카테고리 없음

[Euron] 2week_Attention Is All You Need

yejjji 2025. 3. 17. 12:19

0. Abstract

기존의 sequence transduction(시퀀스 변환) 모델들은 복잡한 recurrent(순환) 이나 convolutional neural(합성곱) 신경망에 기반을 둔다.

시퀀스 변환 모델이란?
입력 시퀀스를 받아 출력 시퀀스로 변환하는 딥러닝 모델
대표적 활용: NLP(자연어 처리), 음성 인식, 기계 번역

<대표적인 시퀀스 변환 모델>
1. RNN 기반 모델: + 순차적 데이터 학습이 가능함
- 긴 시퀀스에서 기울기 소실 문제
2. CNN 기반 모델: + RNN보다 병렬화 가능하여 빠름
- 장기 의존성 학습 어려움

3. Attention 기반 모델 -> **Transformer**: + 병렬 연산 가능, 연산 속도 빠름, 장기 의존성 학습 가능

- 높은 연산량, 하지만 GPU로 해결

장기 의존성이란?
시퀀스 데이터에서 **멀리 떨어진 요소들 간의 관계를 학습**하는 능력
예) 자연어 처리에서의 장기 의존성
문장: "나는 오늘 너무 피곤해서 **커피를 마셨다**. 그래서 지금은 조금 더 **기운이 난다**."

-> 기운이 난다는 앞 문장의 커피를 마셨다와 관련이 있지만, 중간에 다른 단어들이 많아서 거리가 멀다. RNN 같은 모델들은 이런 장기적인 관계를 학습하지 못할 수도 있다.

일반적으로 encoder(인코더)와 decoder(디코더)로 구성이 되는데, 가장 성능이 좋은 모델들은 encoder와 decoder를 연결하는 attenion(어텐션) mechanism을 포함하고 있다.

본 논문은 이보다 더 간단한 새로운 형태의 모델, **Transformer**을 제안한다.
이 모델은 recurrent와 convolutions 개념을 배제하고, **오직 attention mechanism에만 집중**한 모델이다.

2개의 machine translation(기계 변환) task에서 실험한 결과, **Transformer은 우수한 성능**을 보였고, 특히 병렬 처리 부분에서 훨씬 용이하고, 훈련 시간이 크게 단축된 것을 확인할 수 있었다.
실제로, WMT 2014 English-to-German translation task 에서 28.4 BLEU로, 기존의 가장 우수한 모델인 ensemble보다 2 BLEU가 더 높았다.
WMT 2014 English-to-French translation task 에서 단일 모델 기준에서 41.0 BLEU라는 새로운 최고 점수를 세웠고, 8개의 GPU에서 3.5일을 훈련시키며 기존의 최고 모델들에 비해 적은 훈련 비용으로 학습을 완료하였다.

1. Introduction

RNN 특히, LSTM과 GRU는 언어 모델링과 기계 학습과 같은 sequence m
transduction 문제에서 최고 수준의 모델로 일컬어져 왔다.
Recurrent language 모델과 encoder-decoder 아키텍쳐의 성능을 향상시키
력을 계속 해오고 있다.

Recurrent 모델은 input과 output 시퀀스의 각 신호의 위치를 따라 연산을 수행한다.
위치를 시간 축상의 연산 단계에 정렬시키고, 이전의 hidden state(은닉 상태) 와 위치 input값으로 이루어진 함수인 은닉 상태 시퀀스를 생성한다.

그러나 이러한 순차적 특성으로 인해 병렬화가 어렵고, 긴 시퀀스를 처리할 때에도 심각한 문제가 된다.
특히 메모리 제한으로 인해 여러 샘플을 동시에 학습시키기가 어려워지는 문제가 발생한다.

최근의 연구에서는 연산 효율성을 높이기 위해 factorization tricks 과 conditional computation 같은 기법이 도입되었다.
조건부 연산의 경우 모델 성능도 함께 개선되었다.
하지만, 아직까지도 순차적 연산 방식의 자체적인 근본적인 한계는 여전히 해결되지 않았다.

Attention 메커니즘은 많은 시퀀스 모델링과 순환 모델 작업들에서 중요한 역할을 하며, input값과 output값 시퀀스 사이의 거리에 관계 없이 종속성을 모델링할 수 있게 해준다.

그러나 대부분의 연구에서는, attention 메커니즘이 여전히 순환 신경망과 함께 사용되는 경우가 많았다.

이번 연구에서 Transformer라는 새로운 모델 아키텍처를 제안한다.
Transformer 구조는 **순환 구조를 완전히 배제**하고 **오직 attention 메커니즘을 활용**하여 **input값과 output값 간의 전역 종속성을 학습**하는 방식을 채택한다.
이 구조는 기존의 방법보다 훨씬 높은 병렬 처리 성능을 제공하며, 8개의 P100 GPU에서 단 12시간의 훈련 이후만으로도 기계 번역에서 새로운 최고 성능을 달성할 수 있다는 것을 보여줬다.

2. Background

순차적 연산을 줄이려는 목표는 Extended Neural GPU와 ByteNet, ConvS
기반을 형성한다.
이 모델들은 기본 구조로서 합성곱 신경망을 사용하고, 은닉 표현을 모든 input값과 output 위치들에서 평행하게 계산한다.
2개의 임의의 input값과 output위치들로부터의 신호를 연결하기 위해 필요한 연산의 수는 위치들 간의 거리에 따라 증가한다. 예를 들어, ConvS2S의 경우 선형적으로, ByteNet의 경우 로그적으로 증가한다.
이러한 특성 때문에 먼 위치들 간의 관계를 학습하는 것이 어려워지는 문제가 발생한다.

반면에, Transformer에서는 연산의 수가 상수값으로 줄어든다. 그러나 attention 가중치를 평균화시키는 과정에서 effective resolution(해상도)가 감소하는 문제가 발생한다.

Self-attention(자기 어텐션)이란?
Intra-attention이라고도 말하며, 하나의 시퀀스 내의 서로 다른 위치들 간의 관계를 학습하여 해당 시퀀스의 표현을 계산하는 attention 메커니즘이다.
이 메커니즘은 독해, 추상적 요약, textual entailment, task와 무관한 문장 표현과 같은 다양한 작업에서 성공적으로 사용되어왔다.

End-to-End 메모리 네트워크는 RNN 기반이 아닌, Recurrent Attention 메커니즘을 사용하며, 질문 응답과 언어 모델링과 같은 작업에서 좋은 성능을 보였다.

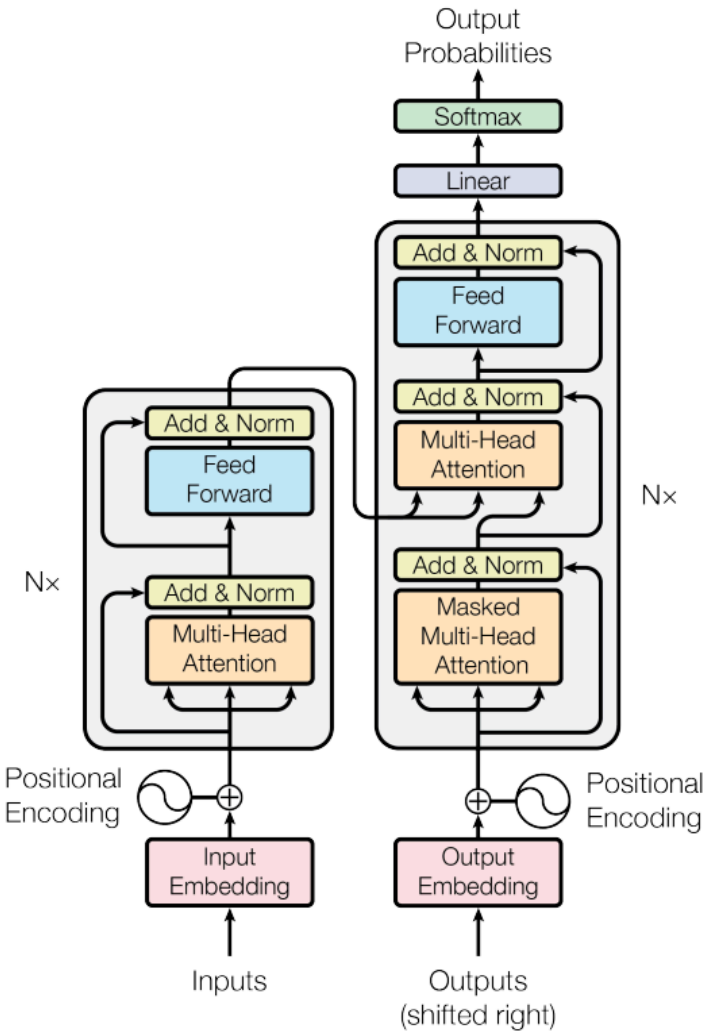
그러나, **Transformer**는 기존의 시퀀스 정렬 방식의 RNN이나 CNN을 전혀 사용하지 않고, 오직 Self-Attention만으로 input과 output의 표현을 계산하는 최초의 Transduction 모델이다.

이어지는 다음 섹션에서는 **Transformer**의 구조를 상세히 설명하고, Self-Attention의 필요성을 논의하며, 기존의 모델들과 비교하여 Transformer가 가지는 장점을 이야기 해보고자 한다.

3. Model Architecture

대부분의 경쟁력 있는 신경망 기반 시퀀스 변환 모델은 Encoder-Decoder 구조를 가지고 있다.
여기서, encoder는 기호 표현들 (x1, ..., xn)로 이루어진 입력 시퀀스를 연속적인 표현들인 z = (z1, ..., zn)로 매핑한다.
이 z를 기반으로 decoder는 하나씩 기호로 이루어진 출력 시퀀스 (y1, ..., yn)를 생성한다.
각 단계에서 모델은 auto-regressive(자기 회귀적) 으로 동작하며, 다음 기호를 생성할 때 이전에 생성된 기호들을 추가 input값으로 사용한다.

Transformer는 전체 아키텍처를 따라 encoder와 decoder 모두에 대해 쌓인 self-attention과 점별 완전 연결층(point-wise, fully connected layers) 을 사용한다.



Transformer 아키텍처 / 왼쪽: encoder, 오른쪽 : decoder

3.1 Encoder and Decoder Stacks

1. Encoder

Encoder는 N = 6개의 동일한 레이어로 구성된다. 각 레이어는 두 개의 하위 레이어를 가지는데, 첫 번째는 다중 헤드 자기 어텐션(Multi-Head Self-Attention) 메커니즘이고, 두 번째는 단순한 위치별 완전 연결 피드포워드 네트워크(Position-wise Fully Connected Feed-Forward Network) 이다.

본 논문은 각 하위 레이어 주변에 residual connection(잔차 연결)을 사용하며, 그 후에 Layer Normalization(층 정규화) 을 적용한다. 즉, 각 하위 레이어의 output은 LayerNorm(x + Sublayer(x))이다.
이때, Sublayer(x)는 해당 하위 레이어에서 구현된 함수이다.

이러한 residual connection을 용이하게 하기 위해, 모델의 모든 하위 레이어와 임베딩 레이어는 출력 차원을 dmodel = 512로 설정하여 output값을 만들어낸다.

2. Decoder

Decoder는 역시나 N = 6개의 동일한 레이어로 구성된다. 각 encoder 레이어의 두 개의 하위 레이어 외에도, decoder는 세 번째 하위 레이어를 추가하여 인코더 스택의 출력에 대해 다중 헤드 어텐션(Multi-Head Attention)을 수행한다. Encoder와 마찬가지로, Decoder도 각 하위 레이어 주위에 잔차 연결(residual connection)을 사용하고, 그 후에 층 정규화(Layer Normalization)를 적용한다.

또한 decoder 스택에서 self-attention 하위 레이어를 수정하여, 위치가 이후의 위치를 참조하지 않도록 한다.
이 masking 기법은 output 임베딩이 한 위치씩 오프셋(offset) 되어 있는 사실과 위치 i에 대한 예측이 오직 i보다 작은 위치에서의 알려진 출력에만 의존하도록 보장합니다.

3.2 Attention

Attenion 함수는 쿼리, 키값 쌍들을 output값에 맵핑하는 방식으로 묘사되고, 쿼리, 키, value, output은 모두 벡터값이다.

Output값은 value값들의 가중합으로 계산이 되는데, 각 value값에 할당된 가중치가 각각 일치하는 키와 쿼리의 compatibility function(호환성 함수)로 계산이 되는 원리이다.

3.2.1 Scaled Dot-Product Attention

본 논문은 특정 attention을 "Scaled Dot-Product Attention"이라고 부른다.
Input값은 쿼리, dk차원을 가진 키값, dv차원을 가진 value로 구성된다.
각 값들의 가중치를 얻기 위해서 모든 키값들에 대해 쿼리를 dot-product(내적)하고, 각각을 sqrt(dk)로 나눠주며, softmax 함수를 적용해준다.

실제로 일련의 쿼리에 대한 attention 함수를 동시에 계산하여 행렬 Q로 묶고, 키와 값도 각각 행렬 K와 V로 함께 묶었다. 본 논문에서는 출력 행렬을 아래의 식과 같이 계산해주었다.

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

가장 공통으로 사용되는 2개의 attention 함수는 **additive attention**과 **dot-attention**이다.

Dot-product attention의 경우 1/sqrt(dk)의 scaling factor를 제외하고는 본 논문의 알고리즘과 동일하다.
Additive attention의 경우 하나의 은닉층으로 feed-forward network를 사용하여 compatibility function을 계산한다.

두 attention function은 이론적 복잡도 측면에서는 거의 차이가 없으나, dot-product attention이 속도가 더 빠르고 공간 효율적이다. 이는 매우 최적화된 행렬 곱셈 형태를 사용하기 때문이다.

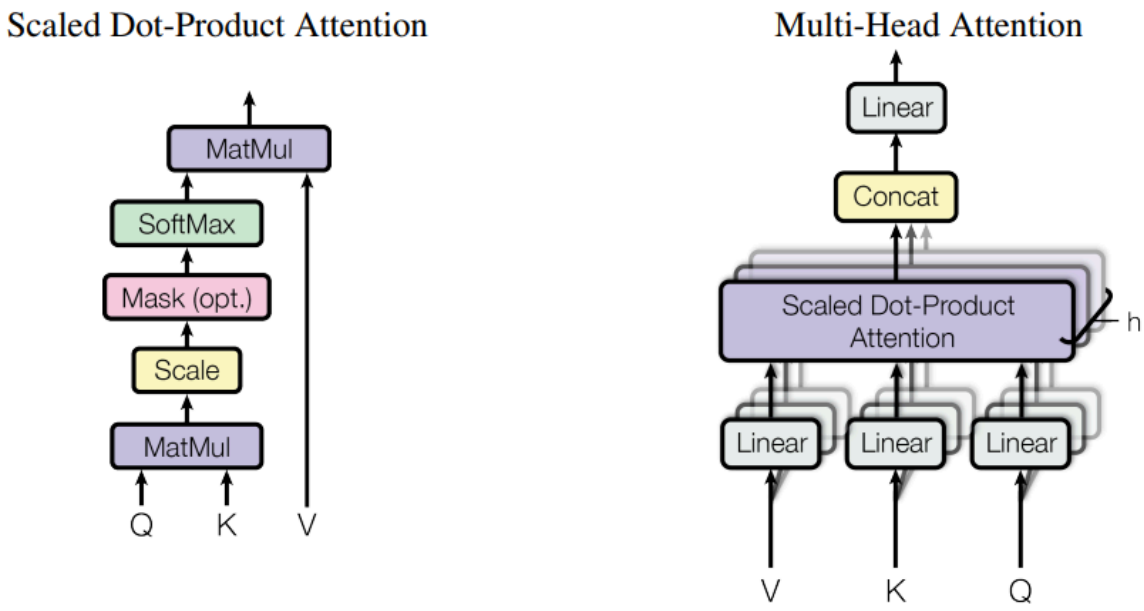
작은 dk값의 경우 2개의 메커니즘에서 비슷하게 수행되는 양상을 가지나, 더 큰 dk값으로 스케일링 없이는 dot-product를 능가한다.
이는 dk값이 클 경우 dot-product의 계산값이 커지고, 이로 인해 softmax function이 큰 값은 매우 커지고 작은 값은 매우 작은 값을 가지는 특성 때문에 특정 값만 매우 큰 중요도를 가지고 나머지는 무시되는 경향이 생긴다.
반대로 dk값이 작을 경우 dot-product의 계산값도 마찬가지로 작아지고, 이로 인해 softmax function의 확률값이 일관된 값을 가지게 되며 가중치가 거의 균등하게 분포하게 된다.

이러한 경향들을 해소하기 위해 1/sqrt(dk)를 곱해주게 된다.
실제로 QKT의 분산값은 분산값에 dk가 곱해진 형태로,dk 값에 영향을 많이 받지만, 1/sqrt(dk)를 QKT에 곱해준 값의 분산값은 분산값의 제곱 형태로 dk 차원에 구애받지 않는다는 것을 확인할 수 있었다.

3.2.2 Multi-Head Attention

dmodel 차원의 키, 값 및 쿼리를 사용하여 단일 주의 함수를 수행하는 대신에, 쿼리, 키 및 값을 각각 dk, dk 및 dv 차원에 대해 학습된 선형 투영을 통해 h배로 선형 투영하는 것이 유익하다는 것을 발견하였다.

이러한 각 투영된 버전의 쿼리, 키 및 값에 대해 주의 함수를 병렬적으로로 수행하여 dv 차원의 출력 값을 산출하였다.
이러한 값들은 연결된 후 다시 한 번 투영되어 output 값이 도출되며, 이는 [illegible] figure2에서 확인할 수 있다.



왼쪽: Scaled Dot-Product Attention / 오른쪽: Multi-Head Attention 모델

Multi-Head Attention 모델은 서로 다른 위치에 있는 다양한 representation subplace 들에 대해 주의를 기울이게 한다.
단일 attention head로 평균을 내는 과정으로 이를 억제할 수 있다.

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

WiQ 는 쿼리 차원, WiK는 키 차원, WiV는 value 차원, Wo는 최종 가중치 행렬을 의미한다.

이 과정에서, h = 8를 병렬 attention층과 head에 적용하였다.

각 head의 차원이 줄어들면서, 총 computational cost가 차원이 가득찬 single-head attention과 비슷해졌다.

3.2.3 Applications of Attention in our Model

Transformer모델은 3개의 방법으로 multi-head attention을 사용한다.

1. encoder-decoder attention 층에서 쿼리들은 이전의 decoder층으로부터
memory 키와 값들은 encoder의 output값으로부터 온다.
이는 decoder에서 모든 위치들이 input sequence에서 모든 위치들을 참조 가능하다는 것이다.
이 원리는 seeuqnce-to-sequence 모델들에서의 전형적인 encoder-decoder attention 메커니즘을 흉내냈다.

2. encoder은 self-attention 층을 포함하는데, encoder의 이전 층의 output값으로부터 모든 키, 값, 쿼리값을 얻는다.
또, encoder은 모든 위치들을 서로 참조해낸다.

3. 비슷하게, decoder도 self-attention을 수행한다. 하지만, auto-regressive한 속성을 막기 위해 leftward 정보의 흐름은 막아야 한다. 즉, 지금의 위치가 다음에 나올 위치는 참조하지 못하도록 막아야 한다.
이를 위해, softmax의 모든 input값을 -무한대로 설정하여 masking해준다.

3.3 Position-wise Feed-Forward Networks

서브적인 층들까지 attention 과정을 적용하기 위해, 각 층들은 서로 완전연결된 feed-forward network 즉, FFN을 포함한다.
이는 각각의 위치에 동일하게 적용된다.
또, ReLu 활성화를 사이에 두고 2개의 선형 변환으로 구성된다.

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

* max()는 ReLu 함수로, 비선형성 학습을 하여 복잡한 패턴 학습도 가능하게 한다.
b1, b2는 편향값으로, 편향값을 더해줌으로서 학습이 유연하게 이루어지게 한다.
W1과 W2의 경우, 학습 가능한 가중치 행렬을 의미한다.

선형 변환은 동일하게 작용하지만, 층과 층으로부터 다른 종류의 파라미터들을 사용한다.

커널 사이즈가 1인 2개의 합성곱으로 또다르게 설명이 가능하다.
Input과 output의 차원성은 512이고, 내부층은 차원성이 2048이다.

3.4 Embeddings and Softmax

다른 시퀀스 변환 모델들처럼 input token과 output token값들을 dmodel 차원의 벡터들로의 변환 위해 embedding 기법을 사용한다.
또, decoder output값을 예측된 다음 token의 확률값으로의 변환을 위해서 선형 변환과 softmax function을 사용한다.

본 논문의 모델에서는 2개의 embedding 층과 pre-softmax 선형 변환 사이에 같은 가중치 행렬을 공유하게 하였고, embedding 층들에서 가중치들을 sqrt(dmodel)로 곱해주었다.

3.5 Positional Encodings

본 논문의 모델은 잔차, 합성곱 개념을 포함하지 않기 때문에, 모델이 sequence 순서대로 사용될 수 있도록 절대적, 상대적 토큰 위치에 대한 개념 적용이 필요하다.
이를 위해 encode와 decoder 스택의 아래 부분에 위치한 input embedding 과정에 "Positional encodings" 과정도 추가해주었다.
Positional emcodings와 embedding 과정은 동일한 차원을 가지기 때문에, 합해주는 연산이 가능하다.

해당 과정에서는, 다양한 frequency에 대해 sine 함수와 cosine 함수를 적용해주었다.

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

Positional encoding의 각 차원은 sine 곡선, 즉 정현파 형태이다. 2*pi 에서 10000*2*pi 까지의 범위의 기하급수를 형성한다.
본 모델이 상대적 위치를 쉽게 학습할 수 있도록 sinusoid 형태의 함수를 선택하였다.

바로 위의 이미지에서 볼 수 있는 2개의 버전이 거의 동일한 결과를 나타내지만, 사인 곡선 버전을 택하였다.

이는 모델이 training 과정에서 발견된 것보다 더 긴 길이의 sequence 길이를 추정할 수 있기 때문이다.

4. Why Self-Attention

이 섹션에서는, self-attention 층의 다양한 측면을 일반적으로 동일한 길이의 심볼 표현 시퀀스 ()를 ()와 같은 길이의 다른 시퀀스로 맵핑하는 데 사용되는 순환 및 합성곱 계층과 비교한다.
예를 들어, 일반적인 시퀀스 변환 encoder나 decoder의 은닉층과 같은 경우가 있다.
Self-attention을 사용하도록 하기 위해 3가지의 요구 사항을 고려한다.

1. 각 층의 총 계산 복잡도
2. 필요한 최소한의 순차적 작업들에 의해 수행되는 병렬화 가능한 계산량
3. 네트워크의 장거리 종속성 간의 경로 길이
이다.

장거리 종속성을 학습하는 것은 시퀀스 변환 작업에서 중요한 부분이다.
종속성을 학습하는 능력에 영향을 주는 하나의 주요한 요소는 네트워크에서 forward(순방향)과 backward(역방향) 신호가 통과해야 하는 경로의 길이이다.

네트워크에서 input 시퀀스와 output 시퀀스의 위치들 간의 길이가 짧을수록 장거리 종속성 학습이 더 쉽다.
또한, 서로 다른 계층 유형으로 구성된 네트워크에서 input과 output 위치 간의 최대 경로 길이를 비교한다.

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

Table 1에서 볼 수 있듯이 self-attention 층은 모든 위치들을 연결하는 데에 상수 개의 순차적 작업 과정이 필요하지만, recurrent 층의 경우 O(n)개의 순차적 작업 과정이 필요하다.
계산 복잡성의 관점에서, 시퀀스 길이 n이 representation 차원 d보다 더 작 attention 층들이 더 계산이 빠르다.
예를 들어, word-piece와 byte-pair 표현과 같이 기계 번역에서 최첨단 모델이 사용되는 경우가 많다.

매우 긴 문장들을 포함하는 작업들의 계산 성능을 향상시키기 위해, self-attention은 각 output 위치를 중심으로 한 입력 시퀀스에서 r 사이즈만 고려하도록 제한된다.
이를 통해, 최대 경로 길이를 O(n/r)로 증가시킬 수 있다.

이 접근법에 대해서는 추후에 더 연구하고자 한다.

커널 너비를 나타내는 k가 n보다 작은 경우 단일 컨볼루션 층은 input과 output의 모든 위치쌍들을 연결하지는 않는다.
그렇게 하기 위해서는 연속적인 커널의 경우 O(n/k) 합성곱 층의 스택이 필요하며, 확장된 합성곱의 경우 O(logk(n))가 필요하다.
이는 네트워크에서 두 위치 사이의 최대 경로의 길이를 증가시킨다.

컨볼루션 층들이 k라는 요소로 인해 일반적으로 recurrent 층보다 더 복잡하다.
하지만, 분리 가능한 컨볼루션의 경우 복잡도를 O(k*n*d + n*d^2)로 상당히 줄인다.
하지만, k=n 의 경우, 분리 가능한 컨볼루션의 복잡도는 self-attention 층과 point-wise feed-forward 층이 결합된 형태와 같다.

추가적인 장점으로, self-attention은 보다 해석 가능한 모델을 산출해낸다.
모델로부터 attention 분포를 파악하고 관련 예시들을 부록에서 다룬다.
단일 attention head들이 다른 작업들을 수행하기 위해 학습하고, 문장의 구문과 의미 구조와 관련된 행동을 보인다.

5. Training

이 섹션에서는 모델들의 훈련 체계를 설명한다.

5.1 Training Data and Batching

4백 50만 개의 문장쌍들로 구성된 standard WMT 2014 English-German d
훈련했다.
문장들은 byte-pair encoding을 활용하여 인코딩되었고, 약 37000 개의 토큰들로
source-target 하였다.

3천 6백 만 개의 문장들로 구성된 상당히 큰 WMT 2014 English-French dataset을 사용하여 특징들을 32000개의 word-piece 어휘로 나누었다.

문장쌍들은 대략적인 분량 길이에 의해 함께 배치되었다.
각각의 훈련 배치는 약 25000개의 특징과 2500개의 타겟 특징들을 포함하는 문장쌍들로 구성된다.

5.2 Hardware and Schedule

8 NVIDIA P100 GPU를 가진 하나의 기계로 훈련을 진행하였다.
각 훈련 단계들은 0.4초가 소요되었고, base model을 총 10000개의 단계로 구성되며 총 12시간 동안 훈련시켰다.

더 큰 모델들에 대해서는 각 단계들이 1초가 소요되었고, 총 30만 개의 단계들을 3.5일 동안 훈련시켰다.

5.3 Optimizer

$$b1 = 0.9, b2 = 0.98, 시그마 = 10^(-9)$$

로 설정한 Adam optimizer을 사용하였다.
바로 아래의 공식에 따라 훈련 과정동안 학습률을 변화시켰다.

$$lrate = d_{\text{model}}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5})$$

첫 warmup-steps 훈련 단계에서 학습률을 선형적으로 증가시키고, 그 후에는 단계수의 역제곱근에 비례하여 줄인다.
본 논문은 warmup-steps 값으로는 4000으로 설정했다.

5.4 Regularization

훈련 과정 동안 3개의 정규화 방법을 고안했다.

1, 2. Residual dropout
서브층의 input값에 추가되고 일반화되기 전에 각 서브층의 output값에 dropout을 적용한다.
게다가, dropout을 encoder와 decoder에 embedding과 위치 인코딩에도 적용하였다.
base model에 Pdrop값으로 0.1을 사용한다.

3. Label Smoothing
훈련 과정 동안, 시그마 값으로 0.1을 적용하여 label smoothing과정을 진행하였다.
이는 모델이 더 확실하지 않은 방법을 배우기 때문에 당황할 수 있으나, 정확도와 BLEU 점수를 향상시킨다.

6. Results

6.1 Machine Translation

WMT 2014 English-to-German 번역 작업에서, 큰 Transformer 모델의 경우 새로운 최고 BLEU 점수로 28.4를 달성하여 기존의 최고 수준의 모델들을 능가하였다.
이 모델의 배열은 아래의 Table 3에 제시되었다.

Table 3: Variations on the Transformer architecture. Unlisted values are identical to those of the base model. All metrics are on the English-to-German translation development set, newstest2013. Listed perplexities are per-wordpiece, according to our byte-pair encoding, and should not be compared to per-word perplexities.

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	positional embedding instead of sinusoids									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

Table 3

훈련 과정은 8 P100 GPU에서 3.5일이 걸렸다.
base 모델은 training cost 관점에서 모든 이전의 모델들, 앙상블 모델들을 능가했다.

WMT 2014 English-to-French 번역 작업에서, 본 모델은 41.0 BLEU 점수로 이전의 단일 모델들을 능가했고, 이전의 최고 수준의 모델의 training cost의 1/4보다도 training cost 값이 작았다.
Dropout 비율로 Pdrop값으로 0.1을 대신 사용하였다.

base 모델들에 대해 5개의 체크포인트들을 평균낸 값으로부터 얻어진 단일 모델을 10분 간격으로 적용시켰다.
큰 모델들에 대해서는 20개의 체크포인트들을 평균화하였고, beam size로 4, length penalty값으로 0.6을 사용하였다.

이러한 하이퍼파라미터들은 development set 실험 후에 선택되었다.
최대 출력 길이를 입력 길이에 50이 더해진 형태, 입력 길이 + 50으로 설정하고, 가능한 경우 조기 종료시켰다.

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

Table 2

바로 위의 Table 2 이미지값이 본 실험의 결과를 요약하고, 다른 모델들과 번역의 질과 train cost를 비교한다.
모델 훈련에 사용된 부동 소수점 연산수를 훈련시간, GPU수, 각 GPU의 지속적인 단일 정밀 부동 소수점 용량 추정치를 곱하여 추정한다.

6.2 Model Variations

Transformer 모델의 다른 구성 요소들의 중요성을 평가하기 위해, base model을 다른 형태로 변형하여, development set인 Newstest2013에서 English-German 번역의 성능의 변화를 측정하였다.
바로 이전의 섹션에서 설명한 것처럼, beam search를 사용하지만, checkpoint들을 평균화하는 과정은 사용하지 않았다.

Table 3: Variations on the Transformer architecture. Unlisted values are identical to those of the base model. All metrics are on the English-to-German translation development set, newstest2013. Listed perplexities are per-wordpiece, according to our byte-pair encoding, and should not be compared to per-word perplexities.

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	param $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	positional embedding instead of sinusoids									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

Table 3

Table 3의 A행에서, attention hear와 attention key값, value 차원 수에 변화를 주고, 3.2.2 섹션에서 설명한 것처럼 계산값을 일정하게 유지시켰다.최고 수준으로 세팅되었을 때보다 단일 head attention이 0.9 BLEU값이 낮을 때, 너무 많은 head들에서 성능이 드롭아웃되었음을 확인할 수 있었다.

Table 3의 B행에서는 attention key의 size dk값을 줄였고, 이로 인해 모델의 성능이 줄었다는 것을 확인할 수 있었다.

Table 3의 C행과 D행에서는 모델이 클수록 더 성능이 좋고 dropout이 과적합을 방지하는 데에 매우 유용하다는 것을 알 수 가 있다.

E행에서는, 사인파 위치를 학습된 위치 embedding값으로 대체하고, base model과 거의 동일한 결과를 낸다는 것을 확인할 수 있었다.

7. Conclusion

이 작업에서, 본 논문은 encoder-decoder 아키텍처에서 가장 일반적으로 사용되는 반복되는 층을 multi-headed self-attention으로 대체하여 완전히 attention을 기반으로 한 최초의 시퀀스 변환 모델인 Transformer을 제시했다.

번역 작업에서, Transformer 모델은 반복, 컨볼루션 층에 기반한 아키텍처
히 훈련이 빠르다.
WMT 2014 English-to-German과 WMT 2014 English-to-French 번역 작업 모두에서, 새로운 최첨단 기술 수준에 도달했다.
이전의 작업에서 Transformer 모델은 모든 이전의 앙상블 모델들을 능가했다.

Attention 기반의 모델들의 미래와 이 모델들을 다른 작업들에 적용하는 계획에 관심이 있다.
Transformer 모델을 텍스트 이외에도 input-output 방식과 이미지, 오디오 및 비디오와 같은 대규모 input과 output을 효율적으로 처리하기 위한 메커니즘 문제들로 확장하고자 한다.
학습을 덜 순차적으로 만드는 것도 또다른 목표이다.

논문 리뷰를 통해 배운 점

1. 그동안 ResNet, CNN 모델에 AI 모델 지식이 국한된 느낌이었는데, 새롭게 Transformer 모델에 대해 알아볼 수 있는 좋은 계기가 되었다.

2. 일반적으로 모델들이 순환 구조를 채택하는 반면에, Transformer 모델의 경우 순환 구조를 완전히 배제하고 오직 attention 메커니즘을 활용하여 input과 output 간의 전역 종속성을 학습하는 방식을 활용한다는 것이 가장 큰 차이이다.

3. Transformer 모델이 기존의 모델들의 성능을 능가하는 모델이라는 것을 다양한 수치값으로 확인할 수 있었다.

4. 일반적으로는 신호끼리 연결할 때 필요한 연산의 수가 위치들 간의 거리에 따라 증가하는 데 반해, Transformer 모델이 신호를 연결할 때에는 연산의 수가 상수값의 형태로 그 수가 줄어든다는 장점이 있다.

논문 리뷰하며 생긴 의문

1. Transformer이 Label smoothing 과정을 진행하는데, 더 확실하지 않은 방법으로 학습한다는 것이 무슨 의미일까?

A. Label smoothing은 모델이 예측할 때 확신을 덜 가지도록 하는 기법이다.
주로 분류 문제에서 사용되며, 모델이 특정 클래스에 너무 확신을 가지는 것을 방지한다.
이를 통해 모델이 과적합(overfitting)을 방지하고, 일반화 능력을 향상시킬 수 있게 한다.

예를 들어, 어떤 샘플의 진짜 레이블이 클래스 1이라면, 그 레이블에 0.9를 할당하고, 다른 클래스들에는 0.1을 균등하게 나누는 것이다.
이렇게 하면 모델이 예측할 때 "클래스 1"에 너무 집중하지 않고, 다른 클래스에 대해서도 약간의 확신을 가지게 된다.

모델이 레이블에 대해 너무 확신하지 않게 되므로, 일종의 불확실성을 배우게 되고, 이로 인해 모델이 다양한 가능성을 고려하며 예측을 한다.
처음에는 이러한 방식이 직관적이지 않아 모델의 성능이 떨어지는 것처럼 보일 수 있지만,
실제로는 과적합을 방지하고 일반화 능력을 높여서 최종적인 정확도나 BLEU 점수 등 성능 지표가 향상된다.

2. Table 3 에서 Transformer 모델의 성능이 가장 좋다는 건 확인할 수 있는데, Transformer모델의 파라미터 수가 다른 모델들보다 지나치게 높은데 그러면 계산 시간이나 계산량이 너무 많아져서 성능이 떨어질 수 있는 거 아닐까?

A. 모델의 파라미터 수가 많다고 해서 무조건 성능이 높아지는 것은 아니다. 하지만, 일반적으로 더 좋은 성능을 낼 가능성 이 높다.

첫 번째 이유는 모델의 표현력이 증가하기 때문이다.
신경망 모델에서 파라미터 수가 많아지면 더 복잡한 패턴을 학습할 수 있다고 한다.
Transformer에서 하이퍼파라미터를 증가시키면 모델이 더 정교한 문맥적 정보를 학습할 수 있을 것이다.

두 번째 이유는, 파라미터가 많은 모델이 더 큰 데이터를 학습하면 일반적으로 BLEU 점수가 상승하고, PPL이 감소하는 경향이 있기 때문이다.
이는 실제로 Table3에서 확인할 수 있다.

하지만, 실제로 파라미터의 수가 많아지면 연산량이 증가해서 훈련 시간도 오래 걸리고, 메모리 사용량이 증가하여 더 많은 GPU 자원이 필요하다.
또, 적은 양의 데이터셋에서 너무 큰 모델을 사용할 경우 훈련 데이터에 과도 있으므로 큰 모델에서는 충분히 많은 양의 훈련 데이터가 필요하다.

큰 모델은 일반적으로 더 높은 표현력을 가지므로 BLEU가 올라가고 PPL이 낮아질 가능성이 높다는 장점이 있으나,
파라미터의 수가 증가하면서 계산 비용, 메모리 사용량, 데이터 크기 등을 고려해야 한다.

'카테고리 없음'의 다른글

이전글 [Euron] 1week_Deep Residual Learning for Image Recognition

현재글 : [Euron] 2week_Attention Is All You Need