Attention is all you need

상태	시작 전

0.

성능 좋은 시퀀스 변환 모델은 대체로 인코더와 디코더를 포함한 복잡한 RNN 또는 CNN 신경망에 기반을 두고 있음

해당 논문은 recurrence와 convolution 을 전부 제외하고 오직 attention mechanism에만 기반한 Transformer라는 간단한 모델을 제안.

abstraction에서 제안한 모델의 특징은 아래와 같이 요약할 수 있음

- 어텐션 기법으로 재귀적으로 각각의 시퀀스를 처리하지 않고 오직 행렬 곱을 이용해서 병렬적으로 시퀀스 데이터를 처리하기 때문에 전보다 훨씬 더 빠른 처리가 가능함
- WMT 2014 data set을 이용해서 영어를 독일어로 번역하는 작업, 영어를 불어로 번역하는 작업에서 훨씬 개선된 성능을 보여줌
- 크거나 한정된 학습 데이터를 가지고서도 다른 task들에 성공적으로 일반화될 수 있음을 보임

1. Introduction

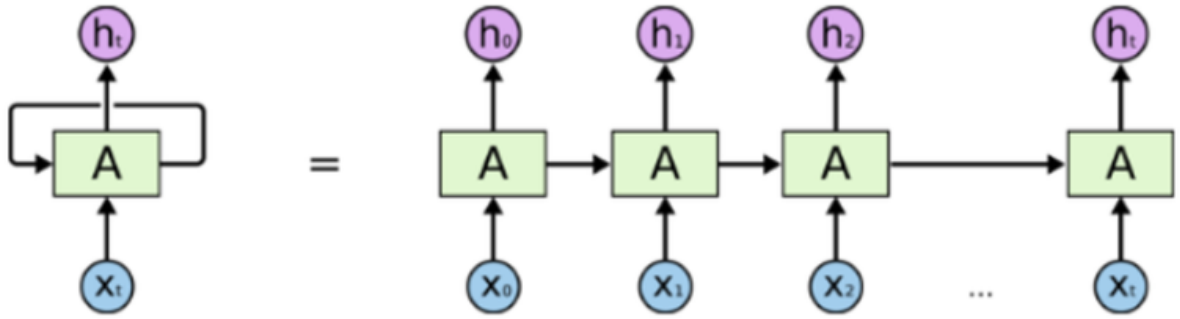
> 순환 모델의 병렬처리 문제

RNN, LSTM, GRU(Gate Recurrent Unit) 등은 언어 모델링 및 기계 번역과 같은 sequence 모델링 및 변환 문제에서 뛰어난 성과를 보이는 Recurrent 모델

Recurrent 모델은 보통 입력과 출력의 symbol position에 따라 계산을 수행

계산 단계에서 위치를 정렬하고 이전 상태 ht-1과 위치 t의 함수인 은닉상태 ht를 생성

[기본적인 RNN 구조]



이는 시퀀스가 길수록 메모리 문제로 인해 병렬적 처리가 어렵게 하고, 최근 들어 모델의 성능 자체는 비약적으로 상승했지만 위의 문제는 해결되지 않음

> ***Attention mechanism 제시***

Attention mechanism은 입력과 출력 시퀀스 거리와 관계없이 의존성을 모델링할 수 있으나 거의 대부분의 경우 순환 네트워크와 함께 사용되고 있어 효율적인 병렬화 불가능

→ 따라서 이 논문에서 순환없이 입력값과 출력값 간 전역 의존성을 모델링할 수 있는 Attention mechanism만을 사용한 모델 구조인 Transformer 제안

2. Background

Extended Neural GPU, ByteNet, ConvS2S 에서도 연속적 연산을 줄이기 위한 연구가 이루어졌는데, 모두 CNN을 기본 구성 요소로 사용

- 이러한 모델들은 입출력 간 관련성을 파악하기 위해 거리에 따라(선형 또는 로그 비례) 계산량이 증가
- 따라서 입력값과 출력 값의 거리가 멀수록 의존성을 알기 어려움

반면 Transformer에서는 Multi-Head Attention을 통해 상수 시간의 계산만으로 가능

(i) Self-attenion

- 말 그대로 자신에게 수행하는 어텐션 기법으로 단일 시퀀스 안에서 서로 다른 위치에 있는 요소들의 의존성을 찾아냄
- 메커니즘 독해, 추상적 요약, 텍스트 포함, 학습 과제, 독립적인 문장 표현을 포함한 다양한 task에서 성공적으로 사용됨

(ii) End-to-end memory network

- 시퀀스가 배열된 recurrence보다 recurrent attention mechanism 기반

- 간단한 언어 질문 답변 및 언어 모델링 작업에서 좋은 성능을 보임

> ***장거리 의존성(Long term dependency) 문제***

대부분의 자연어처리는 Encoder-decoder 구조를 가지는 recurrent model인 RNN, CNN 등이 주로 사용됨

→ 문장의 순차적인 특성이 유지되지만 먼거리에 있는 의존성을 알기 취약하다는 단점이 존재

- RNN의 경우 시퀀스 길이가 길어질수록 정보 압축 문제 존재
- CNN의 경우 합성곱의 필터 크기를 넘어서는 문맥은 알아내기 어려움

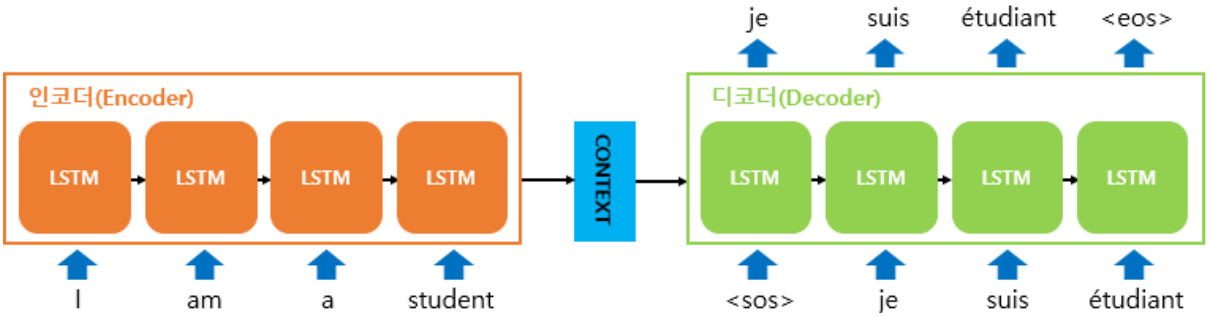
반면 Transformer

- 순환없이 Attention mechanism만을 이용해 의존성을 찾을 수 있음
- Self-attention에만 의존하는 최초 모델

3. Model Architecture

> ***seq2seq 구조***

경쟁력 있는 neural sequence transduction 모델은 대부분 인코더-디코더 구조 (seq2seq)



이전에 생성된 출력을 다음 단계에서 사용

- 이전 단계가 완료되어야 다음 단계를 수행할 수 있음
- 병렬적 처리 불가

> ***Transformer 구조***

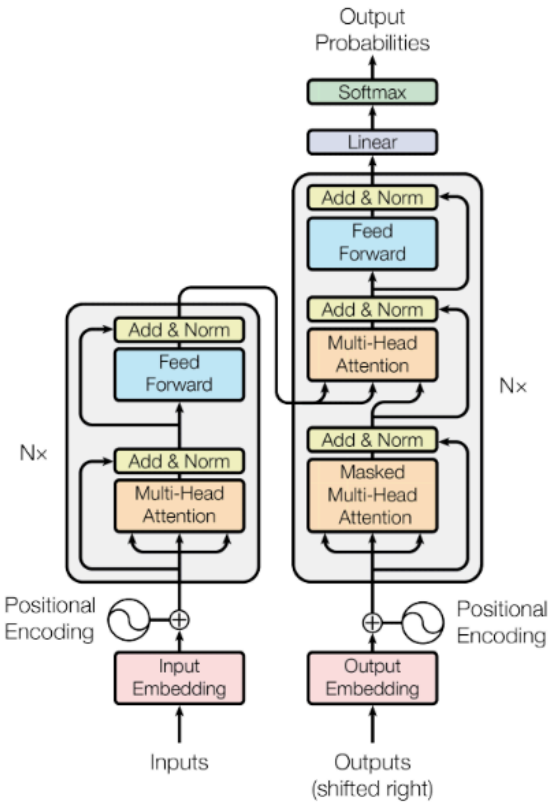


Figure 1: The Transformer - model architecture.

3.1 Encoder and Decoder Stacks

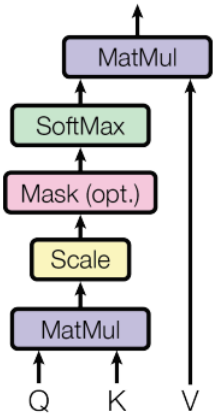
인코더와 디코더 모두에 대해 Self-Attention과 point-wise FC layer를 쌓아올려 사용

- 인코더 : 6개의 동일한 레이어로 이루어져 있으며 하나의 인코더는 Self-Attention layer와 Feed Forward Neural Network라는 두 개의 Sub layer로 이루어져 있음
- 디코더 : 6개의 동일한 레이어, 각 레이어는 인코더가 Sub layer로 가진 Self-Attention layer와 Feed Forward Neural Network 외에 하나의 레이어를 더 가짐👉🏻 인코더의 stack 출력에 대해 Multi head Attention 수행

3.2 Attention

(i) Scaled Dot-Product Attention

Scaled Dot-Product Attention



> Q: 영향을 받는 벡터
>
> K: 영향을 주는 벡터
>
> V: 주는 영향의 가중치 벡터

다음과 같이 계산

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

Output : value의 가중합으로 계산

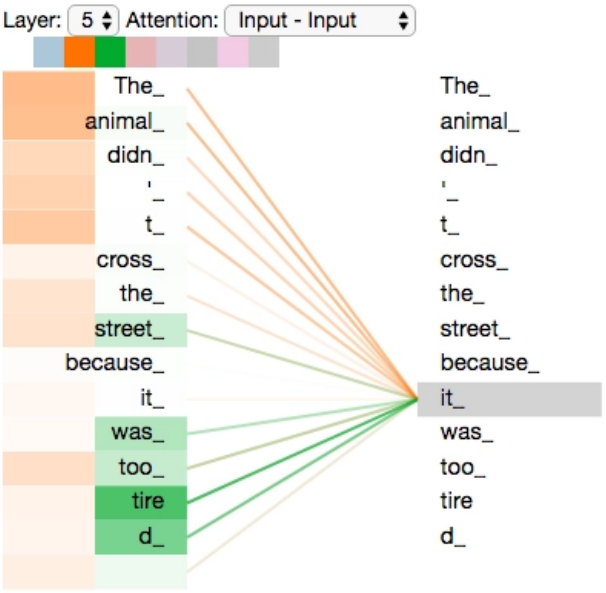
가중합에 이용되는 가중치 : query와 연관된 key의 호환성 함수에 의해 계산

< 계산 과정 >
(1) 워드 임베딩에 가중치를 곱해서 Query, Key, Value를 계산
(2) Query * Key = attention score
: 값이 높을 수록 연관성이 높고, 낮을 수록 연관성이 낮다.
(3) key 차원수로 나누고 softmax 적용
softmax 결과 값은 key값에 해당하는 단어가 현재 단어에 어느정도 연관성이 있는지 나타냄
(4) 문장 속에서 지닌 입력 워드의 값 = softmax 값과 value 값을 곱하여 다 더함
→ 문장에서 연관성이 반영된 값이기 때문에 문맥을 알 수 있도록 함

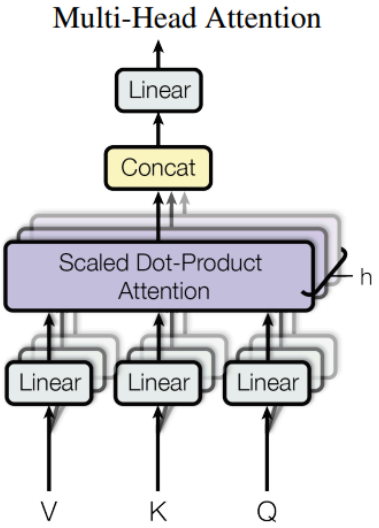
(ii) Multi-Head Attention

" 모델이 다양한 관점에서 문장을 해석할 수 있도록 하는 역할 "

아래 그림에서 문장에서 it이라는 단어의 attention이 The Animal일지, Tired일지 모델이 헷갈릴 수 있음



Multi-Head Attention은 Head (논문에서는 8개)를 두어 모델이 Head 개수만큼의 Scaled dot product Attention 연산을 수행할 수 있게 하여 모델이 다양한 관점의 Attention Map을 만들도록 함



각 헤드의 차원 수가 줄어들었으므로 전체 계산량은 single-head attention과 비슷한 수준

(iii) Applications of Attention in our Model

Transformer는 multi-head attention을 3가지 방법으로 사용

- **Encoder-decoder attention** :이전 decoder 레이어에서 오는 query들과 encoder의 출력으로 나오는 memory key, value들과의 attention → 이는 decoder의 모든 위치에서 input sequence의 모든 위치를 참조할 수 있도록 함
- **Self-attention in encoder** :Encoder의 각 위치들은 이전 레이어의 모든 위치들을 참조할 수 있음
- **Self-attention in decoder** :decoder의 각 위치들은 decoder 내의 다른 위치들을 참조할 수 있는데, 이전부터 자신 위치까지만을 참조할 수 있음 → auto-regressive 성질을 살리면서도 정보가 잘못 흐르는 것을 막기 위해서

3.3 Position-wise Feed-Forward Networks

- Attention layer과 함께 fully connected feed-forward network가 사용됨, 인코더 및 디코더의 각 계층에 개별적으로 위치

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

> 사이에 ReLU 활성화가 있는 두 개의 선형 변환으로 구성
>
> *input과 output의 차원은 512, 은닉층의 차원은 2048*

3.4 Embeddings and Softmax

- 다른 sequence transduction 모델과 마찬가지로, input과 ouput token을 embedding layer를 거쳐서 사용
- 이렇게 생성된 embedded vector는 semantic한 특성을 잘 나타내게 됨
- 또한 input embedding 과 output embedding에서 weight matrix를 서로 share하여 사용

3.5 Positional Encoding

> *토큰의 상대적인 위치(또는 절대적인 위치)에 대한 정보를 제공하기 위한 역할*

Transformer는 Recurrent model을 사용하지 않고 오직 Attention mechanism만을 사용하여 만들기 때문에 Sequence 정보를 담아낼 수가 없음

→ 따라서 별도로 이러한 sequence 정보를 데이터에 추가해주어야 하는데 이 역할을 하는 것이 바로 "positional encoding"

- positional encoding으로 선택할 수 있는 방식은 다양한데, 논문에서는 sine과 cosine 함수를 사용

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

pos는 position, i는 dimension

고정된 오프셋 k에 대해 PEpos+k가 PEpos의 선형 함수로 표현될 수 있기 때문에 모델이 쉽게 상대적인 위치를 참조할 수 있을 것이라 가정했기 때문

4. Why Self-Attention

recurrence, convolution과 비교했을 때 장점을 다음과 같이 정리할 수 있다.

- **하나의 레이어 당 전체 연산 복잡도 감소**: RNN 계열에서는 하지 못했던 병렬 처리 연산의 양을 대폭 늘려 자연스레 학습 시간 감소
- **Long term dependency의 문제점 해결**: 트랜스포머 모델은 대응관계가 있는 토큰들 간의 물리적인 거리값들 중 최댓값이 다른 모델에 비해 매우 짧아 '장기간 의존성'을 잘 학습할 수 있고 시퀀스 변환 문제도 잘 해결할 수 있음
- **해석 가능한 모델**: 'Attention' 이라는 가중치를 시각화하여 토큰들 간의 대응관계를 눈으로 직접 확인 가능

→ 다른 모델에 비해 트랜스포머는 모델이 내뱉은 결과를 해석할 수 있다는 장점