Transformer 구조 조사 및 분석

(feat. attention is all you need)

과목명	딥러닝프레임워크 [00]
담당교수	최인엽 교수님
학과	ICT 공학부 소프트웨어학과
학번, 이름	202184050 정가현
학번, 이름	202184031 송민아
제출일	2024. 06. 04

1. 서론

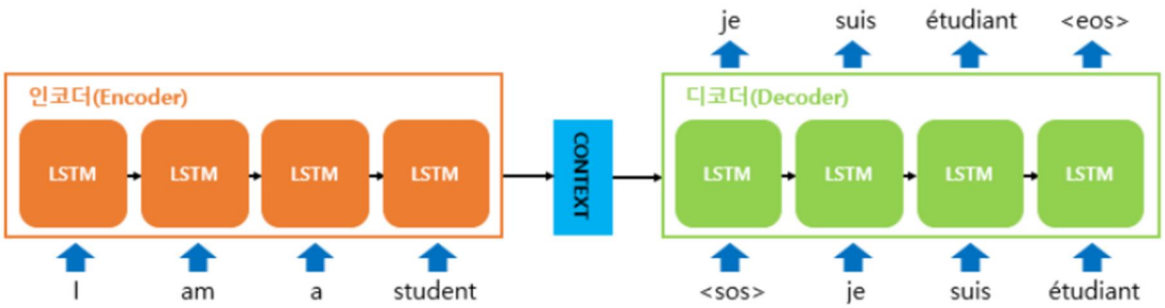
Transformer 모델이란 ?

2017년 구글이 발표한 논문인 "Attention is all you need"에서 나온 모델로 기존의 seq2seq의 구조인 인코더-디코더를 따르면서도, 논문의 이름처럼 어텐션(Attention)만으로 구현한 모델이다. 이 모델은 RNN을 사용하지 않고, 인코더-디코더 구조를 설계하였음에도 번역 성능에서도 RNN보다 우수한 성능을 보여준다.

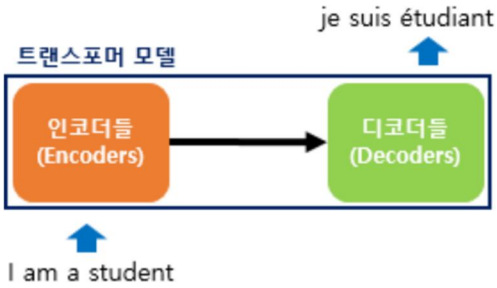
seq2seq vs Transformer

Transformer와 seq2seq 둘다 Encoder와 Deocder 구조인

입력 seq → Encoder → **Vector** → Decoder → 출력 seq의 구조를 가진다.



다만, seq2seq는 Vector를 Context vector라고 부르는데, 이 Context vector는 Encoder의 입력 문장의 모든 단어들을 순차적으로 입력 받고 모든 언어를 압축한 단 하나의 정보이다.



또한, Transformer는 Vector를 Intermediate Representation (중간 표현) 이라고 부른다. Intermediate Representation는 Transformer 아키텍쳐에서 각각의 Encoder와 Decoder, 그리고 Encoder와 Decoder를 연결하는 값을 의미한다. 각각의 Encoder, Decoder 레이어에서 생성되는 데이터의 내부적 표현이고, 입력 데이터를 모델이 이해할 수 있는 형태로 변환하는 과정에서 생성된다. 중간 표현은 입력 seq의 토큰 개수만큼 생성되고, 입력 seq의 토큰과 중간 표현의 각각의 벡터들은 **1:1 대응**이다. Word2Vec과 같이 중간 표현의 각각의 벡터들은 대응되는 입력 토큰의 위치적 문맥적 정보 내포하여 Self Attention으로 주변 토큰과의 관계를 고려한다.

즉, Seq2Seq의 **Vector**는 길이가 고정되고 Transformer의 **Vector**는 입력 seq 토큰 개수에 종속된다.

Transformer 모델의 중요성

1. 병렬성

효율성 : Transformer 모델은 순차적으로 데이터를 처리하는 RNN과 달리, 모든 입력 데이터를 동시에 병렬로 처리할 수 있습니다. 이는 학습과 추론 속도를 크게 향상시킨다.

확장성 : 병렬 처리 덕분에 Transformer 모델은 대규모 데이터와 복잡한 작업에 쉽게 적용할 수 있다.

2. Attention 메커니즘

Self-Attention : Transformer는 Self-Attention 메커니즘을 사용하여 입력 시퀀스 내의 모든 단어 간의 관계를 학습한다. 이는 단어의 의미를 더 잘 이해하고, 문맥을 고려한 더 정교한 표현을 생성하는 데 도움을 준다.

Long-Range Dependencies : 긴 문장이나 문맥에서도 중요한 정보를 잃지 않고 학습할 수 있다.

3. 모듈화와 유연성

Encoder-Decoder 구조 : Transformer는 인코더와 디코더로 구성되어 있어 다양한 작업에 쉽게 맞출 수 있다. 인코더는 입력 데이터를 이해하고, 디코더는 출력 데이터를 생성한다.

모듈형 설계 : 다양한 구성 요소를 추가하거나 수정하여 특정 작업에 맞게 모델을 조정할 수 있다.

NLP및 기타 분야에서의 역할

1. NLP

기계 번역 : Transformer는 구글 번역(Google Translate) 등에서 사용되며, 문장의 문맥을 더 잘 이해하고, 더 자연스러운 번역을 제공한다.

텍스트 요약 : 긴 문서에서 중요한 내용을 요약하는 데 사용된다. Attention 메커니즘은 중요한 정보를 선택하는 데 매우 효과적이다.

질의 응답 시스템 : 질문에 대해 정확한 답변을 제공하는 데 사용된다. 예를 들어, BERT와 같은 Transformer 기반 모델은 구글 검색의 질의 응답 기능에 활용된다.

문서 분류 및 감정 분석 : 문서나 리뷰의 내용을 분류하거나 감정을 분석하는 작업에서 뛰어난 성능을 발휘한다.

2. 기타 분야

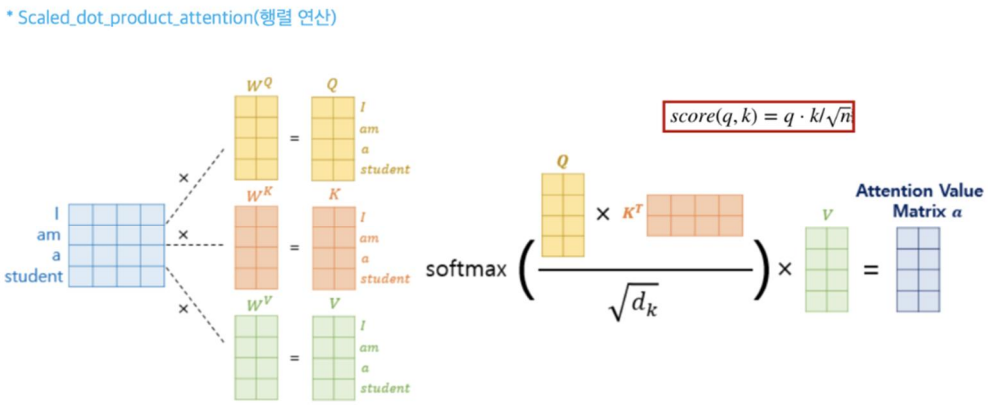
이미지 처리 : 비전 Transformer(ViT)와 같은 모델은 이미지 분류, 객체 검출 등 이미지 처리 작업에서 사용된다. CNN과 비교하여 전역적인 이미지 특징을 잘 캡처할 수 있다.

음성 인식 : Transformer 모델은 음성 인식 및 음성 합성 작업에서도 사용된다. 음성 데이터를 시퀀스로 취급하여, 문맥 정보를 활용한 더 정확한 인식을 가능하게 한다.

생물정보학 : 단백질 구조 예측과 같은 복잡한 생물학적 시퀀스 분석에도 사용된다. 예를 들어, AlphaFold는 Transformer 기반의 모델을 사용하여 단백질 구조를 예측한다.

2. 이론적 배경

Self-Attention



Query, Key, Value 가 동일한 집합을 사용하는 Attention 으로써, 입력 시퀀스의 모든 부분 간의 상호작용을 고려하여 각 토큰이 다른 모든 단어와의 관계를 학습할 수 있도록 한다. 이를 통해 문맥을 더 잘 이해하고, 중요한 정보를 추출할 수 있다.

- **작동 원리**

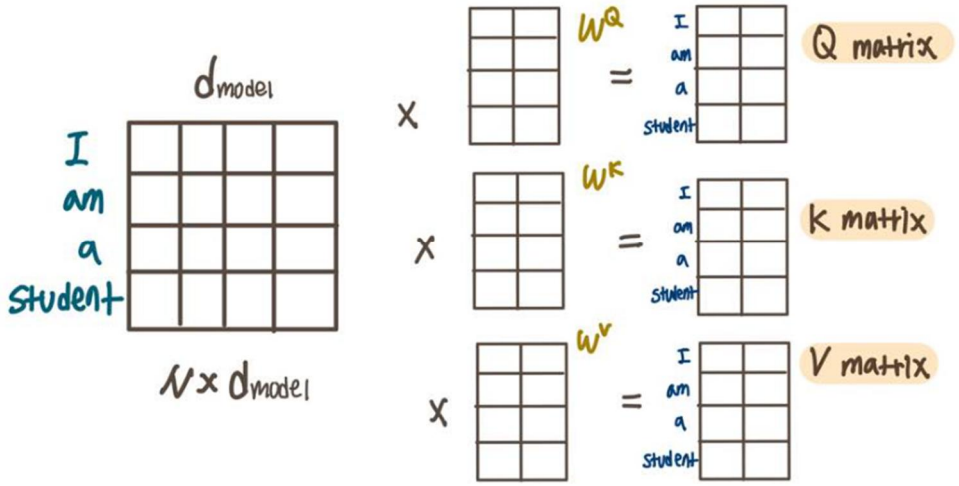
1. 입력 벡터 변환

단어 → Embedding + Positional Encoding → input vector

Embedding (Embedding vector)

텍스트를 실수 벡터 형태(i.e. floating point 숫자들로 구성된 고정된 크기의 배열)로 표현한 결과물을 의미한다. (Positional Encoding 은 다음 장에서 자세하게 다룬다.)

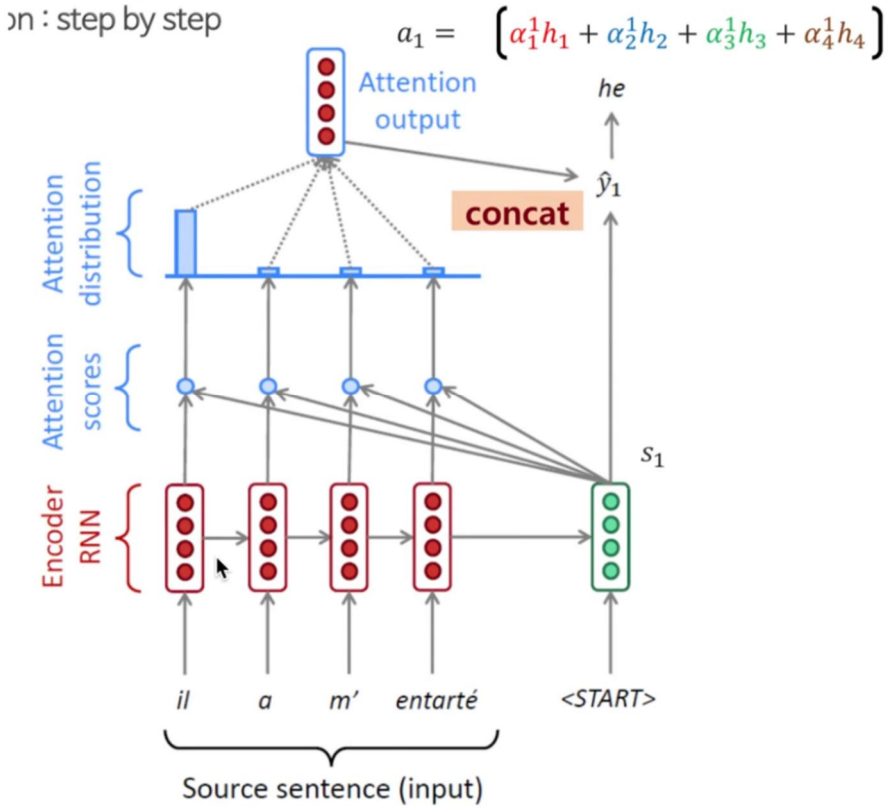
2. Query, Key, Value 벡터 생성



가중치 행렬 Wq, Wk, Wv 하고, Embedding vector 와 행렬 곱을 하여 Query, Key, Value 벡터를 생성한다.

이 행렬들은 Weight matrix 이다. 딥러닝 모델학습 과정을 통해 최적화가 된다. 임베딩된 단어는 벡터이고 실제 한 문장은 행렬이라고 할 수 있다. 행렬*행렬이 되기에 Query, Key, Value 를 한번에 구할 수 있다.

3. Attention value 계산



1. Attention score 계산

Attention score 현재 Decoder 의 t 시점에서 단어를 예측하기 위해, Encoder 의 모든 hidden state 값이 Decoder 의 t 시점 hidden state 와 얼마나 관련있는지 구한 점수이다.

> n 개의 은닉 상태 → n 개의 Attention score
>
> Encoder 의 hidden state 에 따른 Attention score 의 결합 = Energy 값

2. Attention 분포

Attention 분포는 Seq2Seq with Attention 에서 Attention score 의 집합인 Energy 값에 Softmax 를 적용해 얻은 확률 분포이다.

$$\alpha^t = \text{softmax}(e^t)$$

- α^t: Decoder의 t 시점에서의 Attention 분포.
- e^t: Decoder의 t 시점에서의 Attention score.
- Attention 분포를 통해 얻은 각각의 값을 Attention 가중치 라고 부름.
- Encoder의 hidden state가 Decoder의 t 시점에 영향을 주는 정도을 알 수 있음.
- 각 Query(Decoder의 t 시점)에 대해 Key(Enoder의 i번째 hidden state)의 점수(Energy 값)이 정규화됨 ⇒ 특정 Key가 Query와 얼마나 잘 매칭되는지를 반영하는 가중치로 변환.

3. Attention value 계산

Attention value 계산은 Seq2Seq with Attention 에서 Attention 가중치에 Encoder 의 hidden state 의 가중 합한 Attention value 를 계산이다.

$$a^t = \sum_{i=1}^{N} \alpha_i^t h_i$$

- t: Decoder의 현재 출력 인덱스.
- i: Encoder의 hidden state 번호.
- N: Encoder의 hidden state 개수.
- a^t: t번째 Attenion value 값.
- α_i^t : Decoder의 t번째 Attention 분포의 i번째 값. 즉 i 번째 Attention 가중치.
- h_i: Encoder의 i 번째 hidden state 값.

4. 가중합 계산

5. 다음 레이어에 전달

이러한 과정을 통해 Self-Attention 은 입력 시퀀스의 각 원소들 간의 상호작용을 고려하여 모델이 입력의 모든 부분을 동시에 고려할 수 있게 한다. 이는 자연어 처리에서 문장 내의 단어들 간의 관계를 파악하거나 컴퓨터 비전에서 이미지 내의 픽셀들 간의 상호작용을 고려하는 데에 특히 유용하다.

Positional Encoding

Positional Encoding 은 Embedding vector 에 토큰의 위치정보를 포함시킬 목적으로 사용한다. Embedding vector 와 동일한 크기를 가져야 하며, Embedding vector 와 위치 정보를 원소별로 합한다

- 위치정보 부여방법

논문(Transformer: Attention is all you need)에서는 sin, cos 주기함수를 사용한다. 위치 정보를 줄 수 있는 모든 주기함수가 가능하다.

$$PE_{(pos,2i)} = \sin(pos/1000^{2i/d_{model}})$$
$$PE_{(pos,2i+1)} = \cos(pos/1000^{2i/d_{model}})$$

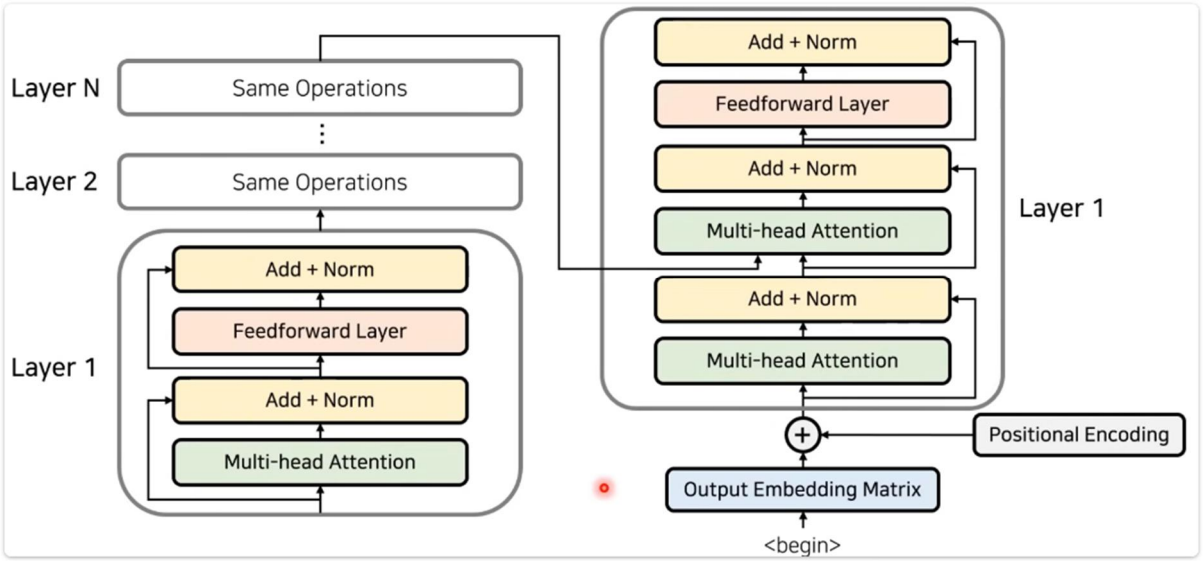
- PE: Positional Encoding의 약자.
- pos: 각각의 토큰 번호.
- i: 각각의 단어에 대한 임베딩 위치.
- d_{model} : Embedding vector의 차원 수.

- 별도의 임베딩 레이어(BERT)

위치 정보를 부여할 수 있는 임베딩 레이어를 이용하여 위치 임베딩을 학습한다.

3. Transformer 구조 분석

Encoder와 Decoder



- **인코더 (encoder)**

Positional Encoding과 Input Embedding Maxtrix를 결합하여 Attention 입력 벡터를 구성한다.

하나의 Encoder Layer는 Multi-head Attetntion FC 레이어로 구성되어 있다. Attention과 FC 레이어 이후에 정규화 (Normalization)를 수행하여 잔차 연결을 수행한다.

(잔차 연결 (Residual connection)과 층 정규화 (Layer normalization)는 다음 장에서 설명한다.)

- **디코더 (decoder)**

Positional Encoding과 Output Embedding Matrix를 결합하여 Attention 입력 벡터를 구성한다.

하나의 Decoder Layer는 두 개의 Multi-head Attention 레이어와 FC 레이어로 구성되어있다. 각 단계 이후에는 정규화를 수행하고, 잔차 연결 수행, 두 번째 Multi-head Attention 레이어에서 Encoder의 마지막 출력값을 입력한다. 이때, Seq2Seq with Attention의 영향으로 소스 문장의 어떤 단어와 연관성을 가지는지 고려한다.

각 Decoder Layer 에서 Attention 과 FC 레이어는 서로 다른 파라미터를 가진다. 각 Decoder Layer 의 입력 및 출력 차원은 동일하다.

Multi-Head Attention

Multi-head Attention 은 Embedding vector 를 head 개로 나누어 병렬적으로 Self Attention 을 수행한다.

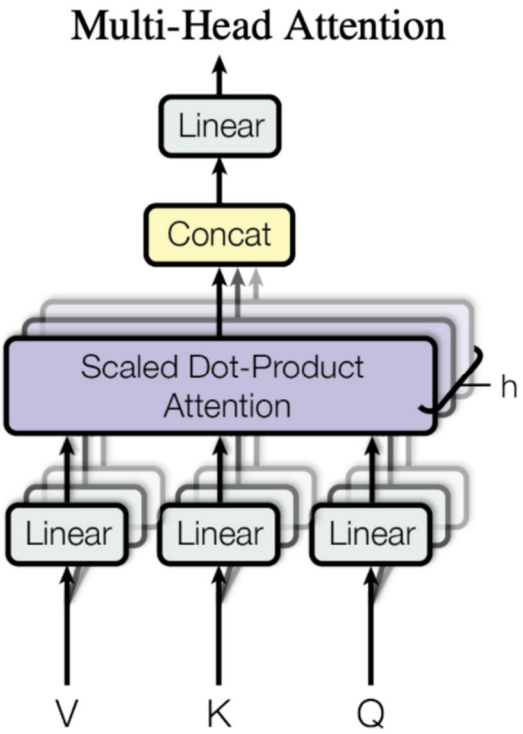
Positional Encoding 을 통해 Embedding vector 가 위치 정보를 기억하고 있어 head 개로 Embedding vector 를 분할 후 결합해도 된다. 할되어 독립적으로(병렬적으로) 실행되는 Self Attention 인스턴스를 **head** 라고 지칭한다.

각각의 head 는 서로 다른 크기의 가중치 행렬 W ⇒ 서로 다른 크기의 Query 연산 값을 가진다. 이때, W 초기화의 무작위성에 의해 각각의 head 는 서로 다른 W 초기값을 가지며 다양성 보장하고 과적합 방지한다.

Multi-head Attention 구성도

1. head 개의 Self Attention을 병렬적으로 수행
2. head 개의 Attention value 를 결합.(concat)
3. 가중치 행렬 W^O를 곱한다 → 출력 벡터 구성

   차원 조절 및 정보 통합 목적이며, 학습 과정에서 경사하강법과 같은 최적화 알고리즘을 통해 최적의 W^O도출한다.



Multi-head Attention 의 하이퍼 파라미터

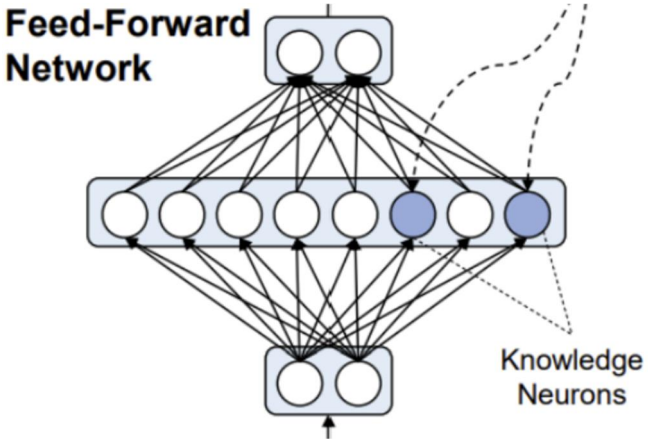
- 가중치 매트릭스 W의 차원: Query, Key, Value의 차원을 결정할 수 있다.
- head 수: head 개수 ⬆ ⇒ 다양성 및 정보 증가, but 계산 비용 ⬆
- 출력 가중치 매트릭스 W^O의 차원: 최종 출력 크기를 결정.

어텐션 헤드를 사용하는 이유와 그 이점

Multi Head Attention 은 기계번역에 큰 도움을 준다. 사람의 문장은 모호할때가 대부분이고 한개의 Attention 으로 인코딩하기가 어렵기때문에 Multi Head Attention 을 사용하여 되도록 연관된 정보를 다른 관점에서 수집하여 모호한 점을 해결한다.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Feed-Forward Networks



Feedforward Layer 는 Transformer 아키텍쳐에서 Multi-head Attention 의 결과를 취합하여 전달하는 역할을 수행하는 Layer 다.

$$\text{FFN}(x) = max(0, xW_1 + b_1)W_2 + b_2$$

- **레이어 구성**

1. FC Layer(가중치 W_1, 편향 b_1)
2. 활성 함수: ReLU 사용.
3. FC Layer(가중치 W_2, 편향 b_2)

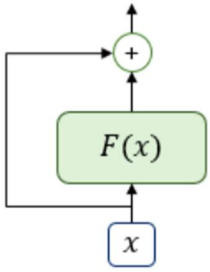
각 위치에서 독립적으로 작동하여, 다층 퍼셉트론 (MLP)과 같은 구조이다. 일반적으로 두 개의 선형 변환과 활성화 함수로 구성하여 네트워크가 더 복잡한 관계를 학습 가능하다. 주로 각 위치에서 단어의 표현을 더 깊게 변환하고 복잡한 패턴을 학습하는데 사용한다.

Residual Connections & Layer Normalization

- **잔차 연결 (Residual connection)**

입력값 x를 타깃값 y로 매핑(mapping)하는 함수 H(x)를 구하는 과정에서 y를 x의 대변으로 보고 H(x)-x를 찾아 나가는 과정을 학습하는 것이다. 즉, 네트워크의 입력과 출력이 더해진 것이 다음 층의 입력으로 사용된다.

$$H(x) = x + F(x)$$



위 그림은 입력 x와 함수 F(x)의 값을 더한 함수 H(x)의 구조를 보이고 있다. 여기서 함수 F(x)는 Transformer에서의 서브층에 해당된다. 위의 그림에 보이듯이 잔차 연결은 서브층의 입력과 출력을 더하는 것이다.

- **층 정규화 (Layer normalization)**

잔차 연결을 거친 후 이어서 층 정규화 과정을 거치게 된다. 잔차 연결의 입력값을 x, 잔차 연결과 층 정규화를 모두 거친 결과 값을 LN이라고 할 경우, 잔차 연결 후 층 정규화 연산을 수식으로 표현하면 다음과 같다.

$$LN = LayerNorm(x + Sublayer(x))$$

층 정규화는 Tensor의 마지막 차원에 대해서 평균과 분산을 구하고, 이를 가지고 어떤 수식을 통해 값을 정규화 하여 학습에 도움을 주게 된다.

각 Encoder Layer에서 Attention과 FC 레이어는 서로 다른 파라미터를 가지며, 각 Encoder Layer의 입력 및 출력 차원은 동일하다. 마지막 Layer의 출력값은 Decoder에 입력한다.