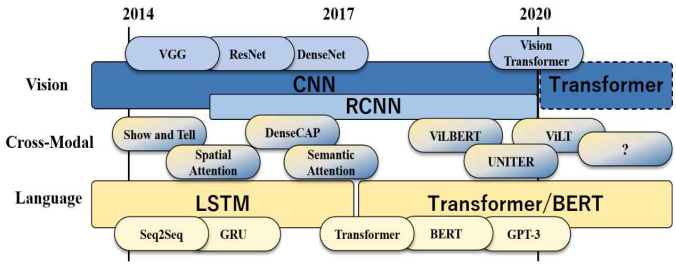
트랜스포머 구조 조사 및 분석

Ⅰ. 서론

1. Transformer 모델의 소개



위 그림은 컴퓨터 비전 및 자연어 처리 분야의 대세 모델들을 표시한 간략한 타임라인입니다. 그림에서 볼 수 있듯이, 트랜스포머는 제안된 후 기존에 가장 인기 있었던 딥러닝 모델인 CNN, RNN을 제치고 대세가 되었습니다.
트랜스포머를 기반으로 한 딥러닝 모델들은 각종 벤치마크에서 신기록을 경신했습니다. 이는 트랜스포머가 Self-Attention에 완전히 의존하기 때문입니다. 트랜스포머는 CNN에서 쓰이는 Convolution 연산 및 RNN에서 쓰이는 Recurrence 연산을 사용하지 않고 Self-Attention 연산에만 의존하는 최초의 모델이었습니다.

2. Transformer 모델이 왜 중요한지, NLP 및 기타 분야에서의 역할

그러면 트랜스포머 모델이 왜 대세가 되었고 중요할까요? 단순하게 생각하면 성능이 뛰어나기 때문입니다.

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Transformer 논문에서는 Self Attention이 좋은 이유에 대해 이렇게 설명합니다.
논문 속 표에서 Self Attention, RNN, CNN, 그리고 Self Attention(restricted)의 성능을 알 수 있는데요
Attention(restricted)은 문장 길이가 너무 길 경우, 일정 길이(r)로 문장을 쪼개 Self-attention을 수행하는 것입니다
위 표를 통해 알 수 있는 Self Attention의 특징은 계산량이 적다는 것입니다.
복잡도에 대해 안다는 가정하에 설명은 따로 안하겠습니다.

그렇다면 이러한 성능이 트랜스포머로 어떤 것들을 할 수 있냐면, 아래 표와 같습니다

모델	예	작업
인코더	알버트, BERT, DistilBERT, ELECTRA, RoBERTa	문장 분류, 개체명 인식, 추출 질문 답변
디코더	CTRL, GPT, GPT-2, 트랜스포머 XL	텍스트 생성
인코더-디코더	BART, T5, 마리안, mBART	요약, 번역, 생성적 질문 답변

Ⅱ. 이론적 배경

1. Self-Attention

이제 구조에 대해 말씀드리겠습니다. 트랜스포머의 핵심 구성 요소는 Self-Attention이기에 Self-Attention의 작동 원리를 설명하겠습니다.

1) Attention 스코어 계산

$$Attention\ Score = Query \cdot Key^T$$

첫 번째 단계에서는 쿼리, 키, 밸류라는세 가지 vector를 생성하고 Query와 Key의 내적을 계산하여 attention score를 얻습니다.

2) Softmax 함수를 통한 가중치 계산

$$Weight = Softmax(Attention\ Score)$$

두 번째 단계에서는 소프트맥스로 attention score를 확률 분포로 변환합니다.

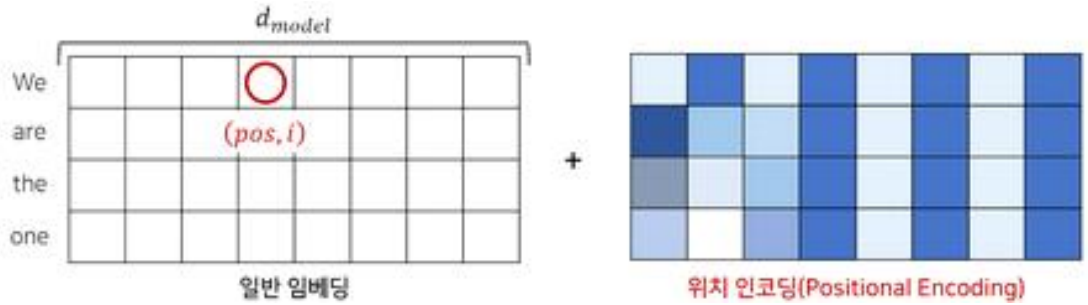
3) 가중치를 적용한 value 계산

$$Output = Weights \times value$$

세 번째 단계에서는 각 단어의 value vector를 가중치와 곱한 뒤 합산하여 Self Attention의 결과를 얻습니다.
이러한 구조를 통해 입력 시퀀스의 각 단어가 다른 모든 단어와의 관계를 고려하여, 문맥을 고려한 표현을 얻을 수 있습니다.

2. Positional Encoding

Transformer 계열의 모든 방법에서 등장하는 Positional Encoding에 대해 살펴보겠습니다. 이는 CNN과 결정적인 차이이기도 합니다.

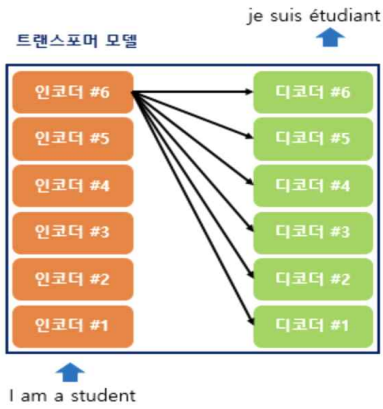


Positional Encoding을 단어 그대로 해석해보면 위치 정보를 입력해주자는 뜻인데요.
문장의 의미를 이해하기 위해서는 단어의 순서 (문장의 단어의 위치 정보)가 매우 중요합니다.
그러나 임베딩 행렬만으로는 순서의 정보를 이해할 수 없기에
Positional Encoding은 임베딩 행렬에 위치 정보를 더하는 방식으로 위치 정보를 처리합니다.

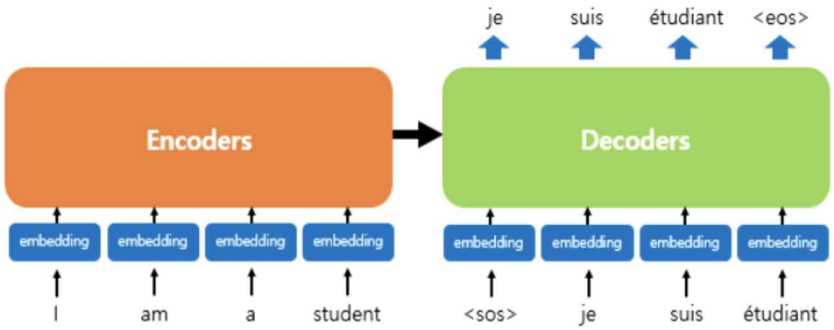
Ⅲ. Transformer 구조 분석

1. Encoder와 Decoder

트랜스포머는 RNN을 사용하지 않지만, 기존 seq2seq처럼 인코더에서 입력 시퀀스를 입력받고, 디코더에서 출력 시퀀스를 출력하는 인코더-디코더 구조를 유지합니다.
다른 점은 인코더와 디코더라는 단위가 N개 존재할 수 있다는 점입니다.

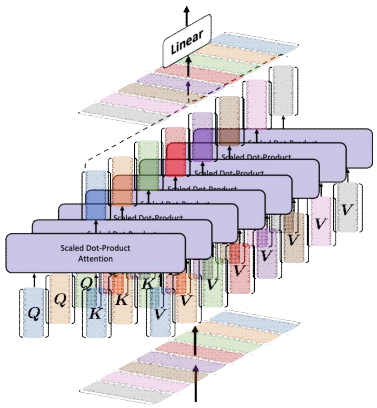


좀 더 설명하자면
이전 seq2seq는 인코더와 디코더에서 각각 하나의 RNN이 t개의 시점(time-step)을 가지는 구조였다면, 트랜스포머는 인코더와 디코더라는 단위가 N개로 구성되는 구조입니다.



decoders는 기존의 seq2seq처럼 시작 심볼 <sos>를 입력으로 받아 종료 심볼 <eos>가 나올 때까지 연산을 진행합니다.
이는 RNN은 사용되지 않지만, 여전히 인코더-디코더 구조는 유지되는 것입니다.
트랜스포머의 인코더와 디코더는 단순히 각 단어의 임베딩 벡터들을 입력받는 것이 아니라 임베딩 벡터에서 조정된 값을 입력받습니다.

2. Multi-Head Attention



트랜스포머는 Self-Attention을 병렬로 h번 학습시키는 Multi-Head Attention 구조로 되어있는데요
그렇다면 왜 굳이 멀티 헤드 구조를 사용하는지 그 이유를 설명하겠습니다.

1. 다양한 표현 학습

각 Attention head는 서로 다른 부분에 집중할 수 있습니다.
이는 모델이 입력 시퀀스의 다양한 측면을 학습할 수 있도록 도와줍니다.

2. 정보 병렬 처리

여러 Attention head를 사용하면 병렬 처리가 가능해지고 각 Attention head는 독립적으로 계산되기 때문에 계산 효율성을 높이고, 모델이 더 빠르게 학습할 수 있도록 도와줍니다.

3. 고차원 공간에서의 유연한 주의 분포

하나의 Attention head만 사용하면 고차원 공간에서의 복잡한 주의 분포를 표현하기에 한계가 있지만
여러 Attention head를 사용하면 고차원 공간을 여러 차원으로 나누어 각 헤드가 다양한 관점에서 주의를 분산시킬 수 있습니다.
그렇게 해서 모델이 더 복잡하고 다양한 패턴을 학습할 수 있게 합니다.

4. 어텐션의 집합적 표현

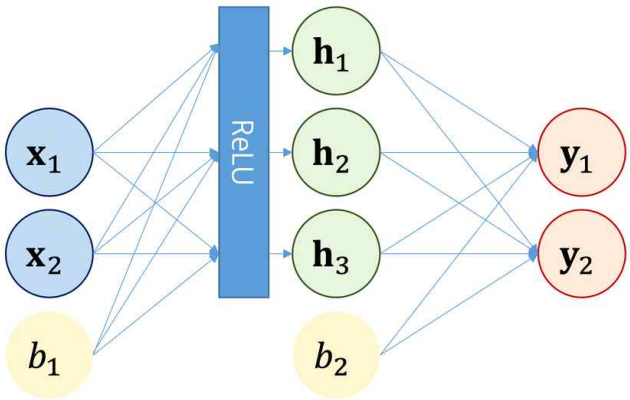
각 Attention head는 서로 다른 가중치 행렬을 학습하여 서로 다른 특징들을 추출합니다. 이를 통해 모델이 다양한 특징을 조합하여 더 정교하고 강력한 표현을 얻을 수 있습니다.

이러한 특징의 이점으로 성능 향상이 있습니다.

실제로 여러 Attention head를 사용하면 모델의 성능이 향상됩니다
예를 들어, Transformer 모델에서는 단일 Attention head보다 다중 Attention

head를 사용했을 때 기계 번역, 자연어 이해 등 다양한 NLP 작업에서 더 우수한 성능을 보였습니다.

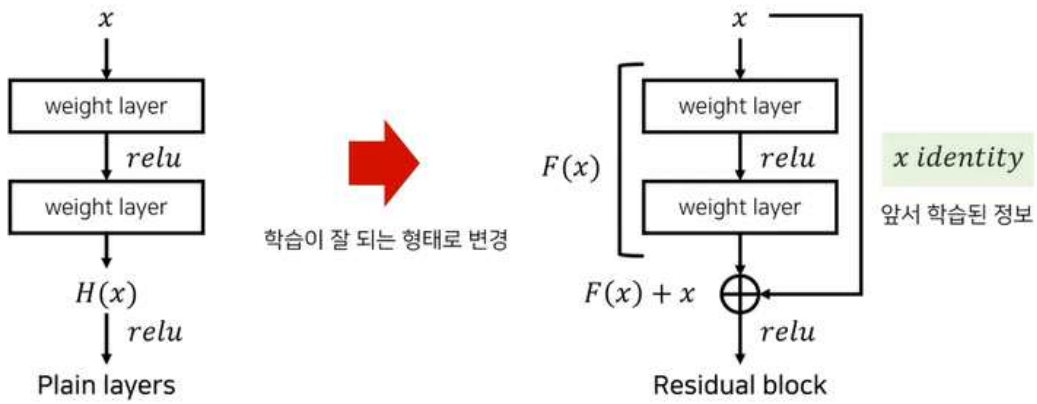
3. Feed-Forward Networks



트랜스포머 구조 내에서 각 인코더와 디코더 블록은 Self-Attention 층 다음에 Feed-Forward Networks 층을 가집니다. 이 층은 어텐션의 결과를 취합하여 전달하는 역할을 수행합니다.

4. Residual Connections & Layer Normalization

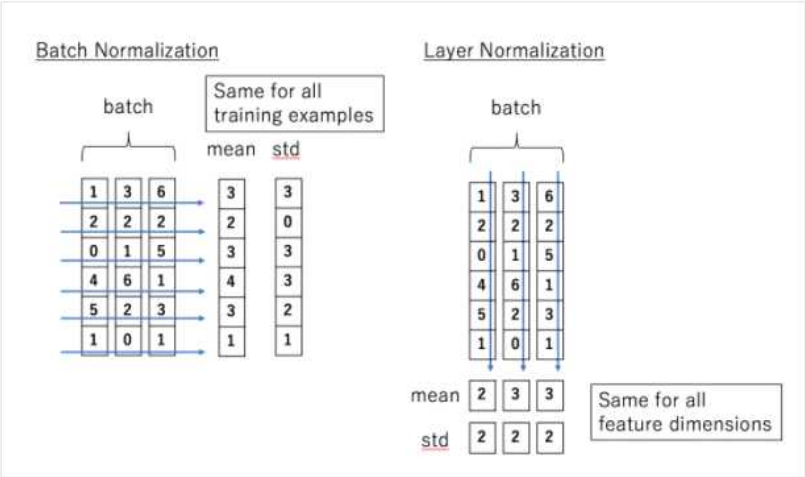
1) *Residual Connections*



일반적인 feed-forward 방식의 신경망에서, input은 앞에 놓인 layer들을 차례차례 통과하게 됩니다.
반면, residual connection은 '건너뛰는' 경로를 제공합니다.

인공신경망을 학습할 때 깊은 모델이 더 좋은 성능을 낼 것이라고 생각할 수 있는데, 일정 깊이에 도달하면 오히려 정확도가 감소하는 경향을 보입니다. 이 문제를 해결하면서 학습을 돕습니다

2) Layer Normalization



Layer Normalization이 무엇인지 설명하자면 평균과 분산으로 정규화해 학습을 효율적으로 만드는 방법입니다.
배치 정규화는 들어보셨을 것이라 생각하는데요.
배치 정규화는 미니 배치 단위로 계산을 하고 Layer Normalization은 input을 기준으로 평균과 분산을 계산하게 됩니다.
mini-batch 단위가 아닌 위와 같은 방법은 세 가지 특징을 가집니다.
이러한 특징들로 안정적인 학습이 진행시킬 수 있습니다.