Transformer

김동원

목차

Transformer

논문 요약 정리 노트

Transformer(Attention is all you need)

RNN, CNN이 아닌 self-attention을 이용한 모델이다.

이전에 seq2seq의 attention은 decoder와 encoder의 hidden state의 유사성을 비교하는 반면,
Transformer에서 self-attention은 입력에서 각각의 단어에 대해 유사성을 계산하는 방법

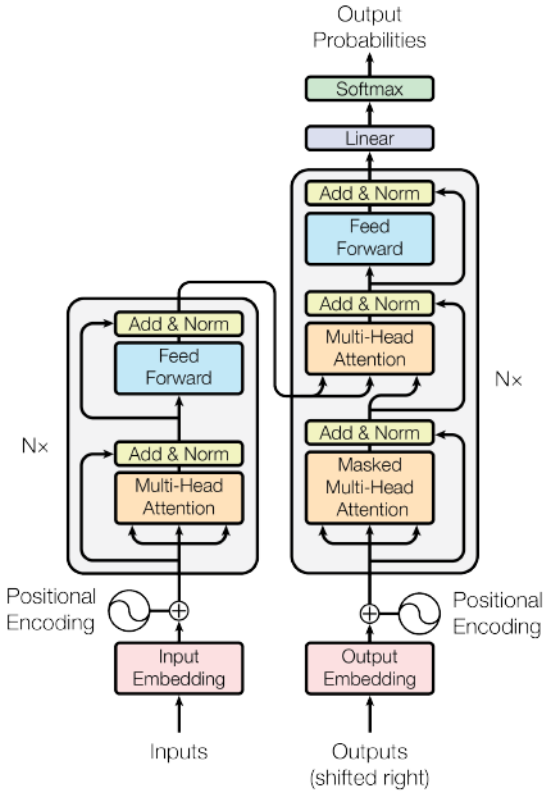


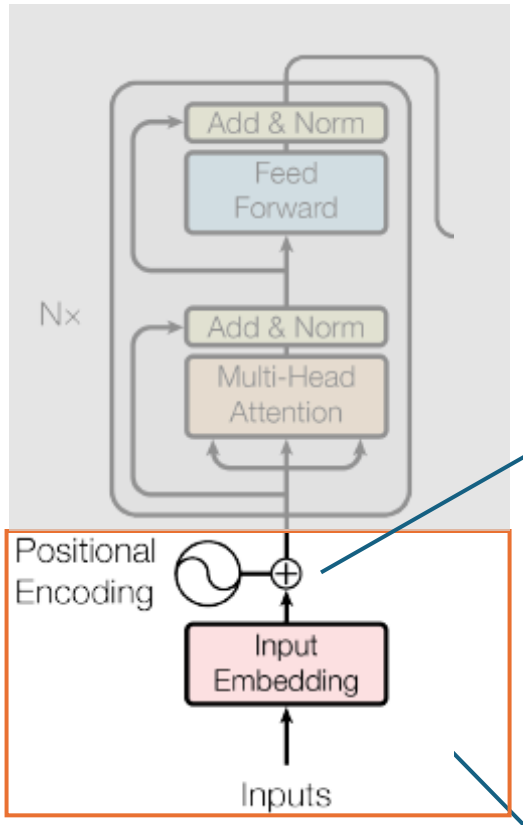
Figure 1: The Transformer - model architecture.

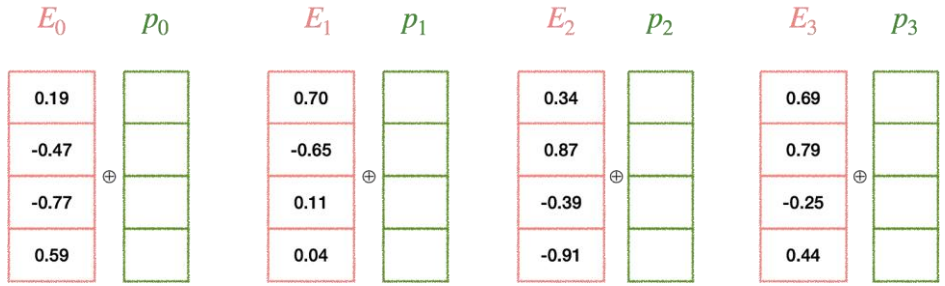
Self-attention이란 입력 문장에서 각각의 단어들에 연관성을 파악하는 것이다.

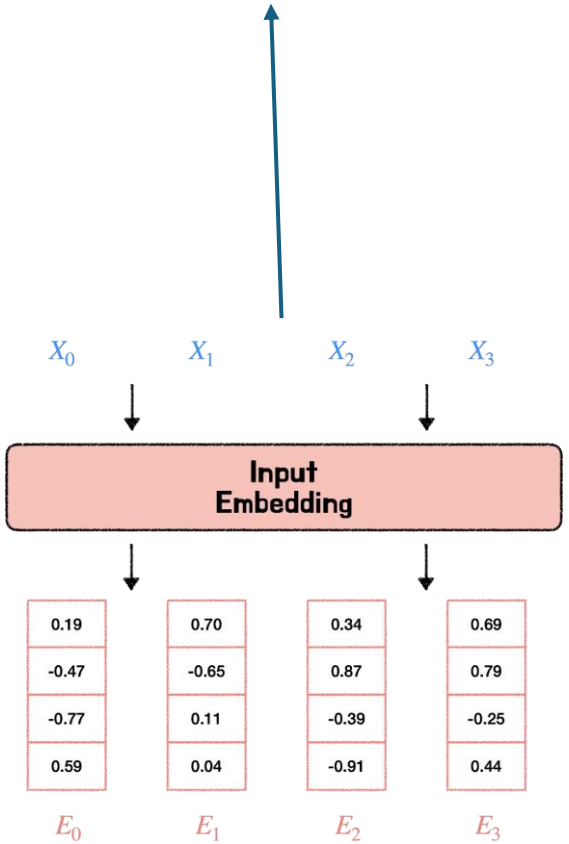
Seq2seq의 attention은 입력과 출력의 단어의 연관성을 파악하는 것이고
transformer의 attention은 문장 내의 단어의 연관성을 파악하는 것이다.

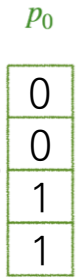
RNN, CNN을 사용하지 않아 순서에 대한 정보를 제공하기 위해 positional encoding과정을 거친다.

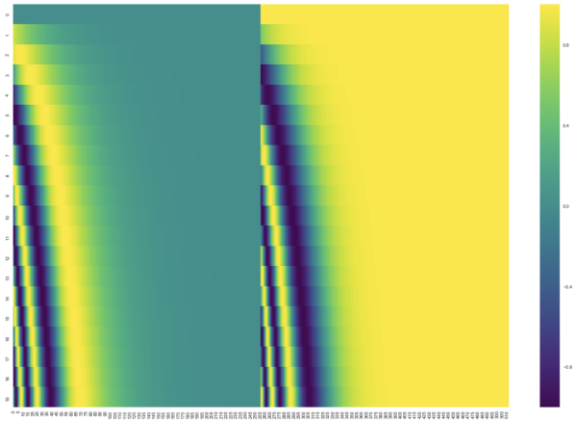
Transformer(Attention is all you need)











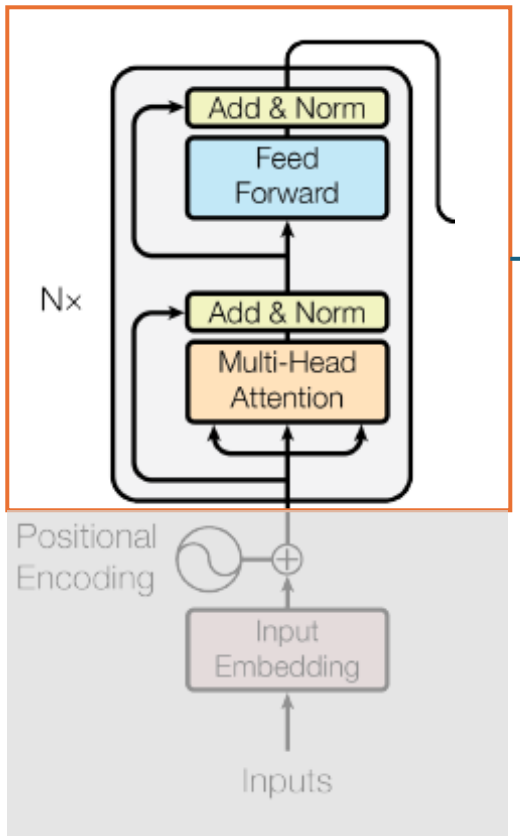
Encoder의 입력은 각 단어의 임베딩 벡터와 위치정보를 포함하는 positional encoding을 통해 구해진다.

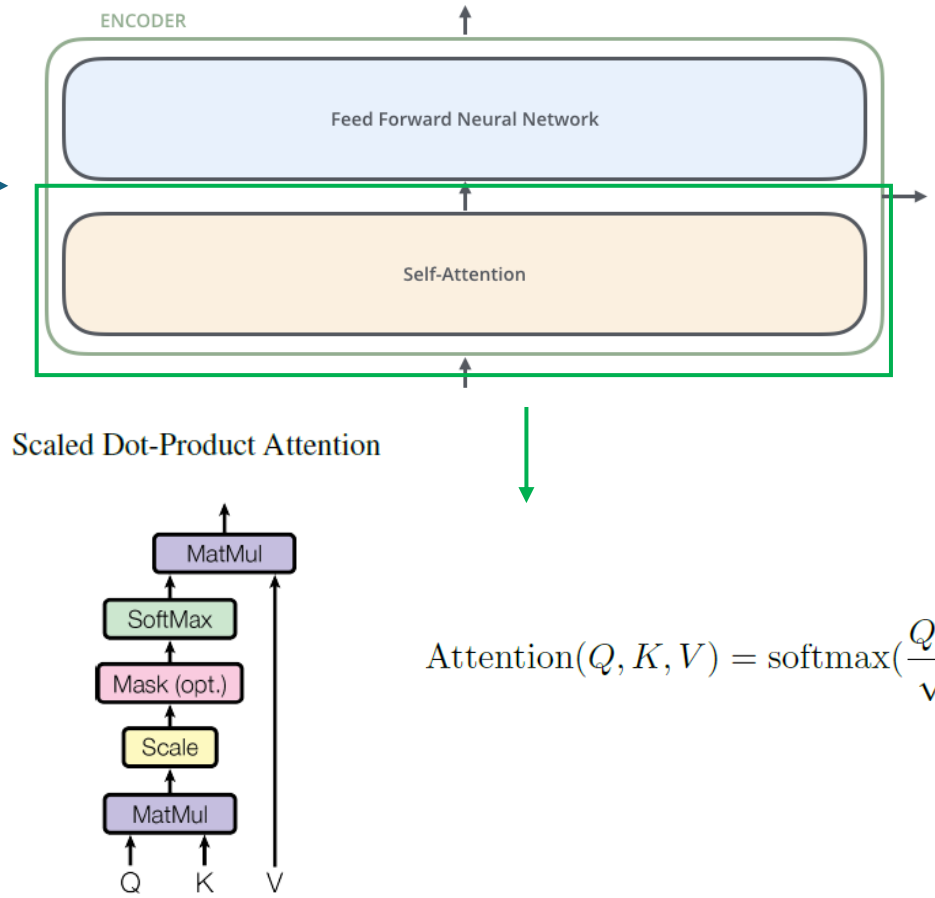
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

Positional encoding 값은 문장이 바뀌어도 같은 값을 가지고, 너무 큰 값을 가지게 되면 의미파악을 위한 값이 희미해 진다.

Transformer(Attention is all you need)

Self-attention



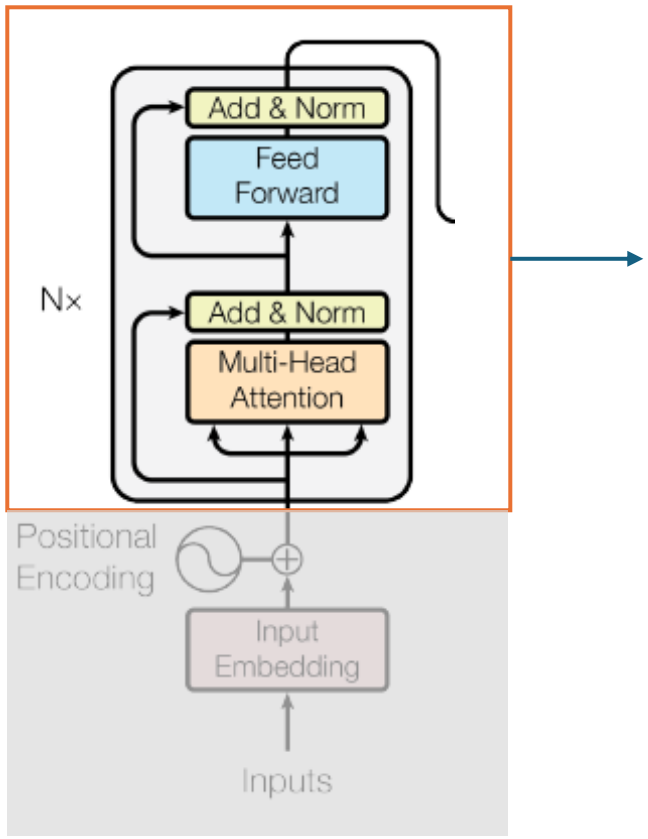


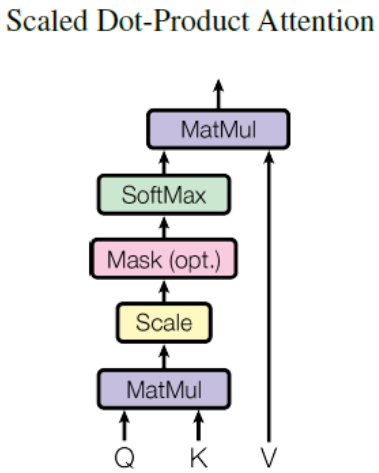
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Self-attention으로 통해 각 문장의 요소들 간의 연관성을 파악하기 위한 query, key, value데이터로 나눈다.

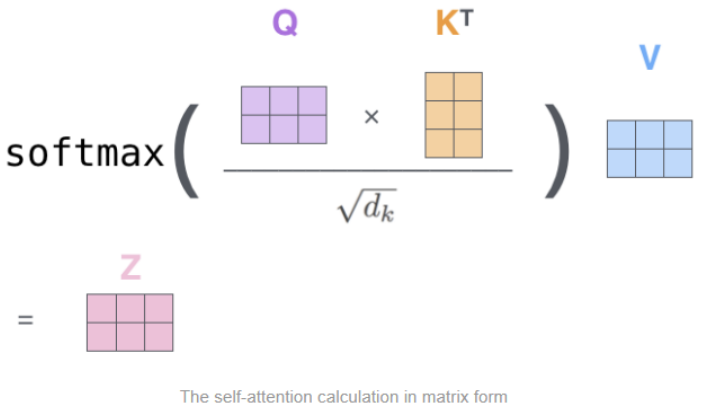
Transformer(Attention is all you need)

Self-attention





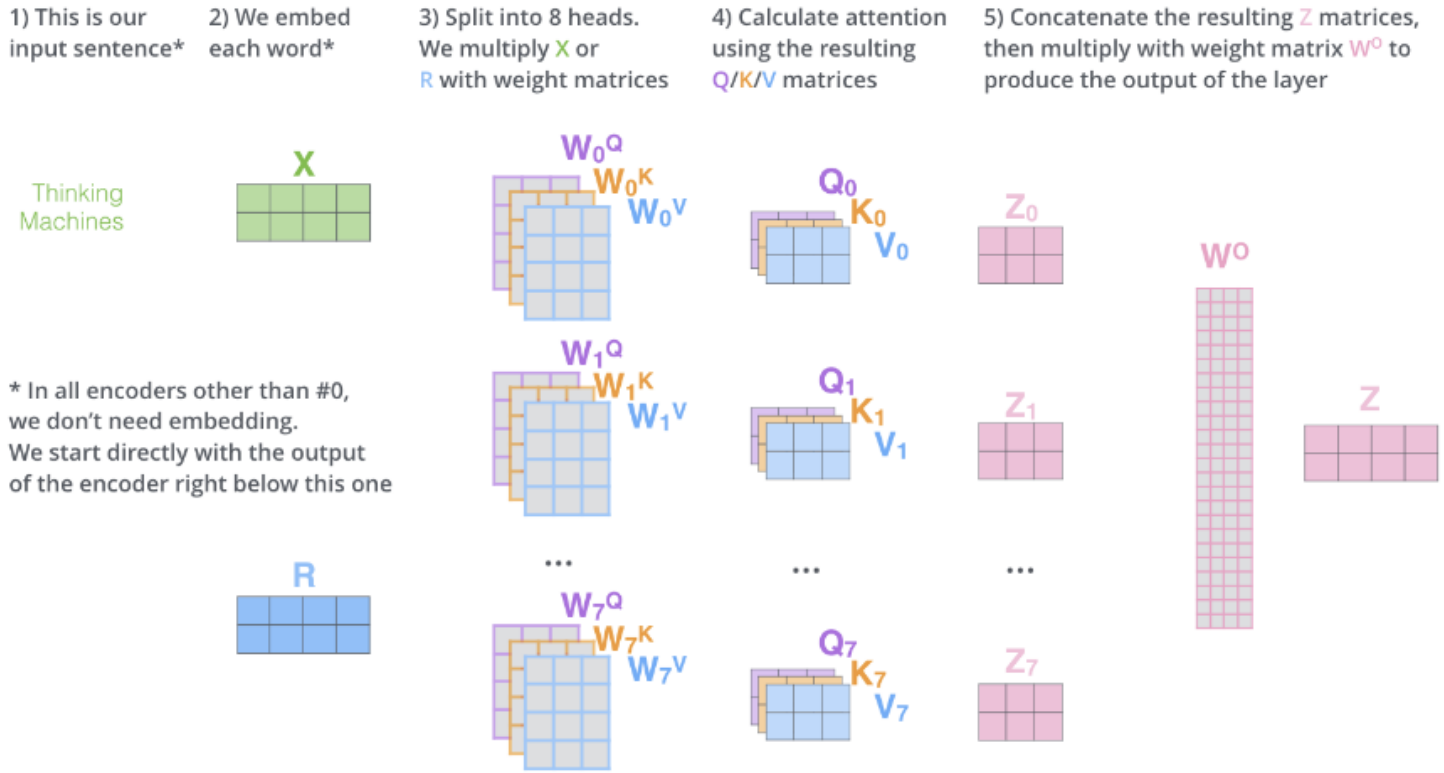
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



The self-attention calculation in matrix form

Transformer(Attention is all you need)

Multi-head-self-attention은 self attention에 비해 를 작은 크기의 query, key, value를 통해 다양하게 학습시키는 방법이다

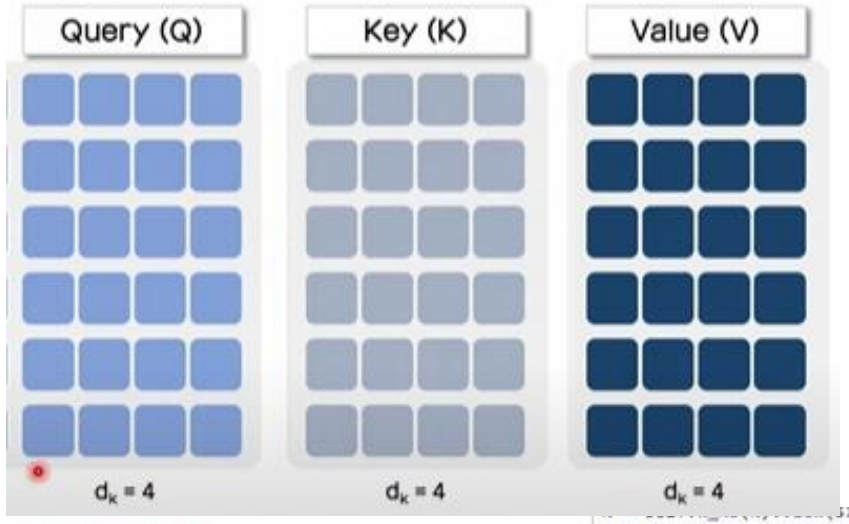


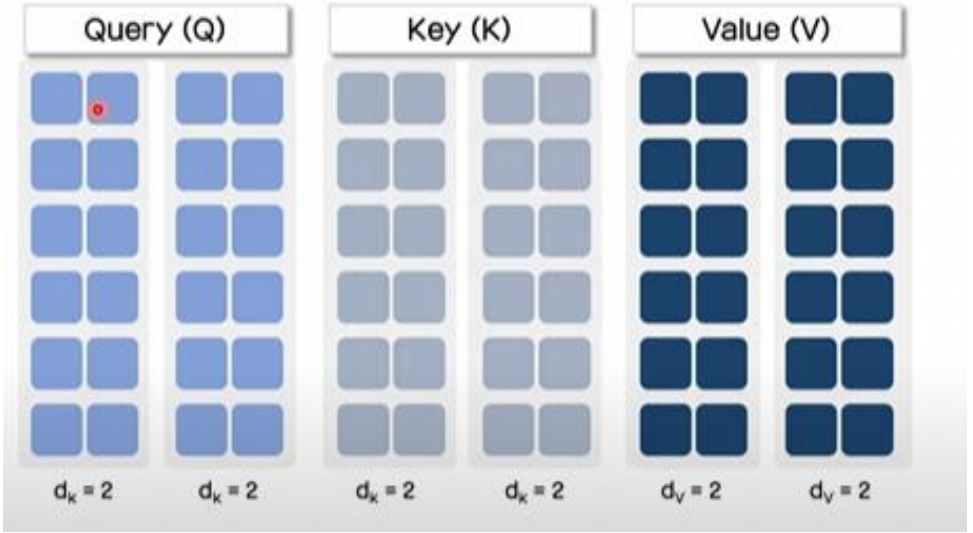
$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$

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

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

$$d_k = d_v = d_{\text{model}}/h = 64.$$





Transformer(Attention is all you need)

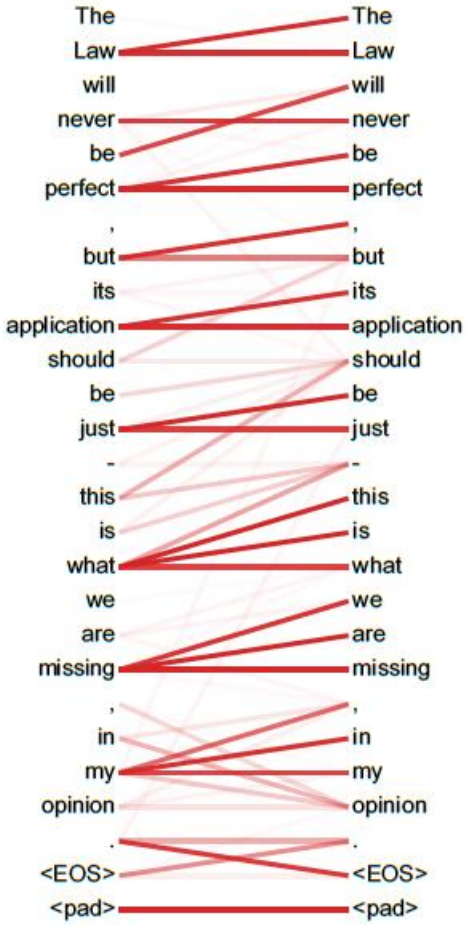
Multi-head-self-attention을 통해 학습하는 것

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

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

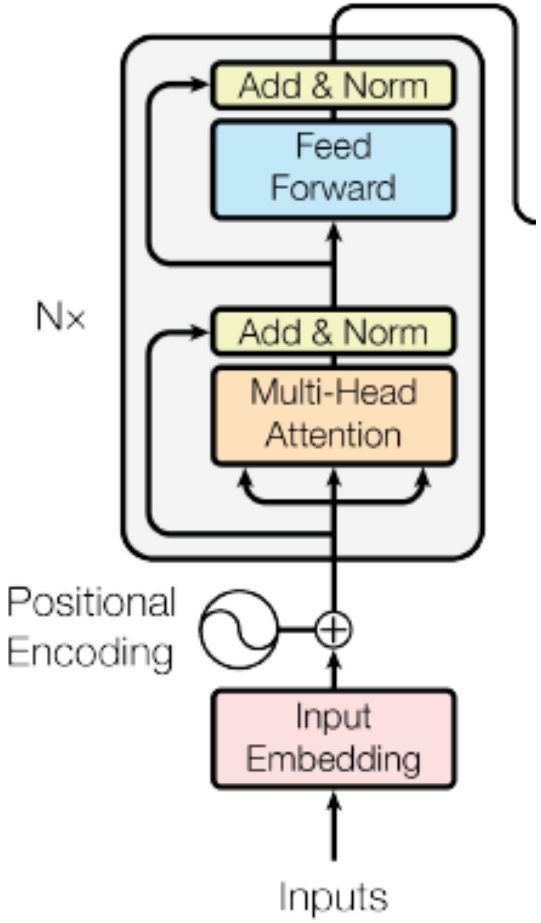
Which do you like better, coffee or tea? - 관계에 집중하는 어텐션

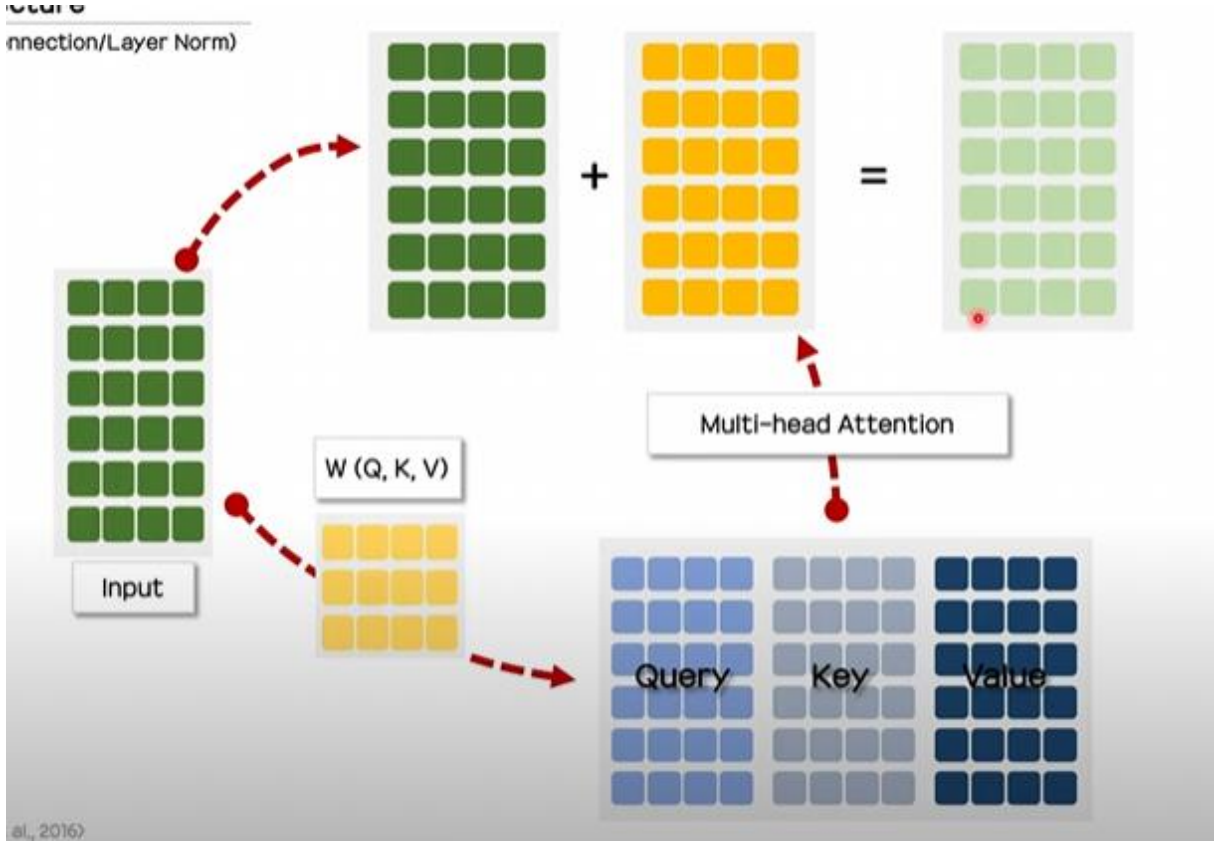
Which do you like better, coffee or tea? - 감조에 집중하는 어텐션



Transformer(Attention is all you need)

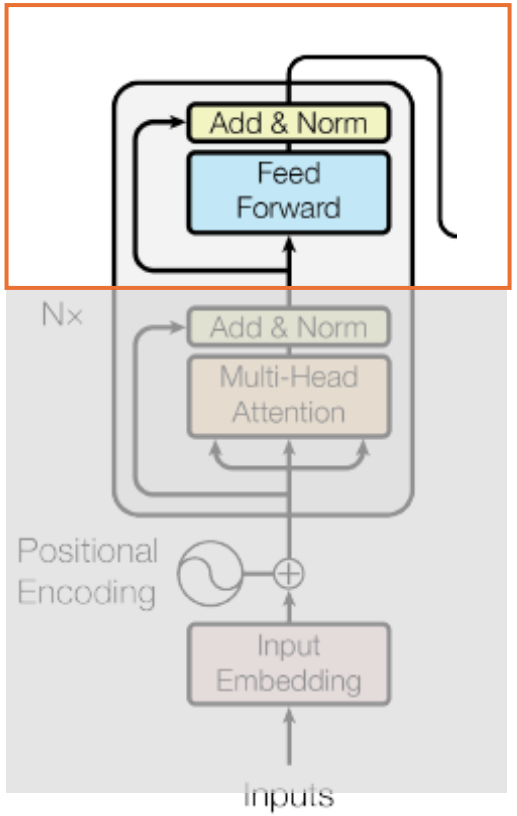
Residual connection을 사용하여 학습의 효율과, 성능을 높였다.

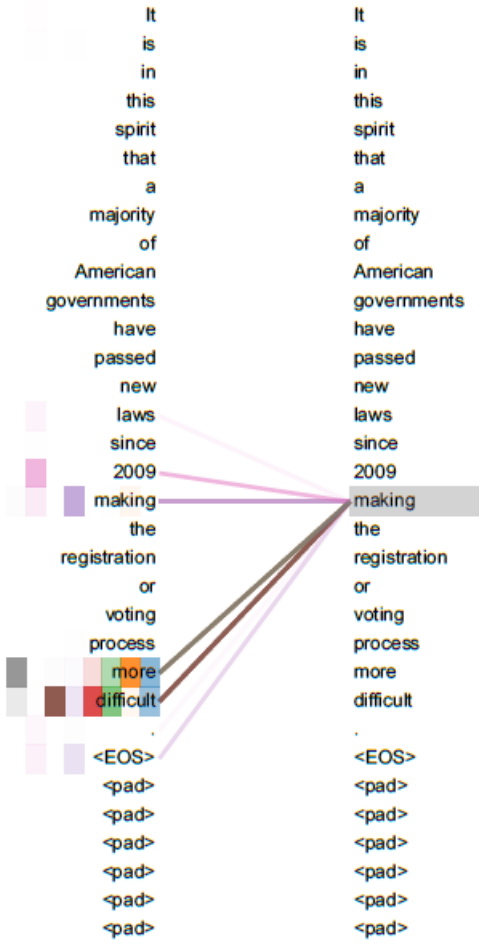




Transformer(Attention is all you need)

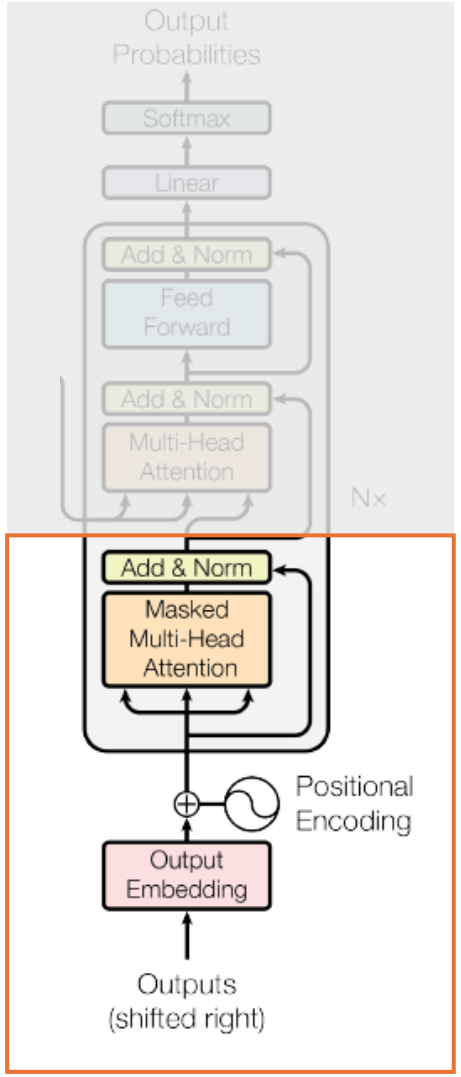
Attention을 거친 벡터는 FFN 네트워크를 거쳐 비선형 변환을 통해 더 복잡한 구조를 학습하게 된다.



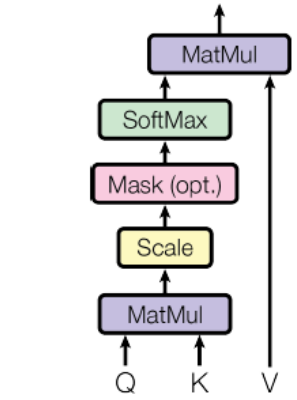


Transformer(Attention is all you need)

Decoder에서는 타겟 문장의 임베딩과 positional encoding과 masked multi head attention을 수행한다.



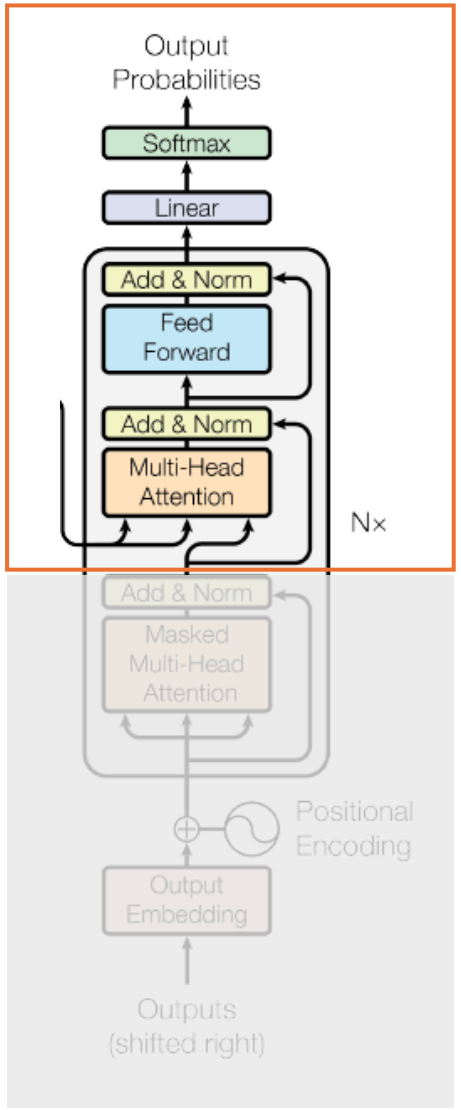
Masked multi-head-attention은 타겟 문장에서 단어를 순서대로 예측하는 것을 고려해서 단어간의 연관성을 파악할 때 이후 단어는 고려하지 않게 하는 방법이다.

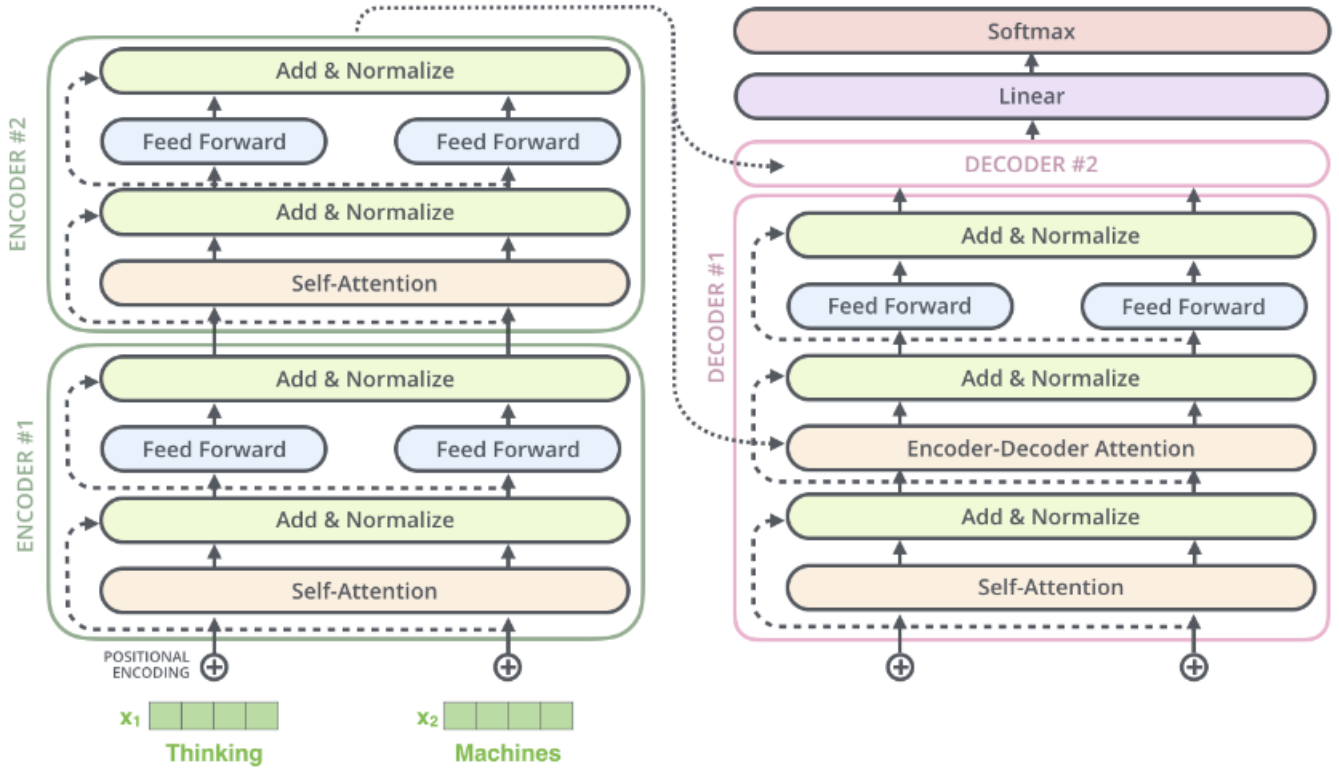


	단어1	단어2	단어3	단어4	단어5
단어1		-1e9	-1e9	-1e9	-1e9
단어2			-1e9	-1e9	-1e9
단어3				-1e9	-1e9
단어4					-1e9
단어5					

이후 결과로 얻은 벡터를 다음에 이어지는 multi-head-attention layer의 query로서 제공하고, Key와 value는 encoder의 마지막 레이어에서 얻은 출력 벡터를 통해 사용한다.

Transformer(Attention is all you need)





Decoder의 최종 결과는 Fully connected layer를 통해 타겟 언어의 단어만큼의 벡터로 변환되고, softmax를 통해 확률을 계산하여 예측하고, cross entropy loss를 구해 학습한다.

Transformer(Attention is all you need)

Regularization을 drop-out과 label-smoothing을 사용하였다.

	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 head와 d_k, d_v에 따른 성능차이

(B)-d_K의 크기에 따른 성능의 차이

(C), (D)-모델의 크기별 성능 차이

(E)-학습된 positional 임베딩을 사용할 경우

Transformer(Attention is all you need)

Self-attention을 사용한 모델의 장점

RNN은 입력을 순차적으로 받고 순차적 연산을 수행하는 반면,

Self-attention은 Sequence를 한번에 입력을 받으므로 순차적인 연산이 적어지고, 병렬적인 연산의 효율이 좋다.

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

참고자료

Transformer 설명이미지(Blossomindy's Research Blog)

[Paper Review] Attention is All You Need (Transformer)

Jay Alammer – transformer