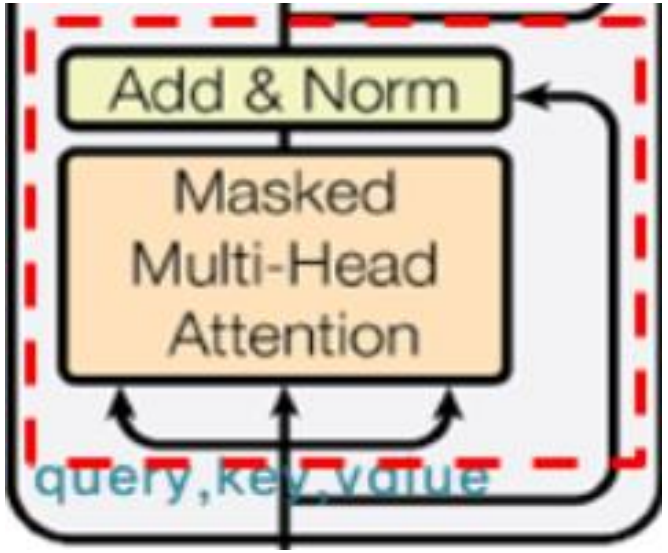
https://pozalabs.github.io/transformer/

Self Attention

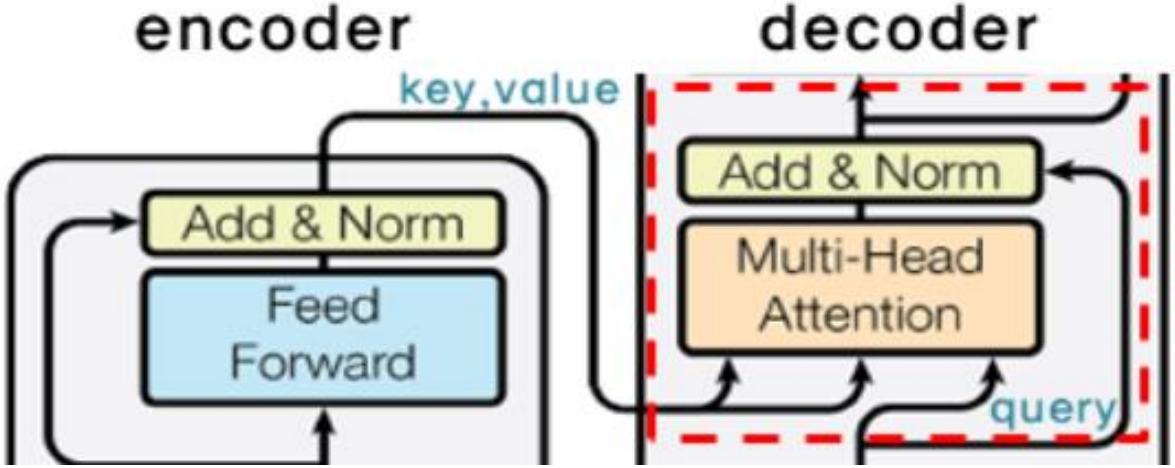
- Encoder Self Attention
  - key, value, query는 모두 encoder의 이전 layer에서 온다
  - 첫 layer라면 embedded vector를 사용한다



- Decoder Self Attention
  - encoder의 self attention과 기본적으로 동일
  - auto-regressive한 특성을 유지하기 위하여 masking out된 scaled dot-product attention을 사용



Enc-Dec Attention

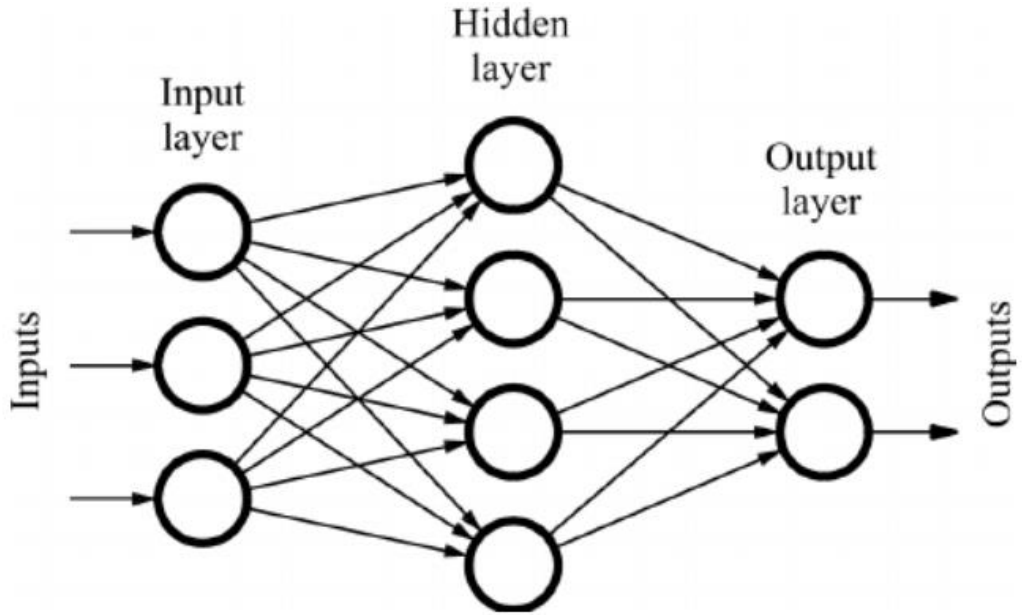


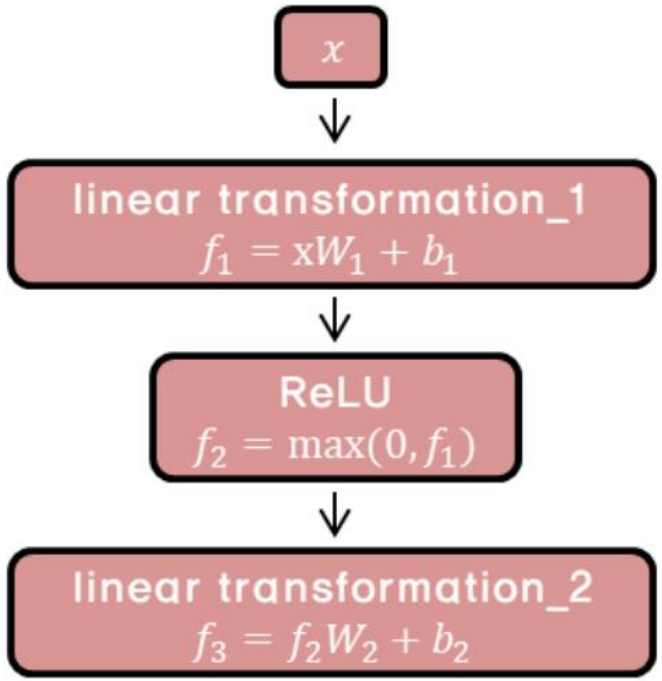
- query들은 이전 decoder layer에서 오고,
  key와 value들은 encoder의 output에서 온다
- query가 decoder sub layer의 output인 이유는
  query라는 것이 조건에 해당하기 때문
- "현재 decoder의 값이 이러한데, 무엇이 output이 되어야 할까?"
- 이 때 query는 이전 sub layer에서 masking 되었으므로, 현재 position까지만 attention을 얻게 됨
- 이는 seq-to-seq의 전형적인 enc-dec machanism을 반영한 것

Postion-Wise Feed-Forward Network

- Position마다, 즉 개별 단어마다 적용되는 Fully Connected Feed-Forward Network
- 두 번의 Linear Transformation과 한 번의 Activation Function(ReLU)로 구성

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





Postional Encoding

- 내용 이해를 못하였습니다. 함께 이야기해 볼까요?
- 넘모 어려워요 ㅜㅜ

https://pozalabs.github.io/transformer/