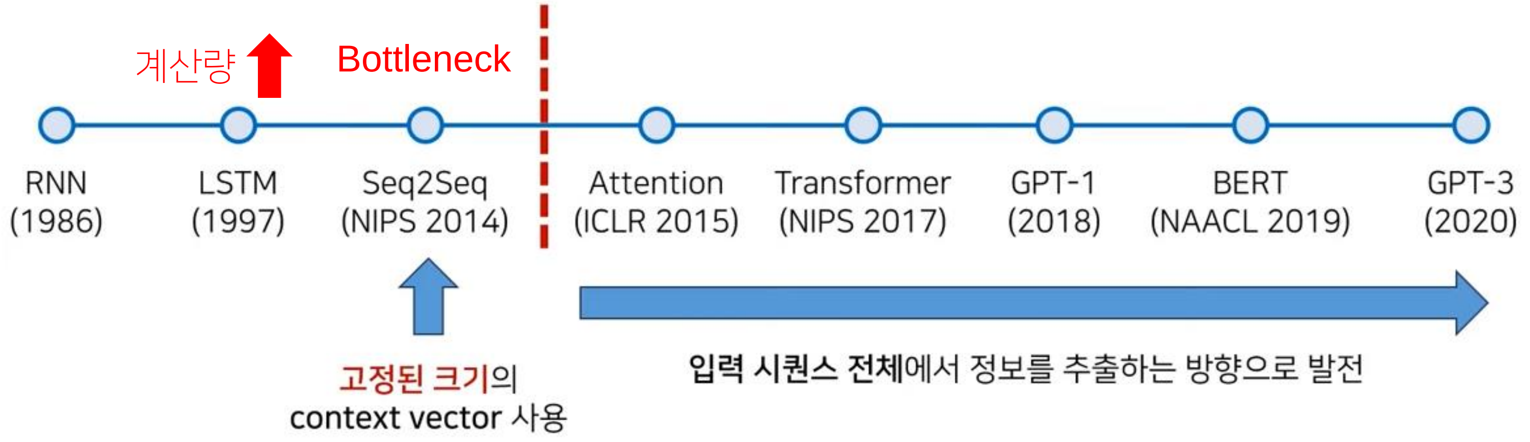
₊·* 2021 SMARCLE Paper Review ˎˊ˗ ·*

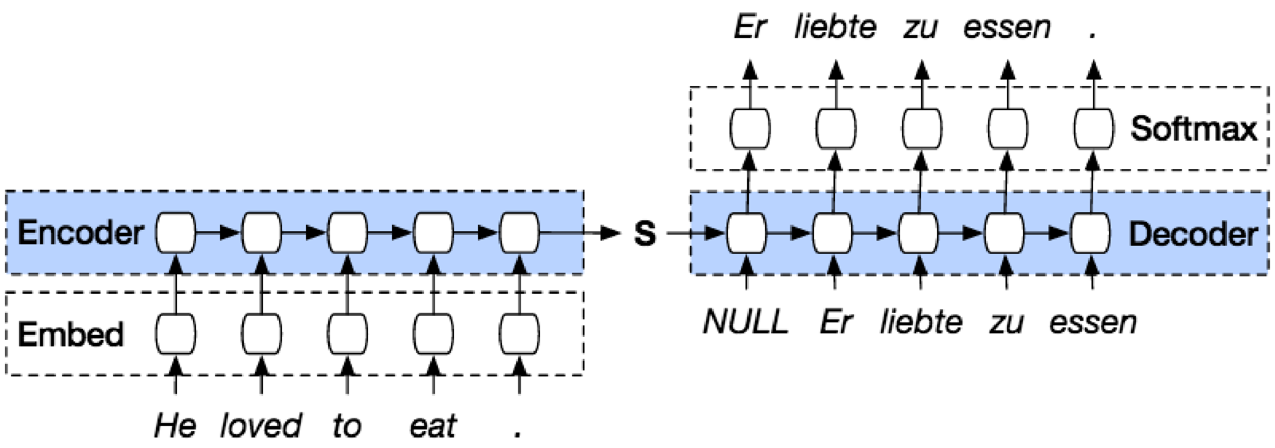
Transformer - Attention Is All You Need (NIPS 2017)

지능기전공학부 무인이동체공학전공
송혜원

SMARCLE

Seq2Seq의 문제점





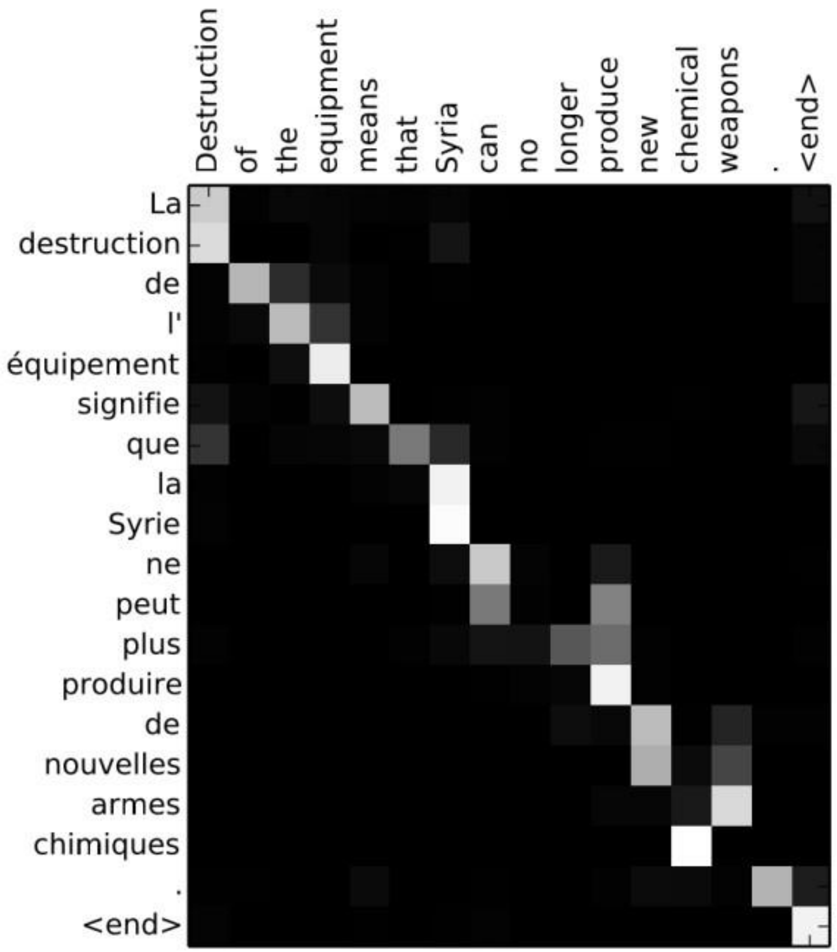
Part 0.

Attention

이전엔 입력을
하나의 context vector로 압축

Attention 기법은
문장 자체를 입력으로!

어떤 단어가 어떤 단어와
가장 연관성이 있는지 알 수 있게 됨



Part 1.

Transformer_Input

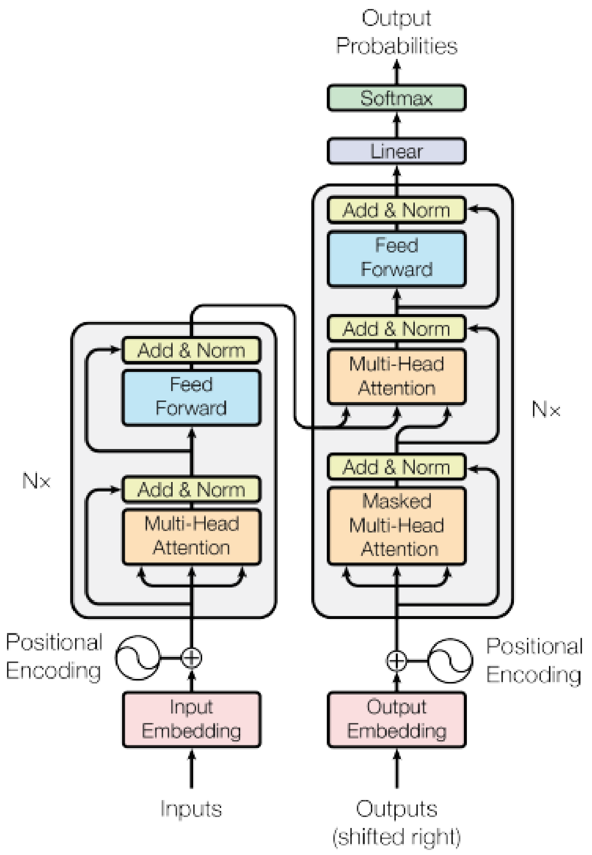


Figure 1: The Transformer - model architecture.

Dog loves potato

Dog				…				
loves				…				
potato				…				

512

문장 전체를 통째로 넣다보니 단어의 위치 정보를 알 수 없음

Positional Encoding 통해 위치 정보를 따로 더해줌!

Part 2.

Transformer_Encoder Self-Attention

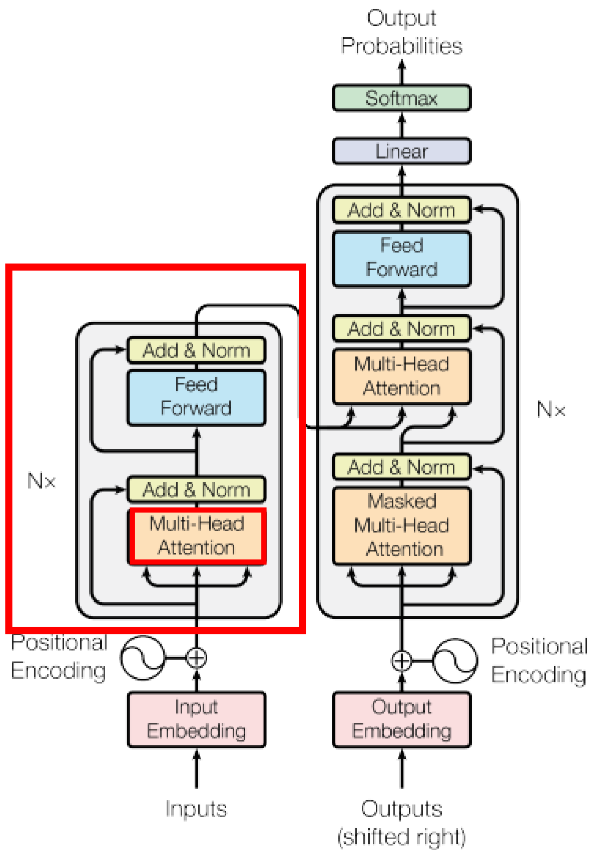
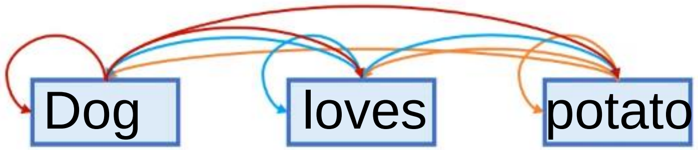


Figure 1: The Transformer - model architecture.

Encoder Self-Attention:



Transformer_Encoder

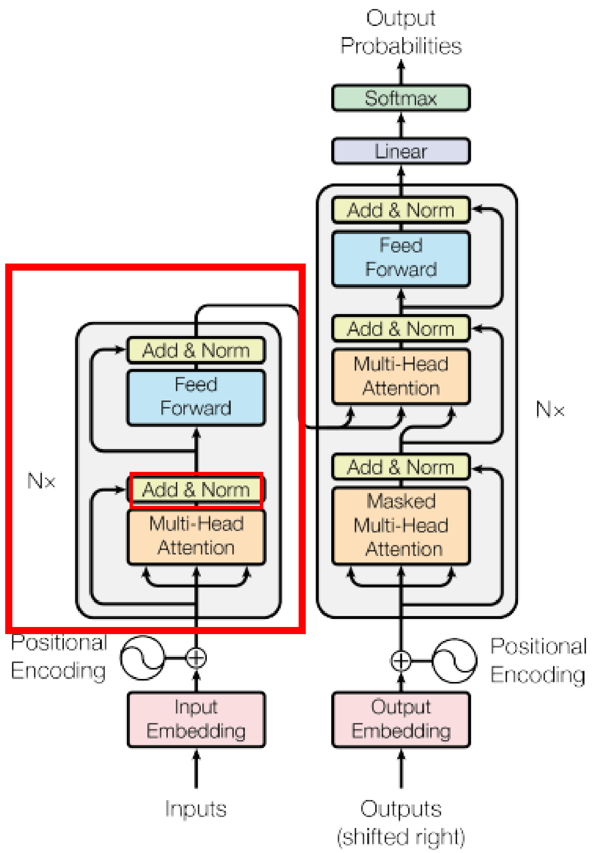
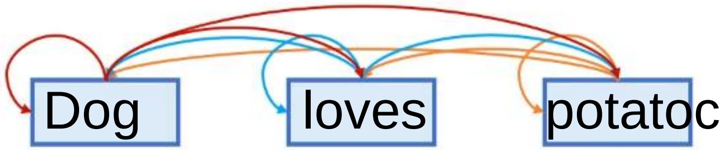


Figure 1: The Transformer - model architecture.

Encoder Self-Attention:



Residual learning

- 특정 layer를 건너뜀
- Gradient vanishing 완화
- Global optima 잘 찾을 수 있음

Transformer_Encoder

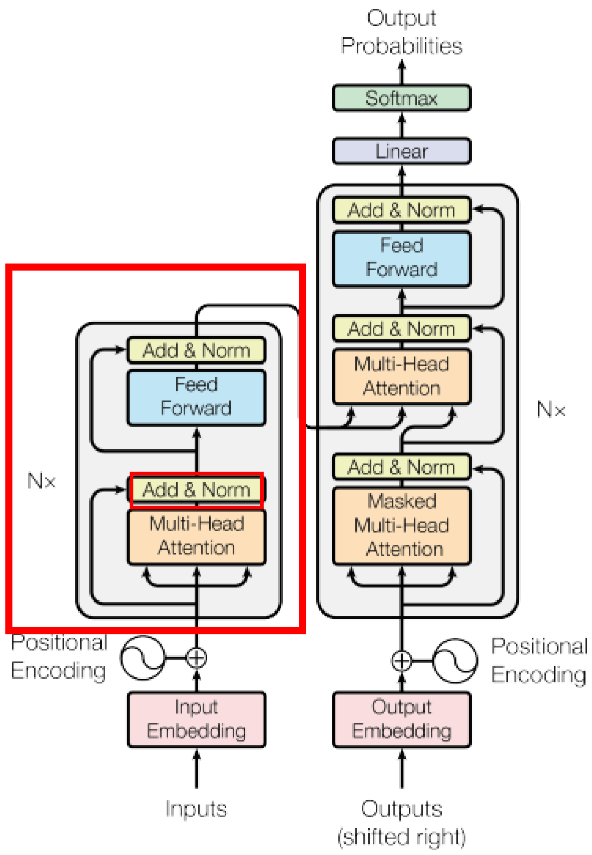
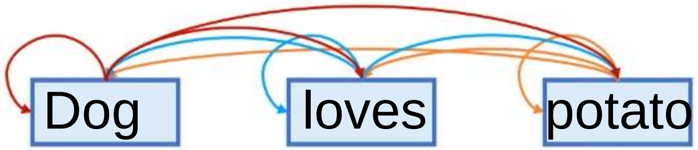


Figure 1: The Transformer - model architecture.

Encoder Self-Attention:



Residual learning

특정 layer를 건너뜀

Gradient vanishing 완화

Global optima 잘 찾을 수 있음

인풋의 다양한 특징 attention 위해 n개의 layer 병렬로 쌓음

Transformer_Masked Decoder Self-Attention

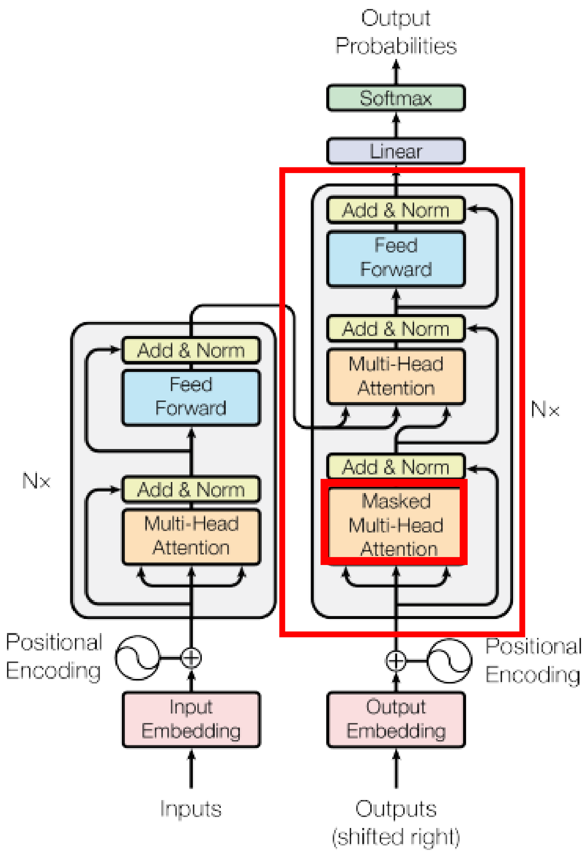


Figure 1: The Transformer - model architecture.

Masked Decoder Self-Attention: 강아지는 감자를 좋아해

Part 3.

Transformer_Masked Decoder Self-Attention

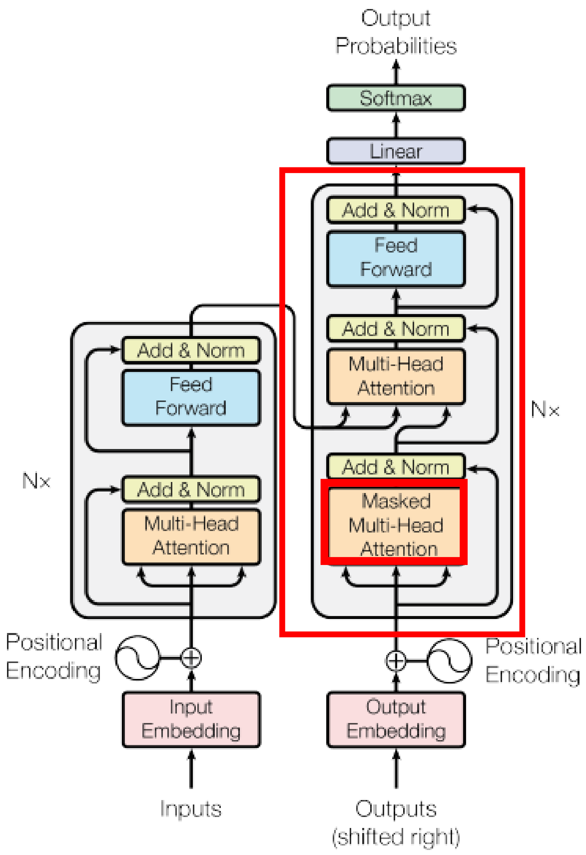
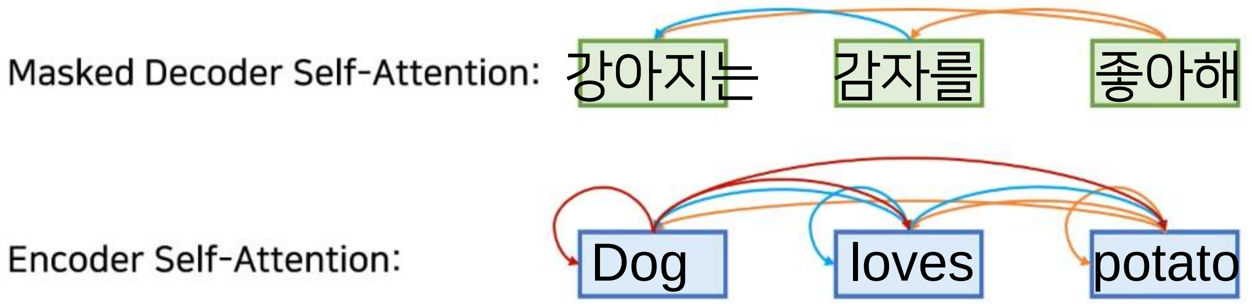


Figure 1: The Transformer - model architecture.



Part 3.

Transformer_Masked Decoder Self-Attention

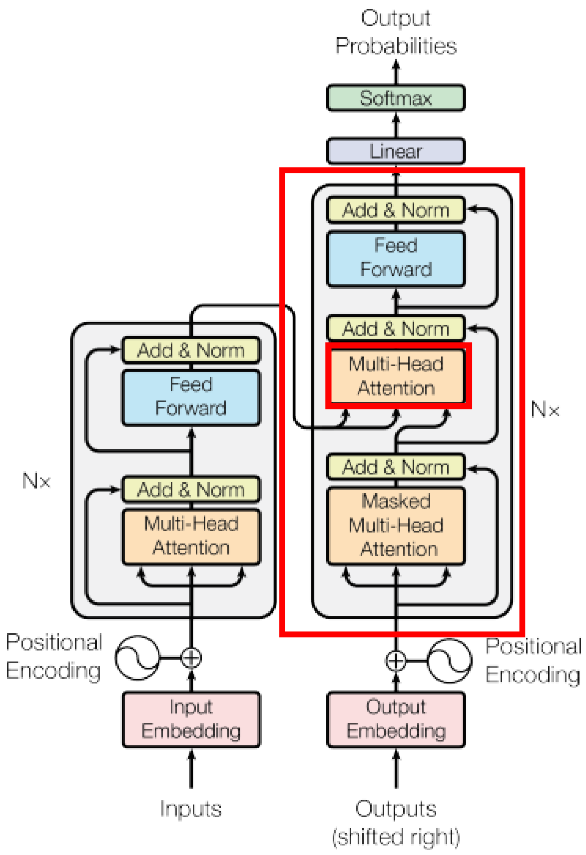
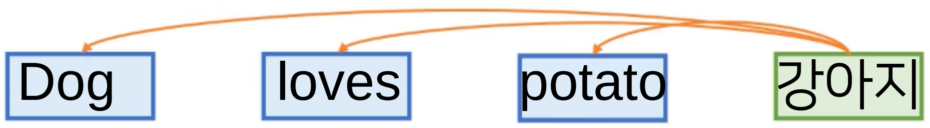


Figure 1: The Transformer - model architecture.

Encoder-Decoder Attention:



- 인코더 파트에서 나온 출력 결과를 디코더마다 적용하여 전적으로 활용

- 시퀀스가 끝날 때까지 반복

Part 4.

Transformer_Multi-Head Attention?

Q(Query) : 물어보는 주체 K(Key) : 대상 V(Value) : 가중치

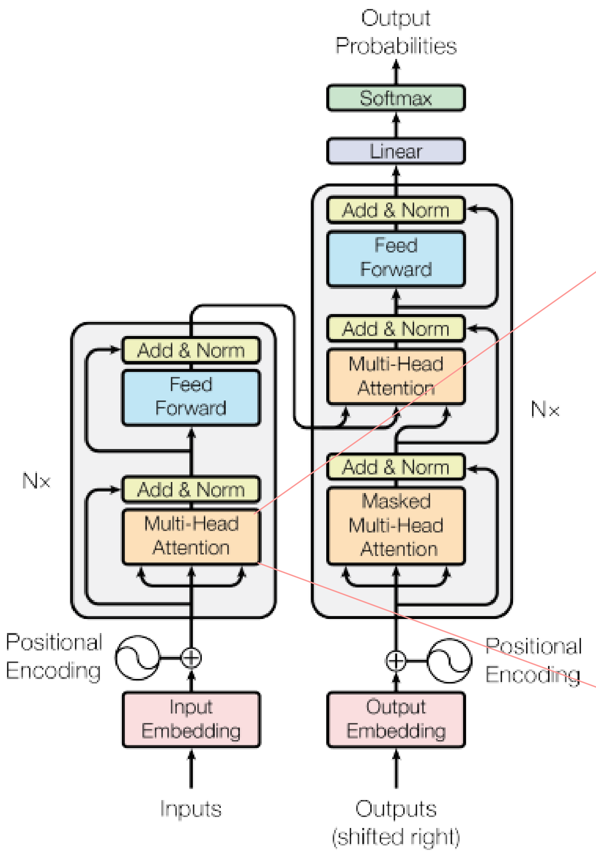
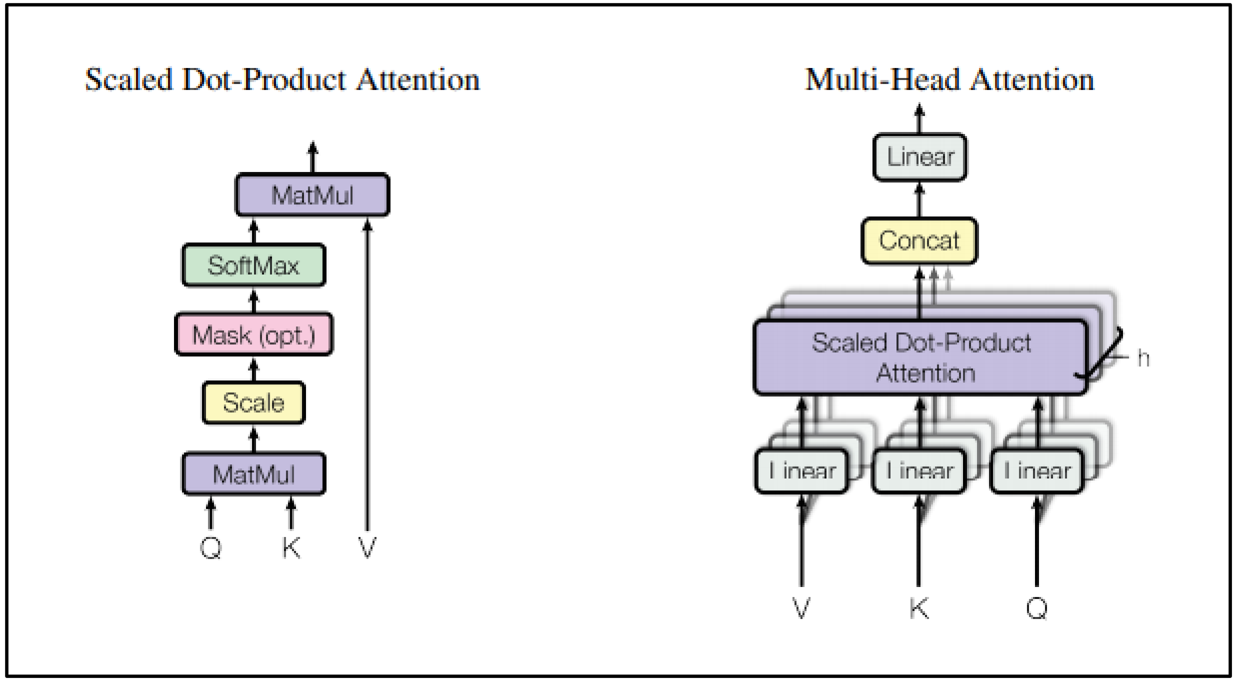


Figure 1: The Transformer - model architecture.



Part 4.

Transformer_Multi-Head Attention?

Q(Query) : 물어보는 주체   K(Key) : 대상   V(Value) : 가중치

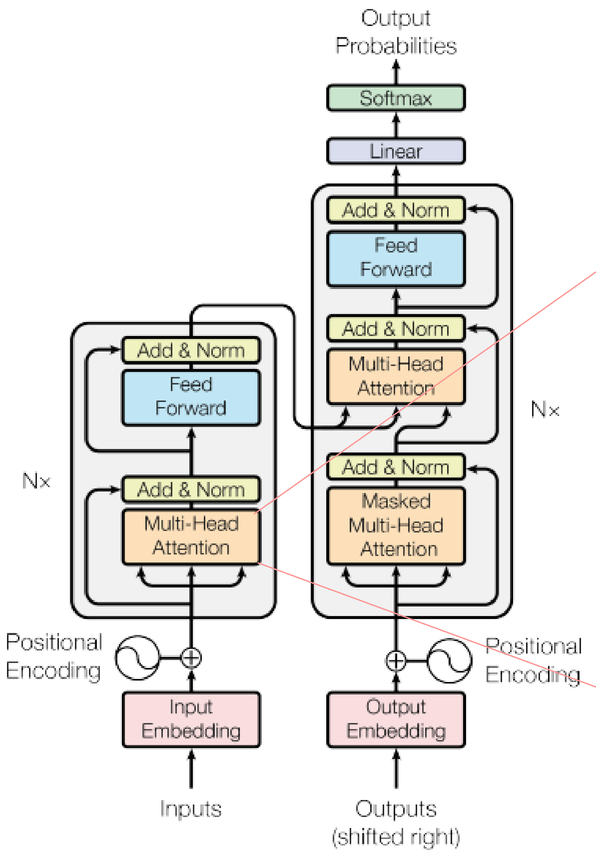
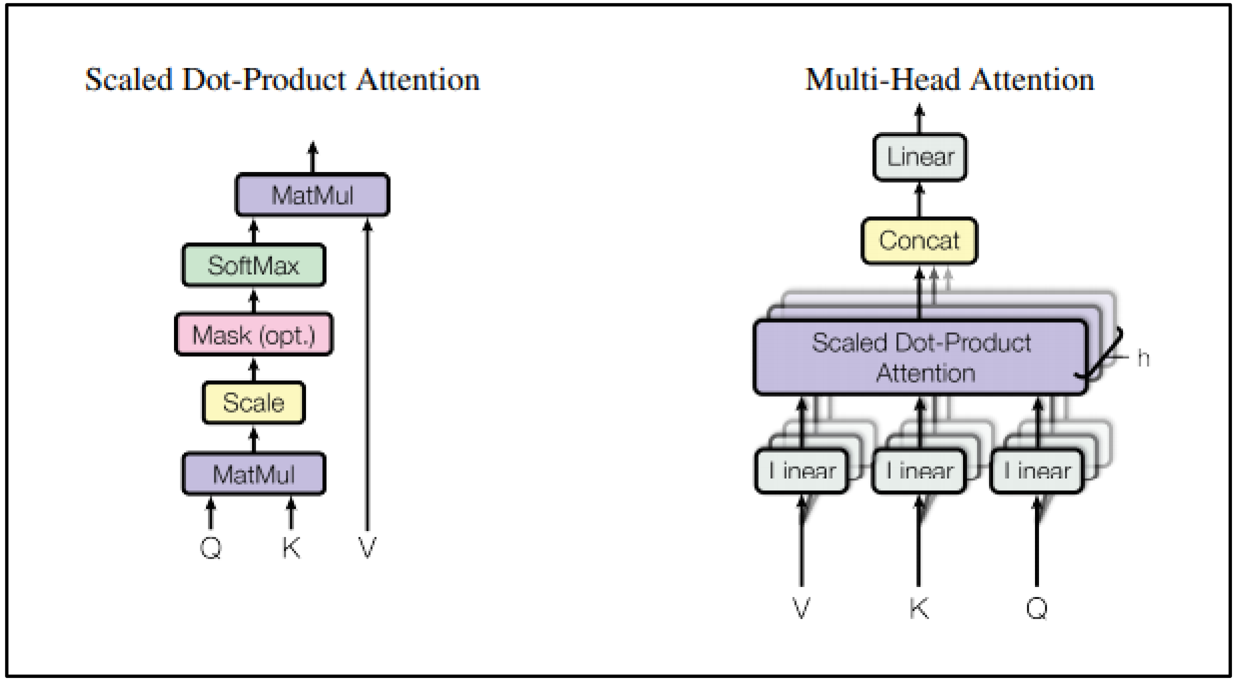
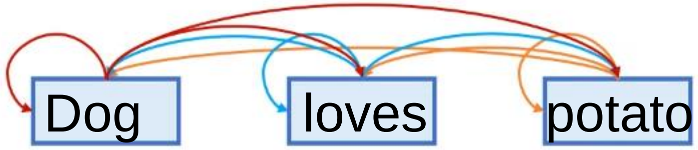


Figure 1: The Transformer - model architecture.



Encoder Self-Attention:



Part 4.

Transformer_Multi-Head Attention?

Q(Query) : 물어보는 주체   K(Key) : 대상   V(Value) : 가중치

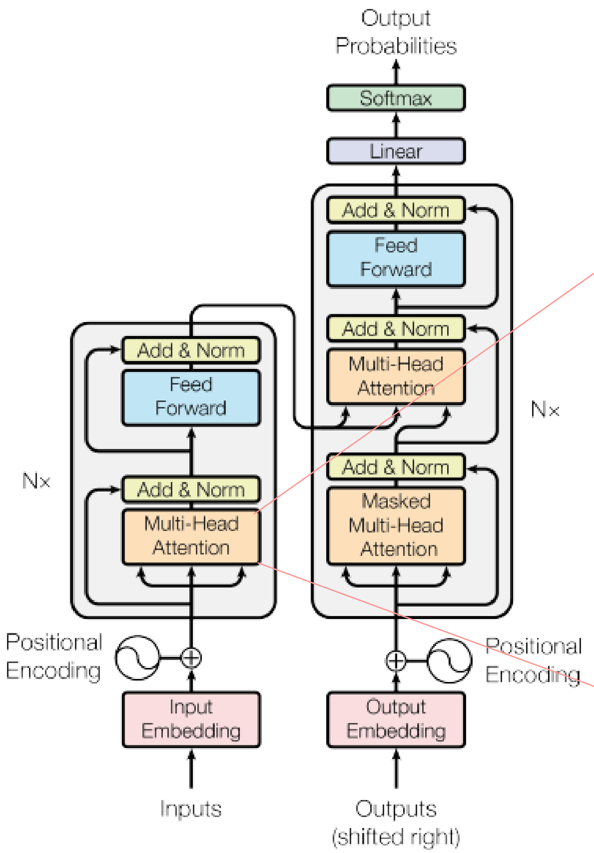
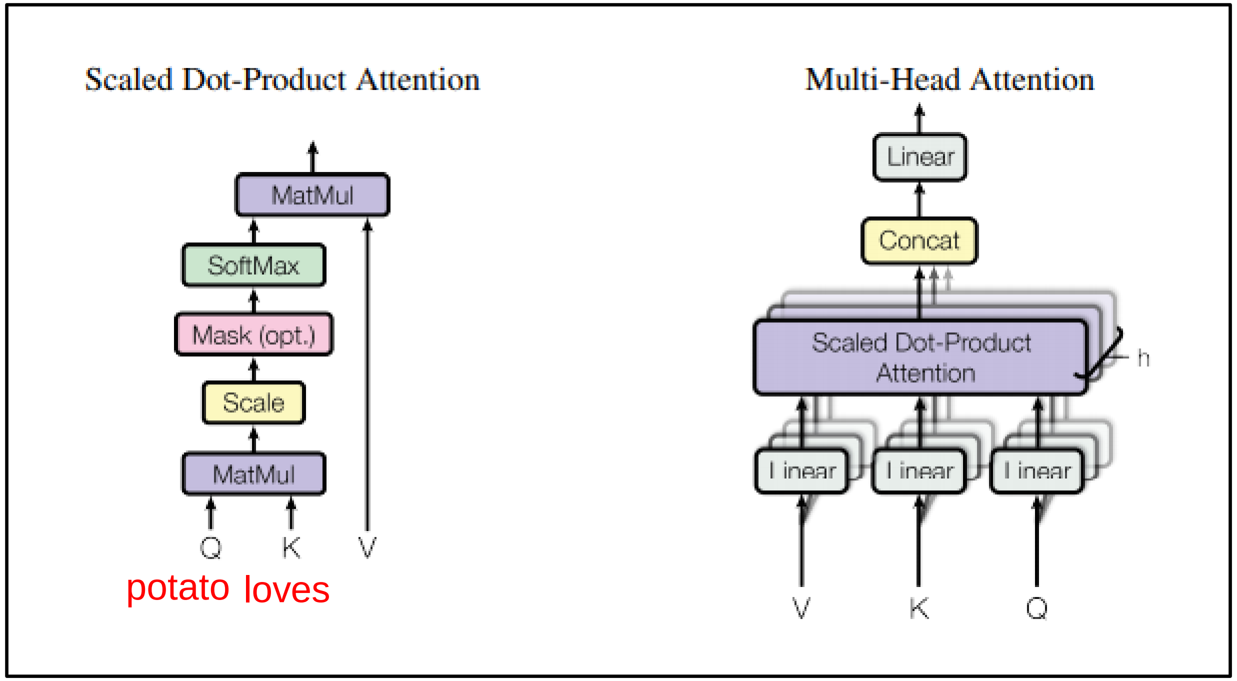
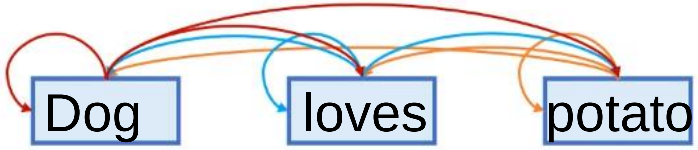


Figure 1: The Transformer - model architecture.



Encoder Self-Attention:



Part 4.

Transformer_Multi-Head Attention?

Q(Query) : 물어보는 주체   K(Key) : 대상   V(Value) : 가중치

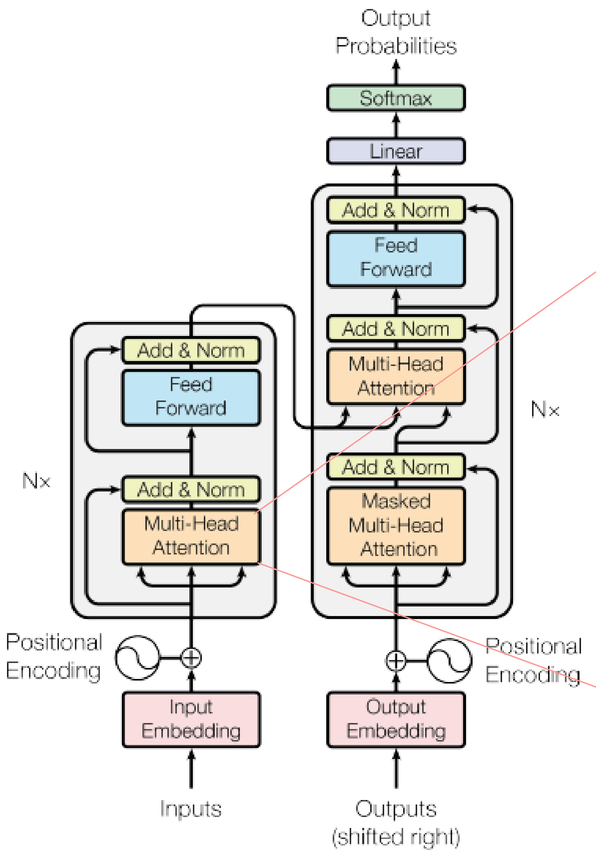
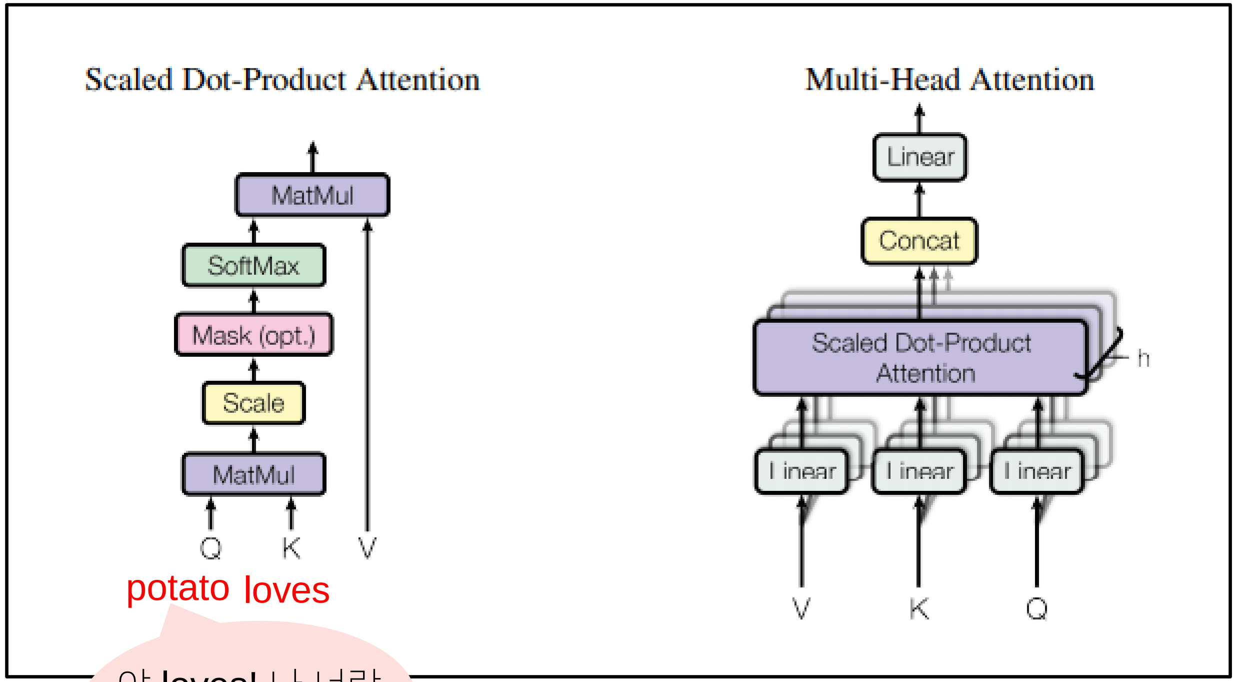
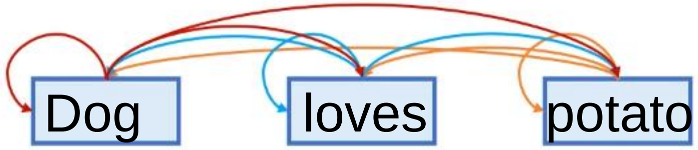


Figure 1: The Transformer - model architecture.



야 loves! 나 너랑
얼마나 관련있어?

Encoder Self-Attention:



Part 4.

Transformer_Multi-Head Attention?

Q(Query) : 물어보는 주체   K(Key) : 대상   V(Value) : 가중치

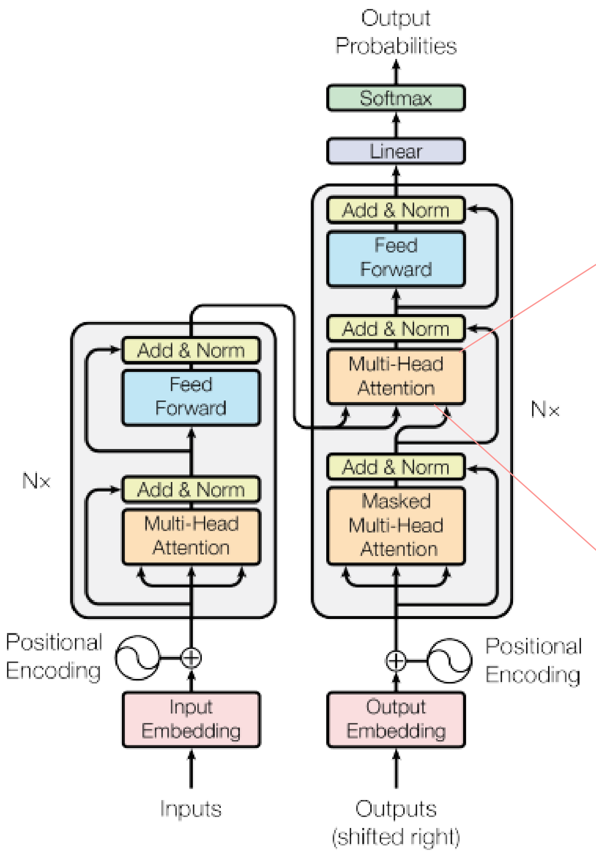
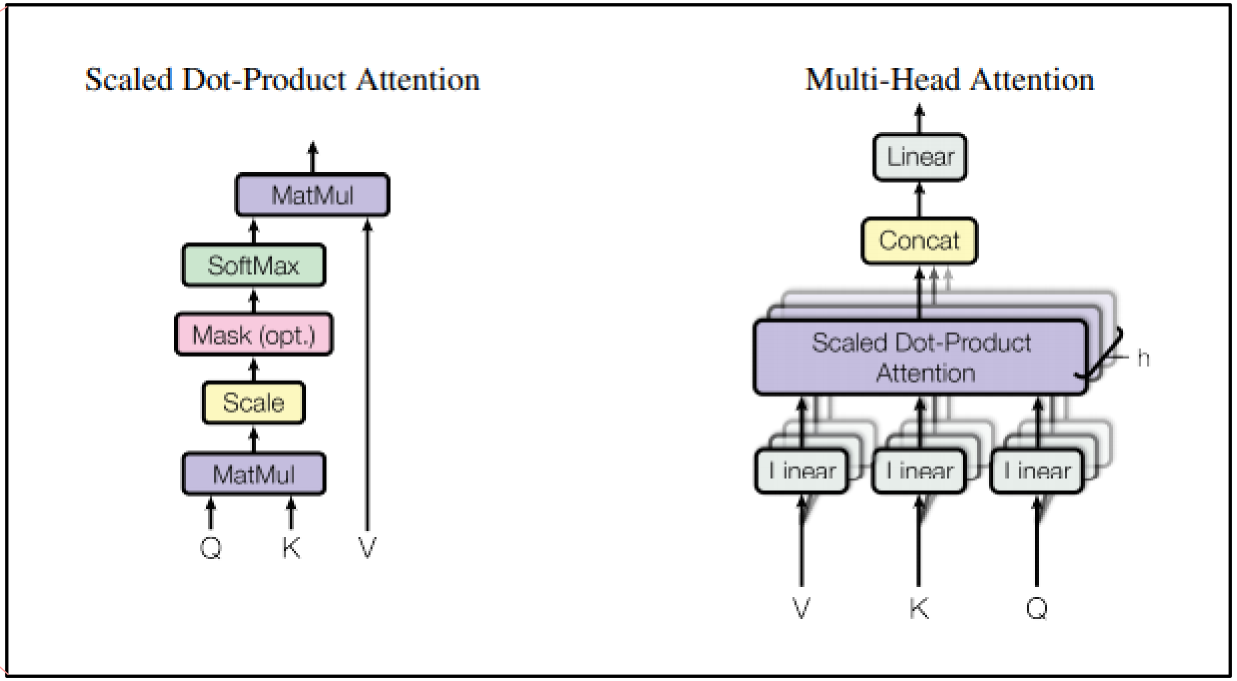
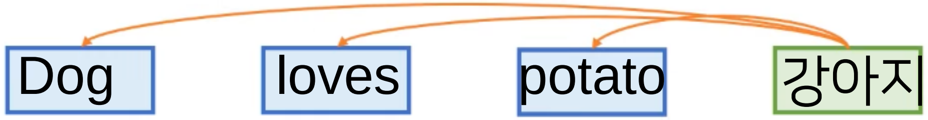


Figure 1: The Transformer - model architecture.



Encoder-Decoder Attention:



Transformer_Multi-Head Attention?

Q(Query) : 물어보는 주체 K(Key) : 대상 V(Value) : 가중치

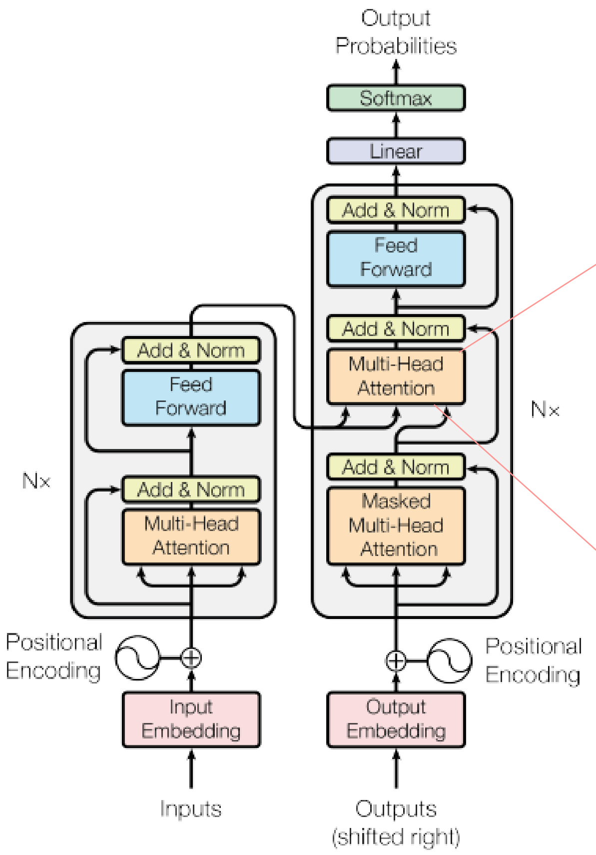
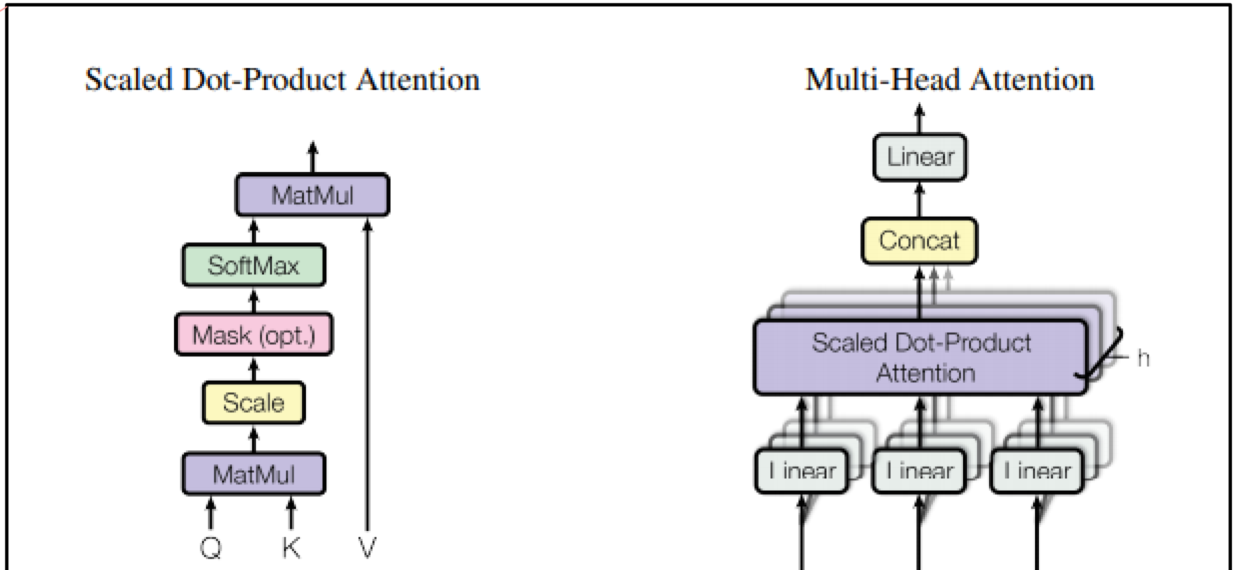


Figure 1: The Transformer - model architecture.



강아지 loves

야 loves! 나 너랑
얼마나 관련있어?

Encoder-Decoder Attention: Dog loves potato 강아지

Transformer_Multi-Head Attention?

Q(Query) : 물어보는 주체 K(Key) : 대상 V(Value) : 가중치

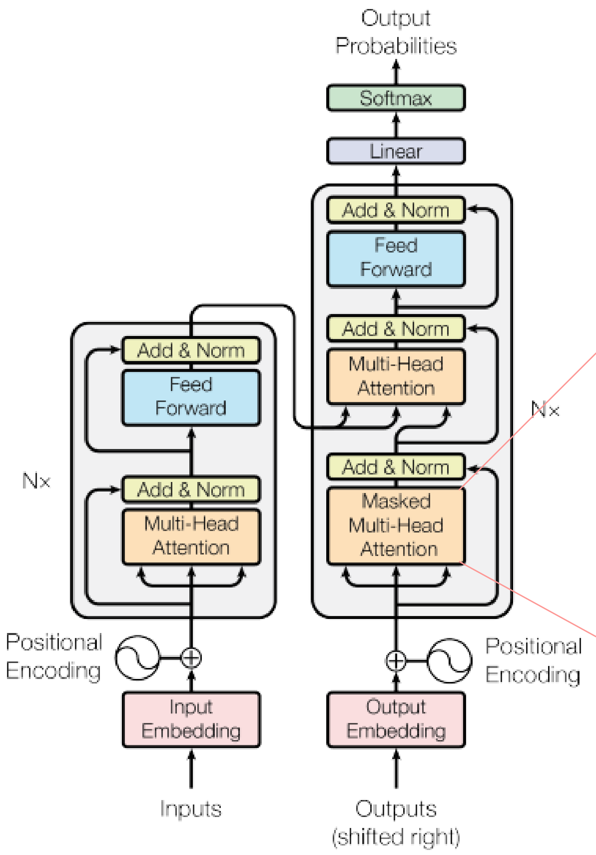
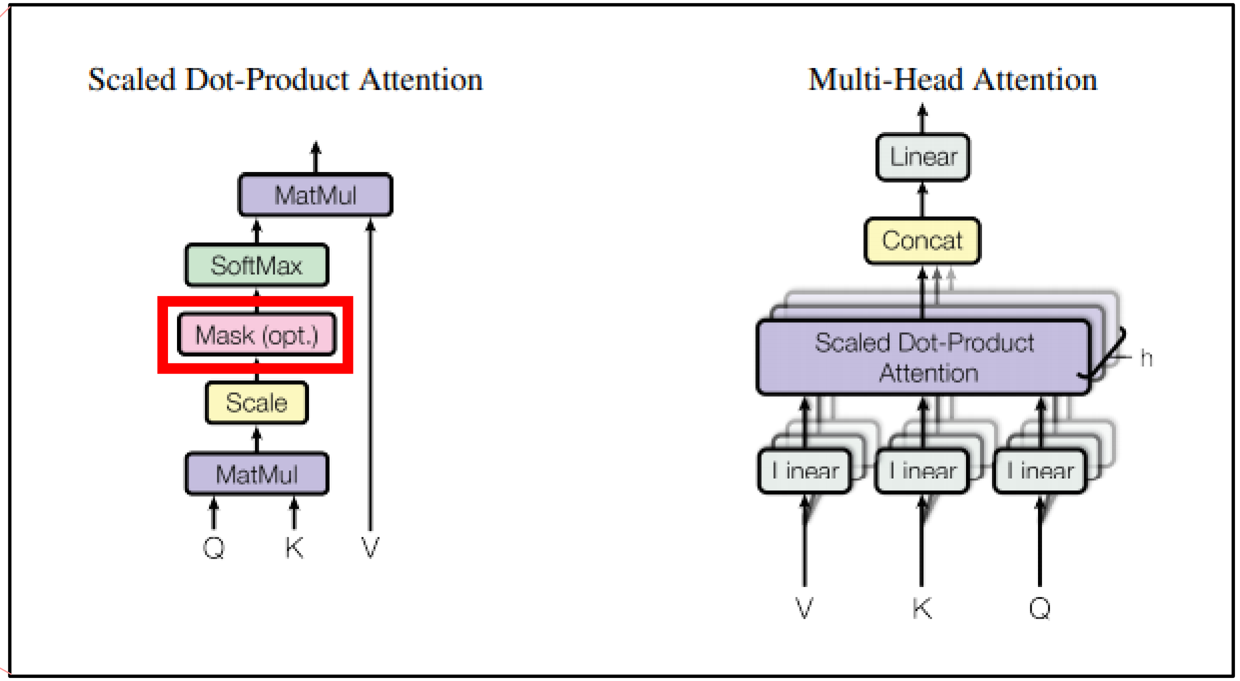


Figure 1: The Transformer - model architecture.



Masked Decoder Self-Attention: 강아지는 감자를 좋아해

Part 4.

Transformer_Multi-Head Attention?

Q(Query) : 물어보는 주체   K(Key) : 대상   V(Value) : 가중치

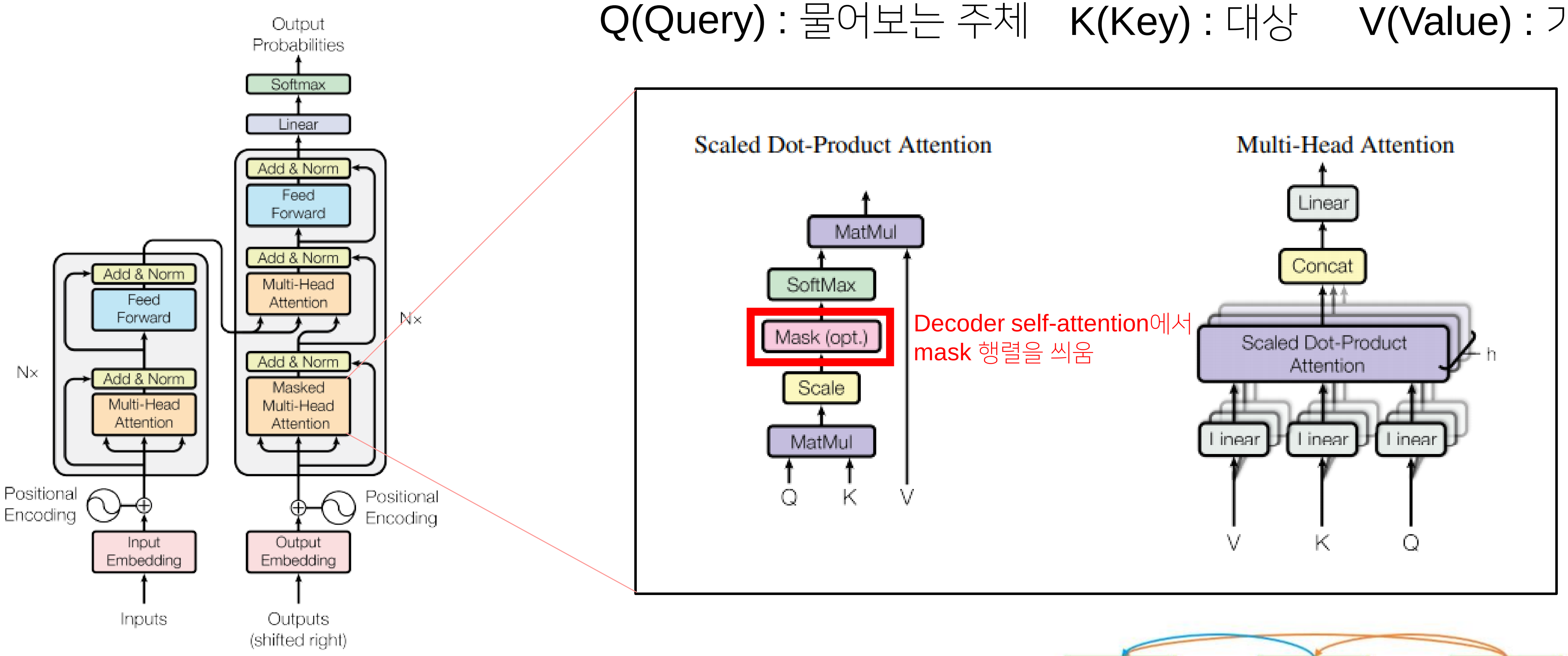


Figure 1: The Transformer - model architecture.

Masked Decoder Self-Attention: 강아지는 감자를 좋아해

Part 5.

Transformer_Positional Encoding

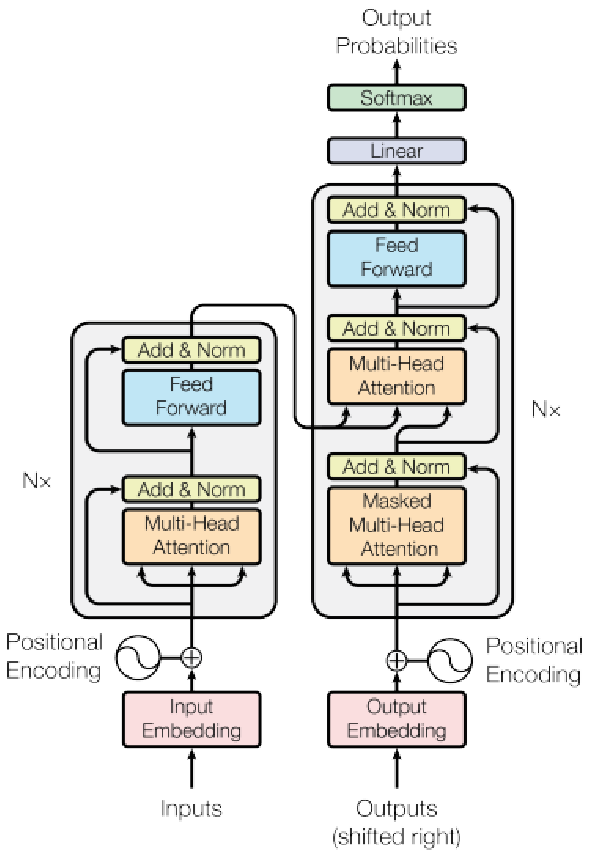


Figure 1: The Transformer - model architecture.

In this work, we use sine and cosine functions of different frequencies:

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

where pos is the position and i is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$. We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}.

We also experimented with using learned positional embeddings [9] instead, and found that the two versions produced nearly identical results (see Table 3 row (E)). We chose the sinusoidal version because it may allow the model to extrapolate to sequence lengths longer than the ones encountered during training.

- 본문에서는 주기함수를 이용해서 단어의 위치를 정해줌
- 이후 모델들은 학습이 가능한 임베딩 레이어를 넣어주는 추세

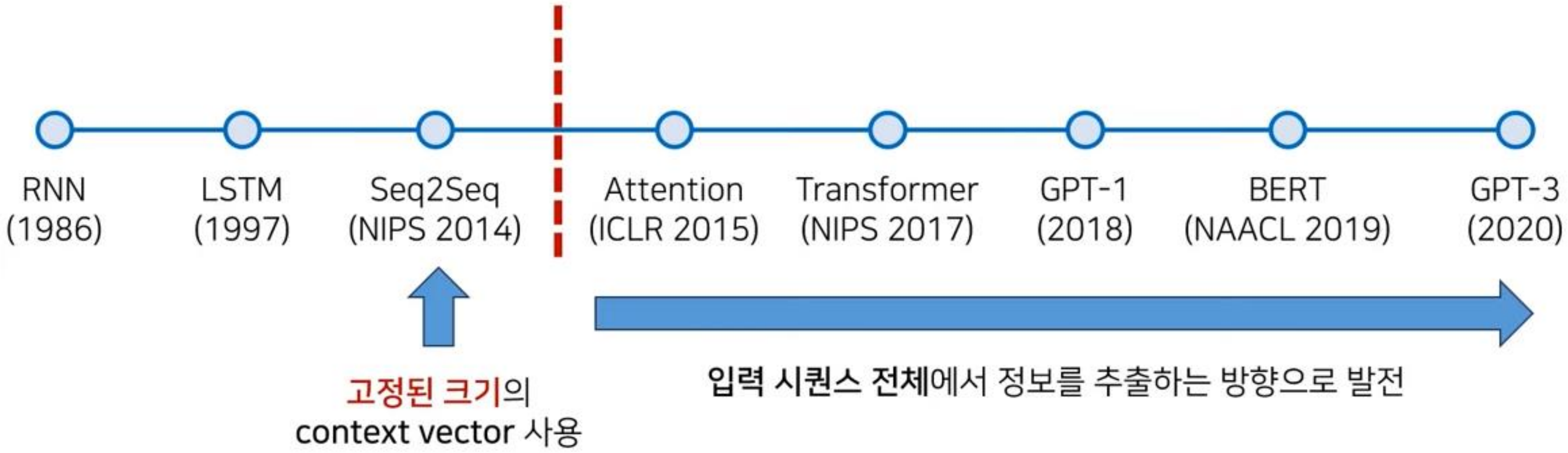
Transformer_Conclusion

SMARCLE

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 $\times 10^6$	
기본	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		
key dimension	(B)					16					5.16	25.1	58	↓수록 bad
						32					5.01	25.4	60	
model 크기	(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	↑수록 good
			1024			128	128				4.66	26.0	168	
				1024							5.12	25.4	53	
				4096							4.75	26.2	90	
dropout	(D)							0.0			5.77	24.6		
								0.2			4.95	25.5		good
									0.0		4.67	25.3		
									0.2		5.47	25.7		
p.e 학습	(E)	positional embedding instead of sinusoids									4.92	25.7		base랑 차이 거의 X
	big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213	

Transformer_TBC..



감사합니다 :-3