👀

Attention is All You Need

Abstract & Introduction

RNN, LSTM, Gated Recurrent Neural Network 는 기존 state-of-the-art. 하지만 sequential 하게 처리하기 대문에 병렬화가 어려움 + batch 크기를 메모리 제한으로 인해 증가시키는 것이 어려움. 본 논문에서는 input, output sequence의 위치와 무관하게 long-range 의존성을 모델링할 수록 해주는 Attention Mechanism만을 사용하여 recurrent, convolutional 연산을 완전히 제거한 Transformer 모델 제안

기존의 순차적 연산을 줄이는 ByteNet, ConvS2S는 CNN을 사용하여 입출력 모든 sequence의 위치에 대해 hidden representation을 병렬로 연산함. 하지만 이런 모델은 임의의 입출력 위치 간의 신호 관계를 모델링하는데 **필요한 연산량이 두 위치간 거리에 의해 증가함** → **각 입력단어가 출력 단어에 영향을 주기 위해선 여러 layer를 거쳐야 함. 이때 CNN의 kernel size가 k 라면, 한 layer를 지나면서 최대 k 개의 단어만 연결 가능. 두 위치 거리가 멀어질 수록 더 많은 layer를 통과해야 함**→ 멀리 떨어진 위치간 dependency를 학습하는 것을 어렵게 만듦.

Transformer에서는 이를 constanct 하게 줄였으나 **어텐션 가중치 averaging으로 인해 정보의 해상도가 감소할 수 있다는 문제 → weigted sum을 결합하기 때문에 개별 단어의 고유한 정보가 사라질 수 있는 문제. 따라서 multi-head로 다양한 어텐션 패턴을 학습** 를 감안하여, multi-head attetion을 도입함

Self-Attention은 단일 시퀀스 내 서로 다른 위치를 연결하여 해당 시퀀스의 representation을 계산하는 매커니즘. Transformer는 **end-to-end memory network → 외부 메모리를 사용하여 특정 단어를 저장하고 이를 활용하여 학습(Transformer에선 입력을 메모리에 저장하고, attention으로 중요한 정보를 검색해서 응답을 생성하는 방식)**는 기존 RNN 기반 시퀀스 정렬 대신 recurrent attention 매커니즘으로 작동하여 입출력 표현 계산하는 **transduction 모델 → 입력시퀀스를 출력 시퀀스로 변환하는 모델을 의미** 임.

Method

sequence transduction (변환) 모델은 encoder-decoder 구조를 따름. 이때 encoder는 입력 시퀀스 x1,..xn으로 부터 연속된 표현 시퀀스 z1,...,zn로 변환. 이후 decoder는 z를

기반으로 출력 시퀀스 y1,..,yn을 한 요소로 생성함. 각 단계에서 모델은 auto-regressive 한 방식으로 다음 출력을 생성할 때 이전에 생성된 symbol을 추가 입력으로 사용함.

Transformer 모델은 이 전체적인 구조를 유지하면서 encoder, decoder 모두에서 누적된 self-attention과 point-wise fully connected layer를 사용

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

시간복잡도 비교

[self-attention vs RNN] 기존의 recurrenr layer 가 시퀀스 변환을 통해 auto-regressive 하게 작동할 수 있는 RNN 은 O(n)의 sequential 연산이 필요함. 하지만 self-attetion은 모든 위치를 연결하는 연산을 constant number내에서 수행 가능함. 또한, sequence 길이 n 이 d 보다 작을 때 recurrent layer 보다 계산 복잡도가 낮아짐.

[self-attention vs CNN] CNN은 kernel size k 가 n 보다 작은 경우, 여러개의 convolutional layer를 쌓아야 하는데, 이때 필요한 layer 의 수는 O(n/k)로 완전 연결이 되기 위해서는 여러개의 계층이 필요함

따라서 self-attention은 RNN, CNN 보다 짧은 경로로 긴 거리 의존성을 학습할 수 있고, RNN과 비교했을 때 병렬화가 쉽고 CNN과 비교했을 때 더 적은 layer를 사용하여 입출력 위치간 연결이 됨

추가적으로 모델이 더 해석 가능함. 개별 attetion head가 각각 다른 역할을 하며, 문장의 구문적, 의미적 구조와 관련된 행동을 보이기 때문

Attetion

Attention Function은 Query, Keys, Values의 집합을 입력받아 output (이때 Q, K, V, output은 모두 vector)을 생성하는 함수로 output은 value의 가중합으로 계산됨. 각 가중치는 Q, V 의 유사도를 나타내는 compatibility function으로 결정됨

일반적으로 사용되는 attention function 는 Additive Attetion과 Dot-product Attetion로 Additive Attention은 feed-forward network와 single network layer로 유사도를 계산, 하지만 **Dot-product Attention 에 비해 메모리 비효율** **→ Additive Attention은 Q, K를 각각 FFN에 통과시켜 비선형 연산이 추가됨, 하지만 Dot-product는 행렬 곱셈 연산으로 GPU에서 최적화+ 단순 내적 연산이므로 O(1)로 행렬 곱셈 하나로 바로 어텐션 가중치를 계산 가능함** , 계산복잡도는 유사

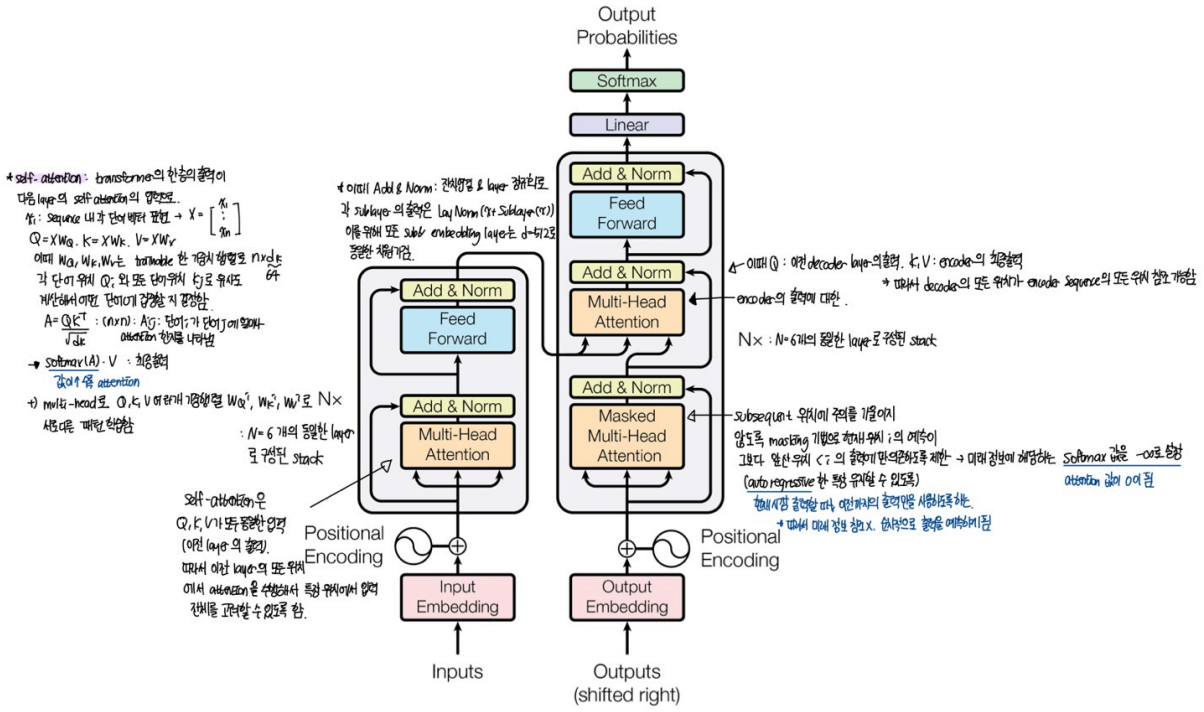

Figure 1: The Transformer - model architecture.

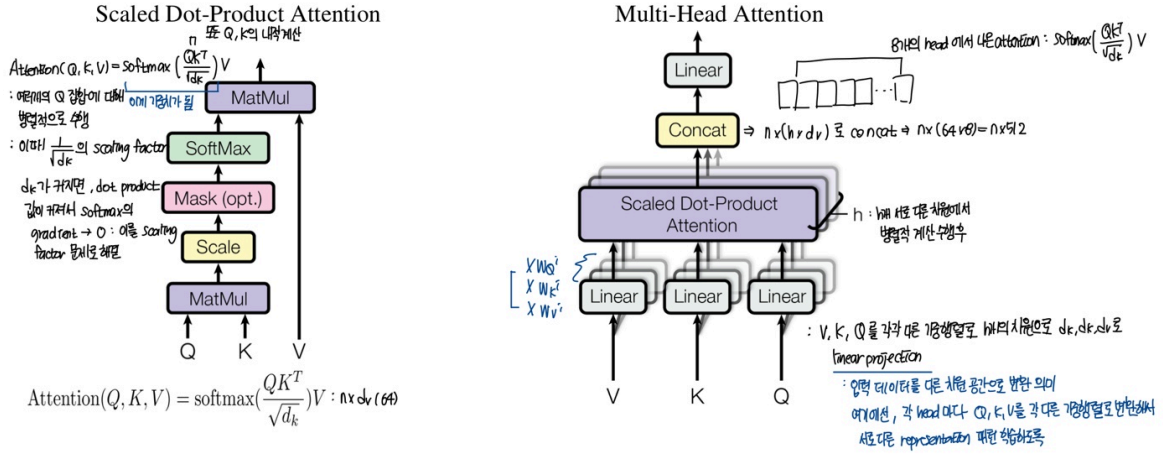

Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

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

FFN은 위치별로 독립적으로 적용되는 두개의 선형변환과 ReLU 활성화 함수로 구성됨

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

Embedding

입력 토큰과 출력 토큰을 d model(논문에선 512) 차원의 벡터로 변환하는 학습된 임베딩을 사용함

decoder의 최종출력을 softmax로 변환해서 다음 토큰의 확률 분포를 예측함. 이때 입출력 임베딩과 softmax 직전 linear projection에서 동일한 가중치 행렬을 공유하고 임베딩 가중치 sqrt(d model)을 곱해줌

Positional Encoding

Transformer는 RNN, CNN을 사용하지 않기 때문에 시퀀스 내 순서 정보를 직접 학습할 방법이 필요함

따라서 입력 embedding에서 positional encoding을 추가하여 순서 정보를 제공함. positional encoding의 차원은 입력 embedding과 동일하므로, encoder, decoder stack의 bottom에서 입력 embedding과 더해지는 방식

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

본 논문에서는 고정된 방식의 positional encoding을 사용함. bc 특정 거리를 유지하는 벡터가 선형변환으로 쉽게 계산될 수 있기 때문 + learned 방식과 동일한 성능을 보임. 이때 사인/코사인 방식이 학습되지 않은 길이의 시퀀스에도 일반화 가능성이 높다고 판단

Experiments

450만개의 setencepairs로 구성된 WMT 2014 English-German 데이터셋 사용. byte-pair-encoding으로 인코딩 후 영어, 독일어가 공유하는 vocab size는 37000개 token이 됨.

문장쌍은 길이가 유사한 것끼리 batch로 학습을 진행. train batch에선 25000 source token과 25000개 target token이 포함됨.

Adam optimizer를 사용 이때 learning rate는 10^(-9)로 공식 따라 조정되도록 (초반 4000 step에서는 선형적으로 증가시키고, 이후에는 step의 역제곱근 로 감소시킴)

P(dropout)=0.1 로 encoder-decoder에서 임베딩과 postional encoding sum, sub-layer 출력 후 layer noramlization 전에 residual dropout 적용

train 때, label smoothing 0.1 적용

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**	$2.3 \cdot 10^{19}$	

모든 개별&앙상블 모델 뛰어넘음, 훈련 비용 상대적 저렴

추가적인 실험으로, 마지막 5개의 checkpoint를 평균내어 최종 모델 생성 (big model의 경우 20개의 마지막 checkpoint를 평균냄)

	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

Transformer의 각 구성 요소가 성능에 미치는 영향을 평가

(A) : Attention head 수 + Key/Value Dimension 변화 → head 수가 너무 많으면 성능 저하

(B) : Attetion K dk 변화 → 너무 줄여도 성능 저하 (유사도 계산이 쉽지 않기 때문. 어느 정도 정교한 함수가 필요)

(C), (D) : dropout을 적용한 것이 overfitting 방지에 효과적임이 입증

(E), (F) : positional embedding을 학습 가능한 위치 임베딩으로 대체했을 때, base model과 비슷한 성능을 보임

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

영어 구성 구문 분석 에서의 challenge :출력 결과의 구조적 제약, 출력 시퀀스 길이가 입력보다 길어짐(따라서 inference에서 출력 최대 길이=입력길이 + 300으로 설정). 적은 데이터셋 환경

task-specific tuning 없이도 transformer가 뛰어난 성능을 보임을 확인..!

Discussion & Conclusion

기존 encoder-decoder에서 사용되던 recurrent layer를 multi-head attention layer로 대체하여 Attention 만을 기반으로 한 최초의 Sequence 변환 모델 Transformer을 소개

Text외의 Modality(이미지, 오디오, 비디오 등)에 적용하고 더욱 효율적으로 대규모 입출력을 처리하기 이한 local, 제한된 attention 매커니즘 연구 계획

Generation 과정을 non-sequential 하게 만드는 것 또한 연구 계획