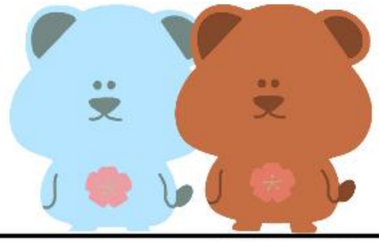
Attention Is All You Need

(Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin)

이은상

EWHA EURON

목차



EWHA EURON

Introduction

EWHA EURON

#01 Introduction

RNN, LSTM, GRU neural networks는 sequence modeling과 transduction problems에서 뛰어난 성능을 보임

그러나 이전 결과를 입력으로 받는 순차적 특성으로 인해 병렬 처리가 불가능하다는 단점 지님

Factorization tricks, conditional computation을 통해 전보다 계산 효율을 상승시켰지만, 근본적인 제약은 아직 그대로 남음

Attention mechanism은 거리에 상관 없이 dependency 모델링이 가능함
→ 다양한 작업에서 강력한 시퀀스 모델링 및 transduction models의 필수 요소가 됨

기존의 attention mechanisms는 대부분 recurrent network와 함께 사용됨

#01 Introduction

본 논문에서는 recurrence를 사용하지 않는 model architecture인 Transformer 제안

Recurrence 대신 global dependencies를 측정하기 위해 전체적으로 attention mechanism을 사용함

Transformer는 병렬처리를 할 수 있었고, 적은 시간 동안 훈련을 한 후 translation quality에서 new state of the art를 달성함

Background

EWHA EURON

#02 Background

순차적 연산을 줄이는 것이 목표인 Extended Neural GPU, ByteNet 및 ConvS2S는 모두 convolution neural networks를 기본 구성 요소로 사용하여 모든 입력 및 출력 위치에 대해 병렬로 숨겨진 표현 계산
→ 필요한 연산 수가 위치 간 거리에 따라 증가
→ 먼 위치 간 종속성 학습하기 어려워짐

Transformer는 이를 constant number로 줄임
그러나 resolution이 줄어드는 문제 발생
→ Multi-Head Attention으로 counteract

* ConvS2S : 선형적으로 증가
* ByteNet : log로 증가

#02 Background

Self-attention
시퀀스의 서로 다른 위치를 관련시켜 시퀀스의 표현을 계산

읽기 이해, 요약, 텍스트 의미 추론 및 작업 독립적인 문장 표현 학습을 포함한 다양한 작업에서 성공적으로 사용됨

End-to-end memory networks
sequence-aligned recurrence 대신 recurrent attention mechanism 에 기반

Simple-language question answering과 언어 모델링 task에서 우수한 성능 보임

#02 Background

Transformer는 시퀀스에 정렬된 RNN이나 convolution을 사용하지 않고 입력 및 출력의 표현을 계산하는 데 Self-attention에 완전히 의존하는 최초의 transduction model

Model Architecture

EWHA EURON

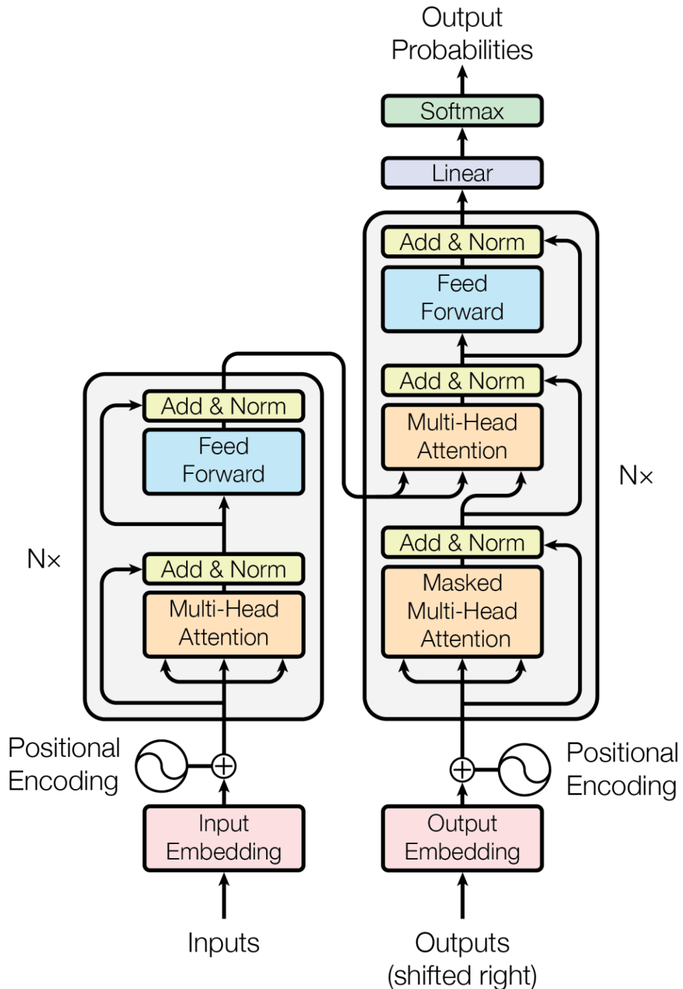
#03 Model Architecture

좋은 성능을 보이는 Neural sequence transduction model들은 대부분 encoder-decoder 구조를 지님
Transformer 또한 이 구조를 따라가고 있고, 그 내부는 self-attention과 fully connected layer만으로 구성되어 있음

Encoder
6개의 같은 레이어로 구성
입력부터 출력까지 512차원으로 고정
- Multi-head self-attention mechanism
- Feed-forward network

두 sub-layer 모두 각각 residual connection, layer normalization 사용

Decoder
- Masked Multi-Head Decoder
- Multi-Head Encoder-Decoder Attention (encoder의 출력을 입력 받음)
- Feed-Forward



#03 Model Architecture

Attention

같은 문장 내에서 단어들의 관계를 나타냄
Query(Q), Key(K), Value(V)로 표현

영어: I love her.
한국어: 나는 그녀를 사랑한다.

Query : 나는
Key : (I, love, her) – 연관성을 찾는 대상

Q와 K의 유사성을 계산하여 유사한 만큼의 Value값 가져올 수 있음

#03 Model Architecture

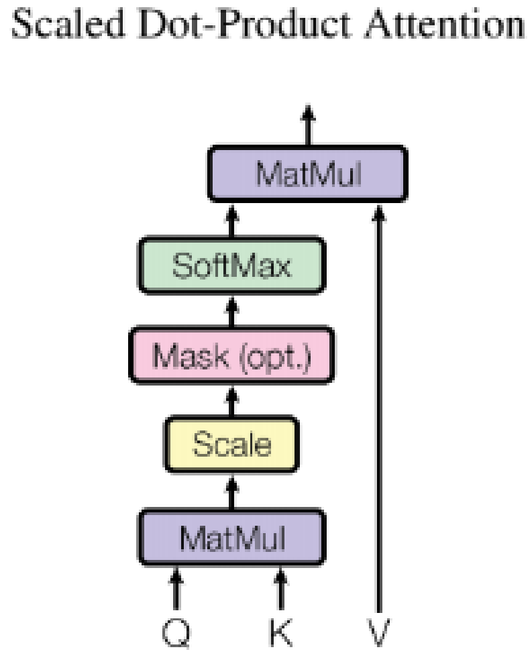
Scaled Dot-Product Attention

Dot-product attention은 matrix를 통해 최적화된 연산을 구현할 수 있기 때문에 훨씬 빠르고 공간 효율적

dk가 너무 큰 경우 행렬 연산 값도 커지며 softmax의 기울기 영역이 작아질 수 있어 dk의 루트만큼 나누어 softmax를 취하도록 함

* Query와 key의 dimension은 dk
* Value의 dimension은 dv

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

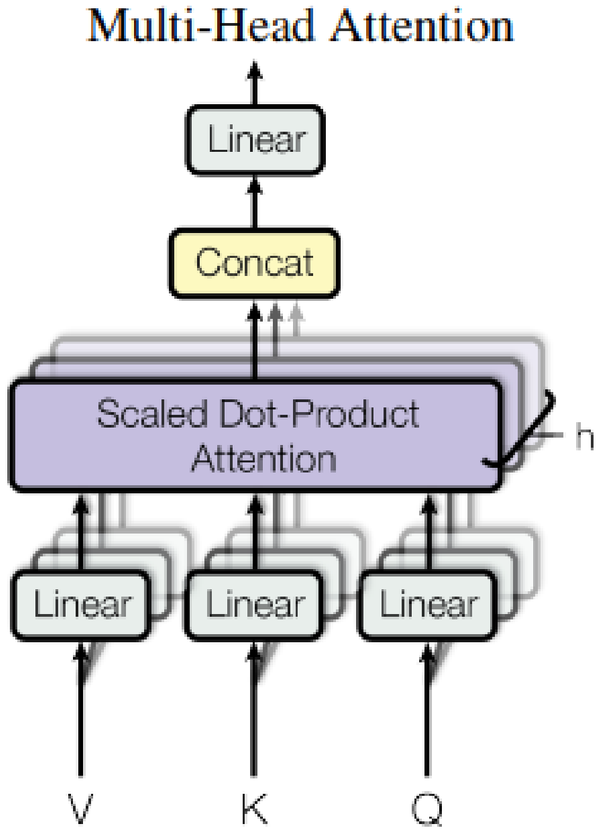


#03 Model Architecture

Multi-Head Attention

d_model 차원의 Q, K, V 사용하여 single attention 수행하는 대신, 각각 dk, dk, dv 차원에 대해 학습된 서로 다른 linear projection 사용하여 Q, K, V를 h회 linear projection

여러 개로 나눈 Maulti-Head들이 서로 다른
Representation Subspaces를 학습하여
다양한 유형의 종속성을 가져 다양한 정보를 결합할 수 있음



Which do you like better, coffee or tea? - 문장 타입에 집중하는 어텐션

Which do you like better, coffee or tea? - 명사에 집중하는 어텐션

Which do you like better, coffee or tea? - 관계에 집중하는 어텐션

Which do you like better, coffee or tea? - 강조에 집중하는 어텐션

#03 Model Architecture

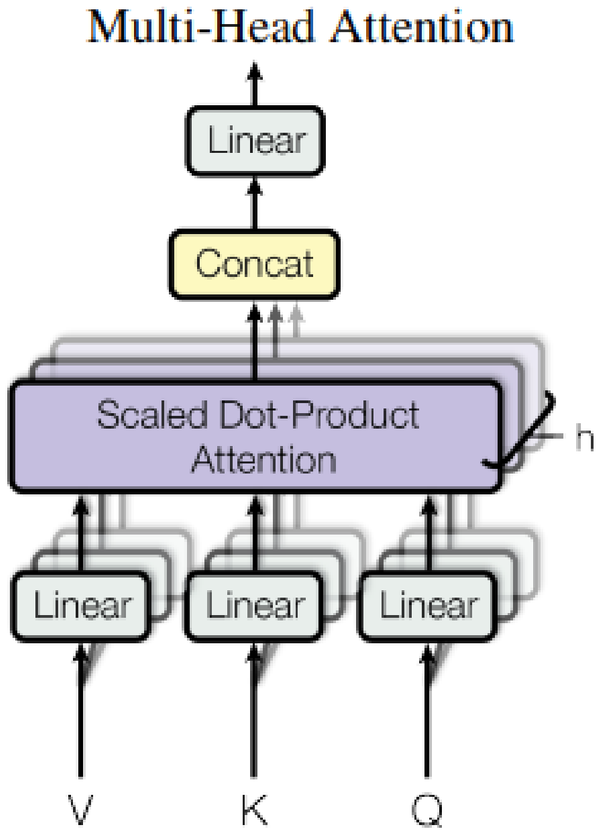
Multi-Head Attention

서로 다른 head들이 서로 다른 어텐션을 가지는 방식으로 동작

원래의 d_model차원을 h개로 나누어 병렬연산한 것이기 때문에 총 계산 비용은 하나의 head를 사용했을 때와 동일

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

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.



#03 Model Architecture

Applications of Attention in our Model

Encoder Self-Attention
Q, K, V 모두 Encoder로부터 가져옴
각 레이어들은 이전 레이어의 출력을 입력으로 받아 이전 레이어에 있는 모든 position에 대응 가능
같은 위치의 Q마다 같은 위치의 K, V들로 attention 값 구함

Masked Decoder Self-Attention
전반적 목표는 Encoder와 동일
출력값이 들어옴(출력값을 input으로 받음)
Sequence model의 Auto-Regressive property 보존 위해 이후에 나올 단어들 참조 안함
(이후의 단어들을 참조하는 것은 일종의 치팅이기 때문)

Encoder-Decoder Attention
Encoder의 마지막 레이어에서 출력된 K, V 사용하고 이전 Decoder에서 Q 사용
Decoder의 sequence들이 Encoder의 sequence들과 어떤 연관을 가지는지 학습 가능

#03 Model Architecture

Position-wise Reed-Forward Networks

Encoder와 Decoder 모두 지니고 있음

2개의 linear layer가 이어져 있고, 사이에 ReLU Activation 사용

Input, output 모두 동일하게 512차원, 중간 inner-layer의 차원은 2048차원

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

#03 Model Architecture

Embeddings and Softmax

학습된 embedding 통해 input token과 output token d_model차원의 벡터로 변환

Decoder 단 출력에서 linear transformation 거치고 Softmax function 위해 토큰들의 확률값 계산

두 개의 임베딩은 linear transformation으로 동일한 weight matrix 공유함

Inner layer에서 이러한 Weights에 $\sqrt{d_{model}}$ 곱함

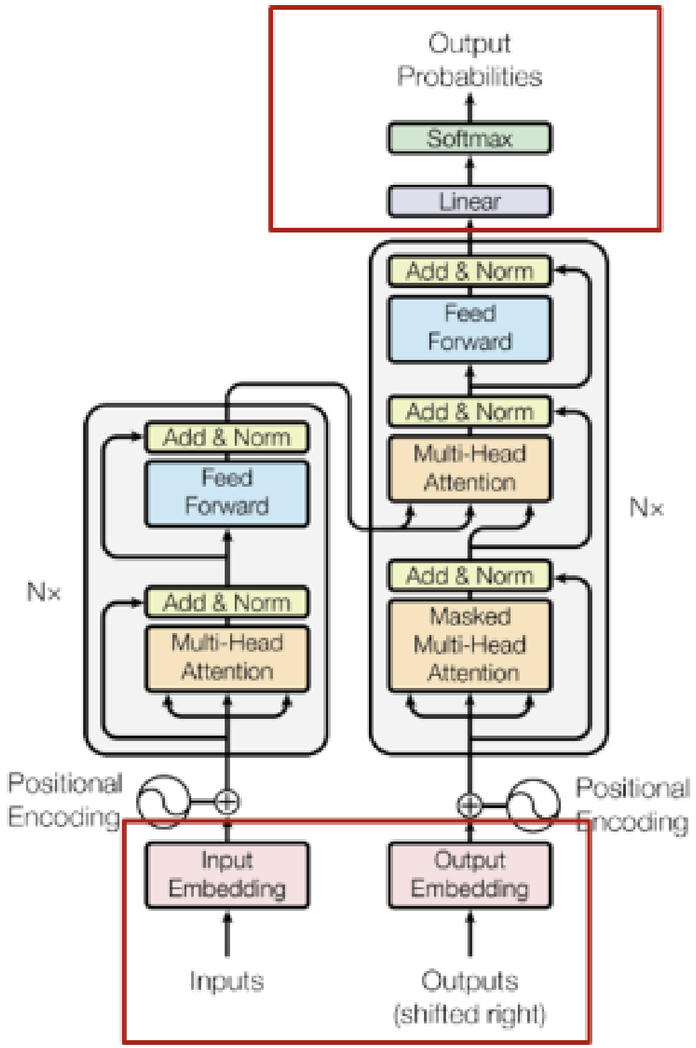


Figure 1: The Transformer - model architecture.

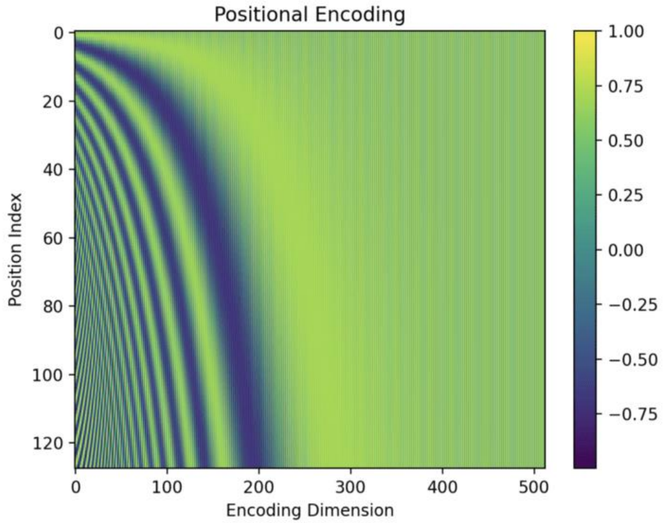
#03 Model Architecture

Positional Encoding

Sequence의 순서를 사용하기 위하여 positional encoding 사용

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

이 중 sinusoidal version이 훈련 중 더 긴 시퀀스 길이 추정 가능
→ sinusoidal version 사용



Why Self-Attention

#04 Why Self-Attention

total computational complexity per layer

amount of computation that can be parallelized

path length between long-range dependencies in the network

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

Training

#05 Training

Training Data and Batching

WMT 2014 English-German dataset & WMT 2014 English-French dataset으로 훈련
영어-독일어는 byte-pair encoding 사용
영어-불어는 word-piece vocabulary 사용
문장 쌍은 근사적인 시퀀스 길이에 따라 배치되었으며, 각 배치에는 약 25000개의 소스 토큰과 25000개의
대상 토큰이 포함되도록 함

Hardware and Schedule
8개의 NVIDIA P100 GPU가 장착된 한 대의 컴퓨터에서 훈련
기본 모델의 경우 훈련 단계당 약 0.4초 소요, 총 100000 단계(12시간) 훈련
대현 모델의 경우 단계 시간 1.0초, 300000단계(3.5) 훈련

#05 Training

Optimizer
Adam optimizer 사용하되, 특정 hyperparameter 적용
훈련 단계에 따라 학습률을 변화시키되, 처음에는 선형적으로 증가하고 이후 단계 수의 역 제곱근에 비례하여 감소
Warmup steps value는 4000으로 설정

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

$$\beta_1 = 0.9, \beta_2 = 0.98 \text{ and } \epsilon = 10^{-9}$$

Regularization
Residual Dropout
각 하위 레이어의 출력과 임베딩 위치 및 위치 인코딩의 합에 드롭아웃 적용
기본 모델의 경우 드롭아웃 비율 0.1
Label Smoothing
훈련 중 적용 값은 0.1로 설정
모델이 더 불확실하게 학습하지만 정확도와 BLEU 점수 향상시킴

Results

#06 Results

좋은 성능을 보임

너무 많은 head는 성능이 저하됨을 보임

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	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

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

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

Conclusion

EWHA EURON

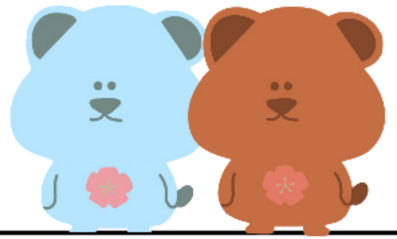
#07 Conclusion

본 논문에서는 encoder-decoder architecture에서 가장 일반적으로 사용되는 recurrent layer를 multi-headed self-attention으로 완전히 대체한 최초의 시퀀스 변환 모델인 Transformer를 제시함

번역 작업에서 Transformer는 recurrence 또는 convolution layer를 기반으로 하는 architecture보다 훨씬 빠르게 훈련될 수 있었음
→ 이전 최고의 모델을 능가하는 state of art의 성능을 보임

미래에 attention based models의 연구가 계속 될 것임

THANK YOU



EWHA EURON