Attention Is All You Need

소프트웨어학부 2017012333 이수아

CONTETNS

배경지식

- Sequential computation
- Long-term dependency problem
- Attention mechanism

연구주제

요약

결론

배경지식

Sequential computation

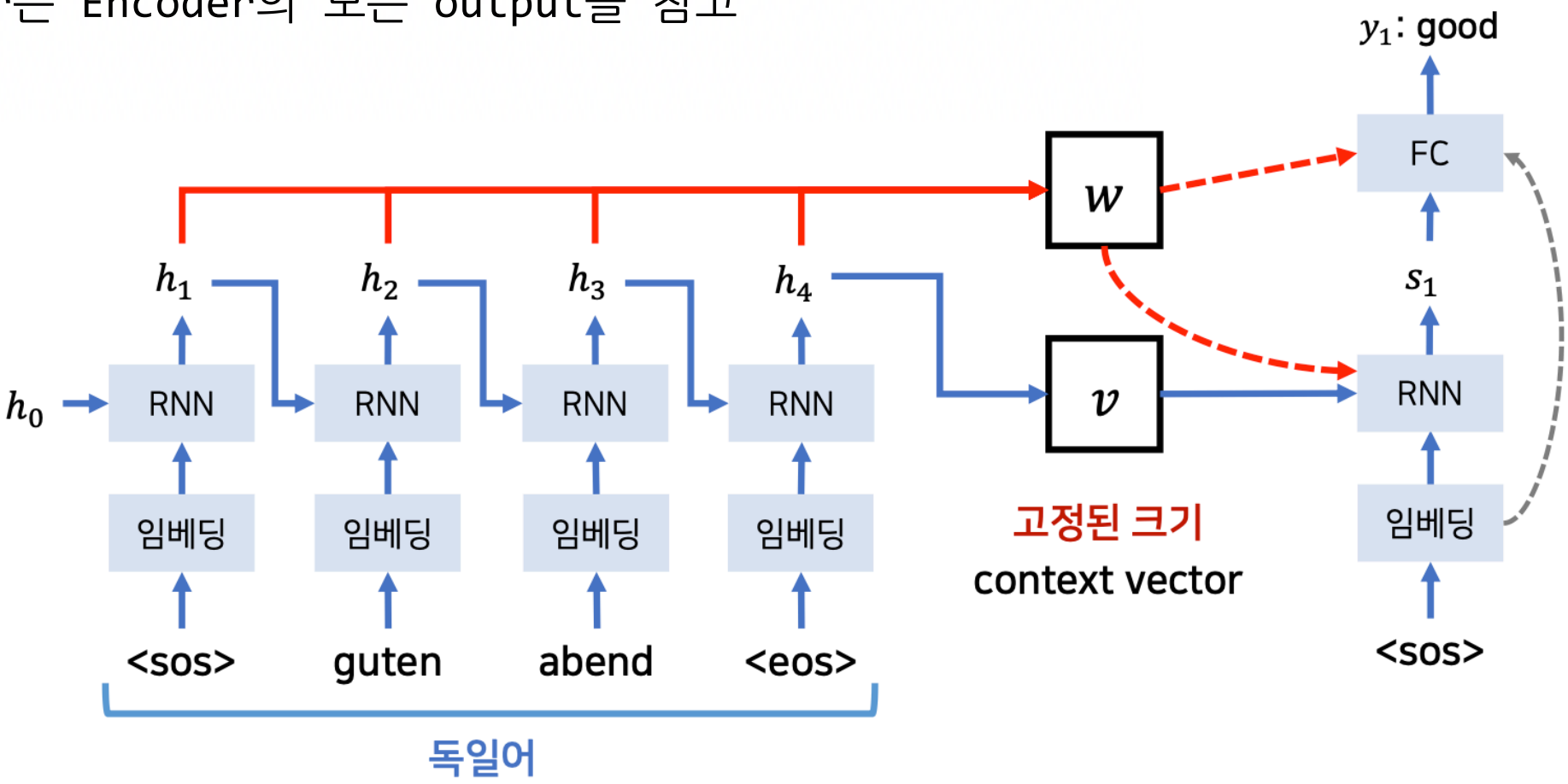
sequence to sequence → Encoder-Decoder 구조의 RNN

Long-term dependency problem

어떤 정보와 다른 정보 사이의 거리가 멀 때 해당 정보를 이용하지 못하는 것

Attention mechanism

Decoder는 Encoder의 모든 output을 참고



연구주제

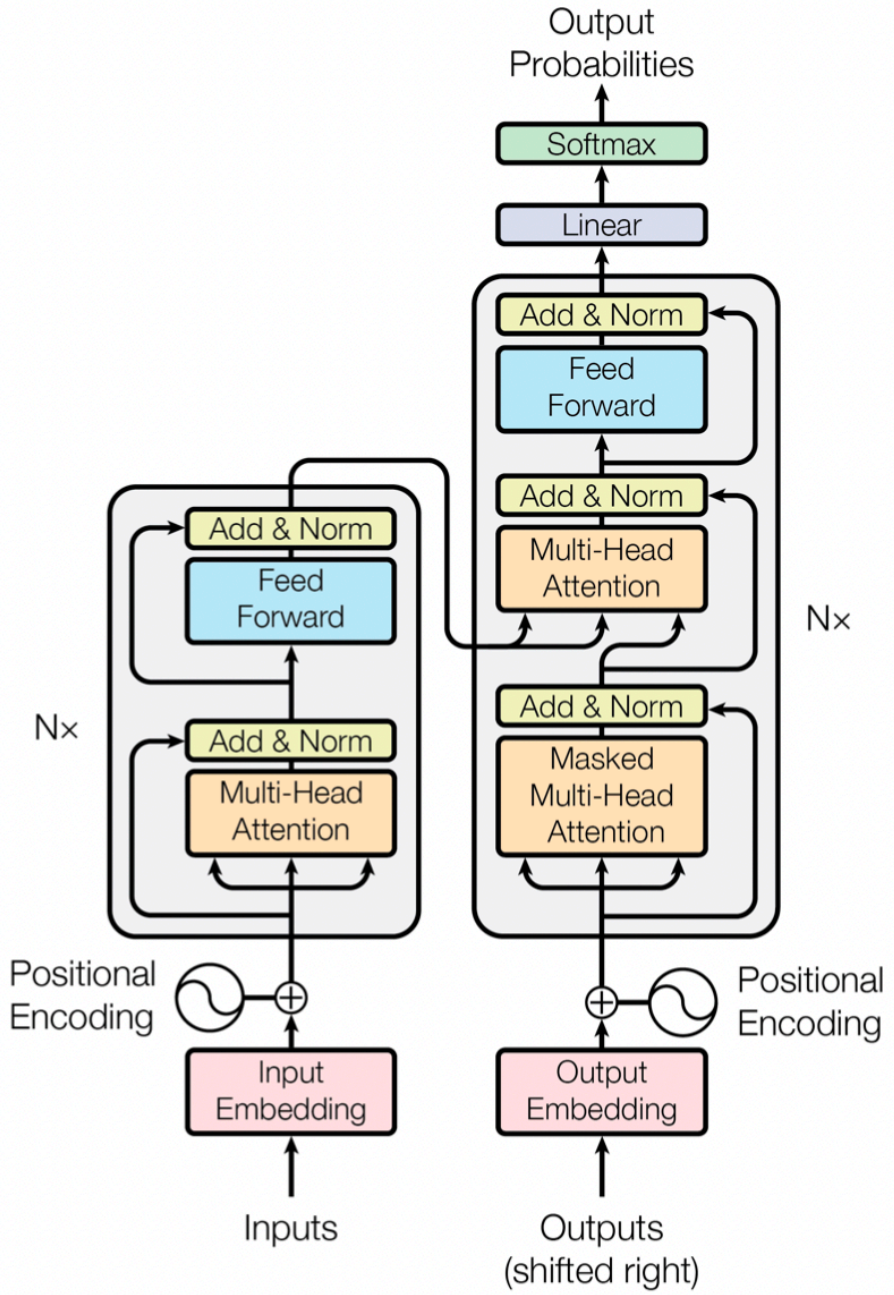
Transformer

RNN, CNN 사용 X

➡ Positional Encoding 사용

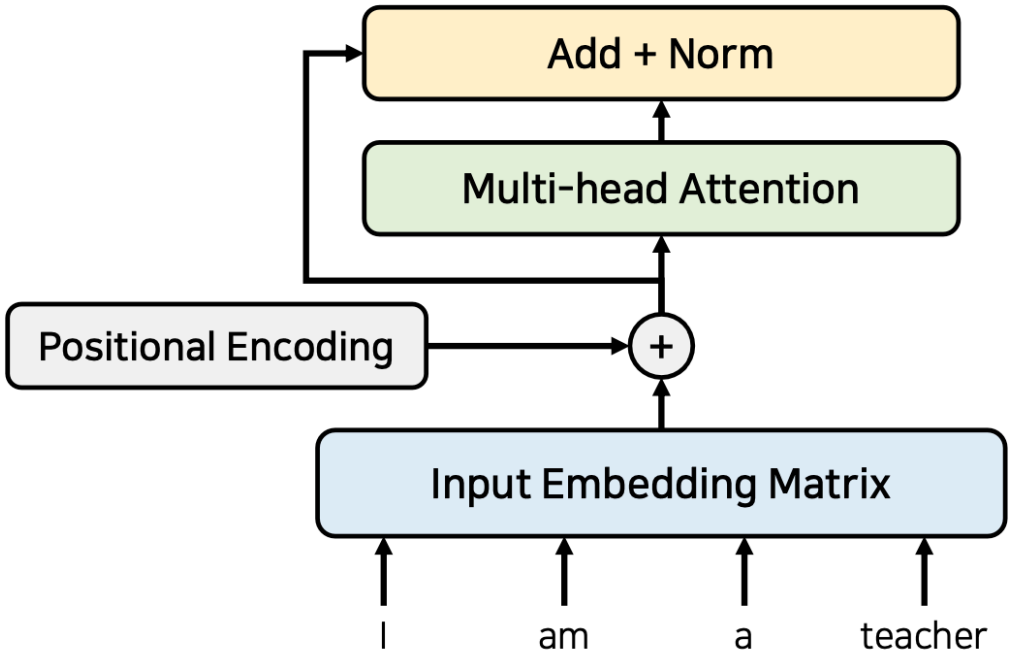
Encoder – Decoder로 구성

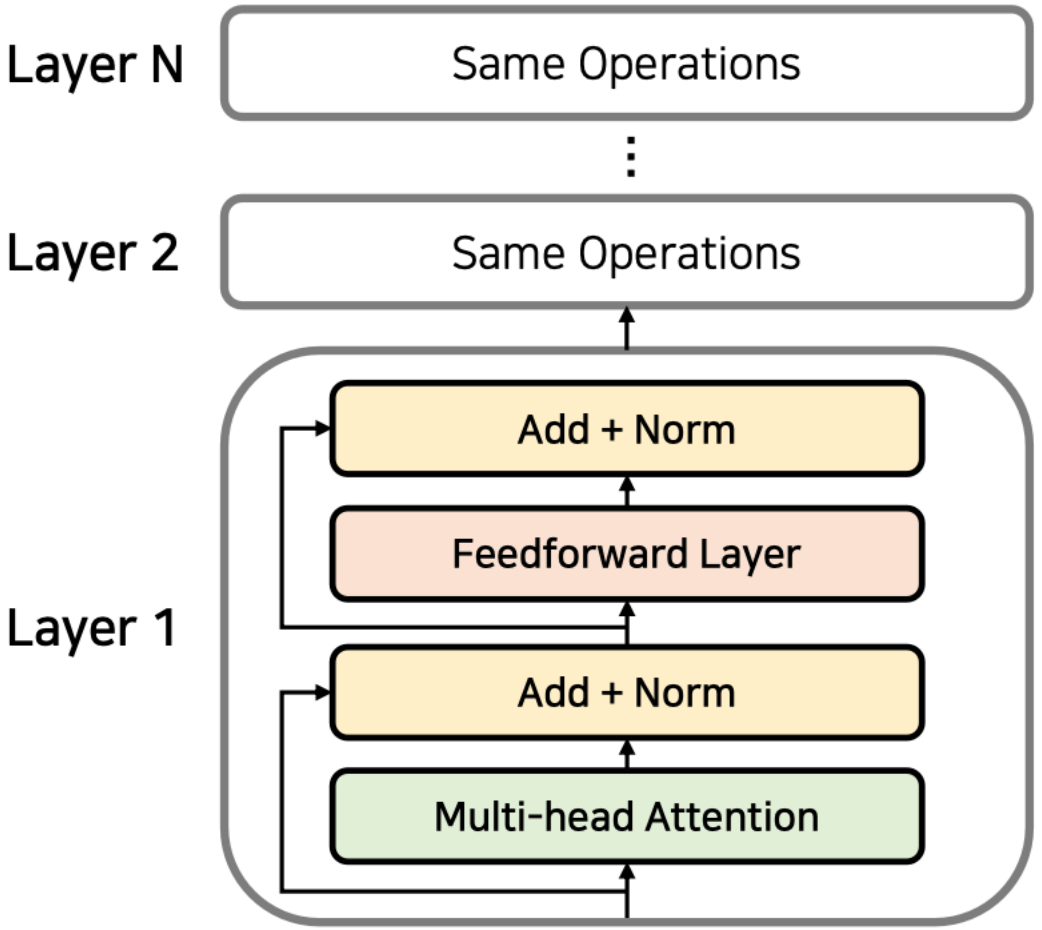
➡ Attention 과정 반복



요약

Transformer: Encoder

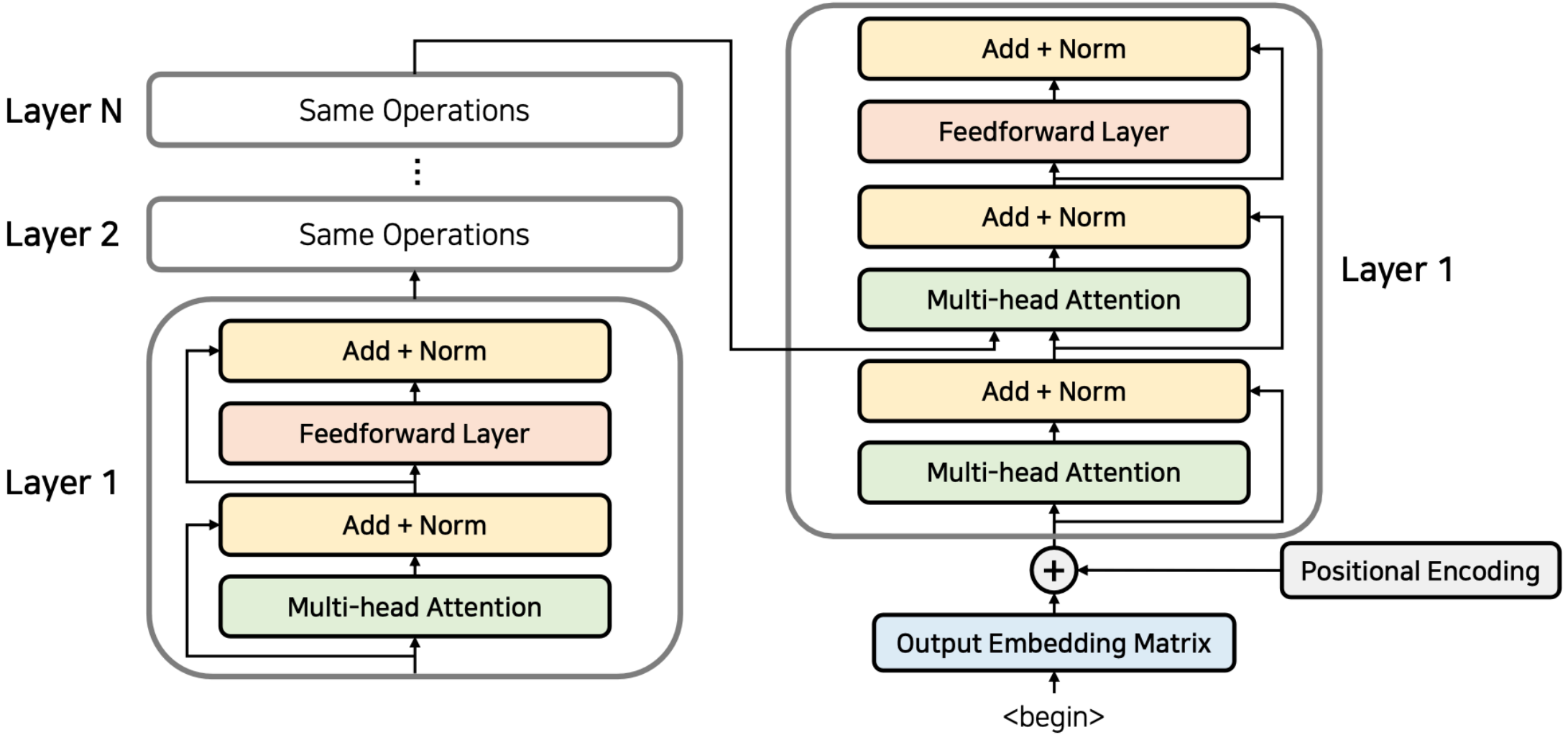




Residual connection 사용

요약

Transformer: Encoder & Decoder



요약

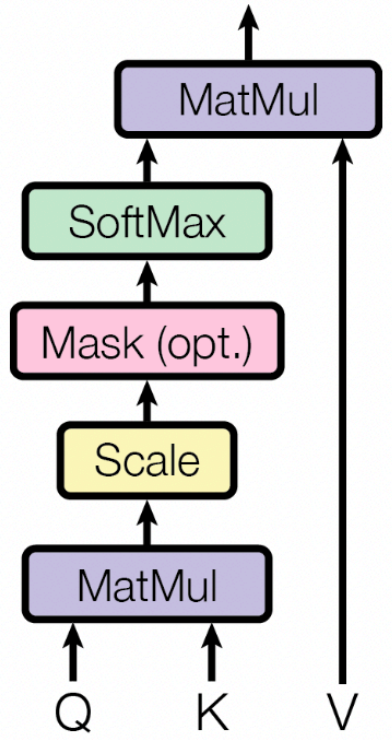
Multi-Head Attention

Attention 3가지 input
- Q: Query
- K: Key
- V: Value

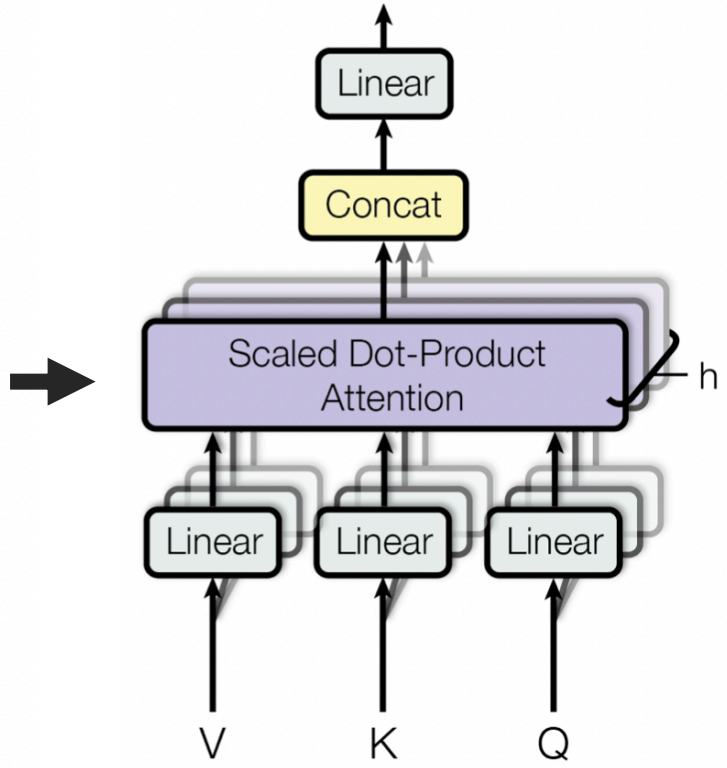
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

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

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



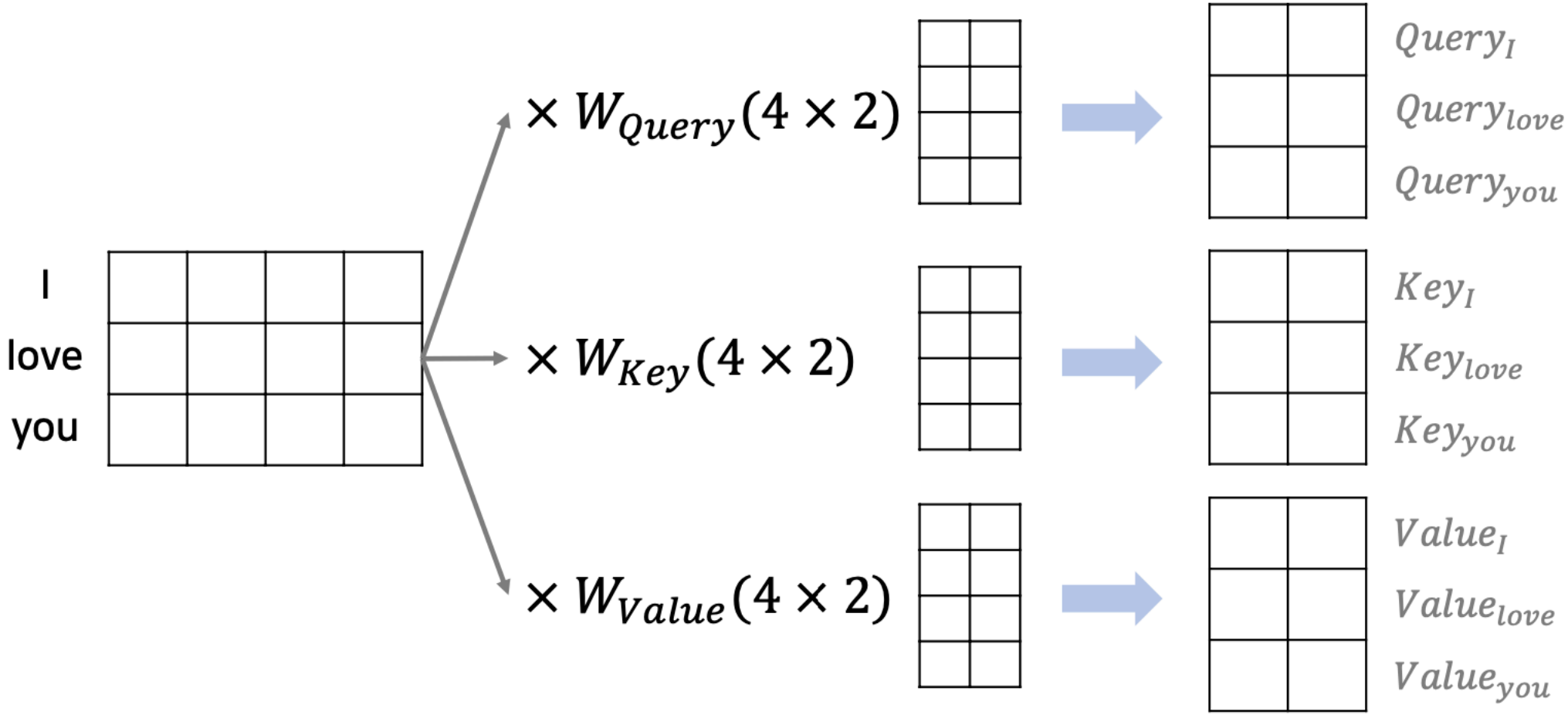
Scaled Dot-Product Attention



Multi-Head Attention

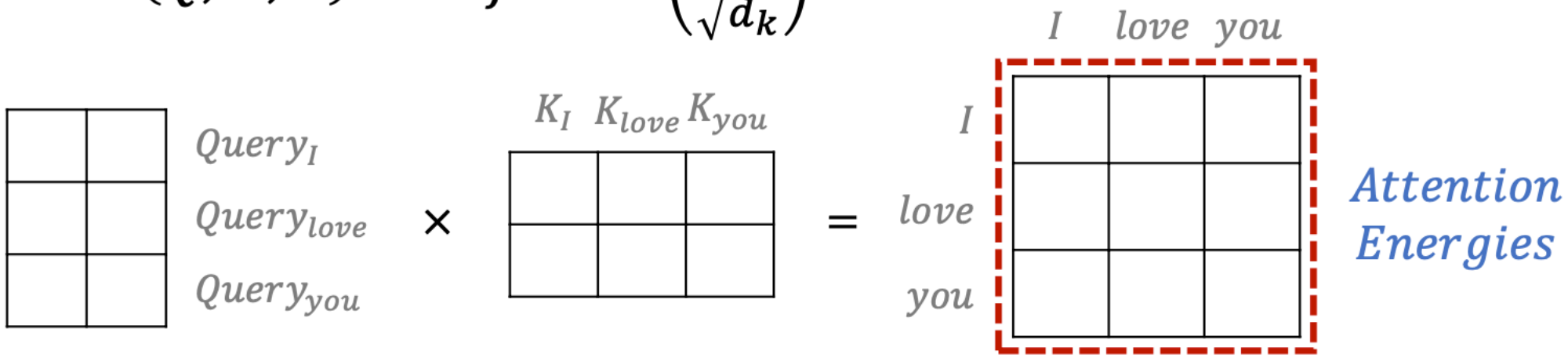
요약

Transformer

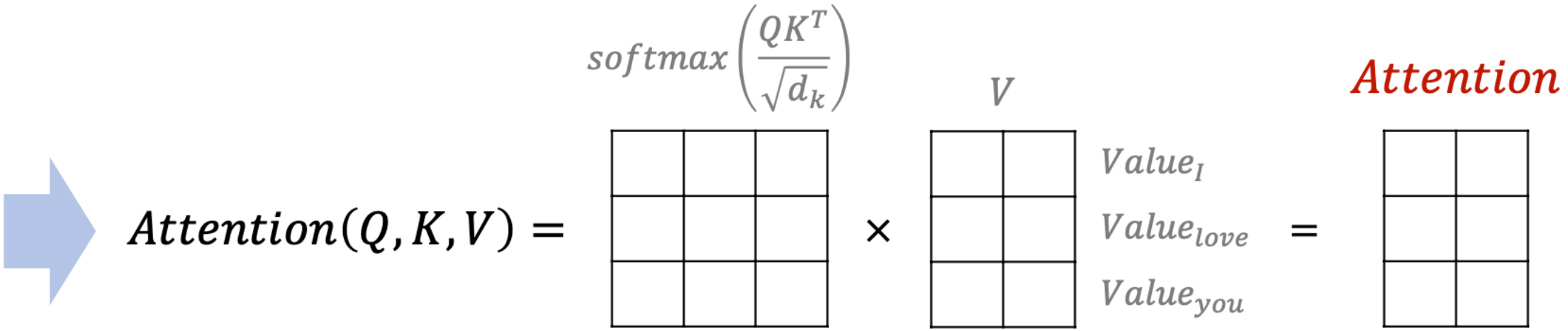


요약

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



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



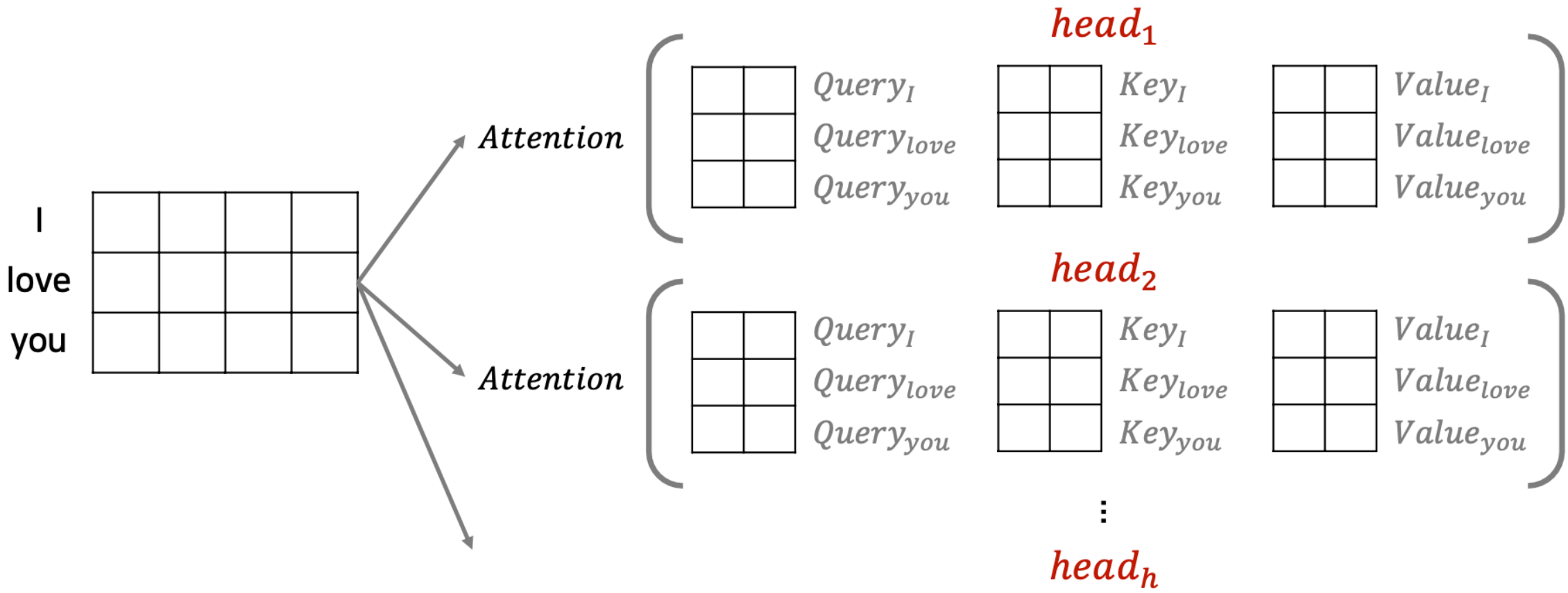
요약

Mask Matrix

마스크 값으로 **음수 무한**의 값을 넣어 $softmax$ 함수의 출력이 0%에 가까워지도록 한다.

요약

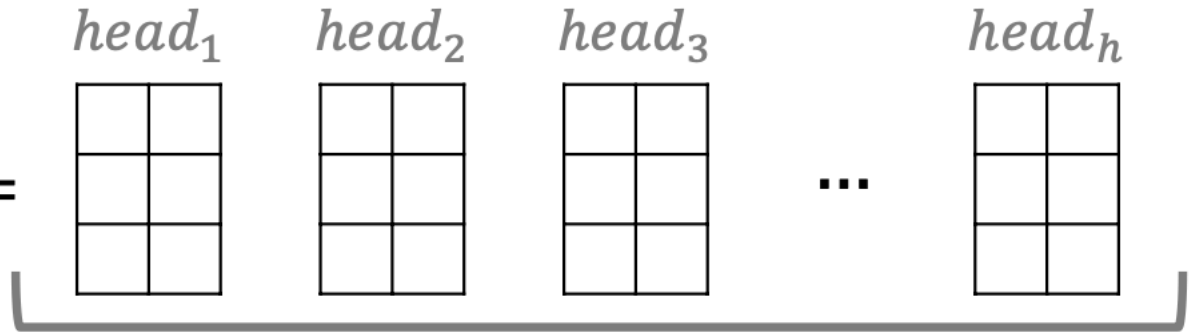
$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W^O$$



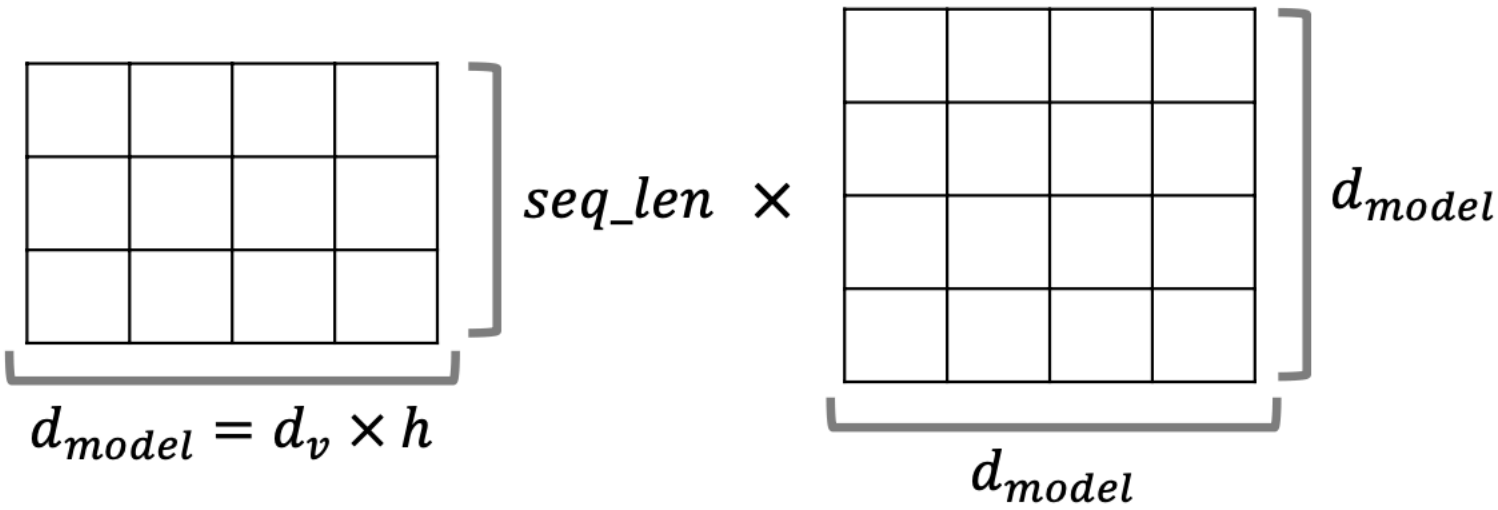
요약

$MultiHead(Q,K,V)$를 수행한 뒤에도 차원(dimension)이 동일하게 유지

$$Concat(head_1, \dots, head_h) =$$



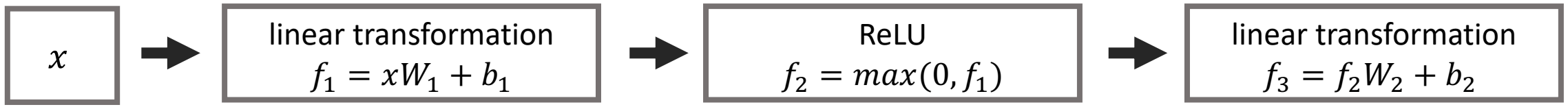
$$MultiHead(Q,K,V) =$$



요약

Position-wise Feed-Forward Networks

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



결론

Training

- Adam optimizer
- $lrate = d_{model}^{-0.5} \cdot \min\left(step_{num}{}^{-0.5}, step_{num} \cdot warmup_{steps}{}^{-1.5}\right)$
- Residual Dropout
- Label Smoothing

결론

Experiments

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

결론

Experiments

	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

# of heads

# of key dimension

model size

Dropout

결론

Conclusion

- Transformer는 recurrence를 이용하지 않고도 빠르고 정확하게 sequential data를 처리할 수 있는 model.
- Encoder와 Decoder에서 attention을 통해 query와 가장 밀접한 연관성을 가지는 value를 강조할 수 있고 병렬화가 가능해진 것입니다.
- 기계번역 뿐만 아니라 다양한 task에 대해 적용가능성 높다.

참고

References

- https://pozalabs.github.io/transformer/
- https://github.com/ndb796/Deep-Learning-Paper-Review-and-Practice

끝