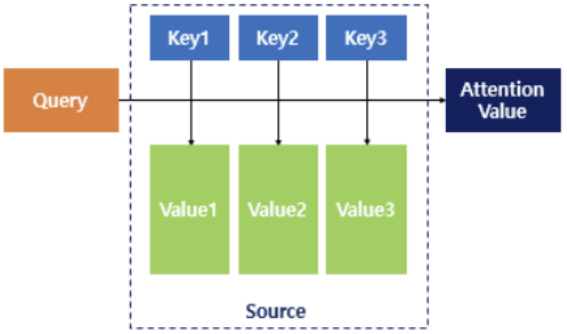
Transformers

Transformers 이전의 모델들

- **RNN**
  - Sequence 데이터를 위한 모델
  - 순차적 구조로 데이터 처리
  - 단점: 긴 문장에서 기울기 소실·폭발로 장기 의존성 문제 발생
- **LSTM**
  - Cell State와 Forget/Input/Output 게이트 도입으로 장기 의존성 문제 개선
  - 단점: 구조 복잡, 연산량 증가
- **GRU**
  - Reset/Update 게이트만 사용해 단순화
  - 연산 효율성 ↑
  - 성능이 LSTM과 유사
- **Seq2Seq**
  - Encoder–Decoder 구조
  - Context Vector에 모든 정보를 압축 → 병목현상 발생

Attention (Seq2Seq)



Attention(Q, K, V) = Attention Value

- 입력 전체 중 관련 있는 부분에 더 집중(가중치 부여)

Transformer

구조

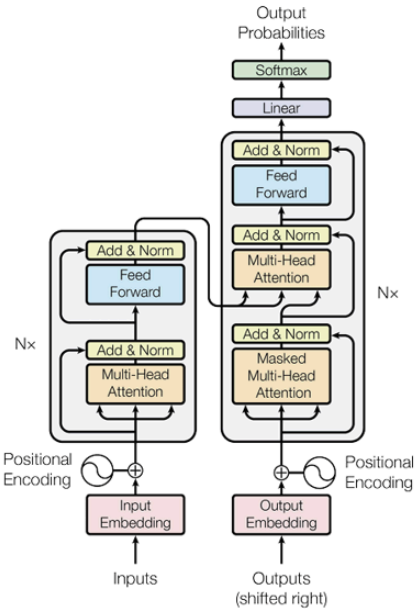


Figure 1: The Transformer - model architecture.

(Input into Encoder)

- Embedding layer: 토큰을 학습 가능한 벡터로 변환 → 선형변환으로 Q, K, V 생성
- Positional Encoding : sin, cos 함수를 이용하여 위치(pos)에 따라 임베딩 벡터에 더함
  → 토큰 간의 순서 정보 반영

🌟 Encoder

1. Self-Attention
   a. 문장 내의 단어들 간의 연관성 학습 (단어 간 유사도)
2. Scaled Dot-Product Attention
   a. Attention 계산을 행렬 연산으로 일괄 처리

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

3. Multi-Head Attention

🌟 Decoder

1. Masked Multi-Head Attention
   a. Decoder는 단어를 순차적으로 출력해야 하기 때문에,
   b. 미래 단어를 참조하지 못하도록 Masking 처리 (Look-Ahead Mask)
   c. Attention Score에서는 미래 위치에 해당하는 값은 -inf 로 바꾸어 softmax 후 0이 되도록 함
2. Position-wise Feed-Foward Networks

a. 여러 개의 Head가 Self-Attention을 수행하고

b. 각 Head의 결과를 합하여 반영

→ 다양한 관점에서 정보를 동시에 봄

a. 각 토큰 벡터에 독립적으로 동일한 두 번의 Linear + ReLU 활성화 함수 적용

3. Residual Connection & Layer Normalization
   a. Residual Connection
   b. Layer Normalization