



[논문 리뷰] Transformer [Attention] [2017]



Attention Mechanisms에 집중한 새로운 Architecture인 Transformer를 제안합니다.

실험에서는 품질이 우수하면서도, **병렬 학습이 가능하다는 것을 증명**하였습니다.
또한 Transformer는 다양한 Task에서도 data를 잘 학습하여, 일반화됨을 보여줍니다.

Introduction

Recurrent language model과 Encoder-Decoder의 한계를 넓히기 위하여, 수많은 노력이 계속되었습니다.

Model들은 Input과 Output의 위치에 따라 부분적으로 계산을 진행합니다.

본질적으로 **Sequential**한 특성은 **훈련과정에서의 병렬화를 배제**하는데, 이러한 메모리 제약에 의해 샘플간 배치화를 제한하여, 더 긴 길이의 **시퀀스를 처리할수록 Critical한 문제가 발생**합니다.

최근에 **factorization tricks**과 **conditional Computation**을 사용함으로써, 상당한 개선을 가져왔습니다.

하지만, 여전히 근본적인 문제가 해결되지 않았음을 알 수 있었습니다.

Attention Mechanism은 **Input과 Output Sequence**거리에서 **상관없는 의존성 (dependency)모델링**을 가능하게 함으로 다양한 task의 Sequence Modeling 및 transduction model 등에서 필수적인 부분이 되고 있습니다.



Recurrence를 회피하는 대신에 전적으로 Input과 Output의 dependency를 추출하기 위해 Attention Mechanism의 의존한 Transformer Model을 제안합니다.

Background

이것들은 모든 input, output 포지션에서 병렬적으로 hidden representation을 계산하는 CNN을 basic한 building block으로 사용하였다. 이 모델들에서는 input과 output의 임의의 두 위치로부터의 신호를 관계시키기 위해 필요한 계산 수가 그 위치들간 거리에 따라 증가하는데, ConvS2S에서는 선형적으로, ByteNet에서는 logarithmically하게 증가한다. 이는 거리가 멀어질 수록 dependency를 학습하는데 어려워지게 만든다. 반면 트랜스포머에서는 어떤 **상수 번의 계산만이 요구된다**. 비록 어텐션 가중 position을 평균냄으로써 유효 해상도의 감소라는 cost가 있지만, 이후 살펴볼 **Multi-Head Attention**으로 상쇄시킬 수 있다.

Self-Attention(infra-attention이라고도 불림)은 **Sequence의 표현을 계산**하기 위해, Single Sequence내에서 다른 위치와 연관시키는 **attention Mechanism**이다.

✨ **Self-Attention**은 독해, 요약, 텍스트 수반, 문장 표현등 NLP의 다양한 Task에서 성공으로 사용되었다.



그러나, **Transformer는 Input과 Output의 Representation을 계산**하기 위해, 오로지 Self-Attention에만 의존한 최초의 transduction Model이다.

Model Architecture

✨ Transformer는 전반적인 Encoder-Decoder구조를 사용하는데,

Self-attention, Point-wise, fully connected layer들을 encoder, decoder에 쌓아 사용합니다.

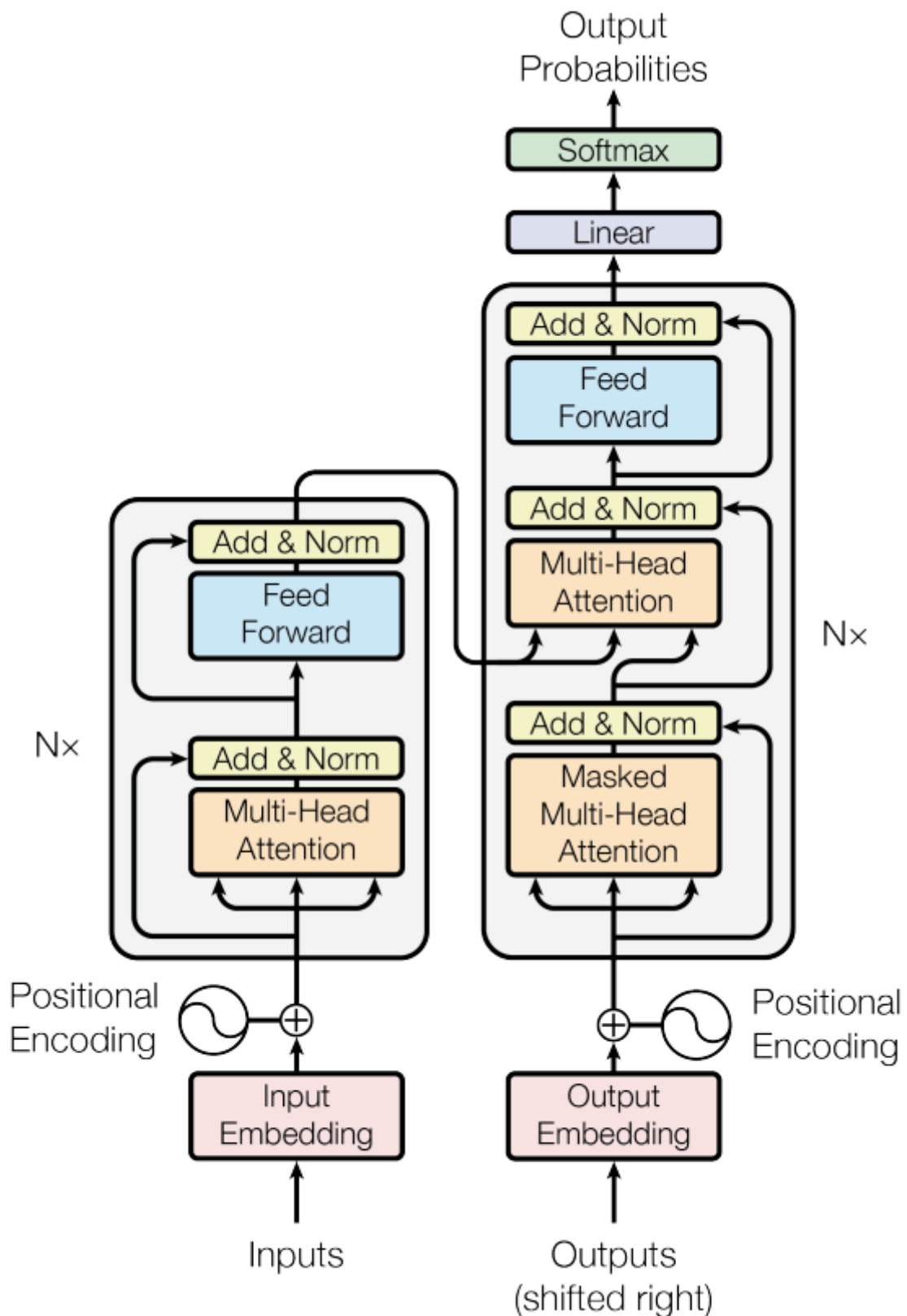


Figure 1: The Transformer - model architecture.

3.1 Encoder and Decoder Stacks

Encoder :

Encoder는 6개의 **layer**가 쌓여 구성되었습니다. 각각의 **layer**는 두 개의 **sub-layer**를 가지고 있습니다.

첫 번째 **layer**는 **multi-head self-attention mechanism**이며,

두 번째 **layer**는 **point-wise fully connected feed-forward network**입니다.

두개의 **sub-layer**마다 각각 **residual connection**을 적용하고, 이후 **layer normalization**을 따르게 합니다.

이 **Residual Connection**의 적용의 편의를 위해, **Embedding**층을 포함한 모든 **layer**들이 **Output Dimension 512**를 갖도록 통일하였습니다.

Decoder :

Decoder 또한 6개의 **layer**가 쌓여 구성되었습니다. 두 가지 **sub-layer**로 구성되어 있으며, **encoder**의 **output**에 대해 **multi-head attention**을 수행하는 **sub-layer**를 추가하였습니다.

Encoder와 유사하게, **residual Connection**과 **layer normalization**을 적용합니다.

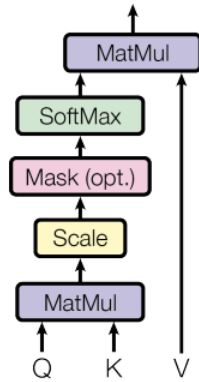
Decoder stack내의 **self-attention layer**를 수정합니다.

(미래의 위치에 접근하는 것을 불가능하고, **이전의 위치들에 대해서만 의존하도록 masking 기법을 활용**합니다.)

3.2 Attention

Attention function은 **query, key, value, output**들이 모두 벡터일 때, **key-value**쌍의 **Output**이 **query**에 **mapping**되게 유사합니다.

Scaled Dot-Product Attention



Multi-Head Attention

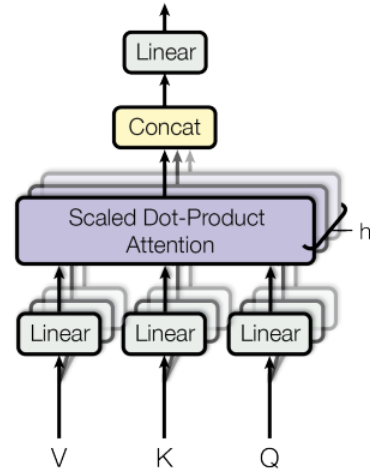


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

3.2.1 Scaled Dot-Product Attention

“Scaled Dot-Product Attention”은 d_k 차원의 key와 d_v 차원의 value들로 구성되어 있습니다.

실제로는 모든 쿼리들이 하나의 행렬 Q로써, 키와 밸류들은 행렬 K와 V로써 packed 되어 동시에 (simultaneously) 어텐션 함수가 계산된다. 어텐션 output 또한 V와 shape가 동일한 행렬이 되며 아래와 같이 계산된다.

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

흔히 사용되는 어텐션 함수는 additive attention과 dot-product attention인데, dot-product attention의 경우 scaling factor로써의 $\frac{1}{\sqrt{d_k}}$ 를 제외하면 본 연구의 제안 알고리즘과 동일하다.

additive attention은 단일 은닉층의 feed-forward network를 이용해 연관성 함수를 계산한다. 두 방법이 이론상 동일한 복잡도를 갖는 반면, 실무에서는 **dot-product attention**이 고도로 최적화된 행렬곱 **code**로 구현될 수 있어 더 빠르고 더 공간효율적이다.

쿼리와 키 벡터의 차원 d_k 가 작은 값일 때는 additive attention과 dot-product attention이 유사한 퍼포먼스를 보이는 반면, d_k 가 클 때 scaling을 적용하지 않으면 additive attention이 dot-product attention의 성능을 앞질렀다.



연구진들은 d_k 가 커질 경우 dot product의 결과값도 커지는 경향이 있습니다. 결과적으로 softmax함수에서 gradient가 매우 작아지는 saturate한 부분으로 QK^T 가 Input 된다고 의심하여, 이것을 방지하기 위해 스케일을 조정하였습니다.

3.2.2 Multi-Head Attention

d_{model} 차원의 key, value, query들을 Single attention function을 performing하기 전에,

Query, key, value를 각각 d_k, d_k, d_v 차원으로 변환하는 서로 다른 h개의 학습 가능한 선형 사영(linear projection)을 진행한 뒤,

각각의 다른 projected version에 대해 병렬적으로 Attention을 진행하는 것이 더 이롭다는 사실을 알았습니다.

멀티헤드 어텐션은 모델이 서로다른 domain들의 서로다른 representation 부분공간(subspace)들로부터 **결합적으로(jointly) 정보에 접근**하도록 한다. 단일 어텐션 헤드에서는, averaging 때문에 이러한 부분공간들로부터의 joint한 정보접근이 억제된다.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

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 $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

이 연구에서 $h = 8$ 의 parallel attention head를 사용하였고, 각 헤드마다 $d_k = d_v = d_{\text{model}} / h = 64$ 의 차원을 설정하였다. 각 헤드에서 감소된 차원 덕분에 단일 헤드로 full 차원 어텐션을 진행할 때와 전체적인 계산 cost가 동일하다.

3.2.3 Applications of Attention in our Model

💡 **Transformer**는 Multi-head attention을 3가지 다른 방식으로 활용합니다.

1. “encoder-decoder attention” layer에 Input되는 Query는 이전 decoder layer로부터 Output으로 오고, Key와 Value는 Encoder의 Output으로부터 옵니다.
decoder의 모든 위치에서 Input Sequence의 모든 위치를 접근하는 것이 허용됩니다.
Seq2Seq Model을 모방한 것이 encoder-decoder attention mechanisms입니다.
2. encoder는 self-attention layer를 포함하고 있습니다. self-attention layer에 input되는 모든 Key, values, queries들은 같은 공간으로부터 옵니다.
각 encoder내의 position은 이전의 모든 encoder layer에 접근하는 것이 가능합니다.
3. 유사하게, decoder 내부의 self-attention층은 모든 decoder층의 position들이 이전 decoder층의 모든 position에 접근하는 것이 가능하도록 합니다. decoder의 auto-regressive 성질을 보존하기 위해, leftward information flow를 방지할 필요가 있습니다. 이를 위해 **masking**을 진행합니다.

3.3 Position-wise Feed-Forward Networks

어텐션 sub-layer에 더하여, 우리의 encoder와 decoder는 각각의 **position**에 대해 독립적으로, 동등하게 적용되는 **fully connected feed-forward network**를 포함한다. 이 층은 두 번의 선형변환(linear transformation)과 그 사이에 ReLU activation을 포함한다. 공식은 아래와 같다.

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

선형변환이 다른 position들에 대해 동일하게 적용되는 반면, 층간(layer to layer)에는 서로 다른 파라미터(W_1, W_2)를 사용한다. 이는 커널사이즈가 1인 convolution을 두번 진행하는 것으로도 생각할 수 있다. input과 output의 차원은 동일하게 $d_{model} = 512$ 이고, FFN 내부의 hidden layer는 2048의 차원을 갖는다.

3.4 Embeddings and Softmax

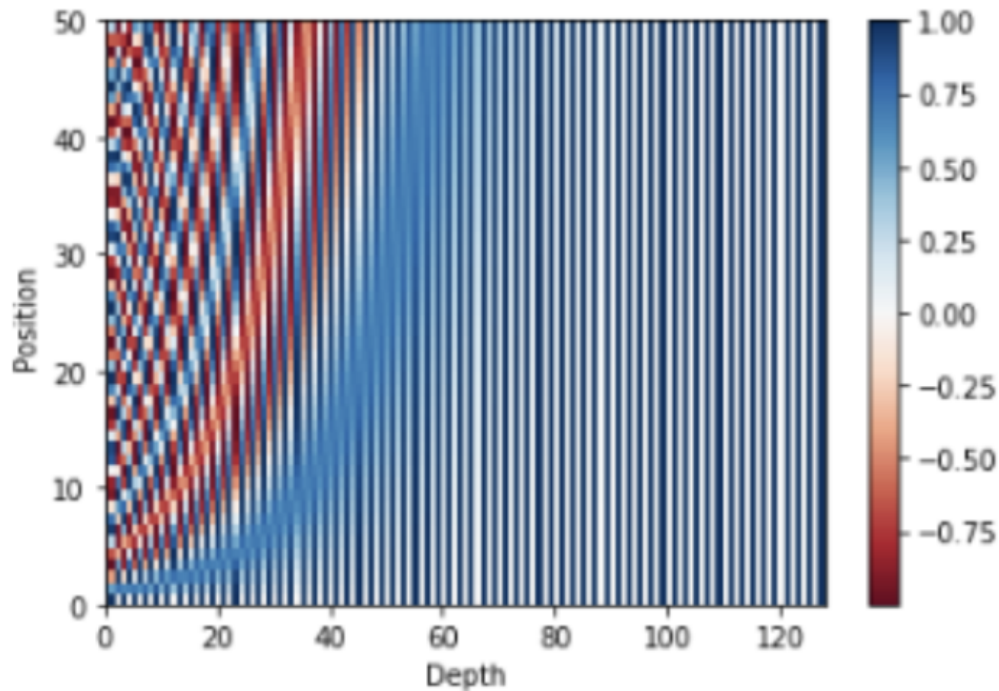
다른 Sequence 변환 모델과 유사하게, Input과 Output을 d_{model} 차원 벡터로 변환하기 위해 학습가능한 임베딩을 사용하였습니다.

또한, decoder의 Output으로, next-token 확률을 예측하기 위해, 학습가능한 선형변환과 소프트맥스 함수를 사용하였습니다.

3.5 Positional Encoding

저자들이 제안하는 모델이 어떠한 recurrence나 convolution도 포함하지 않기 때문에, 모델이 시퀀스의 순서를 반영 및 사용할 수 있도록 시퀀스 내의 토큰들의 상대적/절대적 위치에 관한 정보들을 주입해야만 했다. 이를 위해 우리는 encoder와 decoder의 최 하단에 위치한 input embedding에 "**positional encodings**"를 더하였다. positional encoding은 임베딩과 마찬가지로 d_{model} 차원을 갖게하였고 그렇게해서 두 벡터가 더해질 수 있게하였다. 학습가능한, 혹은 고정된 positional encoding의 선택지는 매우 다양하다.

이 연구에서는 다른 주기의 sine, cosine 함수들을 사용하였다.



각 위치 인코딩의 차원 i 들은 사인 곡선에 대응된다. 이러한 함수를 선택한 이유는 그 어떠한 고정된 offset값 k 에 대해서도 PE_{pos+k} 가 PE_{pos} 의 선형함수로서 표현될 수 있기에 모델이 상대적인 위치를 쉽게 학습할 것이라 가설을 세웠기 때문이다.

Why Self-Attention

Sequence의 Representataion을 동일한 길이의 다른 Sequence의 표현으로 Mapping 할 때 사용된, Recurrent와 Convolution layer와 Self-Attention을 비교하겠습니다.

3가지 이유로 인하여, Self-Attention이 선호됩니다.

1. 하나의 계층당 총 계산 복잡성(total computational Complexity)입니다.
2. 계산을 시행할 때, 병렬화(Parallel)가 가능합니다.
3. 네트워크 내부 long-range dependencies간의 경로의 길이를 입니다.

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

NLP에서는 input되는 시퀀스의 길이 n 이 임베딩 차원 d 보다 작은 경우가 대부분이므로 총당 계산복잡도 측면에서 self-attention이 recurrent나 conv에 비해 좋으며, sequential한 operation이 상수의 복잡도를 가져 선형 복잡도를 갖는 recurrent보다 더 낮다. 또 의존성을 가질 수 있는 maximum path의 길이에 대해서도 self attention은 문장 내의 모든 단어에 한 번에 접근하므로 $O(1)$ 의 복잡도를 가져, recurrent나 conv에 비해 좋다!

또한 부수적인 이익으로 모델에서 어텐션을 이용할 경우 구해진 **attention distribution**이 시각화가 가능하여 사용하는 모델의 해석 가능성(설명력)을 높여준다. 각각의 어텐션 헤드는 서로다른 task들을 수행하도록 분명히 학습될 뿐만 아니라, 문장내의 syntactic하고 semantic한 구조와 관련된 특성을 포착하는 것을 보여준다.

Experiments & Conclusion

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	$2.3 \cdot 10^{19}$	

Label Smoothing During training, we employed label smoothing of value $\epsilon_{ls} = 0.1$ [36]. This hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

본 연구에서는 Transformer를 제시합니다.


Transformer는 오로지 **attention**에 기반을 둔 **first Sequence transduction model**입

니다.

recurrent layer들을 Encoder Decoder 구조인 multi-headed, self-attention로 대체 하였습니다.

Attention Is All You Need

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks in an encoder-decoder configuration. The best performing models

 <https://arxiv.org/abs/1706.03762>




The Illustrated Transformer

이번 글에서 다뤘던 attention seq2seq 모델에 이어, attention 을 활용한 또 다른 모델인 Transformer 모델에 대해 얘기해보려 합니다. 2017 NIPS에서 Google이 소개했던 Transformer는 NLP 학계에서 정말 큰 주목을 끌었는데요, 어떻게 보면 기존의 CNN 과 RNN 이 주를 이뤘던 연구들에서 벗어나 아예 새로운 모델을 제안했기 때문이지 않을까 <https://nlpinkorean.github.io/illustrated-transformer/>

Transformer 논문리뷰 (Attention Is All You Need, NIPS 2017)

논문 Attention Is All You Need의 Training과 Result를 제외한 나머지 부분을 모두 정리(번역..?)했습니다. 오류 지적이나 질문 너무너무 환영입니다 :) convolution이나 recurrent layer 없이 순수 attention 메커니즘만

 <https://vlog.io/@changdaeoh/Transformer-%EB%85%BC%EB%AC%B8%EB%A6%AC%EB%B7%B0>

Paper Review