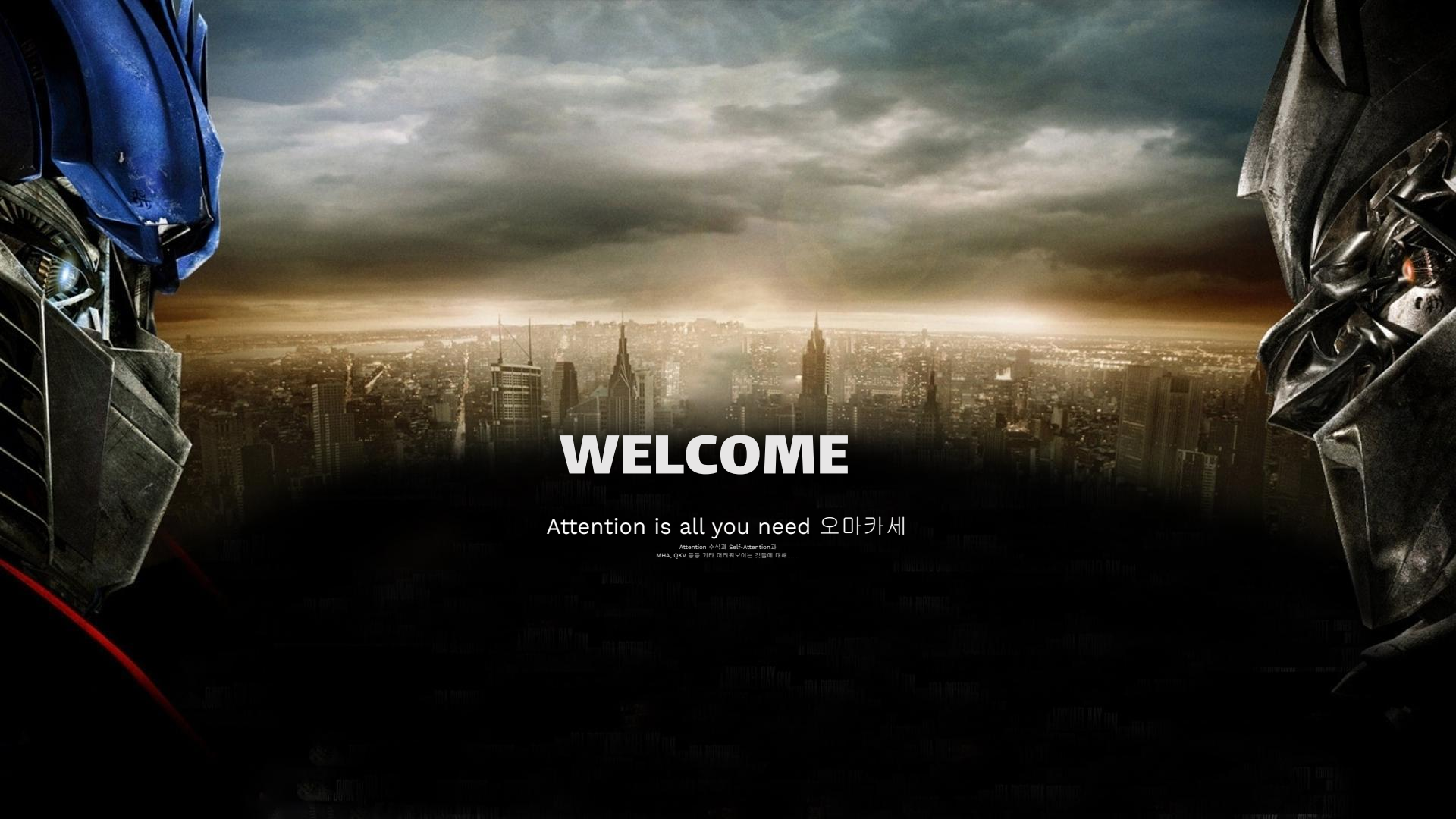

WELCOME
Attention is all you need 오마카세
Attention 수식과 Self-Attention과
MHA, QKV 등등 기타 여러워보이는 것들에 대해.....

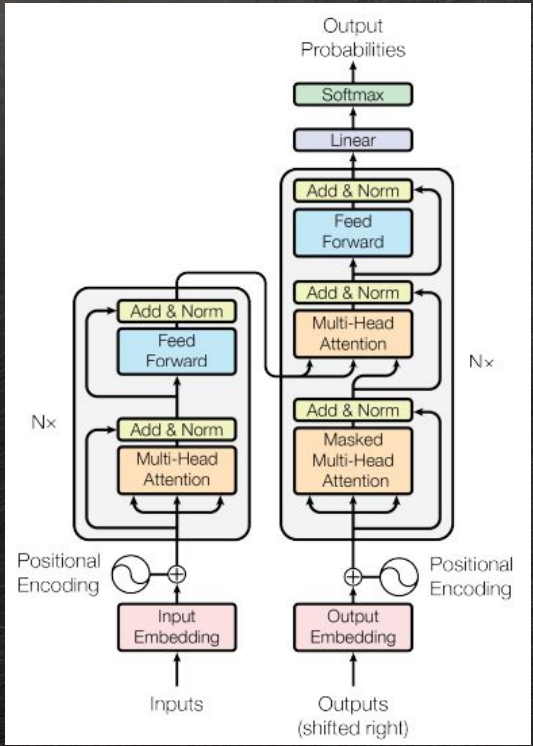
TRANSFORMER

Attention Is All You Need / 🥚🔫

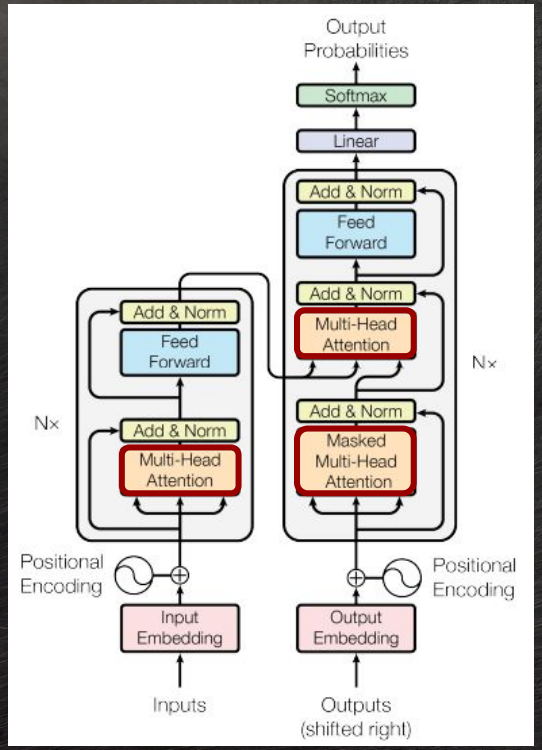
CONTENTS

CONTENTS

Attention

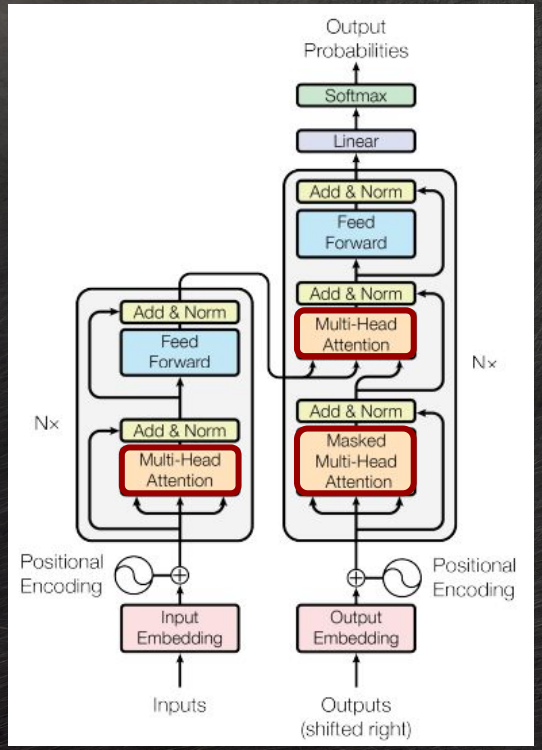

Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

Attention



Attention이 들어가는 부분들

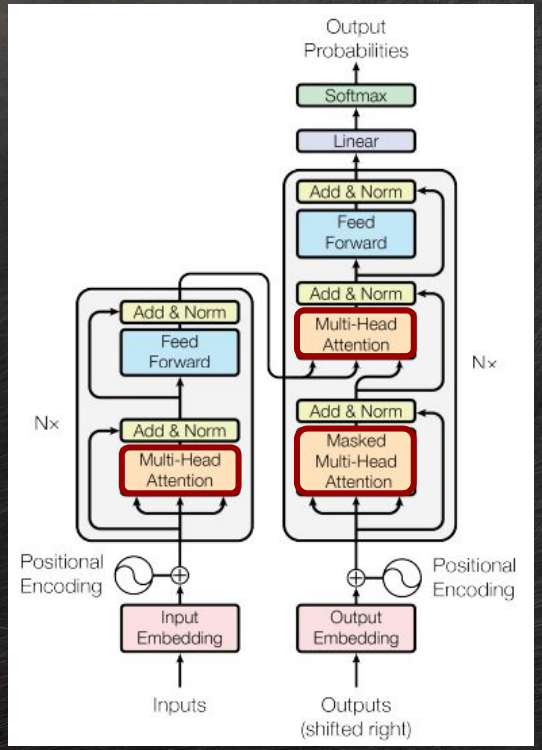
Attention



Attention이 들어가는 부분들

Attention : 주의, 주목

Attention



Attention이 들어가는 부분들

Attention : 주의, 주목

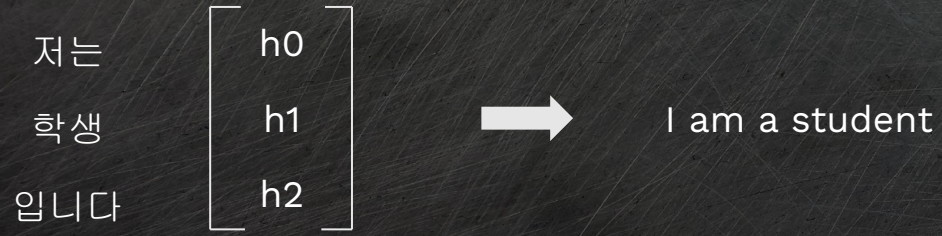
번역 Task

저는 학생입니다. → I am a student

Attention

번역 Task

저는 학생입니다. → I am a student



Attention

번역 Task

저는 학생입니다. → I am a student

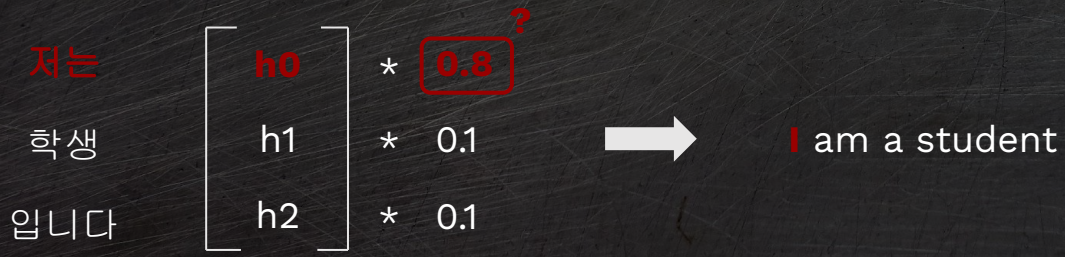
저는
학생
입니다
[h0
h1
h2] → I am a student

저는
학생
입니다
[h0
h1
h2] → I am a student

Attention

번역 Task

저는 학생입니다. → I am a student



Attention

번역 Task

저는 학생입니다. → I am a student

저는 h0
학생 h1
입니다 h2

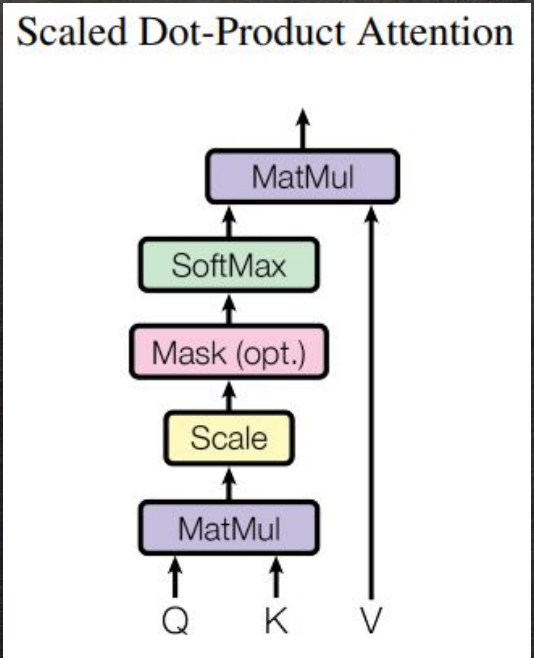
→ I am a student

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

저는 h0 * 0.8 ?
학생 h1 * 0.1
입니다 h2 * 0.1

→ I am a student

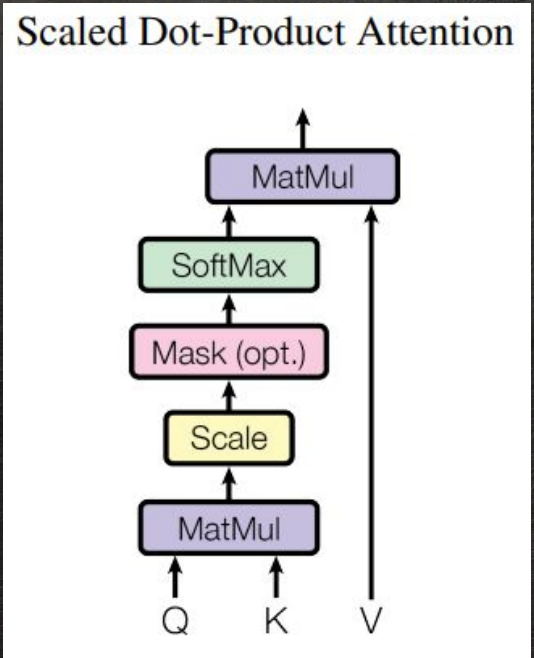
Scaled Dot-Product Attention



Query, Key, Value

mapping a query and a set of key-value pairs

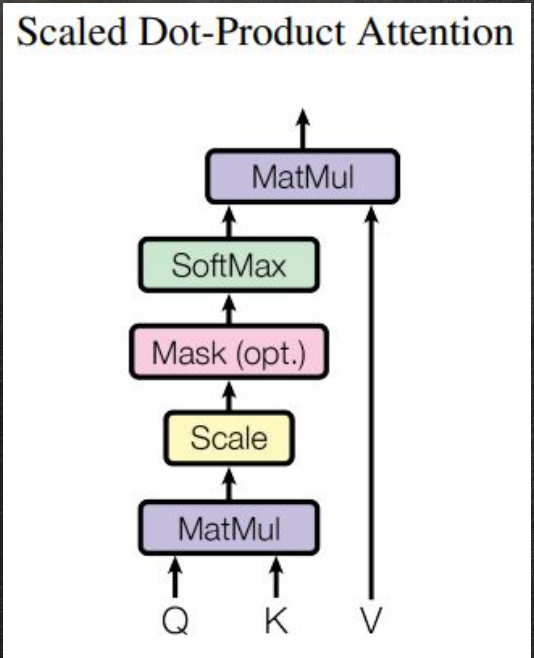
Scaled Dot-Product Attention



Query, {Key, Value}

mapping a query and a set of key-value pairs

Scaled Dot-Product Attention

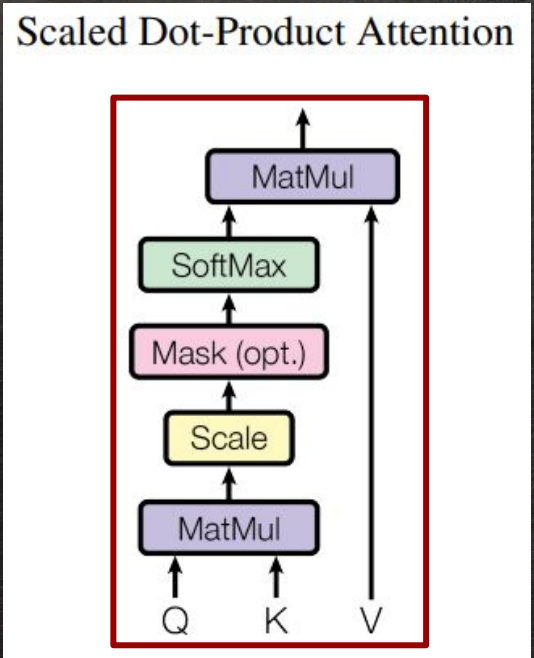


Query, {Key, Value}

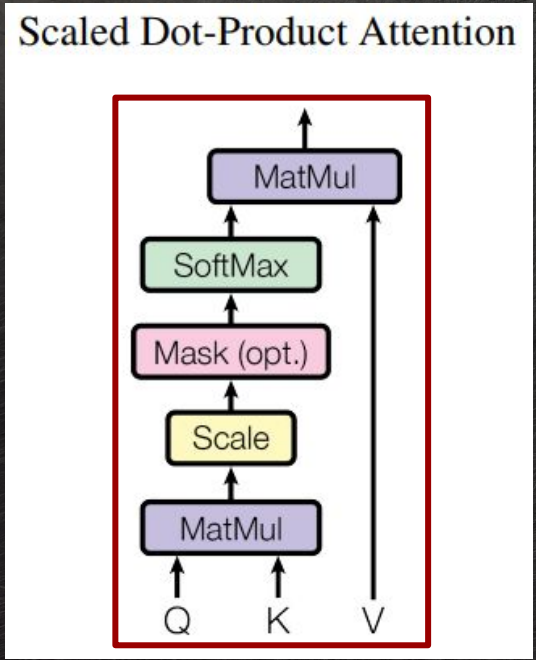
mapping a query and a set of key-value pairs

Query : 분석 대상이 되는 단어에 대한 가중치 벡터
Key : 단어가 Query와의 유사도를 계산하는 데에 사용되는 가중치 벡터
Value : 단어의 의미를 나타내는 가중치 벡터

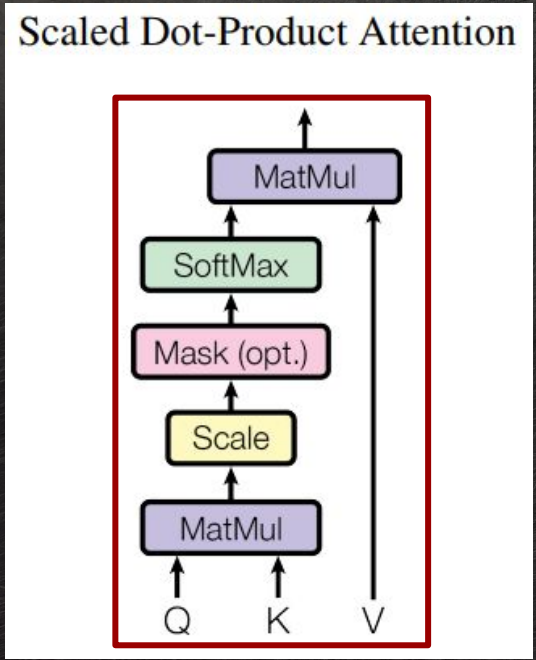
Scaled Dot-Product Attention

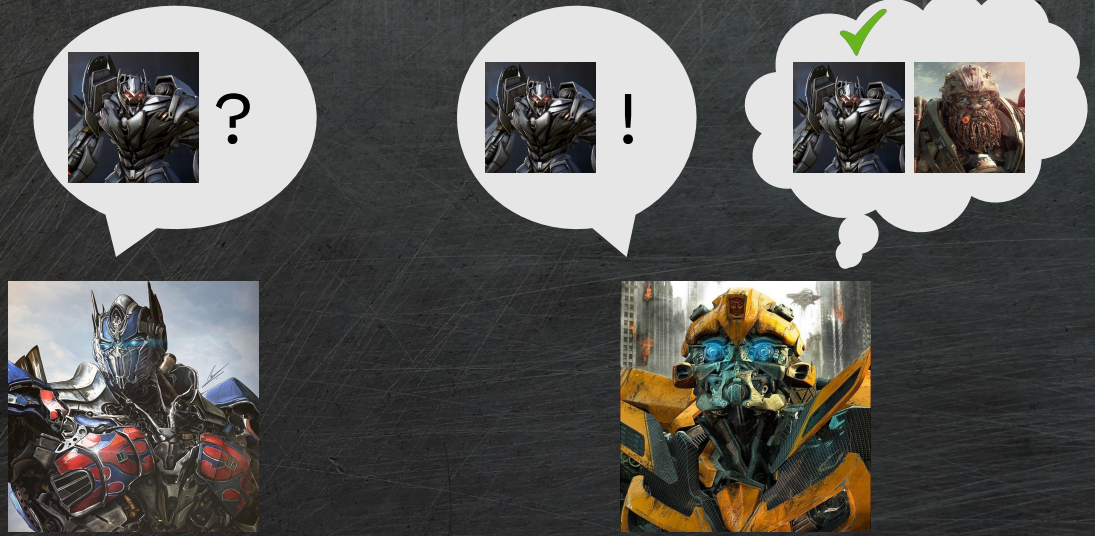


Scaled Dot-Product Attention

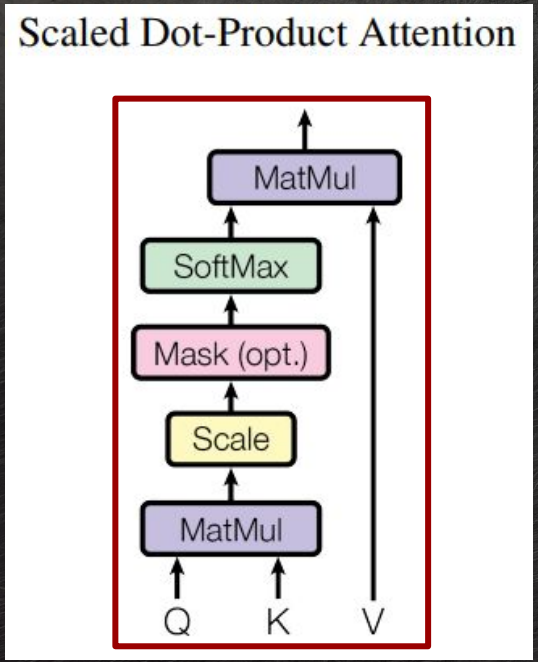


Scaled Dot-Product Attention

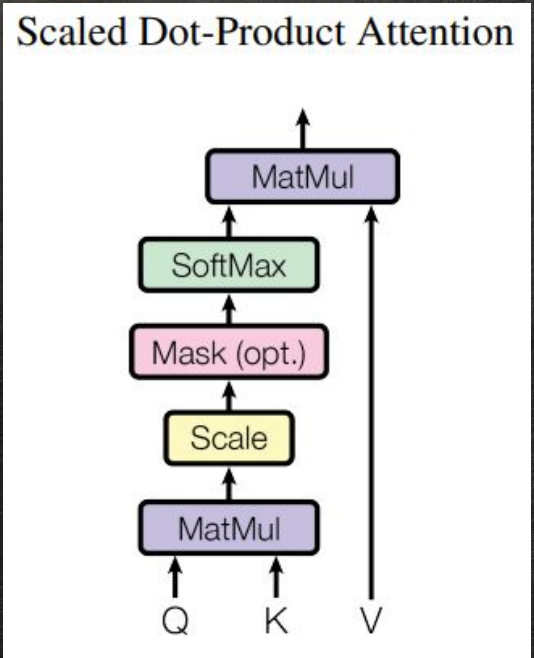




Scaled Dot-Product Attention



Scaled Dot-Product Attention

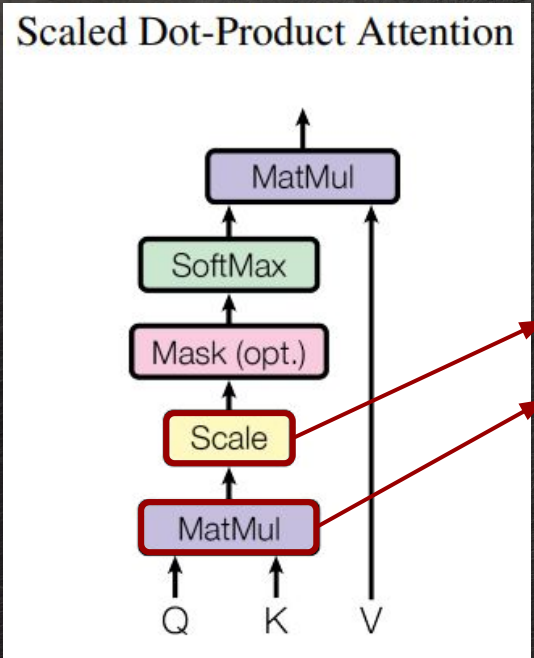


Query, Key, Value

mapping a query and a set of key-value pairs

Scaled Dot-Product Attention

Scaled Dot-Product Attention



Query, Key, Value

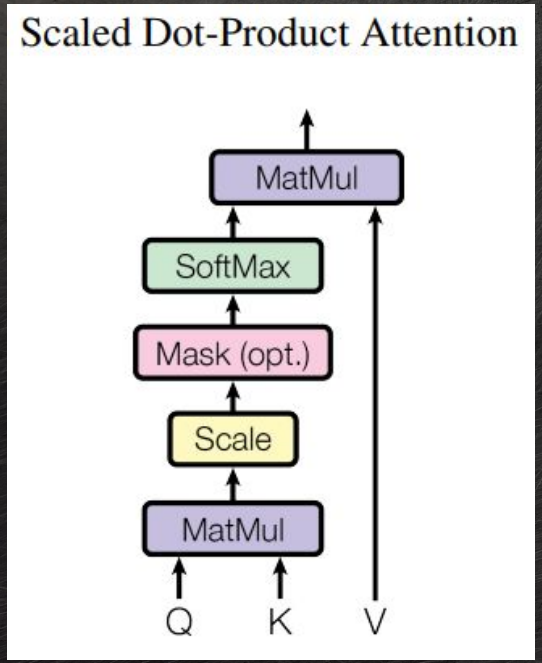
mapping a query and a set of key-value pairs

Scaled Dot-Product Attention

Scaled

Dot-Product : 내적

Scaled Dot-Product Attention

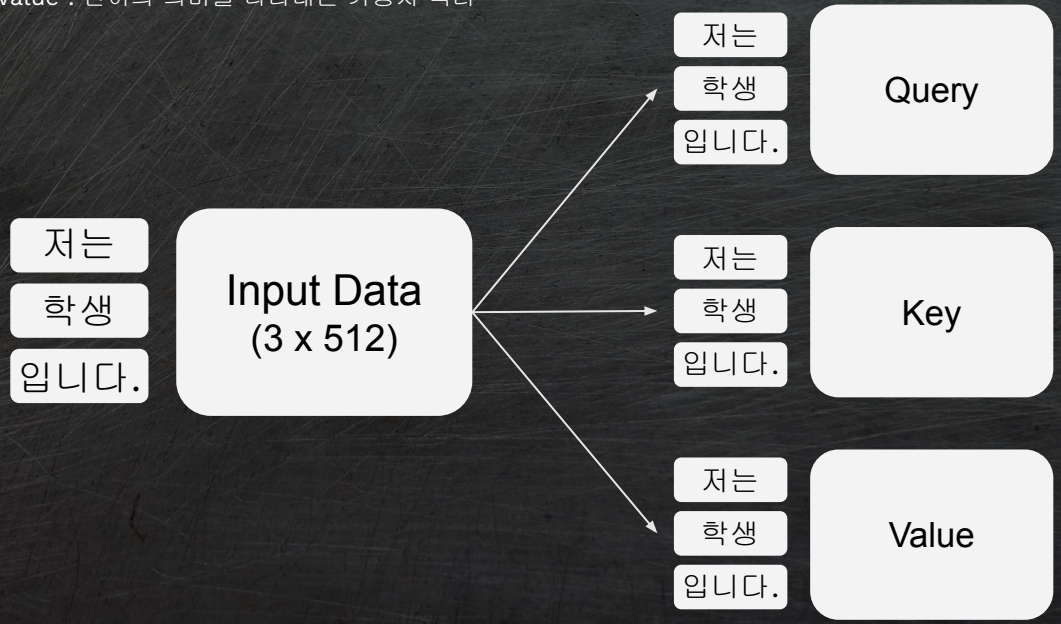


Query, Key, Value

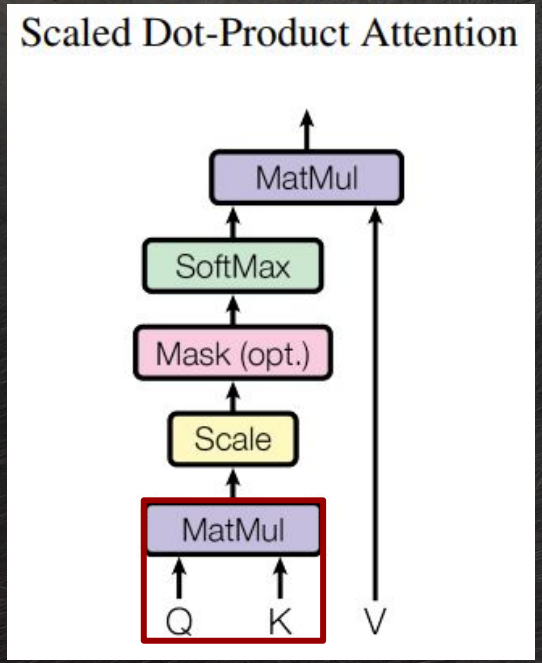
Query : 분석 대상이 되는 단어에 대한 가중치 벡터

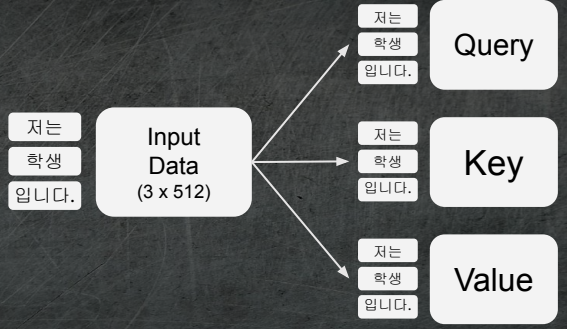
Key : 단어가 Query와의 유사도를 계산하는 데에 사용되는 가중치 벡터

Value : 단어의 의미를 나타내는 가중치 벡터



Scaled Dot-Product Attention





Query, Key, Value

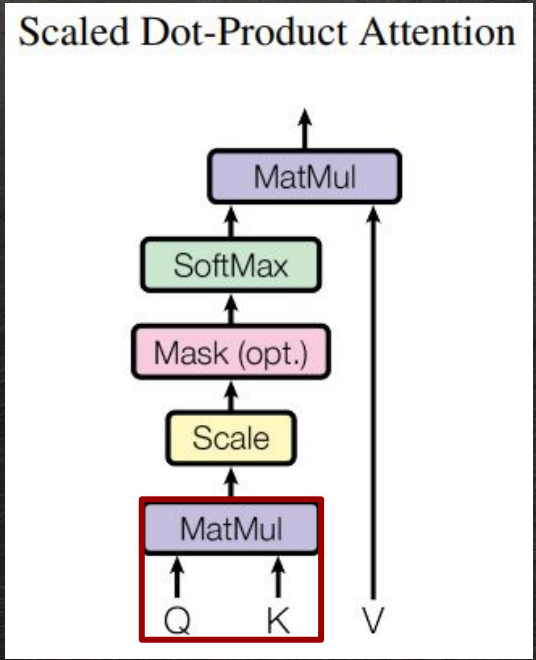
Query : 분석 대상이 되는 단어에 대한 가중치 벡터

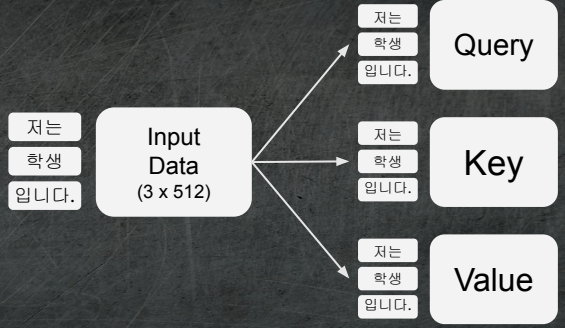
Key : 단어가 Query와의 유사도를 계산하는 데에 사용되는 가중치 벡터

Value : 단어의 의미를 나타내는 가중치 벡터

$$QK^T$$

Scaled Dot-Product Attention





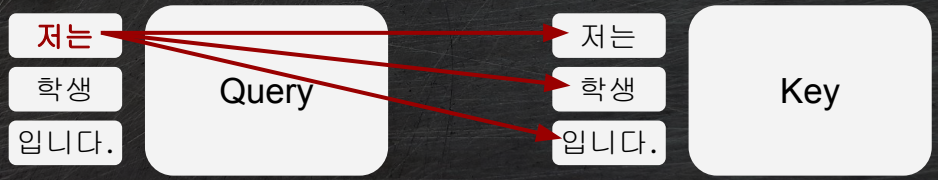
Query, Key, Value

Query : 분석 대상이 되는 단어에 대한 가중치 벡터

Key : 단어가 Query와의 유사도를 계산하는 데에 사용되는 가중치 벡터

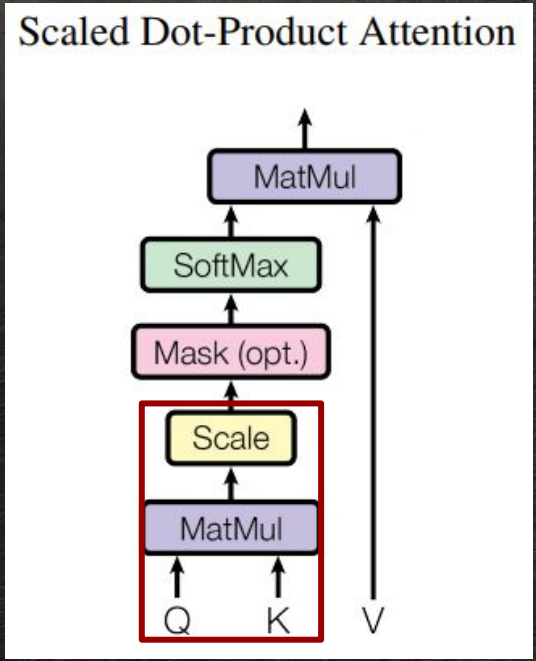
Value : 단어의 의미를 나타내는 가중치 벡터

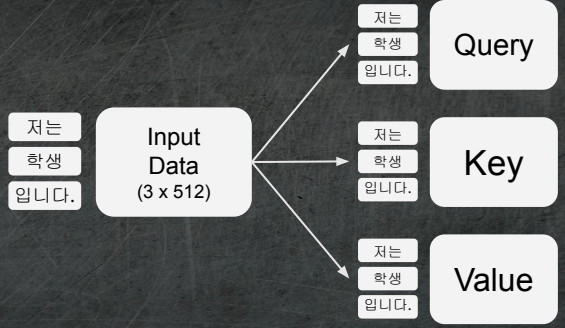
$$QK^T$$



분석 대상이 되는 단어와 문장내의 단어들 사이의 유사도를 구한다.

Scaled Dot-Product Attention





Query, Key, Value

Query : 분석 대상이 되는 단어에 대한 가중치 벡터

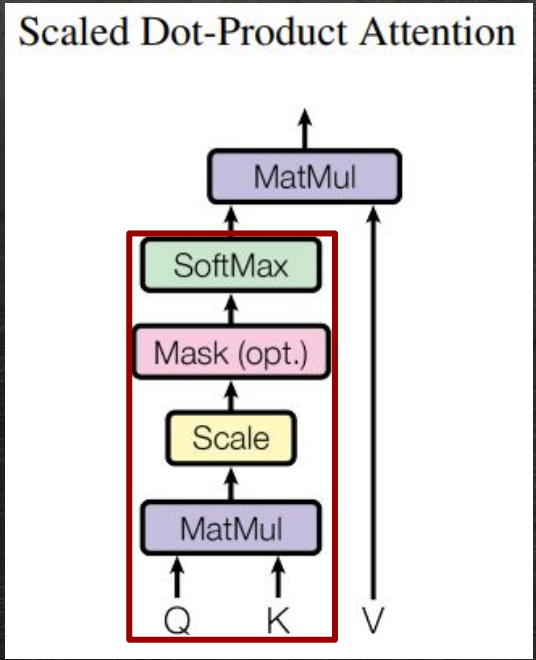
Key : 단어가 Query와의 유사도를 계산하는 데에 사용되는 가중치 벡터

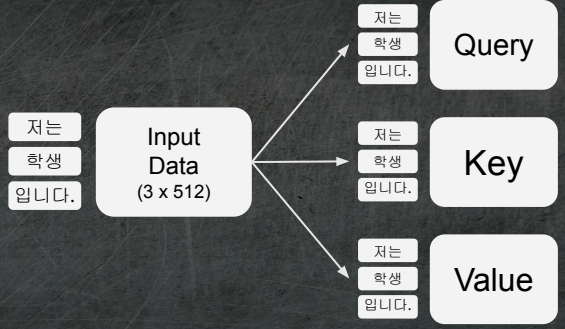
Value : 단어의 의미를 나타내는 가중치 벡터

$$\frac{QK^T}{\sqrt{d_k}}$$ = Attention Score

저는	75점
학생	50점
입니다.	20점

Scaled Dot-Product Attention





Query, Key, Value

Query : 분석 대상이 되는 단어에 대한 가중치 벡터

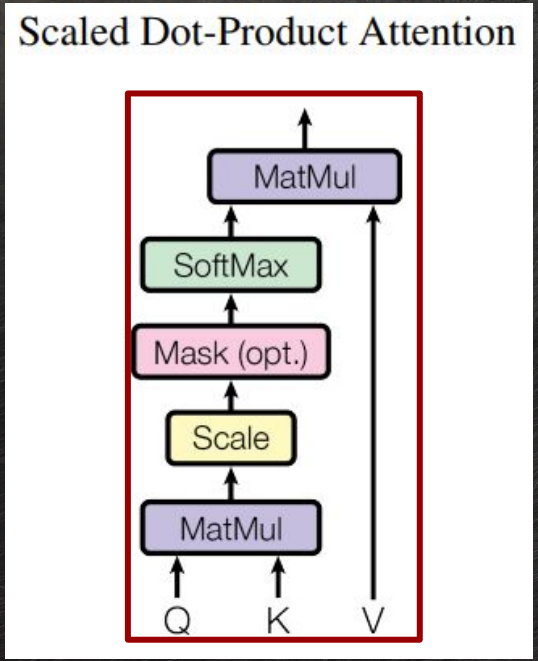
Key : 단어가 Query와의 유사도를 계산하는 데에 사용되는 가중치 벡터

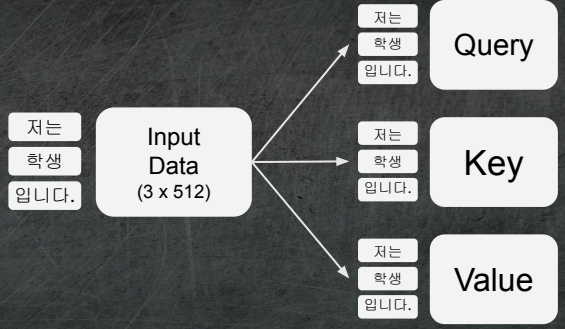
Value : 단어의 의미를 나타내는 가중치 벡터

$$\text{softmax}(\frac{QK^T}{\sqrt{d_k}})$$

가장 높은 유사도를 가진 지식(K)을 선택. 이 지식에는 질문(Q)에 대한 가장 좋은 답

Scaled Dot-Product Attention





Query, Key, Value

Query : 분석 대상이 되는 단어에 대한 가중치 벡터

Key : 단어가 Query와의 유사도를 계산하는 데에 사용되는 가중치 벡터

Value : 단어의 의미를 나타내는 가중치 벡터

QK^T 분석 대상이 되는 단어와 문장내의 단어들 사이의 유사도를 구한다.

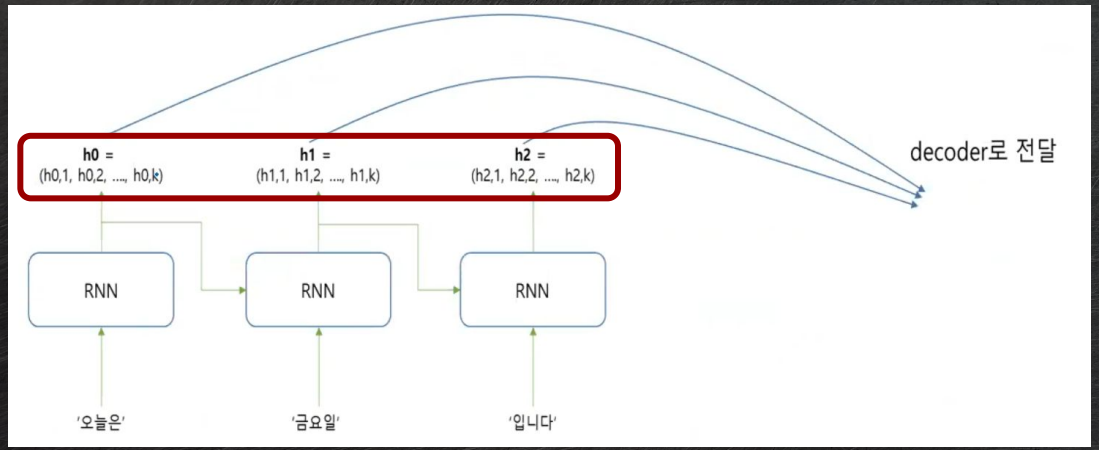
$$\text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

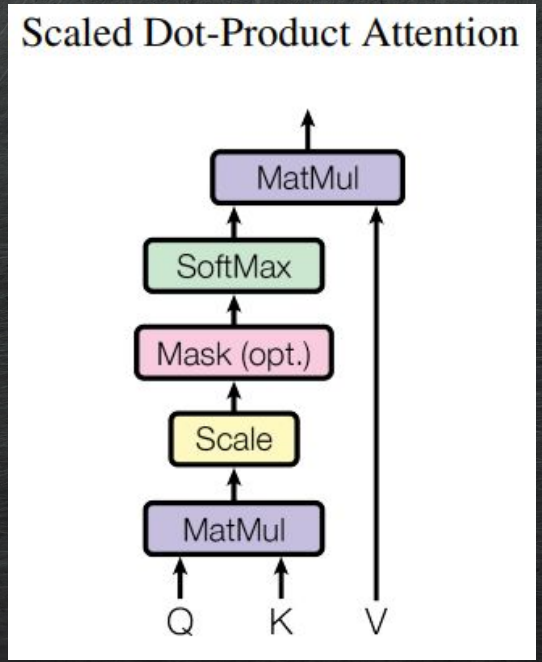
QK의 SoftMax값을 Value와 내적한다

=> 선택된 지식(K)을 기반으로 최종 V를 결정

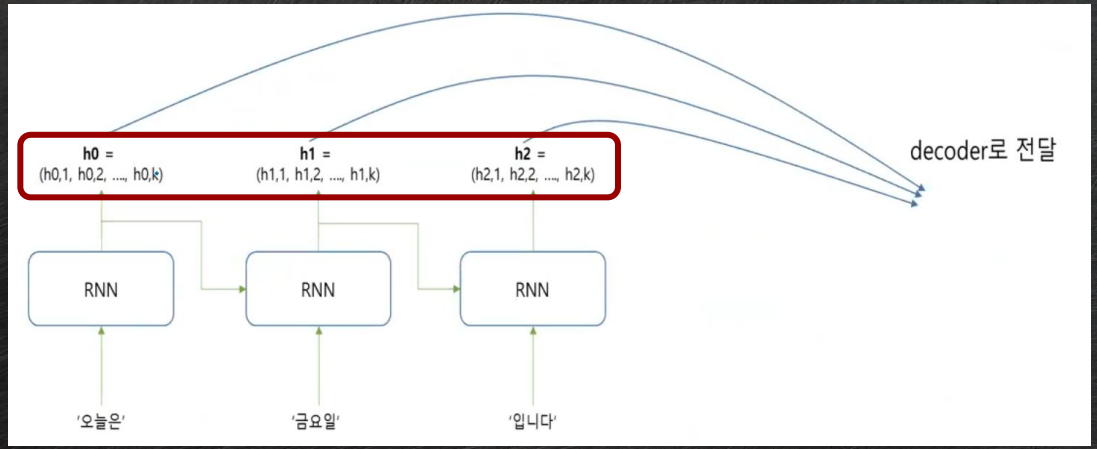
= 특정 단어에 대한 Attention 결과물 도출

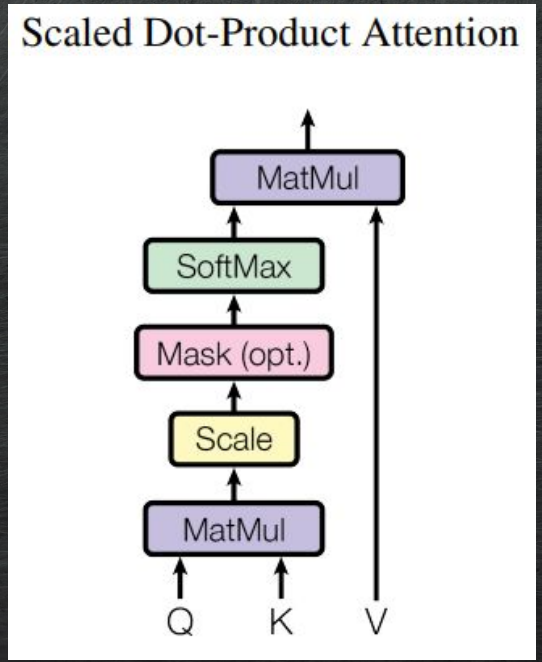
Scaled Dot-Product Attention



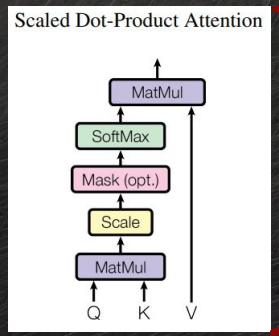


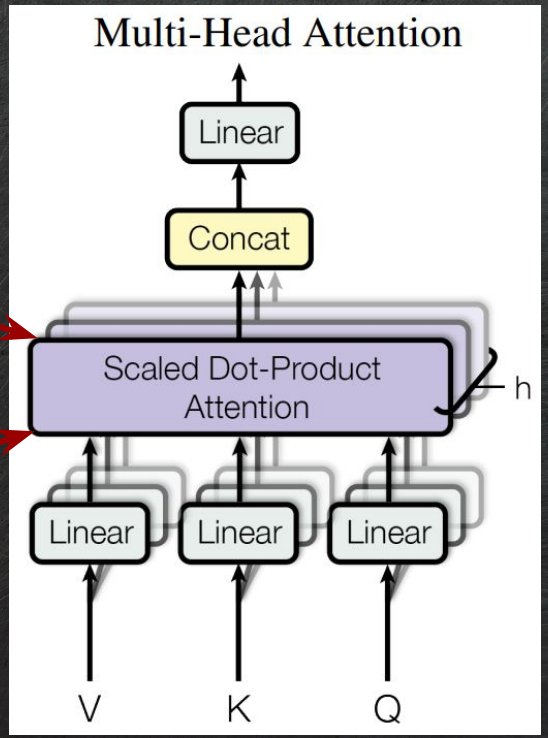
Self Attention



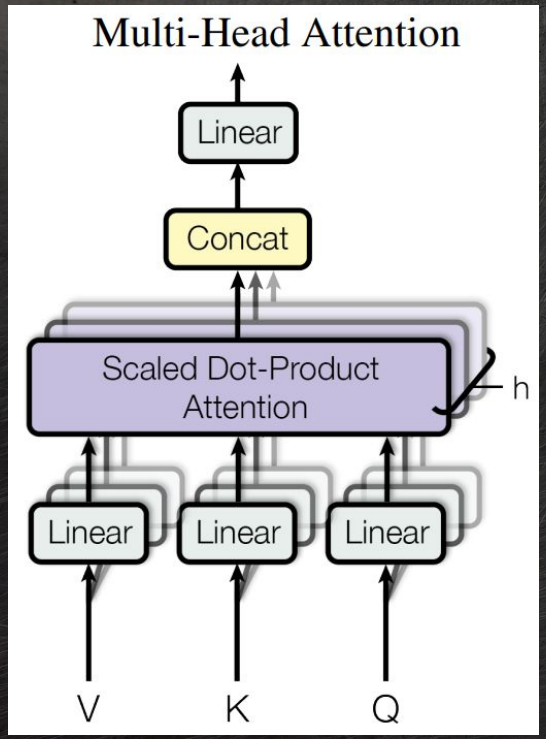


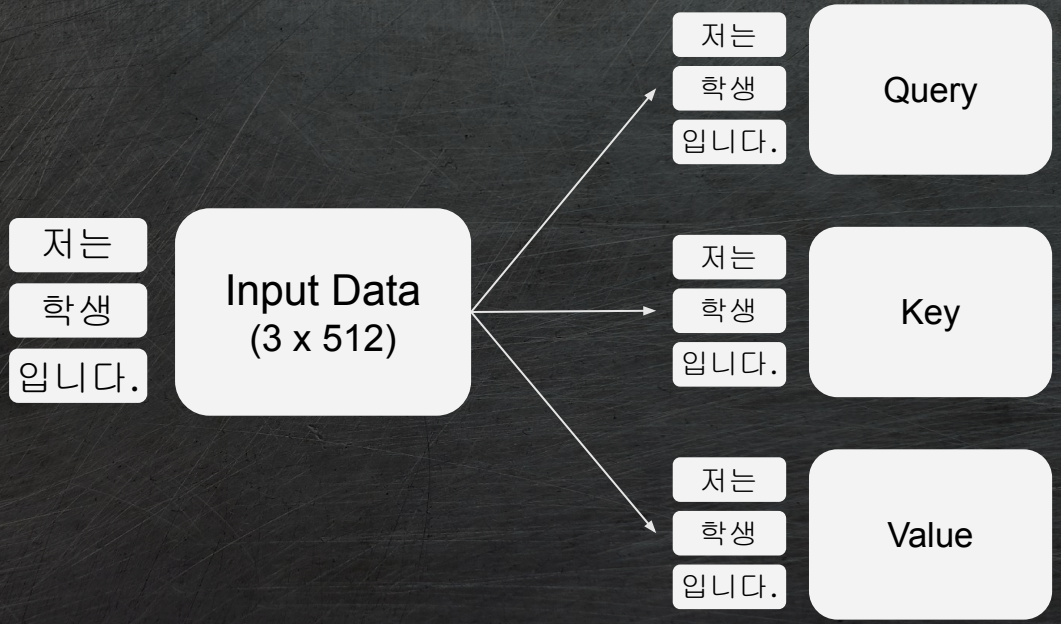
Multi-Head Attention



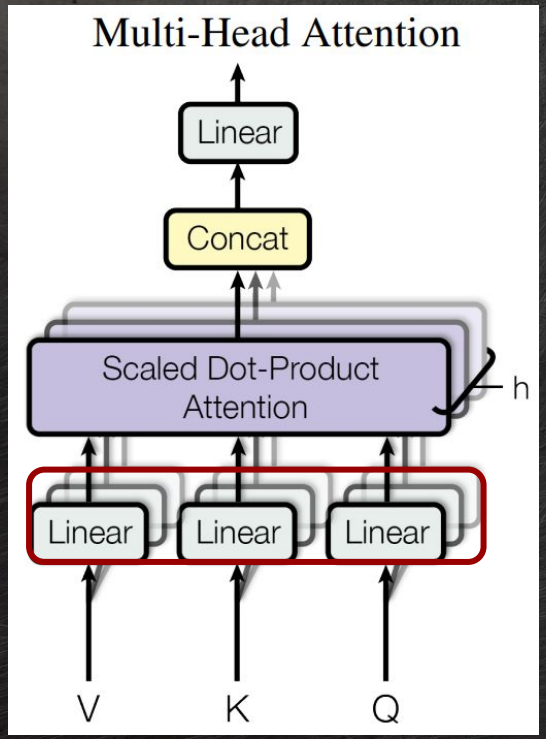


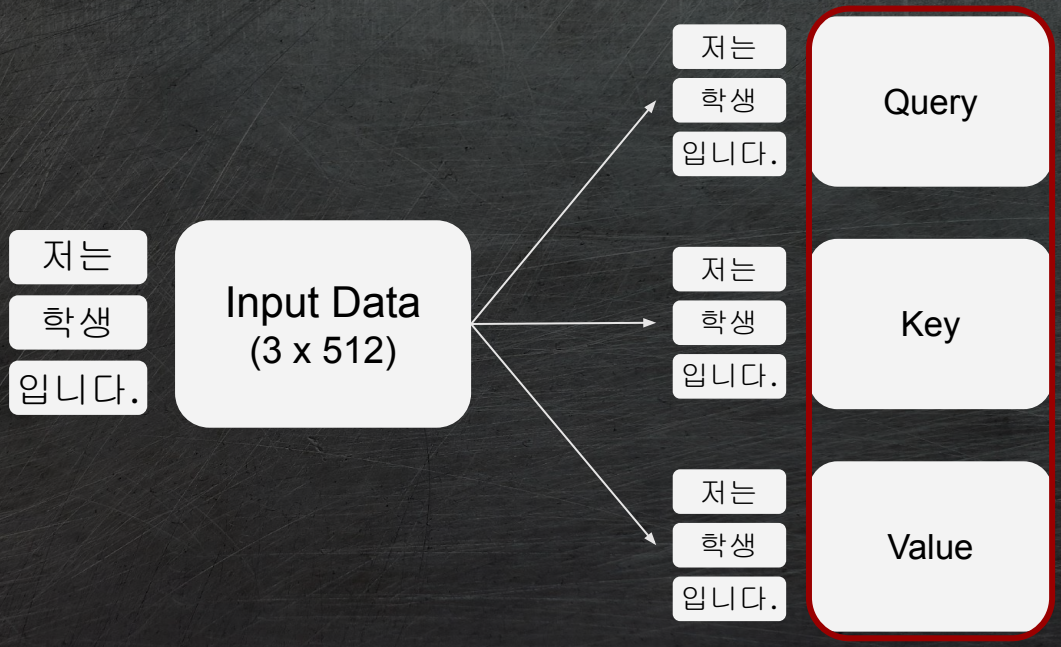
Multi-Head Attention



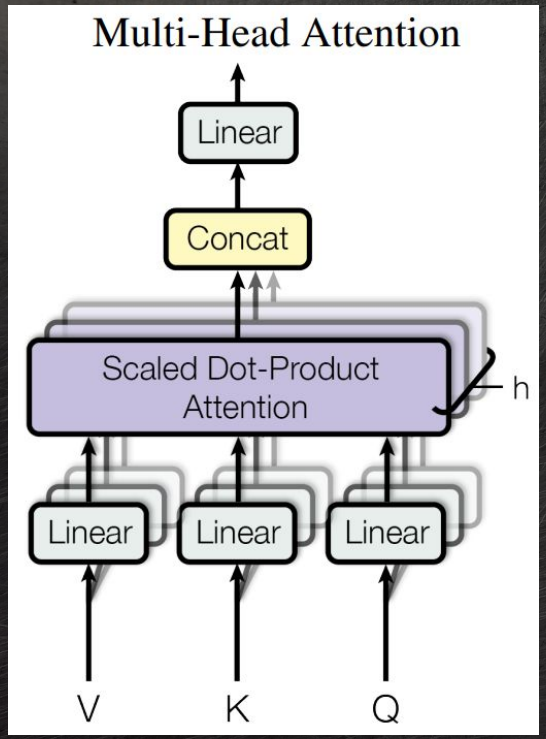


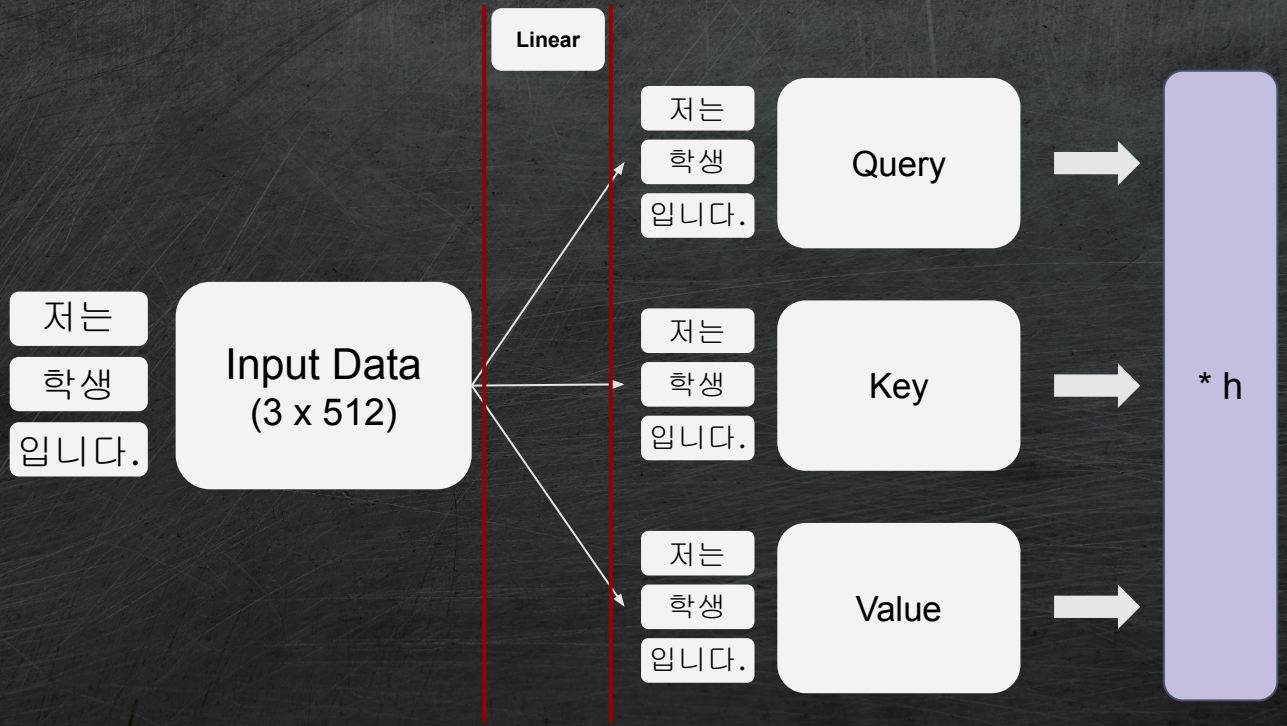
Multi-Head Attention



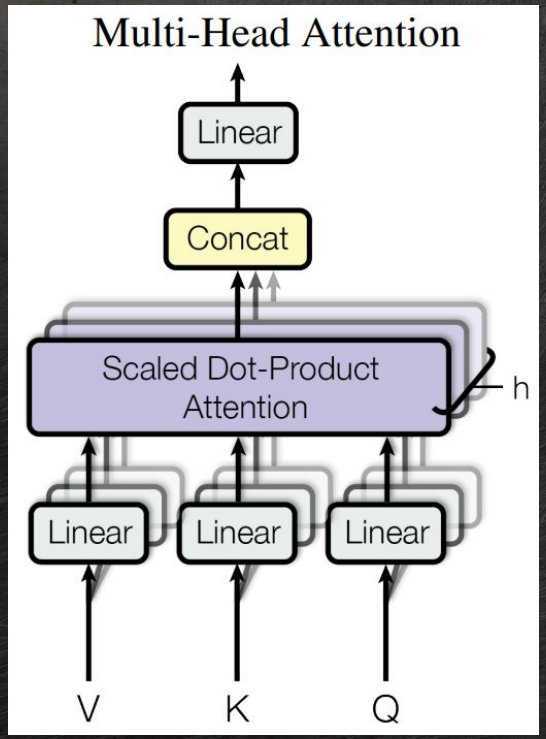


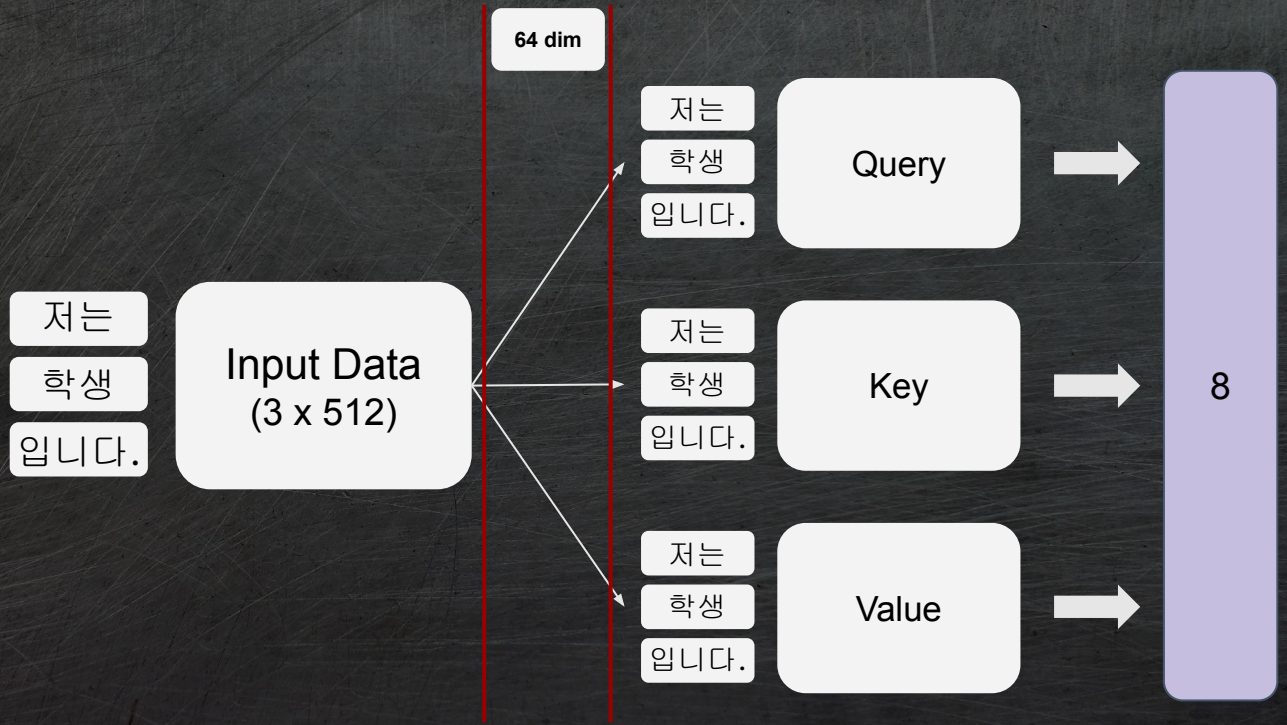
Multi-Head Attention



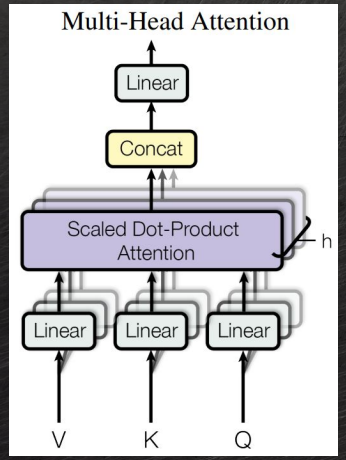


Multi-Head Attention





Multi-Head Attention



저는
학생
입니다.

Input Data
(3 x 512)

64 dim

저는
학생
입니다.

Query

저는
학생
입니다.

Key

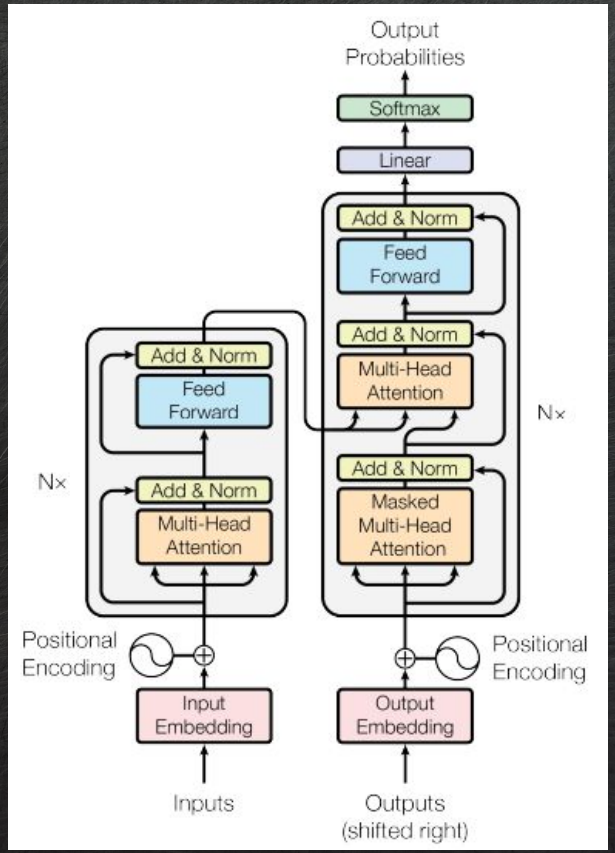
저는
학생
입니다.

Value

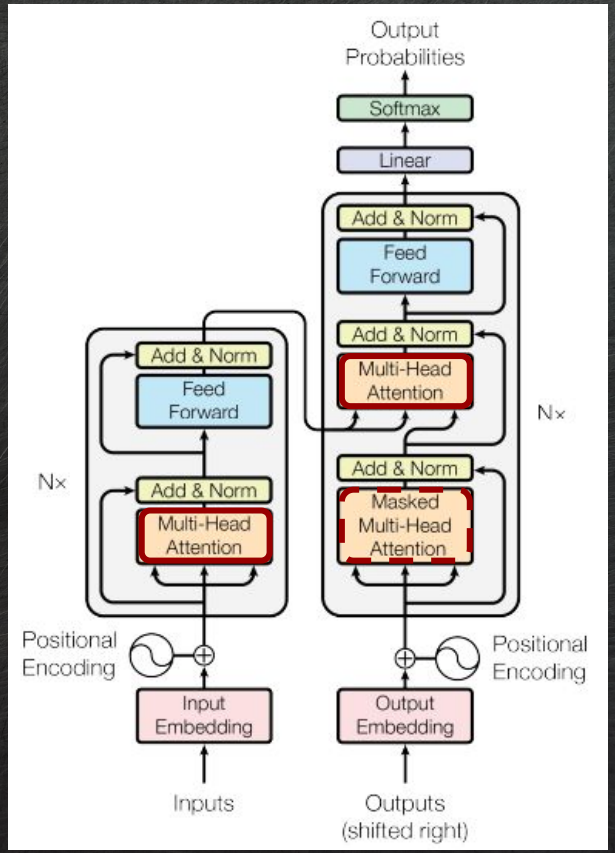
8

512

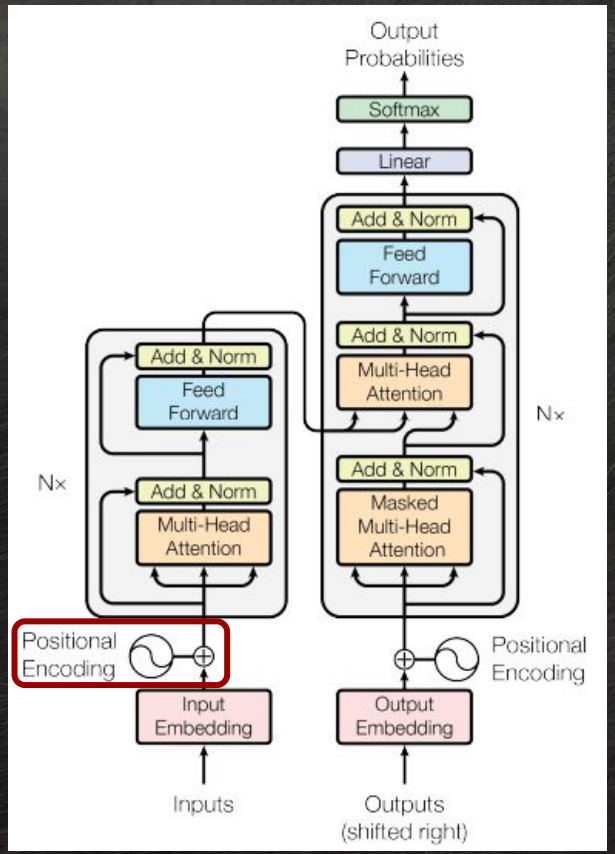
Multi-Head Attention

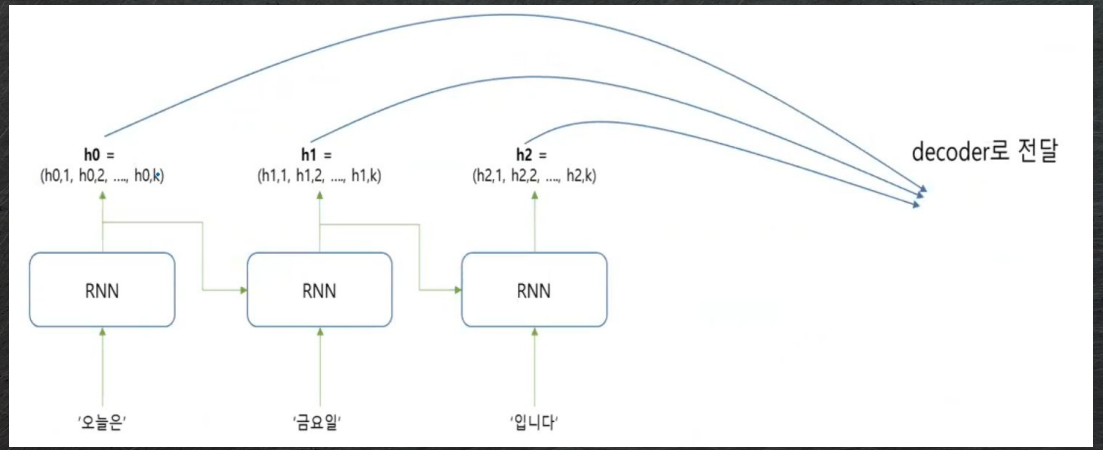


Multi-Head Attention

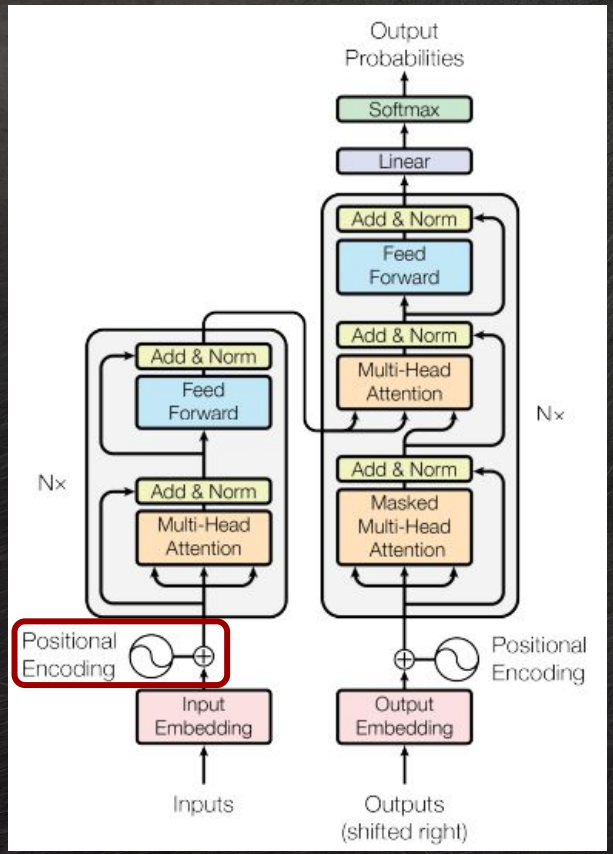


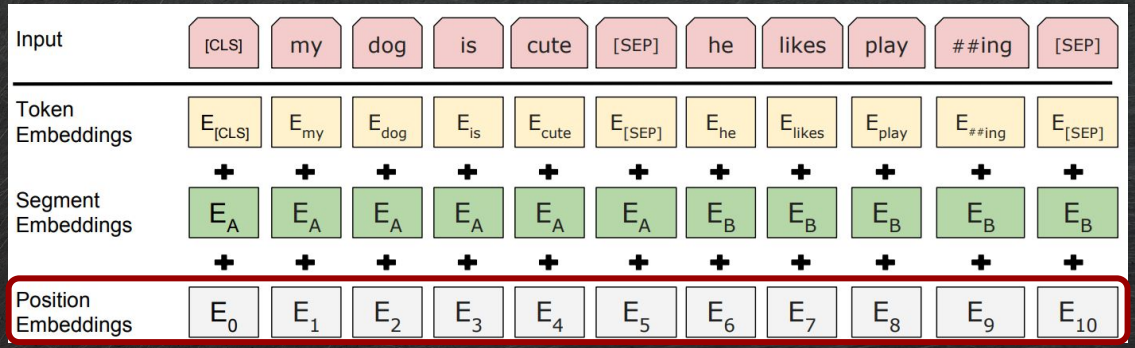
Positional Encoding



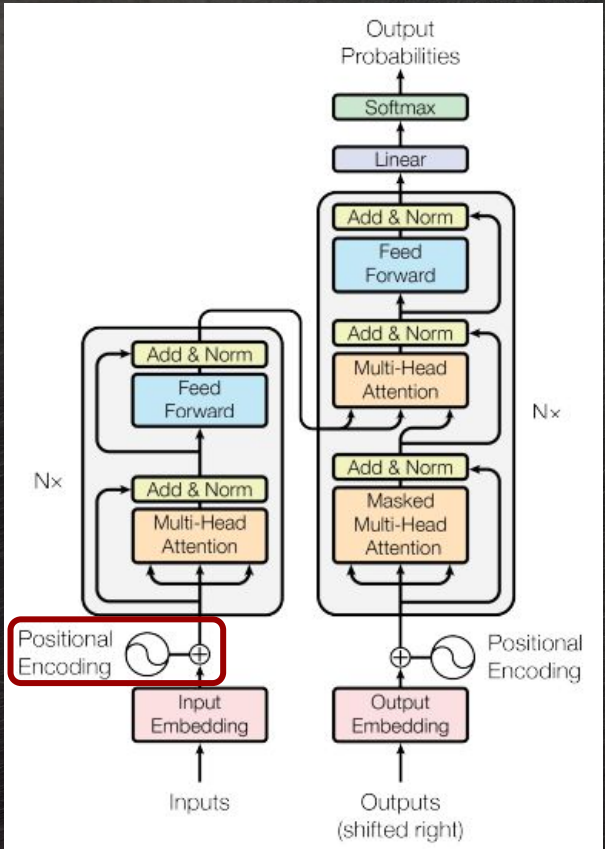


Positional Encoding





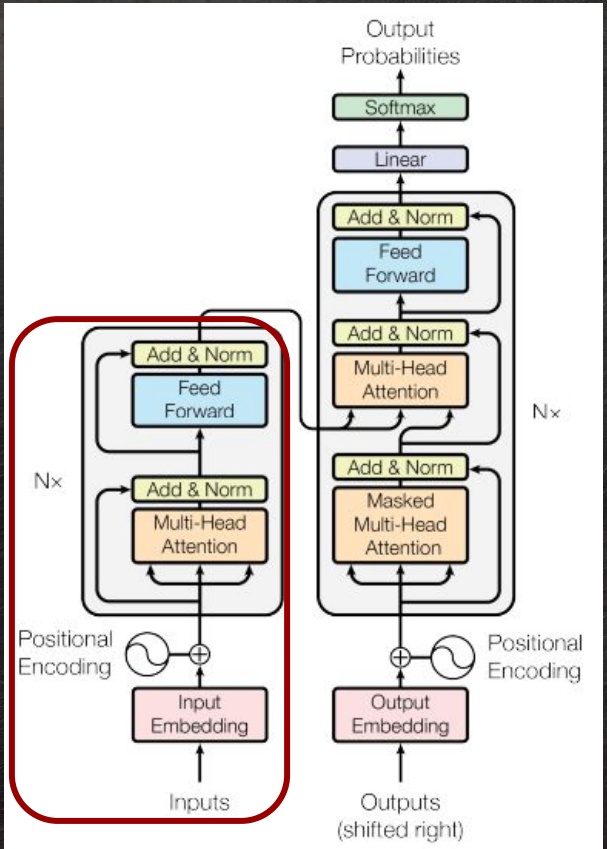
Positional Encoding



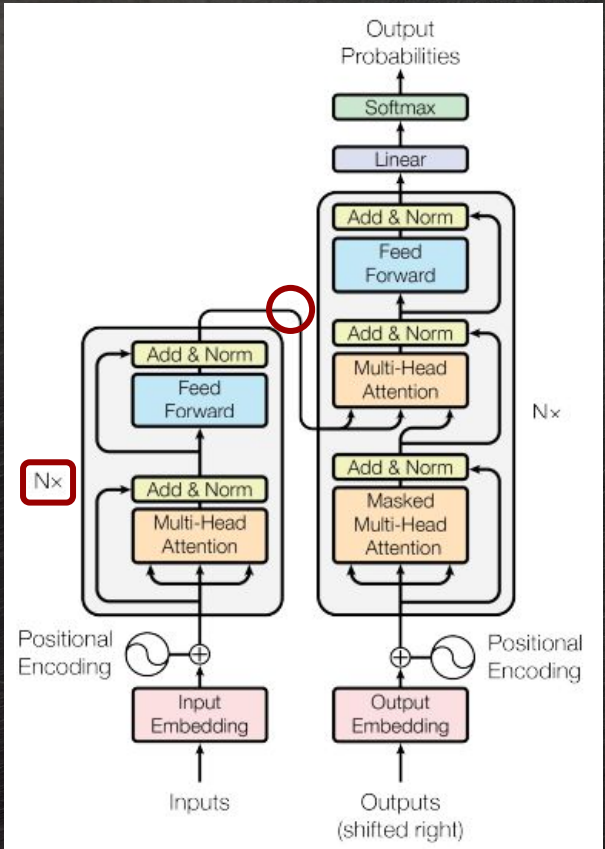
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

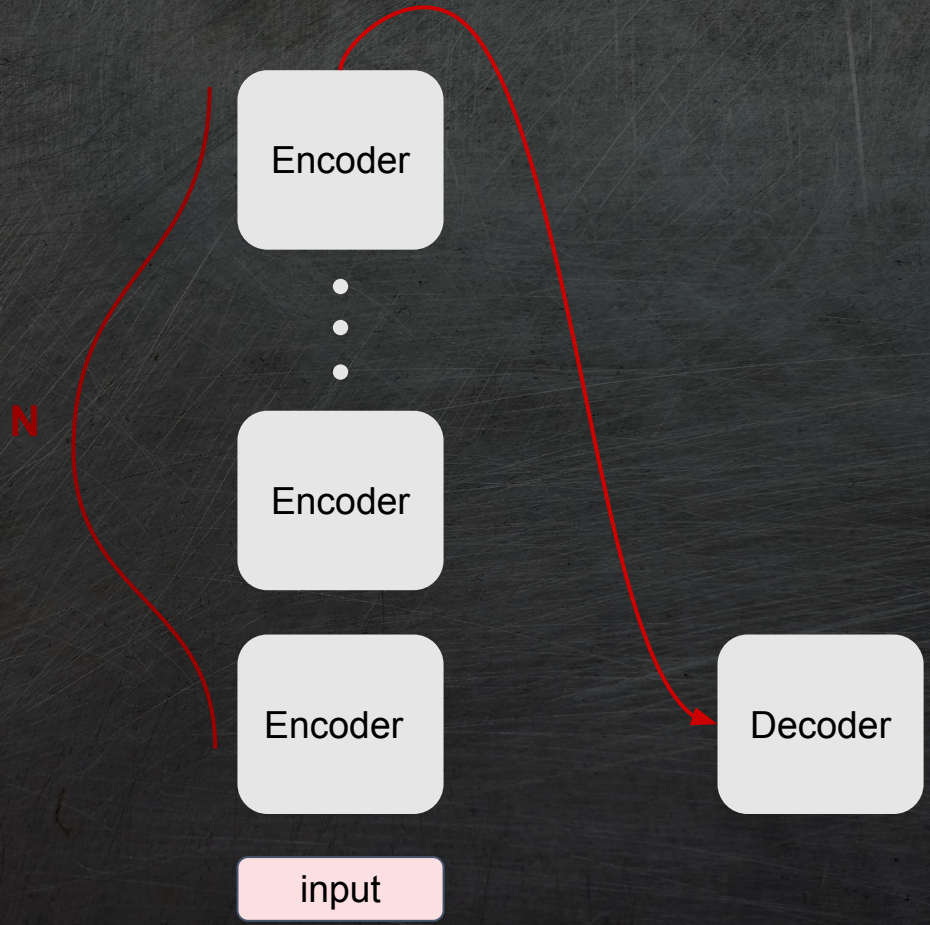
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Model Architecture

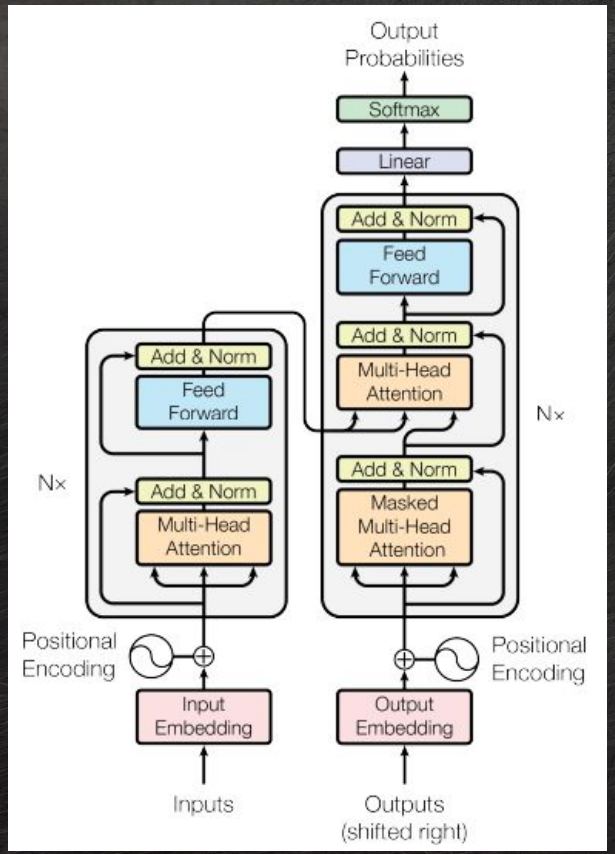


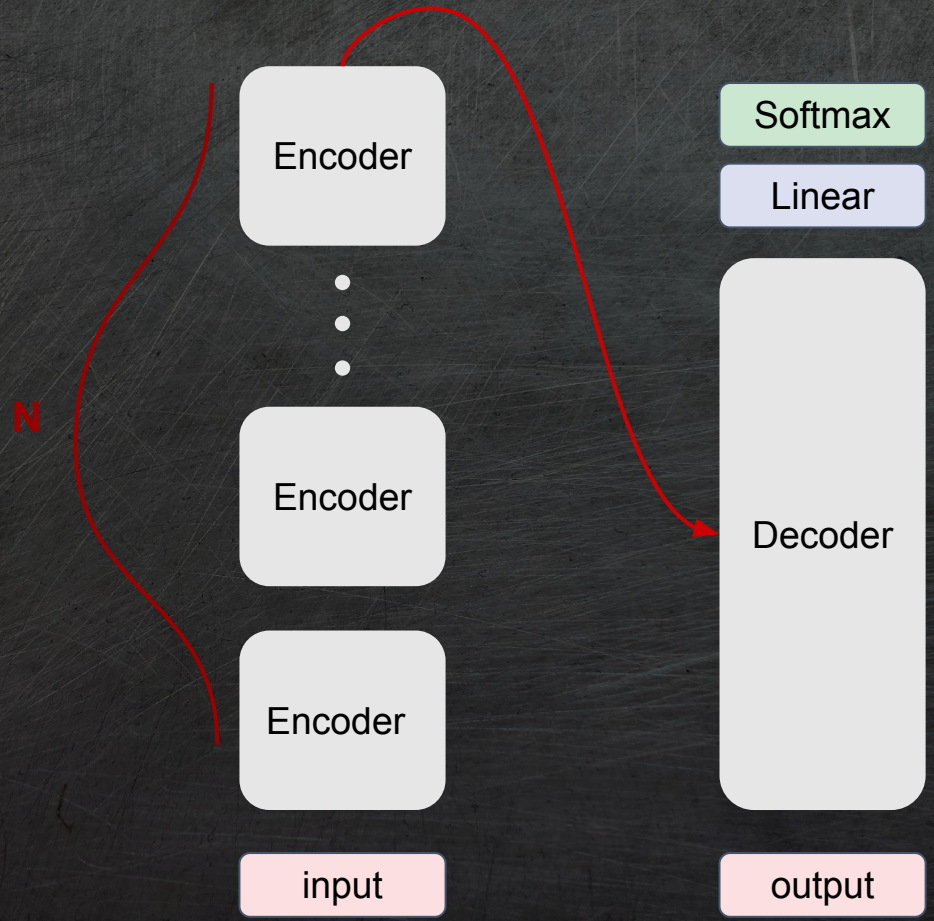
Model Architecture



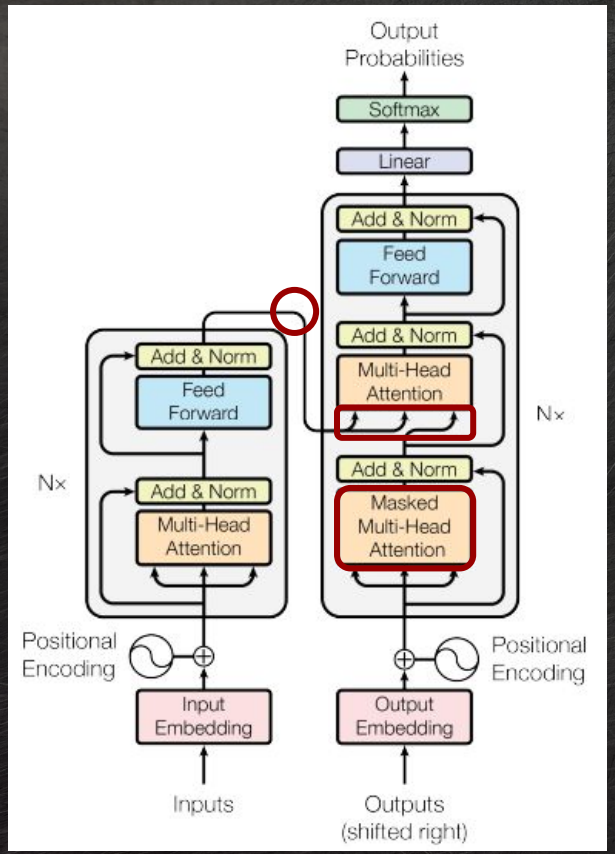


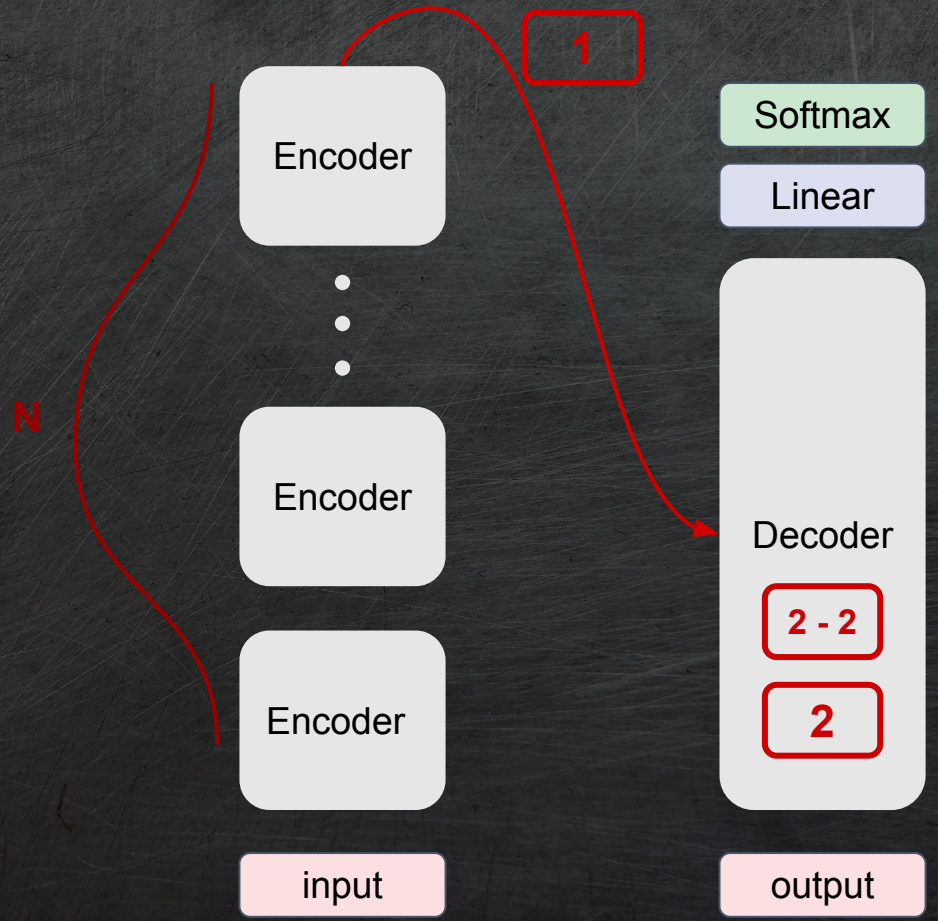
Model Architecture



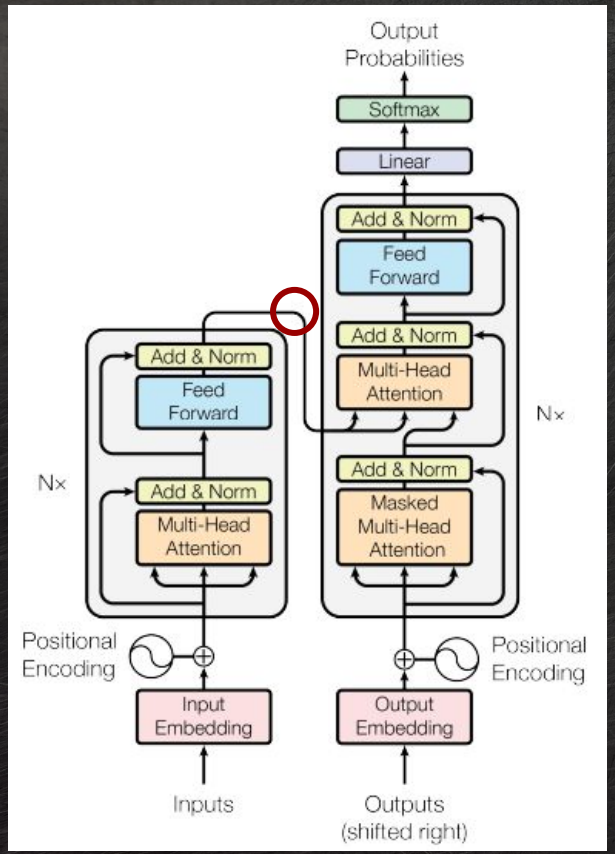


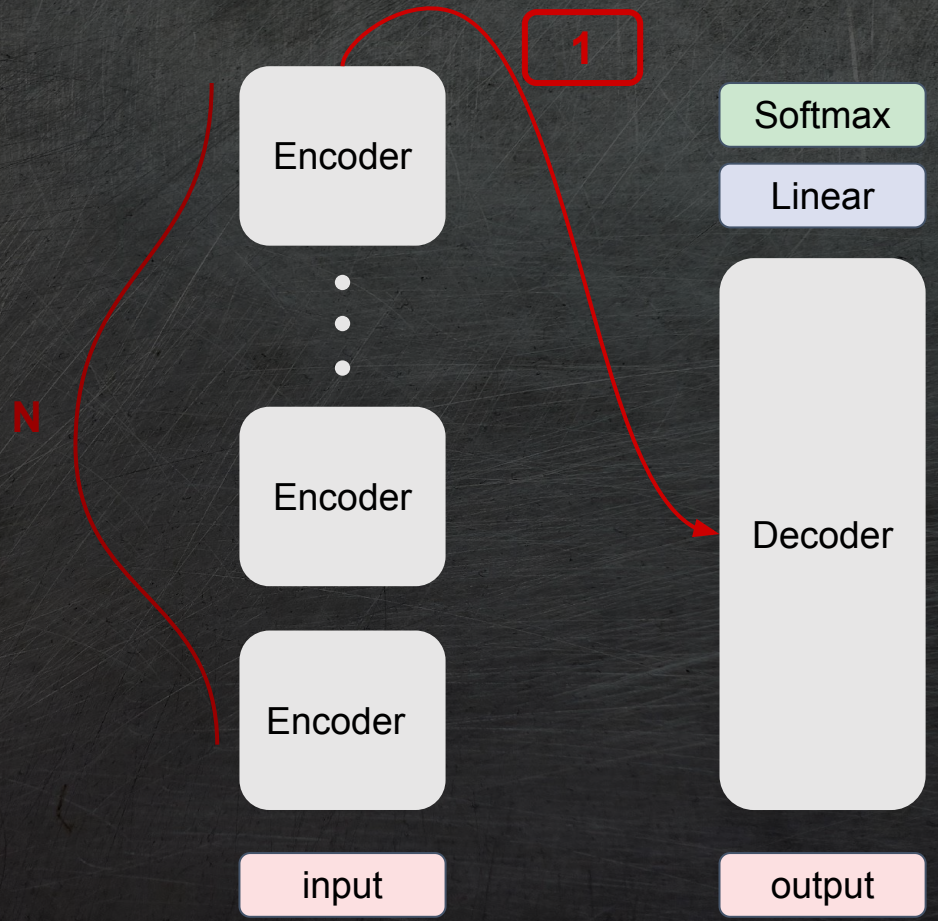
Model Architecture



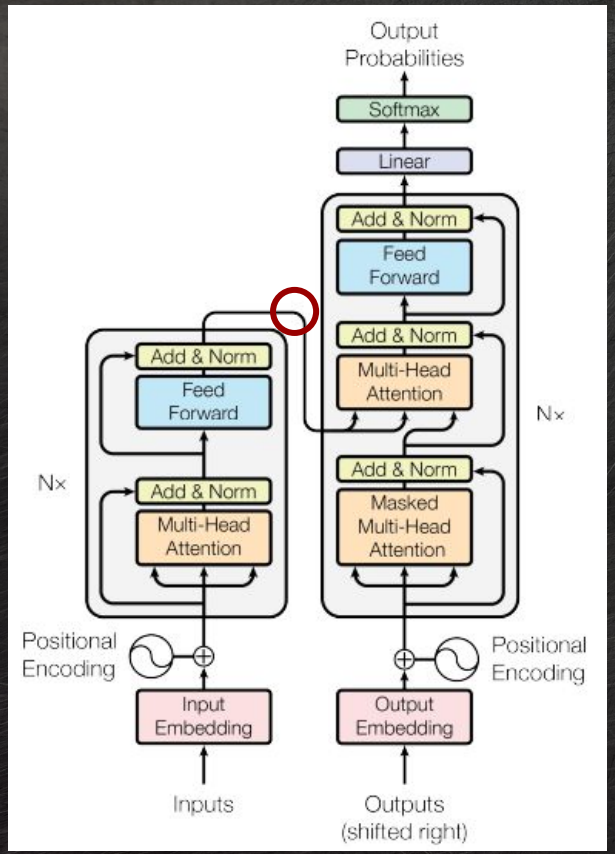


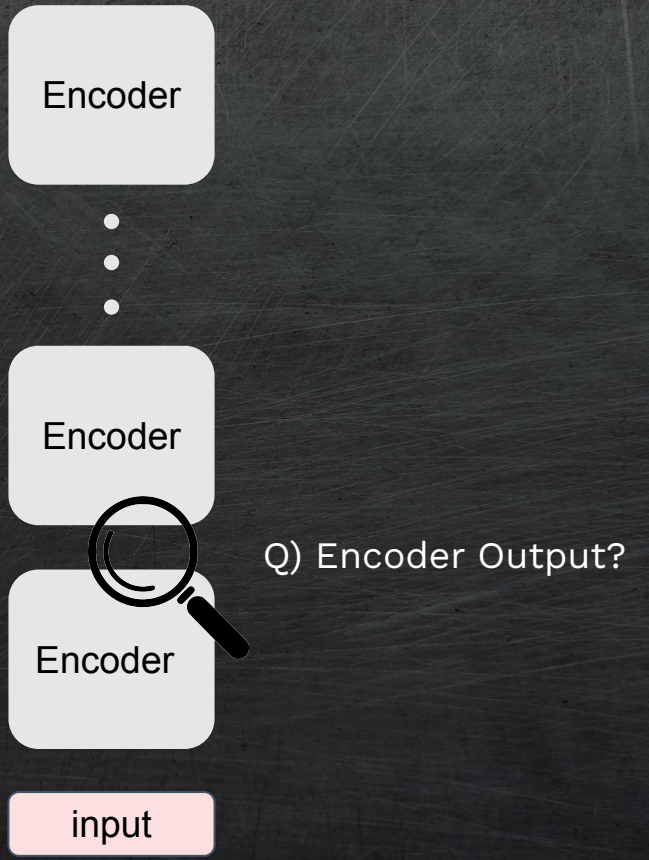
Encoder-Decoder Attention



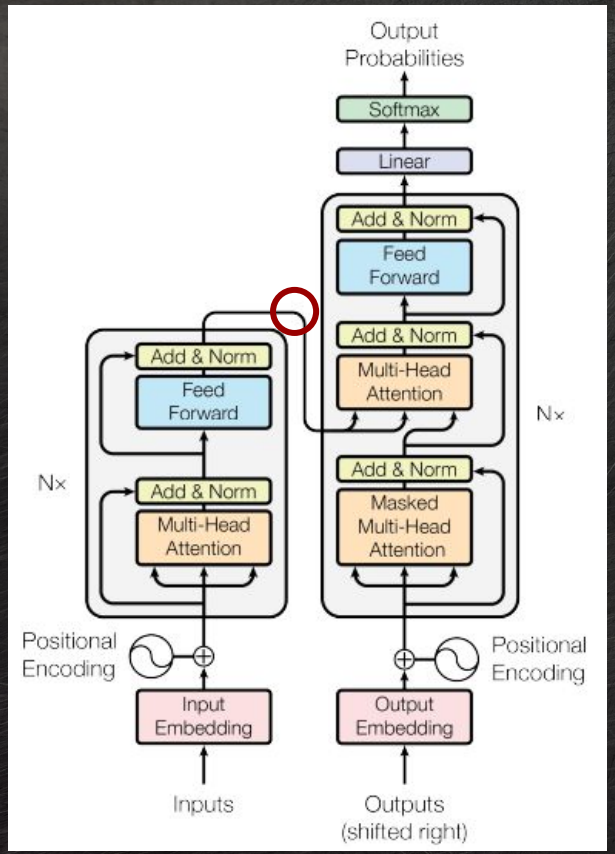


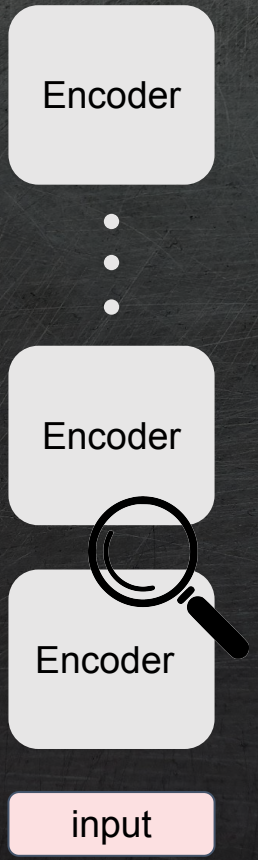
Encoder-Decoder Attention





Encoder-Decoder Attention

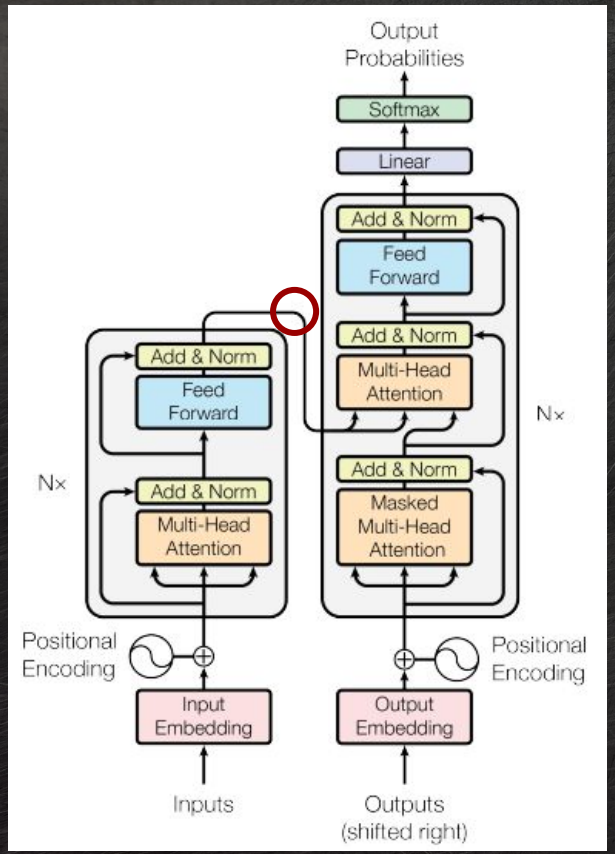


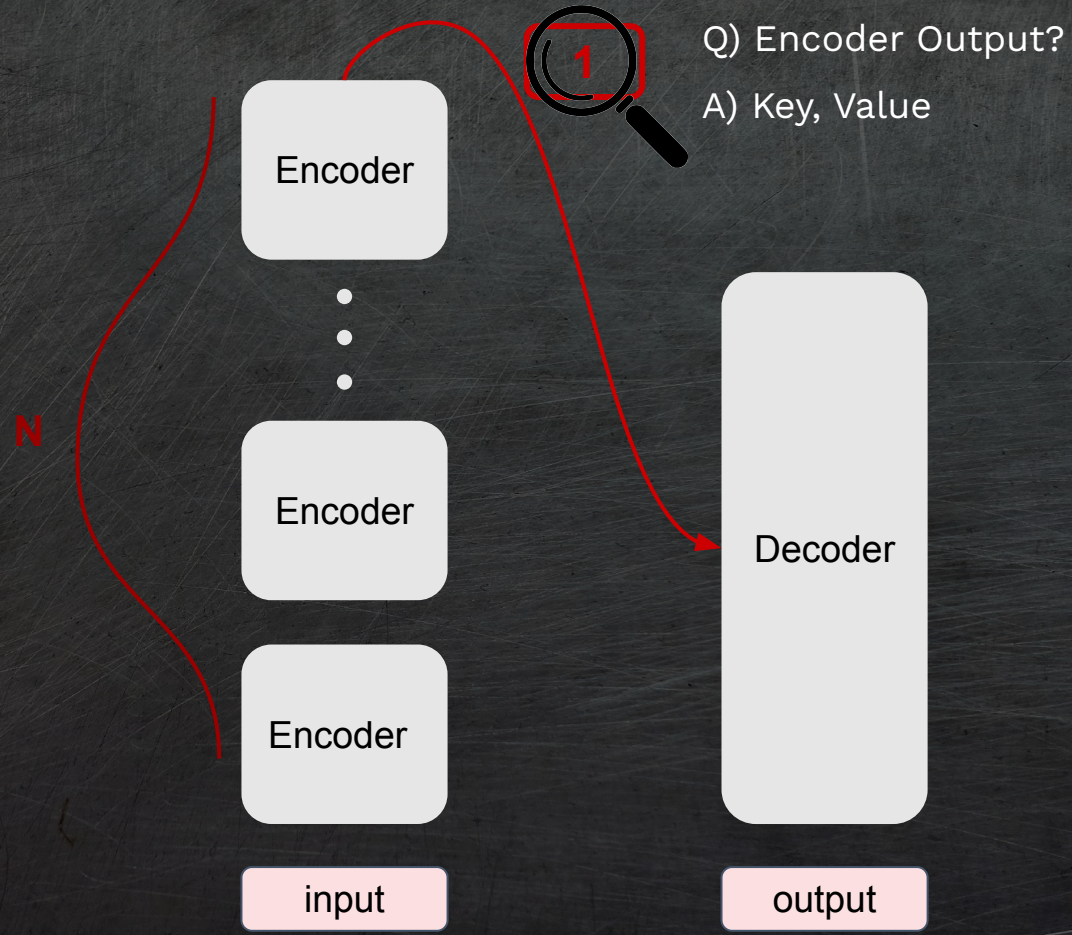


Q) Encoder Output?

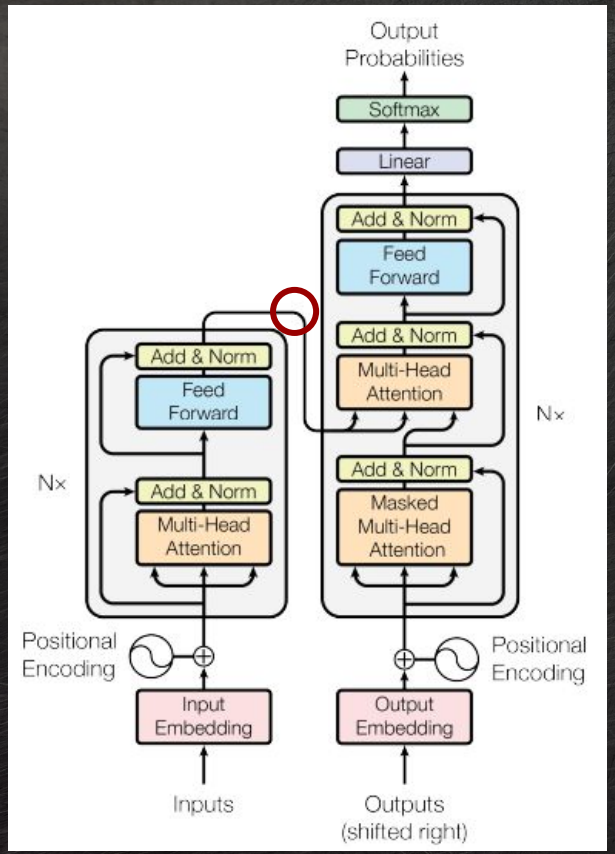
A) Query, Key, Value

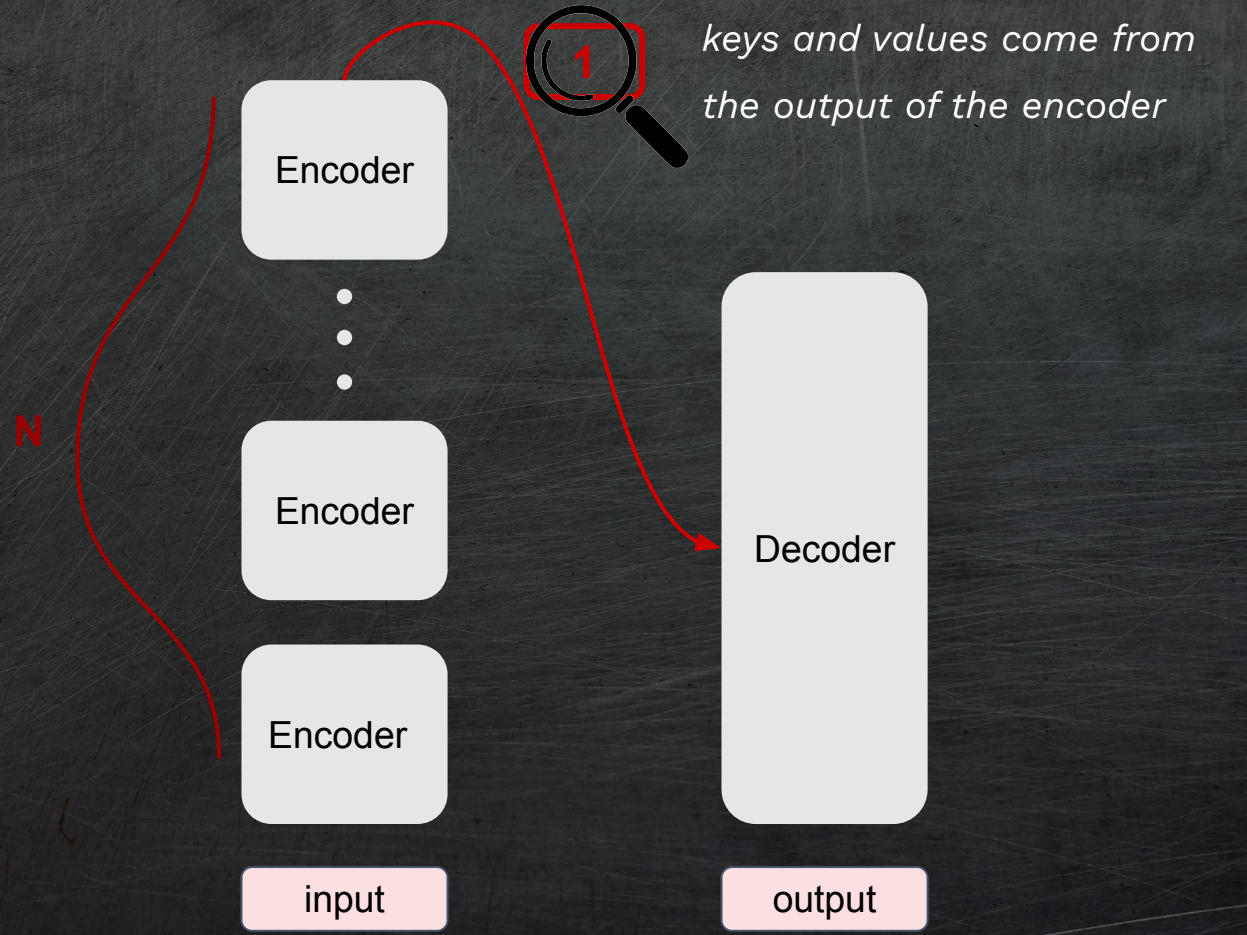
Encoder-Decoder Attention



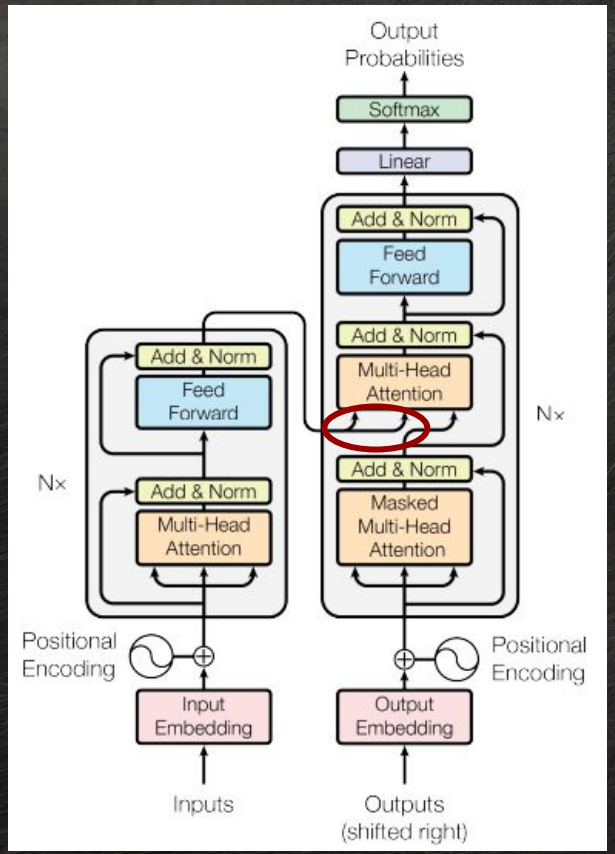


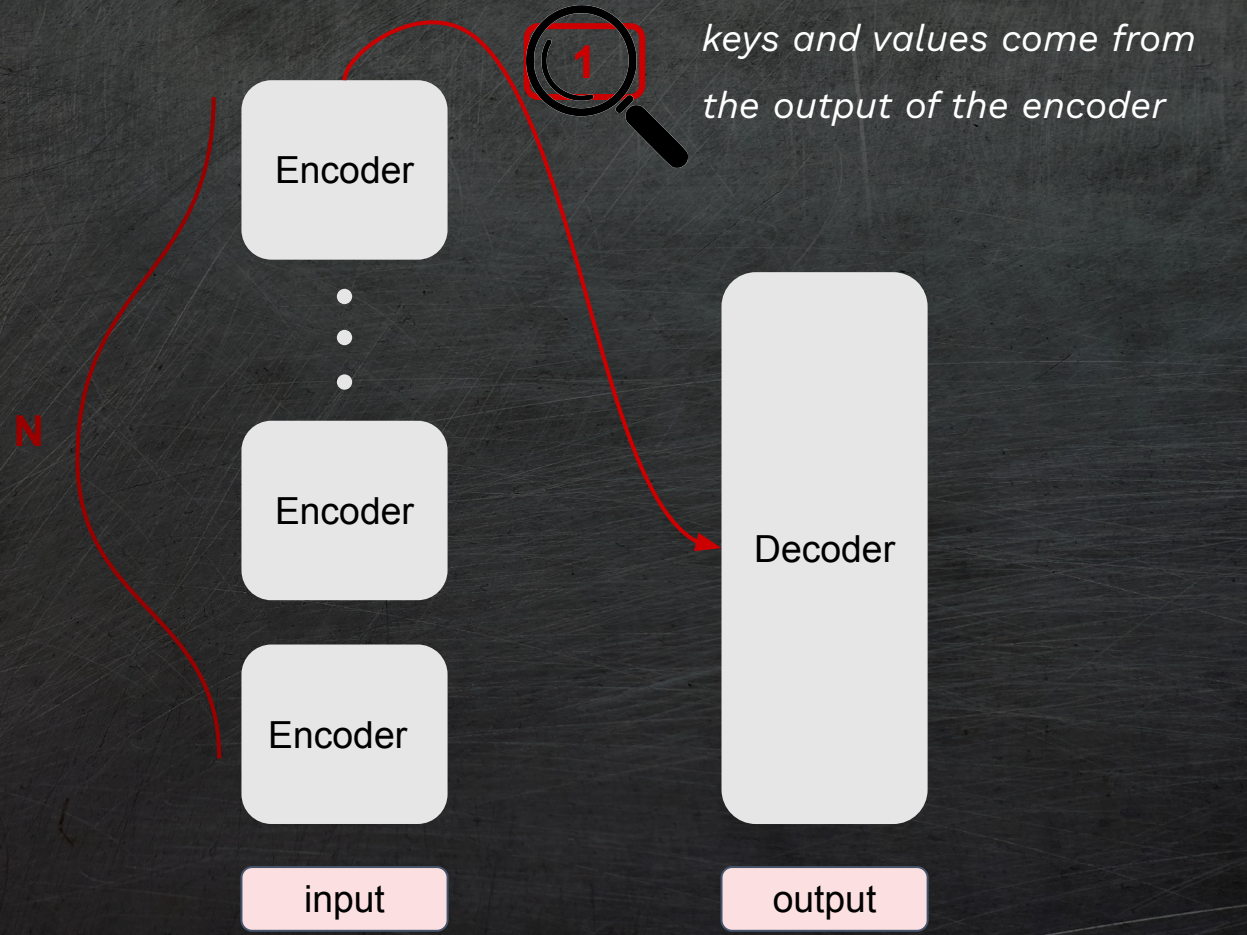
Encoder-Decoder Attention



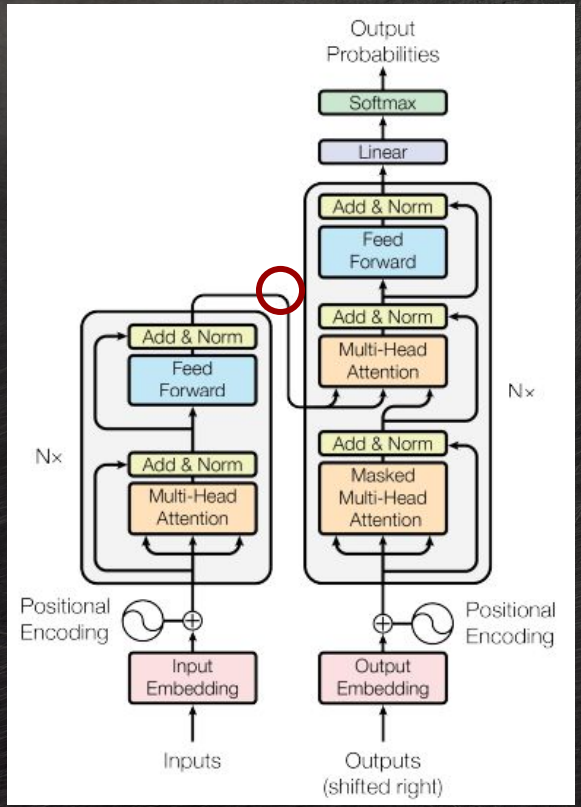


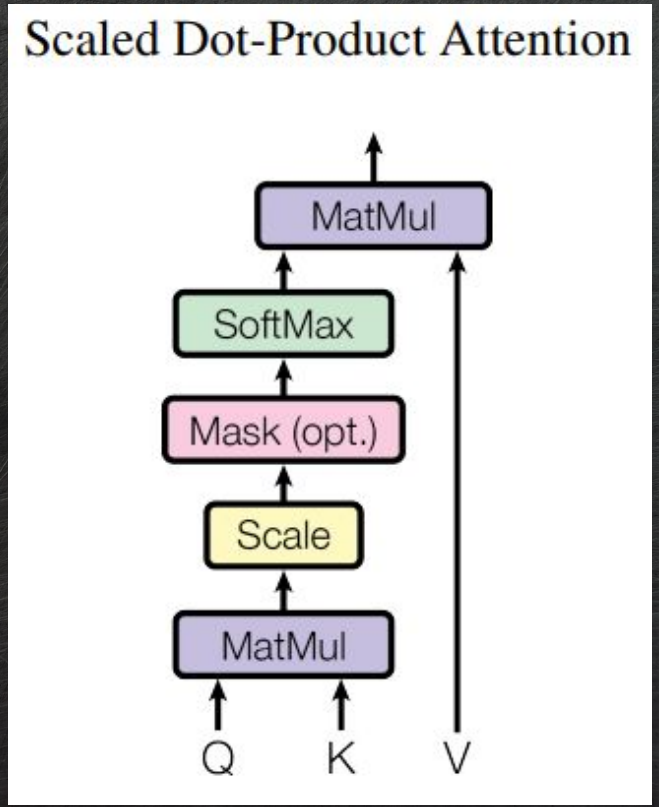
Encoder-Decoder Attention

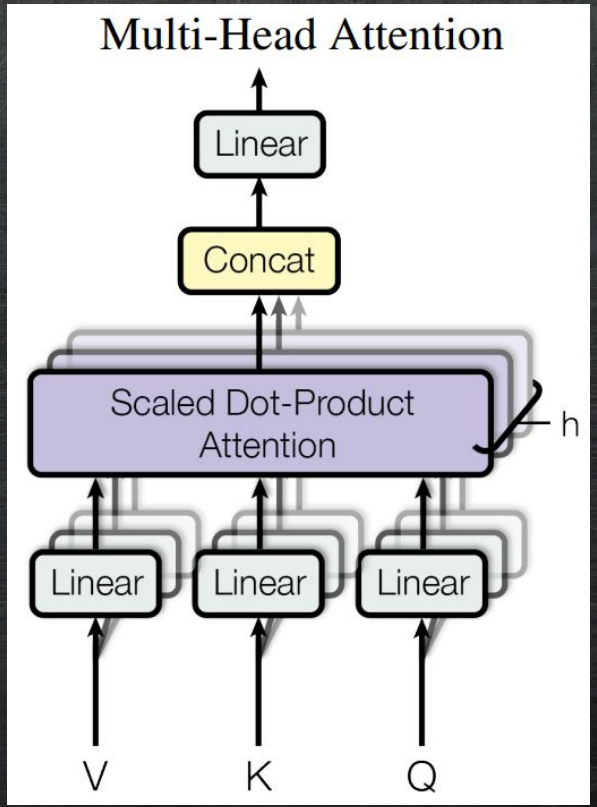




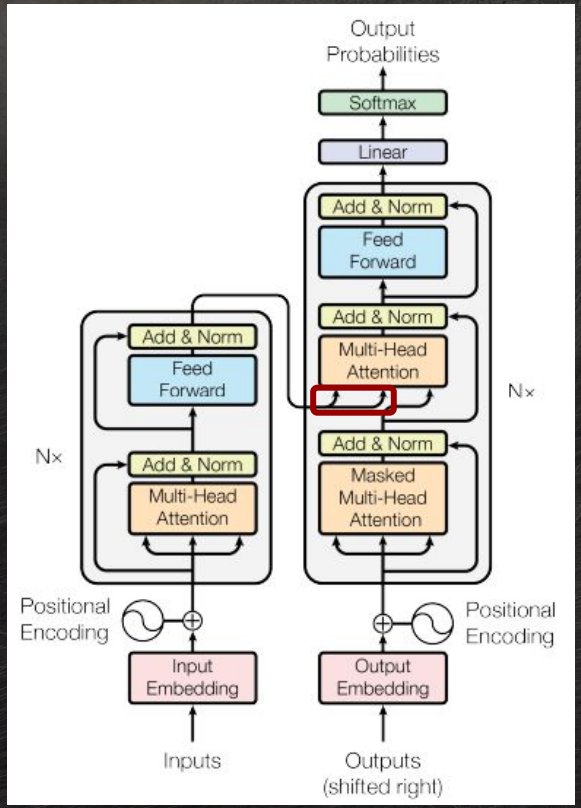
Encoder–Decoder Attention

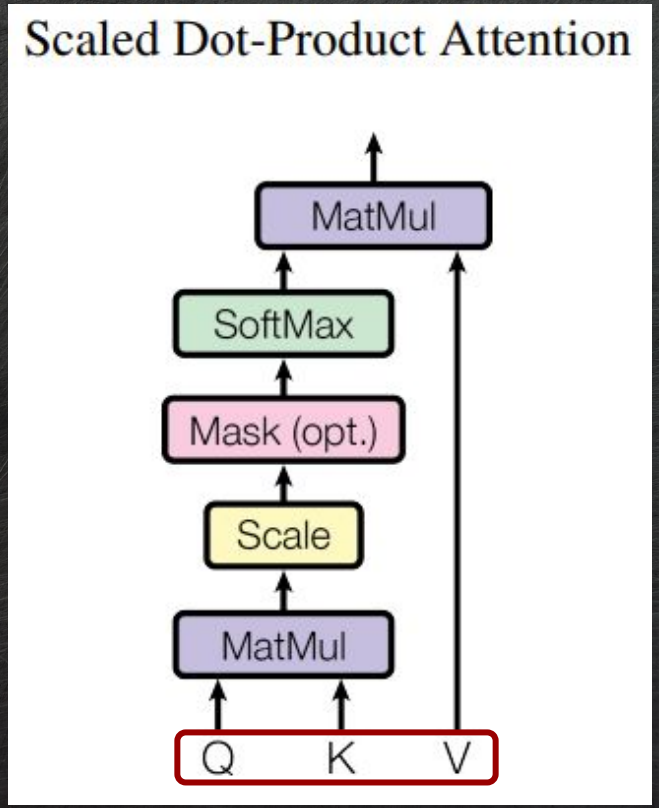


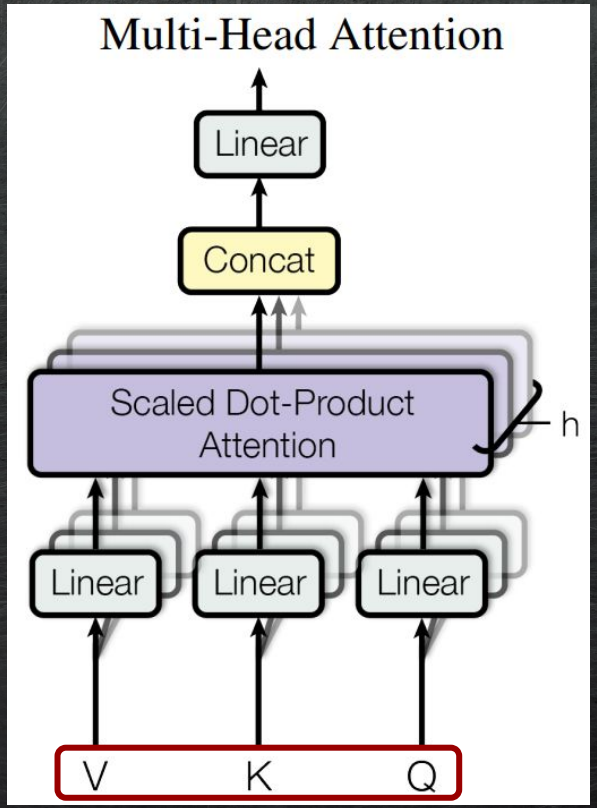




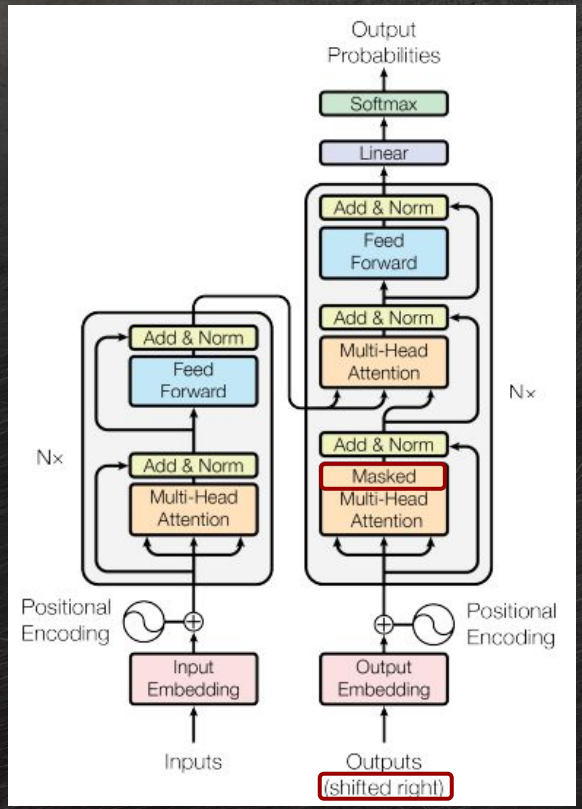
Masked Self-Attention







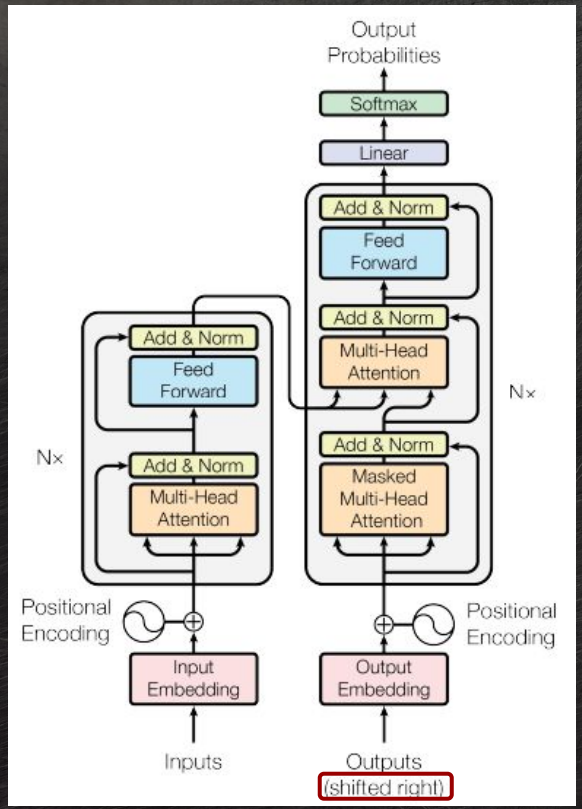
Masked Self-Attention



Shifted right

Masked

Masked Self-Attention



Shifted right

Train 단계이기 때문에, 정답 데이터가 존재.

그래서 ‘I am a student’ 라는 시퀀스 데이터가 존재.

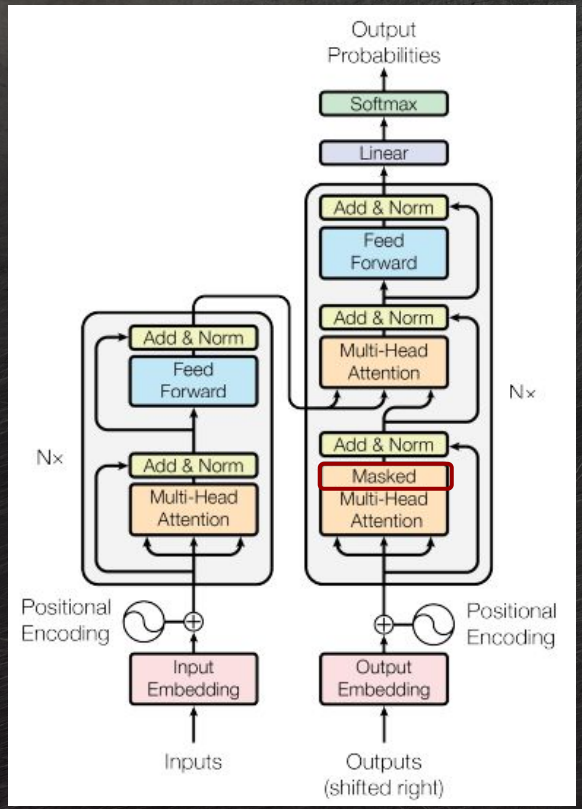
이 시퀀스 데이터를 토큰으로 쪼개서

<SOS>, ‘I’, ‘am’, ‘a’, ‘student’ 로 구성.

<SOS>가 생기면서 정답 데이터의 인덱스가 오른쪽으로 한칸씩 이동

= Shifted right

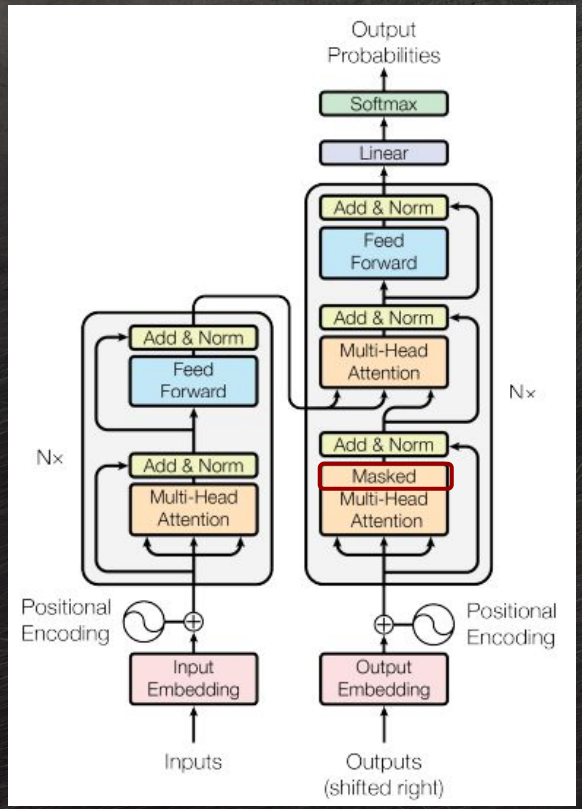
Masked Self-Attention



Masked

	<SOS>	I	am	a	student
<SOS>	7	2	2	2	2
I	1	7	3	2	1
am	1	4	8	4	3
a	1	2	3	6	2
student	1	5	4	1	9

Masked Self-Attention

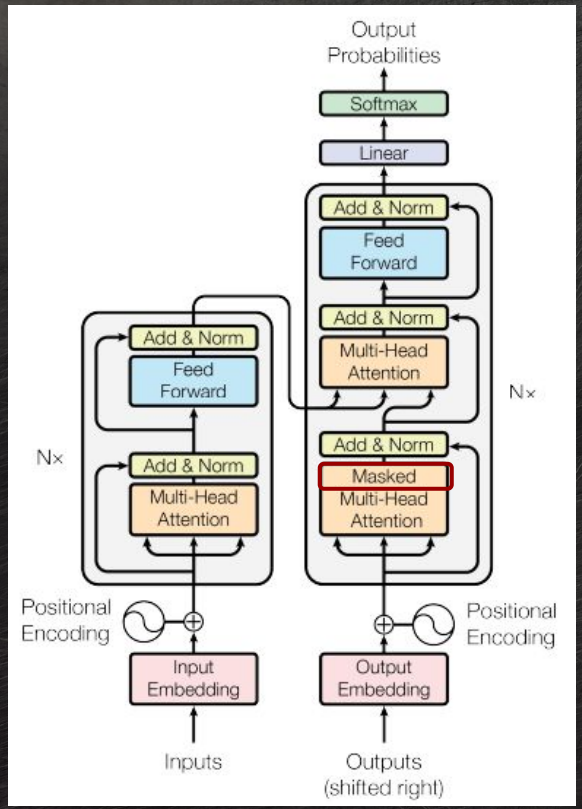


Masked

	<SOS>	I	am	a	student
<SOS>	7	2	2	2	2
I	1	7	3	2	1
am	1	4	8	4	3
a	1	2	3	6	2
student	1	5	4	1	9

	<SOS>	I	am	a	student
<SOS>	7	-∞	-∞	-∞	-∞
I	1	7	-∞	-∞	-∞
am	1	4	8	-∞	-∞
a	1	2	3	6	-∞
student	1	5	4	1	9

Masked Self-Attention



Masked

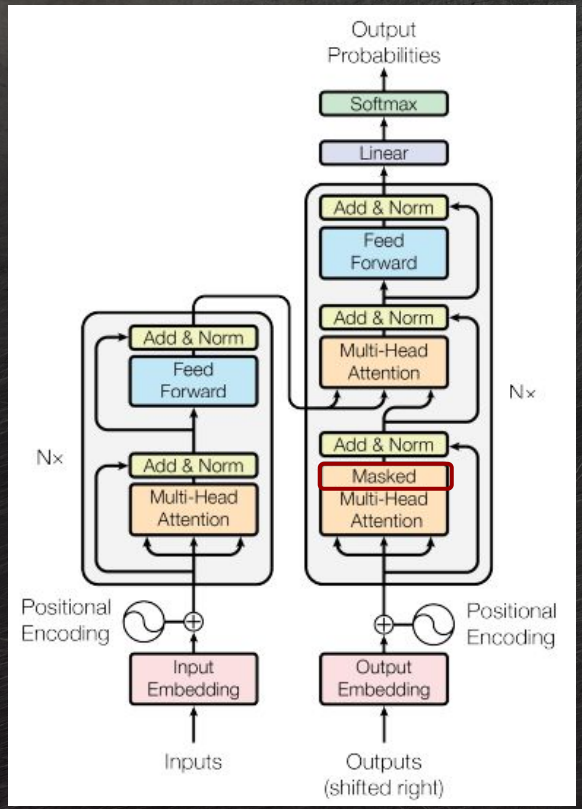
	<SOS>	I	am	a	student
<SOS>	7	2	2	2	2
I	1	7	3	2	1
am	1	4	8	4	3
a	1	2	3	6	2
student	1	5	4	1	9

	<SOS>	I	am	a	student
<SOS>	7	-∞	-∞	-∞	-∞
I	1	7	-∞	-∞	-∞
am	1	4	8	-∞	-∞
a	1	2	3	6	-∞
student	1	5	4	1	9

Q) -무한대는 어디에 적용되는걸까요?

1. Q 2. K 3. V 4. Positional Encoding 5. Attention Score

Masked Self-Attention



Masked

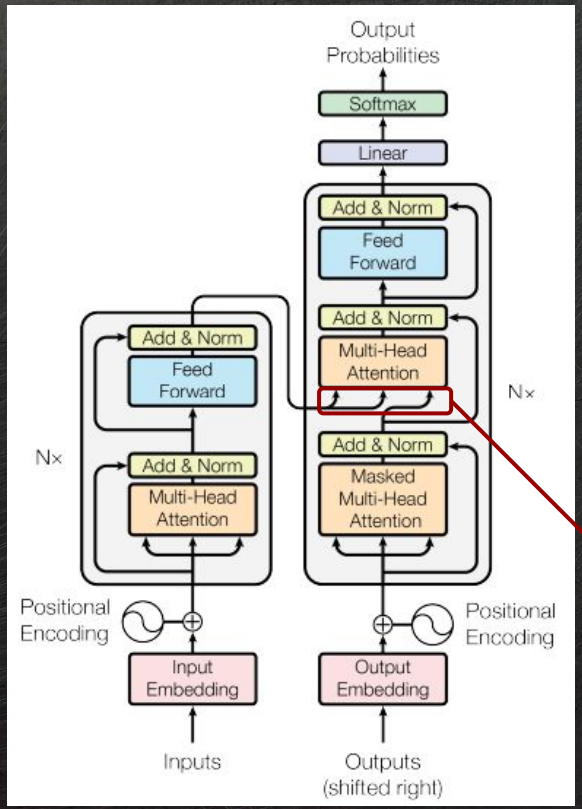
	<SOS>	I	am	a	student
<SOS>	7	2	2	2	2
I	1	7	3	2	1
am	1	4	8	4	3
a	1	2	3	6	2
student	1	5	4	1	9

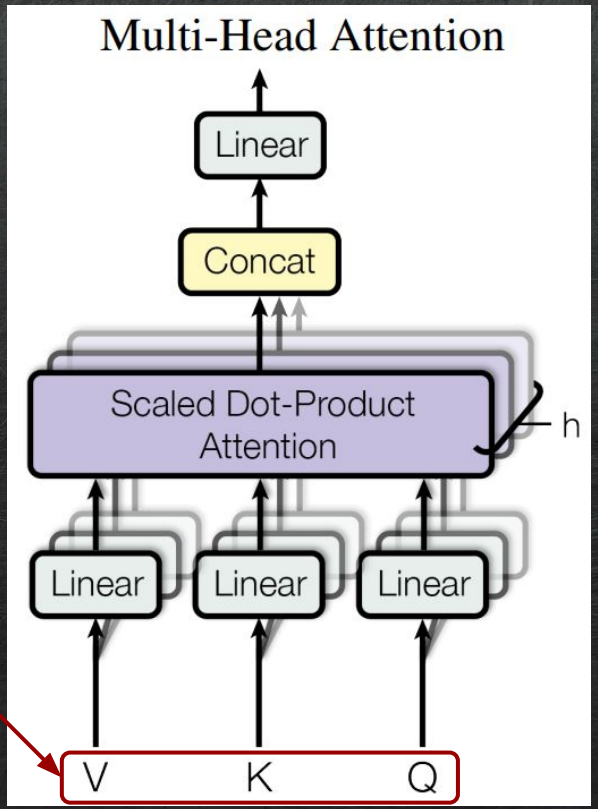
	<SOS>	I	am	a	student
<SOS>	7	-∞	-∞	-∞	-∞
I	1	7	-∞	-∞	-∞
am	1	4	8	-∞	-∞
a	1	2	3	6	-∞
student	1	5	4	1	9

Q) -무한대는 어디에 적용되는걸까요?

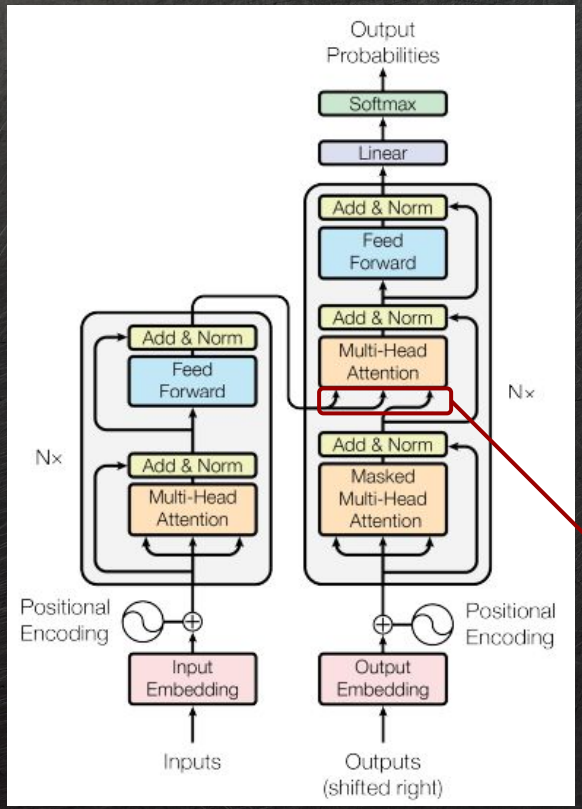
A) 5. Attention Score

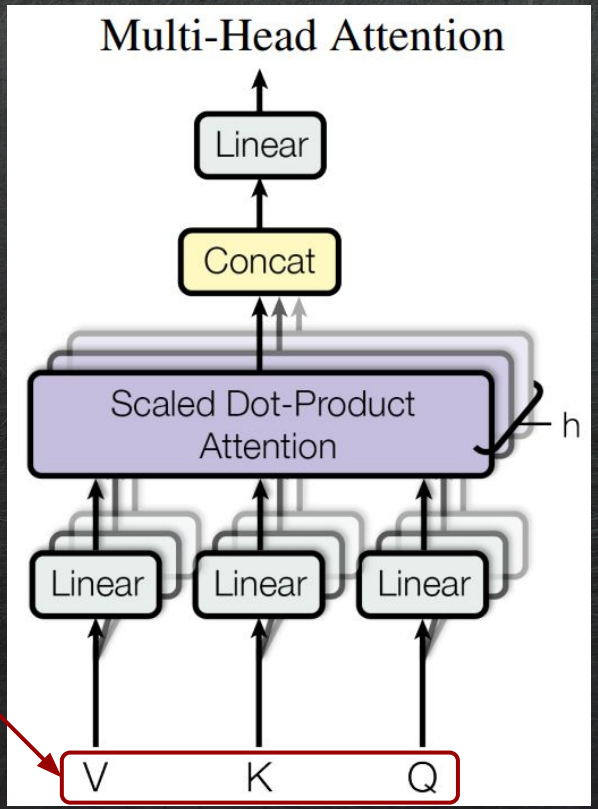
Masked Self-Attention



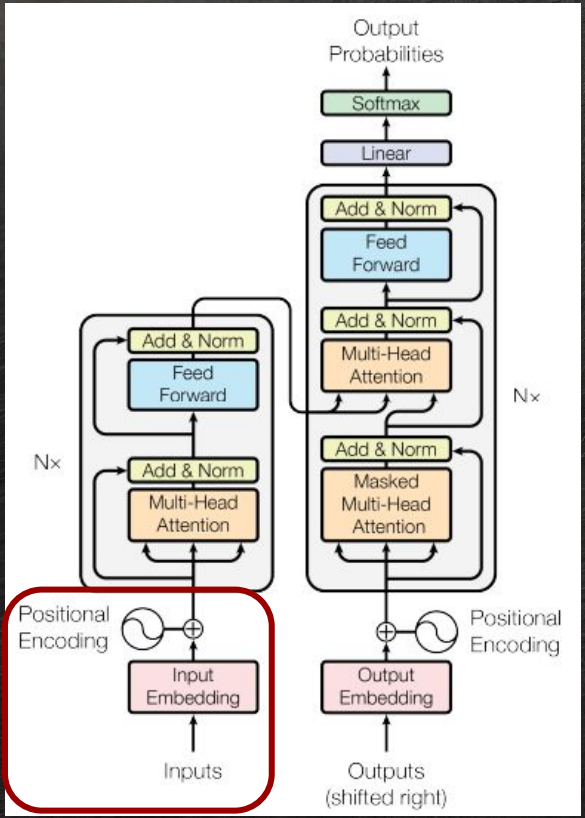


Masked Self-Attention



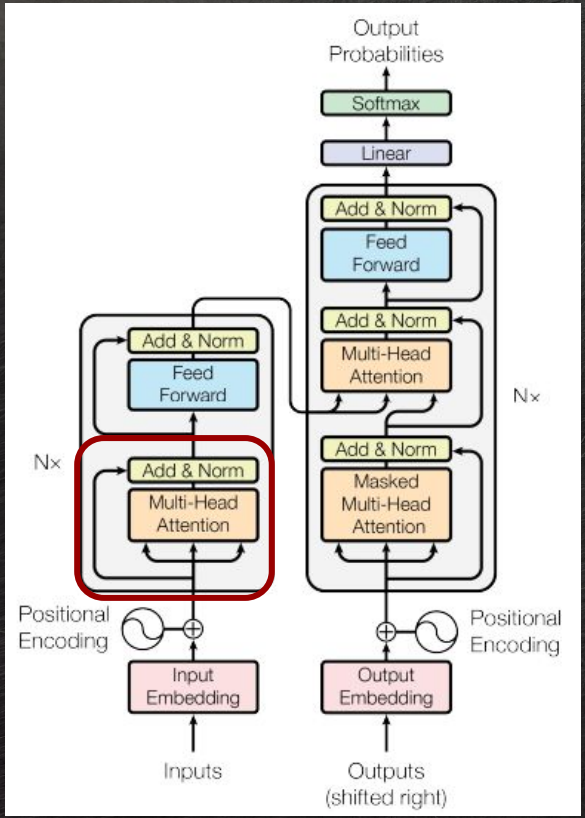


Service



1. 저는 학생입니다.를 input
2. ‘저는 학생입니다’의 각 토큰에 Positional Encoding

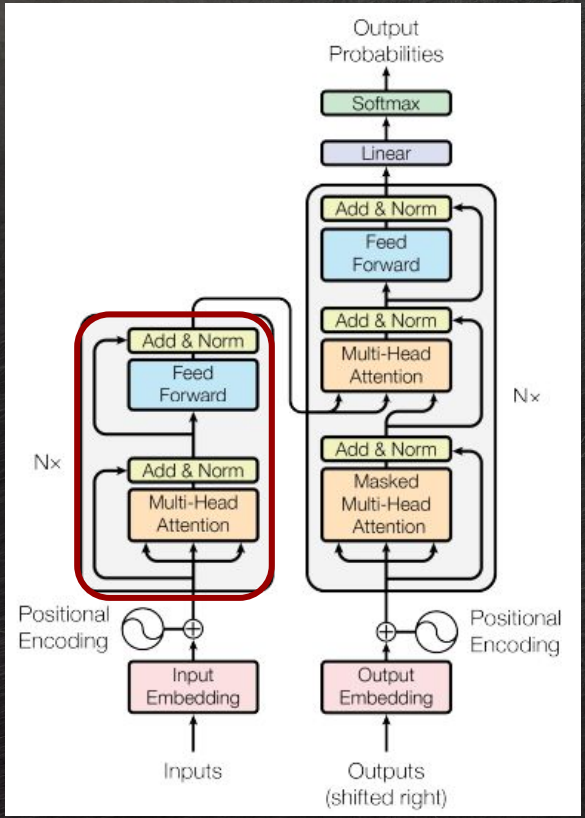
Service



1. 저는 학생입니다.를 input
2. ‘저는 학생입니다’의 각 토큰에 Positional Encoding

1. ‘저는 학생입니다’를 Q, K, V로 변환하여 Attention을 얻음
2. 나온 Q, K, V 행렬을 합연산한 뒤, Layer Norm 진행

Service



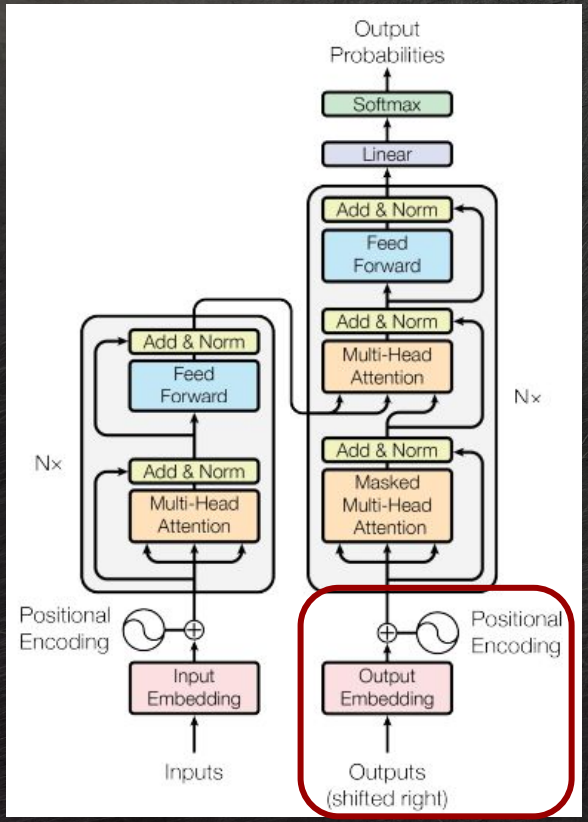
1. 저는 학생입니다.를 input
2. ‘저는 학생입니다’의 각 토큰에 Positional Encoding

1. ‘저는 학생입니다’를 Q, K, V로 변환하여 Attention을 얻음
2. 나온 Q, K, V 행렬을 합연산한 뒤, Layer Norm 진행

1. Fully Connected Feed-Forward With ReLU 진행
2. 나온 Q, K, V를 합연산하고, Layer Norm 진행

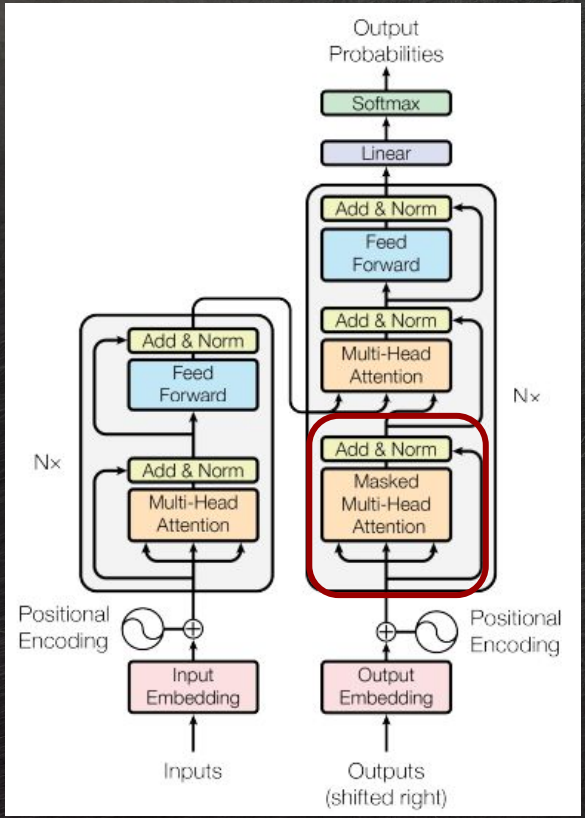
1. 위 과정을 N번 반복 (논문에서는 6회)

Service



1. Output인 'I am a student'가 shifted right됨
2. positional encoding 진행

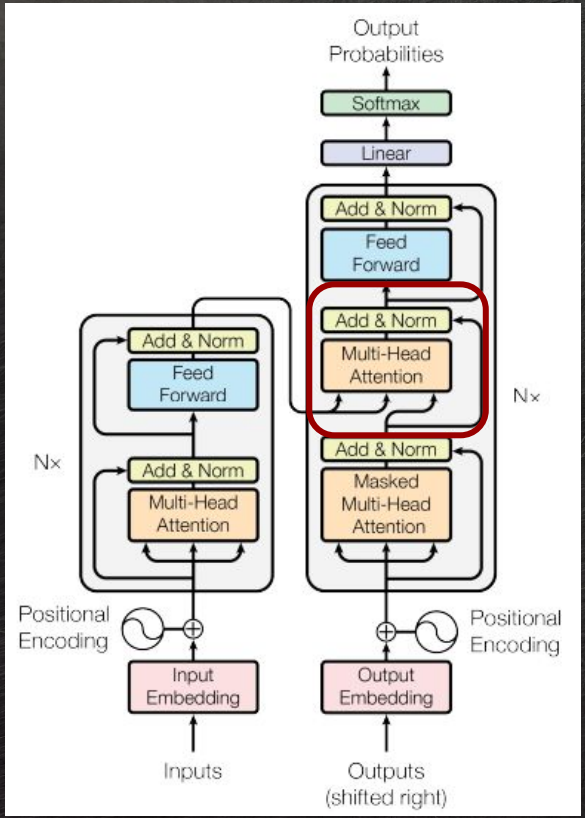
Service



1. Output인 'I am a student'가 shifted right됨
2. positional encoding 진행

1. Masked된 self-attention 진행
2. 나온 Q, K, V값을 합한 뒤, LayerNorm 진행

Service



(Encoder)

1. 위 과정을 N번 반복 (논문에서는 6회)

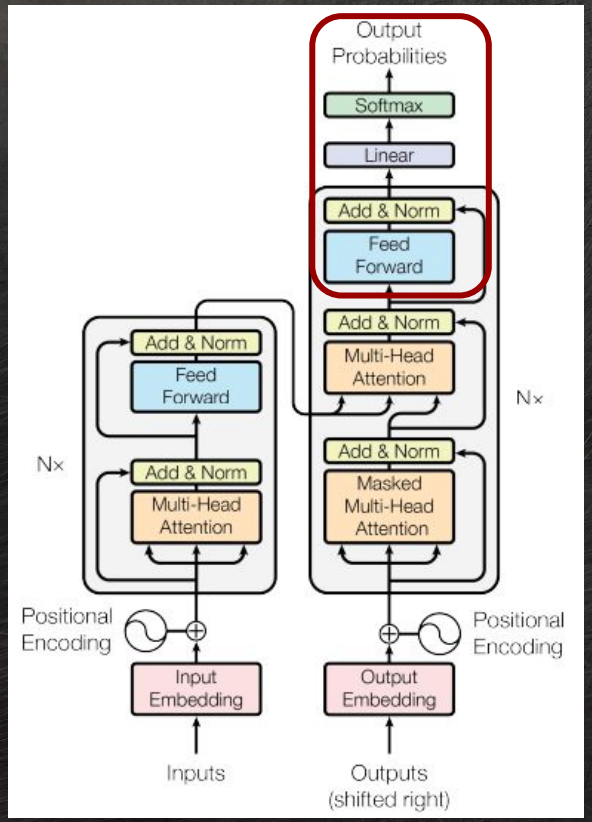
(Decoder)

1. 나온 Q, K, V값을 합한 뒤, LayerNorm 진행

==========

1. Encoder에서 '저는 학생입니다.'의 V, K값을 가져옴
2. Decoder에서 'I am a student'의 Q 값을 가져옴
3. 'I am a student'를 Q로 하는 Encoder-Decoder Attention 진행

Service



(Encoder)

1. 위 과정을 N번 반복 (논문에서는 6회)

(Decoder)

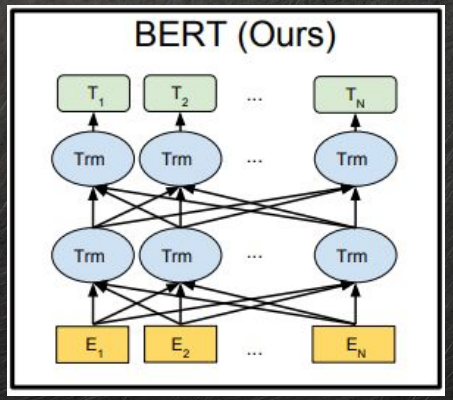
1. 나온 Q, K, V값을 합한 뒤, LayerNorm 진행

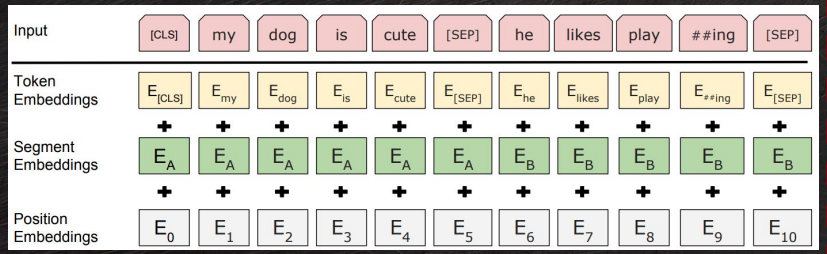
==========

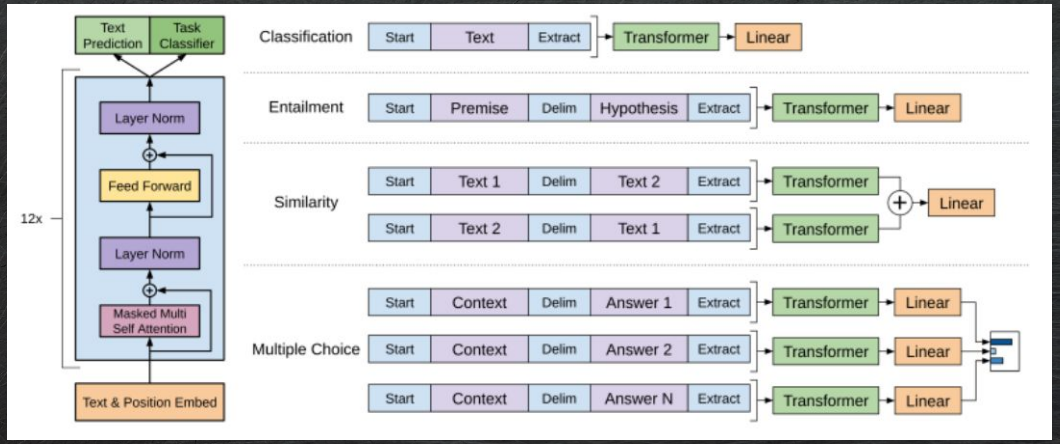
1. Encoder에서 '저는 학생입니다.'의 V, K값을 가져옴
2. Decoder에서 'I am a student'의 Q 값을 가져옴
3. 'I am a student'를 Q로 하는 Encoder-Decoder Attention 진행

1. Fully Connected Feed-Forward with ReLU 진행
2. 합연산 & LayerNorm
3. 위 과정을 N회 반복 후 Decoder를 빠져나감

Service


BERT (Ours)
T1
T2
...
TN
Trm
Trm
...
Trm
Trm
Trm
...
Trm
E1
E2
...
EN


Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
E[CLS]
Emy
Edog
Eis
Ecute
E[SEP]
Ehe
Elikes
Eplay
E##ing
E[SEP]
Segment Embeddings
EA
EB
Position Embeddings
E0
E1
E2
E3
E4
E5
E6
E7
E8
E9
E10


Text Prediction
Task Classifier
12x
Layer Norm
Feed Forward
Layer Norm
Masked Multi Self Attention
Text & Position Embed
Classification
Start
Text
Extract
Transformer
Linear
Entailment
Start
Premise
Delim
Hypothesis
Extract
Transformer
Linear
Similarity
Start
Text 1
Delim
Text 2
Extract
Transformer
Start
Text 2
Delim
Text 1
Extract
Transformer
Linear
Multiple Choice
Start
Context
Delim
Answer 1
Extract
Transformer
Linear
Start
Context
Delim
Answer 2
Extract
Transformer
Linear
Start
Context
Delim
Answer N
Extract
Transformer
Linear

THANK YOU

INSTRUCTION 🥚🔫

조장, PPT 제작, 논문정리, Quest : 손정민

논문정리 및 Quest : 김석영

논문정리 및 Quest : 이효겸

PPT 제작 및 발표 :
윤상현

후기 :

아 진짜 힘들다

근데 재밌었다

발표 오마카세 마음에 드셨으려나