[논문 리뷰]Transformer (Attention Is All You Need)

Abstract

기존 sequence transduction model들은 인코더와 디코더를 포함한 복잡한 recurrent 나 cnn에 기반함

가장 성능이 좋은 모델 또한 attention mechanism으로 인코더와 디코더를 연결한 구조임

"**Transformer**" : 온전히 attention mechanism에만 기반한 구조. (recurrence 나 convolution은 사용하지 않음)

- > 더 parallelizable하고, 훨씬 적은 학습 시간이 걸림

English-to-German , English-to-French translation에서 SOTA 달성함

또한 Transformer는 다른 task에서도 일반적으로 잘 동작함

Introduction

Sequence Modeling과 transduction 문제에서 RNN, long-term memory, gated RNN이 SOTA를 달성해 옴

Recurrent model은 parallelization이 불가능해 longer sequence length에서 치명적임

- 최근 연구에서 factorization trick과 conditional computation을 통해 계산 효율성을 많이 개선
- 특히 conditional computation은 모델 성능도 동시에 개선
- > 그러나 여전히 근본적인 sequential computation의 문제는 남아있음

Attention mechanism은 다양한 분야의 sequence modeling과 transduction model에서 주요하게 다뤄짐

- Attention mechanism은 input과 output sequence간 길이를 신경쓰지 않아도 됨
- > 하지만 여전히 recurrent network와 함께 사용되었음

Transformer

- input과 output간 global dependency를 뽑아내기 위해 recurrence를 사용하지 않고, attention mechanism만을 사용함

- > parallelization이 가능해 적은 시간으로 translation quality에서 SOTA를 달성할 수 있었음

Background

sequential computation을 줄이는 것은 Extended Neural GPU, ByteNet, ConvS2S에서도 다뤄짐

- 이 연구들은 모두 CNN을 basic building block으로 사용함
- input output 거리에서 dependency를 학습하기 어려움
- > Transformer에서는 Multi-Head Attention 으로 상수 시간으로 줄어듦

Self-attenion(=intra-attention)

- reading comprehension, abstractive summarization, textual entailment, learning task, independent sentence representations를 포함한 다양한 task에서 성공적으로 사용됨

End-to-end memory network

- sequence-aligned recurrence 보다 recurrent attention mechanism에 기반함
- simple-language question answering 과 language modeling task에서 좋은 성능을 보임

Transformer는 온전히 self-attention에만 의존한 최초의 transduction model (sequence-aligned RNN이나 Convolution을 사용하지 않음)

Model Architecture

(1) Encoder and Decoder Stacks

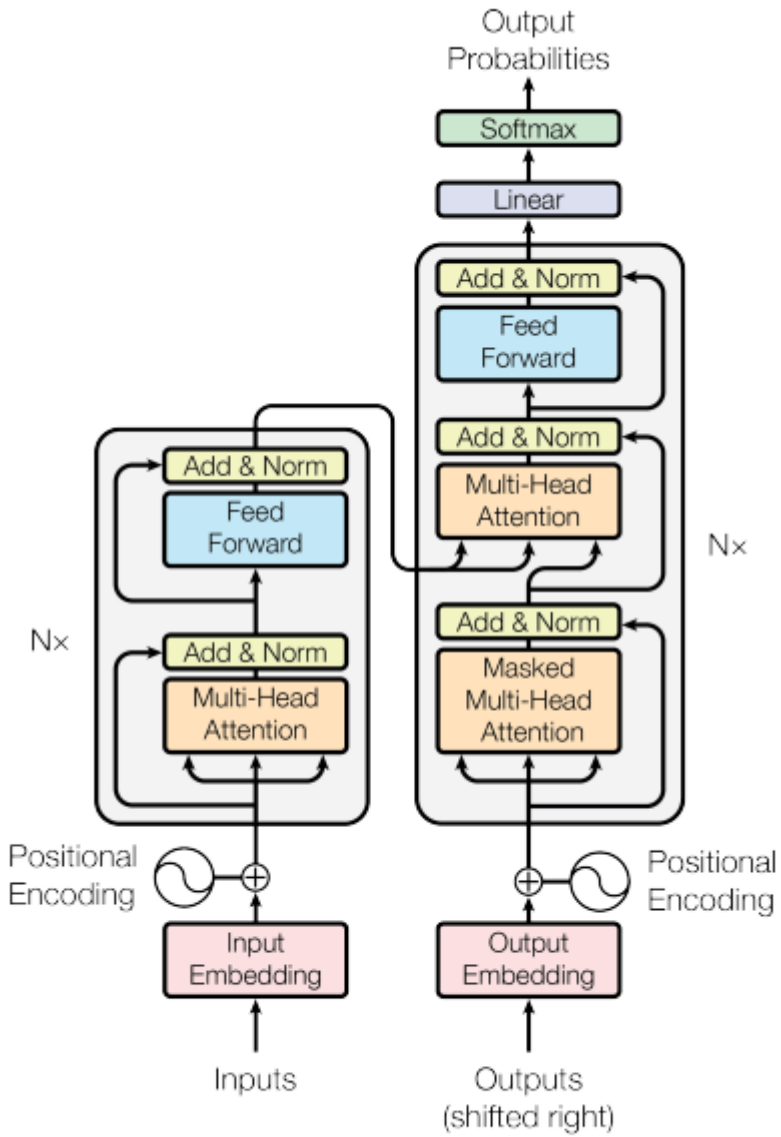


Figure 1: The Transformer - model architecture.

transformer의 구조

Encoder

- Encoder는 6개의 identical layer로 이루어짐
- 각 layer는 두 개의 sub-layer가 있음
- 첫 번째 sub-layer는 multi-head self-attention mechanism
- 두 번째 sub-layer는 간단한 position-wise fully connected feed-forward network
- 각 two sub-layers 마다 layer normalization 후에 residual connection을 사용함

- 즉 각 sub-layer의 결과는 LayerNorm(x+Sublayer(x)) 임
- residual connection을 구현하기 위해, embedding layer를 포함한 모든 sub-layer들의 output은 512 차원임
    - dmodel = 512

Decoder

- Decoder도 마찬가지로 6개의 identical layer로 이루어짐
- 각 Encoder layer의 두 sub-layer에, decoder는 세번째 sub-layer를 추가함
    - encoder stack의 결과에 해당 layer가 multi-head attention을 수행함
- 마찬가지로 residual connection 적용

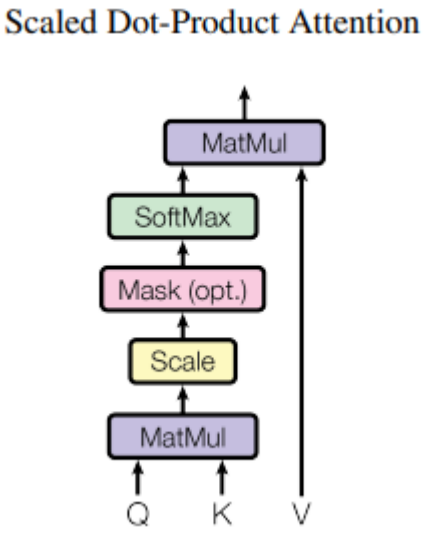
---

(2) Attention

Attention function은 쿼리와 key-value쌍을 output에 매핑함 (query,key,value,output은 모두 vector임)

output은 value들의 weighted sum으로 계산됨

1. Scaled Dot-Product Attention



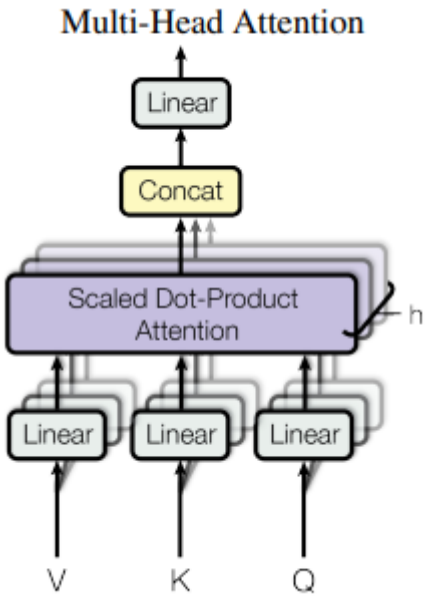
scaled dot-product attention

- Input : query, key의 dimension dk, value의 dimension dv
- 모든 쿼리와 key에 대해 dot product를 계산하고, √dk 로 나눠주고, weight을 적용하기 위해 value에softmax함수를 적용

두 가지 Attention function 이 있음

1. Additive attention : single hidden layer로 feed-forward layer network를 사용해 compatibility function을 계산
2. Dot-product attention : scaling factor (1√dk) 를 제외하면 이 연구에서의 attention 방식과 동일

2. Multi-Head Attention



multi-head attention

Single attention을 dmodel dimensional keys, values, queries에 적용하는 것보다,

queries, keys, values를 h번 서로 다른, 학습된 linear projection으로 linear하게 project하는 게 더 효과적이라는 사실을 알아냄

- > project된 각 값들은 병렬적으로 attention function을 거쳐 dvdimensional output value를 만들어 냄
- > 이 결과들은 다시 합쳐진 다음, 다시 한번 project 되어 최종 결과값을 만듦
- > 각 head마다 차원을 줄이기 때문에, 전체 계산 비용은 전체 차원에 대한 single-head attention과 비슷함

3. Applications of Attention in our Model

Transformer는 세 가지 방법으로 multi-head attention을 사용함

1. 인코더-디코더 attention layers 에서
    - query는 이전 디코더 layer에서 나옴
    - memory key와 value는 인코더의 output에서 나옴
    - > 따라서 디코더의 모든 position이 input sequence의 모든 position을 다룸
    - 전형적인 sequence-to-sequence model에서의 인코더-디코더 attention 방식임
2. 인코더는 self-attention layer를 포함하고 있음
    - self-attention layer에서 key, value, query는 모두 같은 곳(인코더의 이전 layer의 output)에서 나옴
    - 인코더의 각 position은 인코더의 이전 layer의 모든 position을 다룰 수 있음
3. 디코더 또한 self-attention layer를 가짐
    - 마찬가지로, 디코더의 각 position은 해당 position까지 모든 position을 다룰 수 있음
    - 디코더의 leftforward information flow는 auto-regressive property 때문에 막아줘야 할 필요가 있음
    - > 이 연구에서는 scaled-dot product attention에서 모든 softmax의 input value 중 illegal connection에 해당하는 값을 $-\infty$로 masking out해서 구현함

      $-\infty$

---

(3) Position-wise Feed-Forward Networks

인코더 디코더의 각 layer는 fully connected feed-forward network를 가짐

- 이는 각 position에 따로따로, 동일하게 적용됨
- ReLu 활성화 함수를 포함한 두 개의 선형 변환이 포함됨

linear transformation은 다른 position에 대해 동일하지만 layer간 parameter는 다름

---

(4) Embeddings and Softmax

다른 sequence transduction models 처럼, 학습된 임베딩을 사용함

- input 토큰과 output 토큰을 dmodel 의 벡터로 변환하기 위함

decoder output을 예측된 다음 토큰의 확률로 변환하기 위해 선형 변환과 softmax를 사용함

- tranformer에서는, 두 개의 임베딩 layer와 pre-softmax 선형 변환 간, 같은 weight의 matrix를 공유함

---

(5) Positional Encoding

Transformer는 어떤 recurrene, convolution도 사용하지 않기 때문에, sequence의 순서를 사용하기 위해 sequence의 상대적, 절대적 position에 대한 정보를 주입해줘야 함

인코더와 디코더 stack 아래의 input 임베딩에 "**Positional Encoding**"을 추가함

- 즉 positional encoding의 각 차원은 sine 곡선에 해당함
- 모델이 상대적인 position으로 쉽게 배울 수 있을거라 가정하여 위 function을 사용함
    - 어떤 고정된 offset k라도 PEpos+k가PEpos+k로 표현될 수 있기 때문

학습된 Positional Embedding을 사용해 실험을 해보았음

- 두 방식은 거의 같은 결과를 보임
- transformer에선 sine 곡선의 방식을 택함
    - model이 더 긴 sequence 길이를 추론할 수 있게 해주기 때문

Why Self-Attention

recurrent, convolution layer와 self-attention을 비교

1. layer당 전체 계산 복잡도
2. sequential parallelize 할 수 있는 계산의 양
3. network에서 long-range dependency 사이의 path 길이
    - network에서 순회해야하는 forward 와 backward의 path 길이가 이런 dependency를 학습하는 능력에 영향을 주는 주요 요인
    - input과 output sequence에서 position의 조합 간의 path가 짧을수록, long-range dependecy를 학습하기가 쉬움
    - > input과 output position 사이의 최대 path 길이를 비교할 것

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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

self-attention layer는 모든 position을 상수 시간만에 연결함

계산 복잡도 면에서, self-attention layer가 n<d<일 때 recurrent layer보다 빠름

- n : sequence length, d : representation dimensionality
- n<d 인 경우가 machine translation에서의 대부분의 경우에 해당함

아주 긴 sequence의 경우 계산 성능을 개선하기 위해,

self-attention은 input sequence의 neighborhood size를 r로 제한할 수 있음

Convolution layer는 일반적으로 recurrent layer보다 더 비용이 많이 듦

- Separable Convolution의 경우 복잡도를 O(knd+nd2) 까지 줄일 수 있음
- 그러나 k=n 의 경우, transformer 와 같이 self-attention layer와 point-wise feed forward layer의 조합과 복잡도가 같음

self-attention은 더 interpretable한 모델을 만들 수 있음

- attention distribution에 대해 다룸 (논문의 Appendix 참고)
- 각 attention head들은 다양한 task를 잘 수행해내고, 문장의 구문적, 의미적 구조를 잘 연관시키는 성질을 보이기도 함

Training

(1) Training Data and Batching

English-German

- WMT 2014 English-German 데이터셋
- 4.5백만 sentence pairs
- 문장들은 byte-pair 인코딩으로 인코딩 되어있음

English-French

- WMT 2014 English-French 데이터셋
- 36M sentences 와 32000 word-piece vocabulary로 쪼개진 토큰들

(2) Hardware and Schedule

- 8개의 NVIDIA P100 GPU로 학습
- base model은 12시간 동안 (100,000 step) 학습시킴
- big model 은 3.5일 동안 (300,000 step) 학습시킴

(3) Optimizer

- Adam optimizer 사용
- lrate=d−0.5model·min(stepnum−0.5,stepnum·warmupsteps−1.5)

(4) Regularization

세 가지 regularization을 사용함

Residual Dropout

1. 각 sub-layer의 output에 dropout을 적용

2. 임베딩의 합과 positional 인코딩에 dropout 적용

Label Smoothing

3. 학습 중에 label smoothing 적용

Conclusion

(1) Machine translation

WMT 2014 English-to-German translation, English-to-French translation에서 SOTA 달성

(2) Model Variation

(3) English Constituency Parsing

English Constituency Parsing에서도 잘 일반화해서 사용할 수 있는지 실험함

구체적인 tuning 없이도 놀라운 성능을 보임