Transformer (attention is all you need) 리뷰 – 이우진, 김유진, 이연호

https://arxiv.org/pdf/1703.06870

Abstract

- 기존의 시퀀스 변환(sequence transduction) 모델은 인코더와 디코더로 구성된 복잡한 순환 신경망(RNN) 또는 합성곱 신경망(CNN)에 기반
    - 이때, 인코더와 디코더를 연결하기 위해 어텐션 메커니즘을 추가로 활용
- RNN이나 CNN 없이 어텐션 메커니즘만으로 이루어진 새로운 네트워크 아키텍처 **Transformer** 제안
- Transformer 모델은 기존 모델보다 더 높은 품질을 보여주며, 병렬 처리에도 더 효율적이고 학습 시간도 단축

1. Introduction

- 순환 신경망(Recurrent Neural Networks, RNN), 장단기 메모리(Long Short-Term Memory, LSTM), 게이트 순환 신경망(Gated Recurrent Neural Networks, GRU) 등은 시퀀스 모델링과 변환(sequence transduction) 문제에서 최고 성능(state-of-the-art)을 보임.
    - ex) 언어 모델링과 기계 번역 분야
- 순환 신경망 모델은 일반적으로 입력 및 출력 시퀀스의 각 심볼 위치에 따라 연산을 수행
    - 이전의 은닉 상태 h_{t-1}와 현재 위치의 입력을 기반으로 새로운 은닉 상태 h_t를 생성
    - **본질적으로 순차적인 특성**은 학습 시 병렬 처리가 어려움.
- 어텐션 메커니즘(attention mechanism)은 다양한 시퀀스 모델링 및 변환 작업에서 중요한 요소
    - 입력과 출력 시퀀스 내의 거리와 상관없이 의존성을 효과적으로 모델링
- 이 논문에서는 Transformer라는 새로운 모델 아키텍처를 제안
    - **순환 구조를 완전히 배제**하고, 입력과 출력 간의 전역적 의존성(global dependencies)을 모델링하기 위해 **오직 어텐션 메커니즘**에만 의존

- 높은 수준의 병렬 처리가 가능, 8개의 P100 GPU에서 12시간의 학습만으로도 기계 번역 분야의 최고 성능을 달성

2. Background

- 순차적 계산을 줄이려는 목표는 **Extended Neural GPU**, **ByteNet**, **ConvS2S**와 같은 모델의 기반
    - 합성곱 신경망(Convolutional Neural Networks, CNN)을 기본 구성 요소로 사용
    - 입력과 출력의 모든 위치에 대해 병렬로 은닉 표현(hidden representations)을 계산
    - but, **임의의 두 입력 또는 출력 위치 간의 신호를 연결하기 위해 필요한 연산의 수가 위치 간 거리**에 따라 증가 → 멀리 떨어진 위치 간의 의존성(dependency)을 학습하는 것이 어렵다.
        - **ConvS2S**의 경우 선형(linear)적으로 증가하고, **ByteNet**의 경우 로그(logarithmic) 형태로 증가
    - **Transformer**에서는 이러한 연산을 **상수(constant) 수준으로 감소**
        - 어텐션 가중치(attention-weighted)를 평균화하기 때문에, effective resolution이 줄어드는 단점이 있지만, 이를 보완하기 위해 **다중 헤드 어텐션(Multi-Head Attention)** 기법을 적용
- **Self-Attention**
    - **Self-attention (intra-attention)** 메커니즘은, **하나의 시퀀스 내의 서로 다른 위치 간의 관계**를 학습하여 그 시퀀스의 표현을 계산
        - 독해(Reading Comprehension), 추상적 요약(Abstractive Summarization), 텍스트 함의(Textual Entailment), 작업 독립적인 문장 표현 학습(Task-Independent Sentence Representations) 등의 작업에서 성공적으로 사용
- End-to-End Memory Networks
    - 이 네트워크는 시퀀스 정렬 기반 순환 구조 대신 **순환 어텐션 메커니즘(Recurrent Attention Mechanism)**에 기반 ← **어텐션 메커니즘을 반복적으로 적용**한다는 의미에서 recurrent 용어 사용

        - 간단한 언어 질문 응답(Simple-Language Question Answering) 및 언어 모델링(Language Modeling) 작업에서 좋은 성능

- Transformer의 차별점
    - **Transformer**는 입력과 출력의 표현을 계산하기 위해 **시퀀스 정렬 기반의 RNN이나 CNN 없이, 오직 Self-Attention만**을 사용하는 **최초의 변환 (transduction) 모델**

3. Model Architecture

- 대부분의 신경망 시퀀스 변환 모델은 **인코더-디코더 구조**를 가지고 있음.
    - 인코더(encoder)는 입력 시퀀스 (x1, ..., xn)을 받아 이를 연속적인 표현 (sequence of continuous representations)인 z = (z1, ..., zn)으로 매핑
    - 디코더(decoder)는 이 z를 기반으로 출력 시퀀스(**output sequence)** (y1, ..., ym)를 한 번에 한 요소씩 생성
    - 모델은 **auto-regressive** 방식으로 작동, 이는 매 스텝마다 **이전에 생성한 출력 값**을 **추가 입력**으로 사용하여 다음 출력을 생성하는 방식
- Transformer는 이러한 전반적인 구조를 따르면서도, **인코더와 디코더** 모두에 **stacked Self-Attention**과 **Point-wise Fully Connected Layers를** 적용

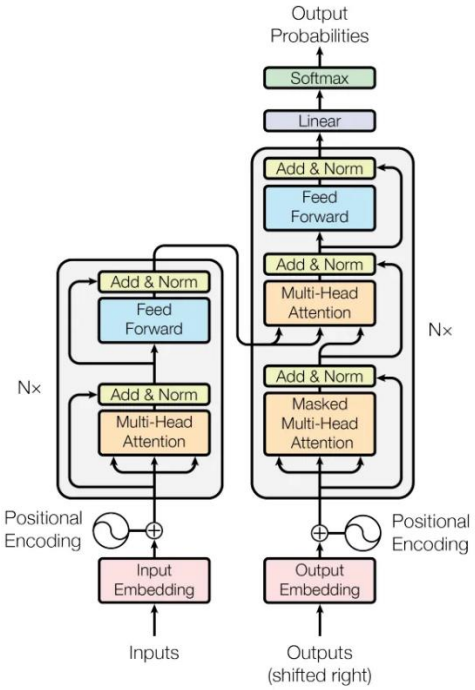


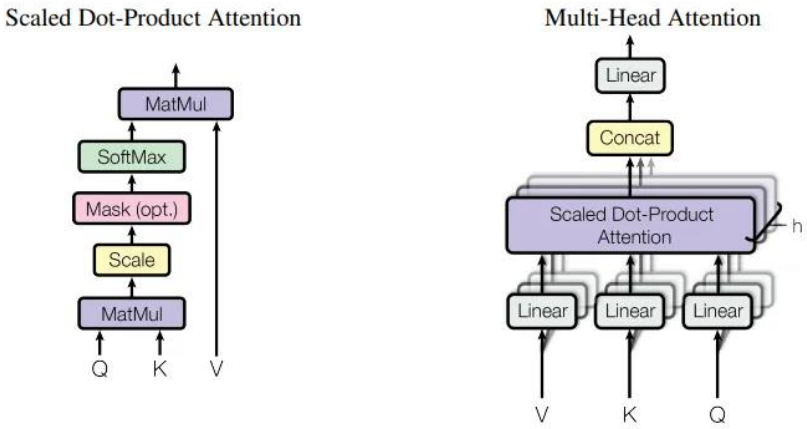
Figure 1: The Transformer - model architecture.

3.1 Encoder and Decoder Stacks

- 인코더(Encoder)
    - 동일한 6개의 레이어(N = 6)로 구성된 스택 구조, 각 레이어는 두 개의 서브 레이어(sub-layer)로 이루어짐.
        - **Multi-Head Self-Attention 메커니즘**
        - **Position-wise Fully Connected Feed-Forward Network**
    - 각 서브 레이어에는 **잔차 연결(Residual Connection)**이 적용되며, 그 뒤에 **레이어 정규화(Layer Normalization)**가 수행
        - Output=LayerNorm(x+Sublayer(x)) ← Sublayer(x)는 해당 서브 레이어가 수행하는 함수
        - 잔차 연결을 효과적으로 적용하기 위해, **모든 서브 레이어와 임베딩 레이어**의 출력 차원은 dmodel=512로 동일하게 유지
- 디코더(Decoder)
    - 디코더 역시 6개의 동일한 레이어(N = 6)로 구성
    - 인코더 레이어의 두 개 서브 레이어에 **추가적으로 세 번째 서브 레이어**가 삽입
        - **Multi-Head Self-Attention**
        - **Position-wise Feed-Forward Network**
        - **Multi-Head Attention over Encoder Output**
    - 인코더와 마찬가지로 **각 서브 레이어에 잔차 연결과 레이어 정규화**가 적용
    - 디코더의 셀프 어텐션 서브 레이어에는 **Masking** 기법이 추가 → 이는 현재 위치(i)가 미래 위치(i+1, i+2, ...)의 정보를 참고하지 못하도록 함.
    - **출력 임베딩은 한 position씩 offset**되어 있어, 위치의 예측은 이전 위치(< i)의 출력에만 의존
        - 따라서, **auto-regressive** 모델의 특성을 유지하면서 **정확한 순차적 예측**을 보장

3.2 Attention

- 어텐션 함수는 Query와 Key-Value pairs를 Output으로 매핑하는 방식으로 정의
    - **쿼리(Query)**, **키(Key)**, **값(Value)**, 그리고 출력(Output)은 모두 **벡터(Vector)** 형태
- 출력은 Value들의 weighted sum으로 계산
    - 각 값에 부여되는 Weight는 해당 값의 Key와 Query 간의 호환성 함수(Compatibility Function)를 통해 결정



3.2.1 Scaled Dot-Product Attention

- 사용하는 어텐션 방식: **Scaled Dot-Product Attention**
- 입력 구성
    - Query와 Key의 차원은 d_k, Value의 차원은 d_v
- 계산 과정
    - Query와 모든 Key의 내적(**Dot Product**)을 구함
    - 이 값을 $₩sqrt{d_k}$로 나누어 **Scaling**
    - **소프트맥스(Softmax)** 함수를 적용해 Value에 대한 Weight를 계산

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

- 어텐션 종류 비교
    - **Additive Attention**: 간단한 피드포워드 신경망을 통해 compatibility 함수를 계산

    - **Dot-Product Attention**: 단순한 내적만으로 compatibility 계산
        - **Dot-Product Attention**이 **속도와 메모리 효율성** 면에서 우수
        - 단, d_k가 큰 경우에는 Additive Attention 더 성능이 좋을 수 있음

- **Scaling**의 필요성
    - d_k가 클수록 내적 값이 매우 커지면서 **소프트맥스의 그래디언트가 매우 작아지는 문제** 발생
    - 이를 해결하기 위해 내적 값을 $₩sqrt{d_k}$로 나눠 **안정성** 확보

3.2.2 Multi-Head Attention

- **single attention function**을 d_{model} 차원의 Query, Key, Value에 적용하는 대신, Query, Key, Value를 h번 선형 변환(linear projection)하여 각각d_k, d_k, d_v 차원으로 매핑하는 것이 더 효과적
    - 변환된 Query, Key, Value에 대해 병렬(parallel)로 어텐션 함수 적용
    - 각 어텐션 결과는 d_v 차원의 출력 값을 생성, 이들을 결합(concatenate)한 뒤 다시 한 번 선형 변환하여 최종 출력을 만듦.
- **Multi-Head Attention의 이점**
    - 다양한 representation subspace에 동시에 집중
    - single head에서는 모든 정보를 평균 내기 때문에, 다양한 정보에 고르게 집중하기 어렵다.

    - $W_i^Q∈₩mathbb{R}^{d_{model} ₩times d_k}$: Query projection matrix
    - $W_i^K∈₩mathbb{R}^{d_{model} ₩times d_k}$: Key projection matrix
    - $W_i^V∈₩mathbb{R}^{d_{model} ₩times d_v}$: Value projection matrix
    - $W^O∈₩mathbb{R}^{hd_v ₩times d_{model}}$: Output matrix
- **구현**
    - **헤드 수(h)**: 8개

- 각 헤드에 대해 $d_k = d_v = ₩frac{d_{model}}{h} = 64$
- 즉, **512차원**을 **8개의 head**로 나누어 **각 head는 64차원**만 처리
- 각 헤드의 차원이 작아지므로, **전체 연산 비용은 single attention head와 비슷**

3.2.3 Applications of Attention in our Model

Transformer가 multi-head attention을 사용하는 방식은 세 가지로 나뉨.

- Key Value는 encoder의 출력에서 나오며 Query는 이전 decoder 층에서 나와 decoder가 모든 입력 시퀀스의 위치에 적용될 수 있게 함.
- Encoder는 self-attention 층을 지니며 이전 encoder 층의 출력에서 모든 Query Keys Values가 나오게 됨. 또한 Encoder의 각 위치는 이전 encoder 층의 모든 위치를 참고함.
- 이는 decoder도 마찬가지이며 auto-regressive 속성을 지키기 위해 scaled dot-production attention의 softmax에 masking(-∞)하는 방식을 택함. *softmax 거치면 0이 되어 미래 정보 확인 불가

3.3 Position-wise Feed-Forward Networks

각 sub-layer는 fully connected feed-forward network를 가짐. 2개의 linear 층과 그 사이에 ReLU 활성화 함수를 사용한 형태로 각 위치에 동일하게 적용. 입출력(d_model)은 512차원으로 내부 층은 2048 차원임.

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

3.4 Embeddings and Softmax

입출력의 변환을 위해 학습된 embedding을 사용, decoder 출력에 적용한 선형변환은 같은 가중치를 사용. 또한 embedding 층에서는 가중치에 sqrt(d_model)만큼을 곱하여 사용.

3.5 Positional Encoding

encoder와 decoder의 input embedding 아래에 위치 정보를 추가함 (동일한 차원). 위치 정보의 인코딩 함수는 다음과 같음. *pos는 위치, i는 차원

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

4 Why Self-Attention?

- 연산 복잡도가 낮으며 연산의 병렬 처리가 가능함.
- 장거리 의존성(long-range dependencies)에 효과적. 앞뒤로 signal을 주고 받을 때의 거리가 다른 모델(Recurrent, Convolutional)보다 짧음.
- 문장의 의미와 문법적 구조와 관련하여 뛰어난 결과를 보이며, 더욱 해석 가능한 모델을 만들어 냄.

5. Training

5.1 Training Data and Batching

1. English-to-German : 450만개의 문장 쌍으로 구성돼 있는 WMT 2014 English-German dataset 과 37000개의 어휘 토큰 사용
2. English-to-French : 3600만개의 문장 쌍으로 구성돼 있는 WMT 2014 English-French dataset과 32000개의 어휘 토큰 사용

sequence 길이가 비슷한 문장 쌍은 같은 batch로 학습을 진행하였다.

5.2 Hardware and Schedule

8개의 NVIDIA P100 GPU 사용

5.3 Optimizer

Adam optimizer를 사용 (β1 = 0.9, β2 = 0.98, ϵ = 10−9)

learning rate 은 아래와 같이 계산하였다.

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

이는 warmup_steps 까지는 선형적으로 증가하고, 이후 step_num의 역제곱근에 비례하여 감소한다.

warmup_steps는 4000으로 설정하였다.

5.4 Regularization

Residual Dropout : $P_{dropout}=0.1$ 으로 설정하였고, dropout은 각 sub layer의 output에 적용되었으며, embedding과 positional encoding 을 더한 후에도 적용되었다.

Label Smoothing : $₩epsilon_{ls}=0.1$으로 설정하였고, 이는 perplexity는 높아지게 하지만 정확도와 BLEU score를 향상시킨다.

6. Results

6.1 Machine Translation

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-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	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**	**41.8**	$2.3 \cdot 10^{19}$	

WMT 2014 English-to-German translation task : 기존 SOTA 모델의 BLEU score 보다 2.0 높은 28.4 달성

WMT 2014 English-to-French translation task : 기존 SOTA 모델 훈련 비용의 1/4 로 BLEU score 41.8 달성

6.2 Model Variations

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)	params $\times 10^6$
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

(A) : attention head 수인 h를 다르게 하여 측정하였다. single head 일 때가 $h=16$ 보다 BLEU score가 0.9 낮다. 또한, head 수가 너무 많아지면 성능이 떨어지는 것을 확인할 수 있다.

(B) : attention key size인 d_{k}가 줄어들면 성능이 떨어진다.

(C) : 큰 모델이 좋은 성능을 보인다.

(D) : dropout이 과적합을 방지하는데 매우 좋다.

(E) : sinusoidal positional encoding 대신 positional embedding 을 사용해도 거의 비슷한 성능을 보인다.

6.3 English Constituency Parsing

4-layer Transformer를 사용하였으며, $d_{model}=1024$로 English consituency parsing task를 진행

1. Wall Street Journal (WSJ) portion of the Penn Treebank의 40K개 의 훈련 문장과 16K개의 토큰 사용

2. BerkleyParser corpora의 17M개의 훈련 문장과 32K개의 토큰 사용

Table 4: The Transformer generalizes well to English constituency parsing (Results are on Section 23 of WSJ)

Parser	Training	WSJ 23 F1
Vinyals & Kaiser el al. (2014) [37]	WSJ only, discriminative	88.3
Petrov et al. (2006) [29]	WSJ only, discriminative	90.4
Zhu et al. (2013) [40]	WSJ only, discriminative	90.4
Dyer et al. (2016) [8]	WSJ only, discriminative	91.7
Transformer (4 layers)	WSJ only, discriminative	91.3
Zhu et al. (2013) [40]	semi-supervised	91.3
Huang & Harper (2009) [14]	semi-supervised	91.3
McClosky et al. (2006) [26]	semi-supervised	92.1
Vinyals & Kaiser el al. (2014) [37]	semi-supervised	92.1
Transformer (4 layers)	semi-supervised	92.7
Luong et al. (2015) [23]	multi-task	93.0
Dyer et al. (2016) [8]	generative	93.3

tuning이 부족했음에도 transformer가 Recurrent Neural Network Grammar 을 제외한 이전 모델들과 비교해 가장 좋은 성능을 보임을 알 수 있다.

7. Conclusion

본 논문에서는 attention 을 기반으로 힌 최초의 sequence transduction 모델인 Transfomer를 소개한다. 기존 인코더-디코더 구조에서 recurrent layers를 multi-headed self-attention으로 대체하였다. Translation task 에서 Transformer가 기존 recurrent/convolution 모델보다 유의미하게 빨라 새로운 SOTA 성능을 보였다. Transformer를 다양한 task에 적용할 수 있음을 기대하고, 텍스트 뿐만 아닌 이미지, 오디오, 비디오 등의 대규모 입출력에서도 활용할 계획이다.