

# 종합 설계

## ( LSTM 모델 구현)

+ Transformer 

# Dataset

write & create

Question Data

Answer Data

Label

**E**  
data

**I**  
data

**E**  
data

**I**  
data

**E/I**

KoGPT-2

## Answer Analysis Model



KoNLPy  
(Morpheme Analysis)

Word Embedding Vector  
values for each Key-word

Sentiment  
Analysis

LSTM

Frequency

Modifier  
(adverbs)

Transformer

Encoder

## 4-BINARY CLASSIFIERS Model

**E/I**  
model

**S/N**  
model

**T/F**  
model

**J/P**  
model

UI

Bot

User

E/I Question

Answer Analysis  
Model

Identifying  
a single MBTI type

User's Answer

Output

: User's MBTI

# 데이터 생성

Index	Answer_Index	Question	Answer	Label
1	1	혼자보다는 다른 사람과 시간을 보내고 싶어 하는가?	그렇다. 다른 사람들과 함께 시간을 보내는 것을 즐기는 편이다.	E/I
	2		난 혼자 있는 시간을 더 선호하는 것 같다. 조용한 게 최고야!	
	...		...	
	250		그래도 사람들과 어울리는 게 더 행복하지?	
2	1	사람들과 만나는 약속이 끝난 후 집에 들어와 혼자 있을 때, 나의 기분은?	약속이 끝나고 집에 돌아와 혼자 있을 때, 대개 나는 안정감과 평온함을 느낀다.	E/I
	2		집에 돌아와 혼자 있는 순간에는 조금 외로움을 느낄 때도 있다.	
	...		...	
	250		집에 돌아와서 집안 정리를 한 뒤 나만의 시간을 갖는 편이다.	
...				

Question Data			write & create		KoGPT-2	합계
E / I	Index	content	직접 작성	chat GPT		
E 	1	새로운 사람들을 만났을 때의 반응은 어떠한가요?			217	217
E 	2	사람들과 함께 있을 때 어떤 느낌을 받나요?		19	69	108
E 	3	혼자 있는 것과 다른 사람들과 함께 있는 것 둘 중 어떤 상황을 더		20		
E 	3	혼자 있는 것과 다른 사람들과 함께 있는 것 둘 중 어떤 상황을 더		30	97	156
E 	3	혼자 있는 것과 다른 사람들과 함께 있는 것 둘 중 어떤 상황을 더		29		
E 	4	당신은 사람이 많이 모인 곳에서 어떻게 행동하나요?	10	20	33	93
E 	4	당신은 사람이 많이 모인 곳에서 어떻게 행동하나요?	10	20		
E 	5	혼자 시간을 보낼 때 주로 어떤 활동을 하나요?		42	33	118
E 	5	혼자 시간을 보낼 때 주로 어떤 활동을 하나요?		43		
E 	6	보통 어떤 방식으로 휴식을 취하는가?		40	33	113
E 	6	보통 어떤 방식으로 휴식을 취하는가?		40		
E 	7	자신의 생일이나 특별한 날을 어떻게 보내고 싶은가?		15	30	59
E 	7	자신의 생일이나 특별한 날을 어떻게 보내고 싶은가?		14		
E 	8	모임에서의 역할은 주로 어떤 것인가요?		18		35
E 	8	모임에서의 역할은 주로 어떤 것인가요?		17		
E 	9	주말에 혼자 집에서 보내는 것과 친구들과 외출하는 것 중 어		20	36	74
E 	9	주말에 혼자 집에서 보내는 것과 친구들과 외출하는 것 중 어		18		
E 	10	사랑 만나기로 했는데 갑자기 약속이 취소되었을 때 기분이 어떠니		19	28	62
E 	10	사랑 만나기로 했는데 갑자기 약속이 취소되었을 때 기분이 어떠니		15		
E 	11	조용한 분위기에서 먼저 나서는 편인가요?		8	23	40
E 	11	조용한 분위기에서 먼저 나서는 편인가요?		9		

총합

1075

## Dataset

write & create

### Question Data

<b>E</b> data	<b>S</b> data	<b>T</b> data	<b>J</b> data
<b>I</b> data	<b>N</b> data	<b>F</b> data	<b>P</b> data

### Answer Data

<b>E</b> data	<b>S</b> data	<b>T</b> data	<b>J</b> data
<b>I</b> data	<b>N</b> data	<b>F</b> data	<b>P</b> data

### Label

<b>E/I</b>
<b>S/N</b>
<b>T/F</b>
<b>J/P</b>

KoGPT-2

EDA

Data Augmentation

Text preprocessing

data = [ Question data\_Answer data\_Label ]

## LSTM

Text preprocessing

Word Embedding  
Model

## Answer Analysis Model



KoNLPy  
(Morpheme Analysis)

Word Embedding Vector  
values for each Key-word

Sentiment  
Analysis

LSTM

Frequency

Modifier  
(adverbs)

## 4-BINARY CLASSIFIERS Model

**E/I**  
model

**S/N**  
model

**T/F**  
model

**J/P**  
model

## UI

Bot

User

E/I Question

Answer Analysis  
Model

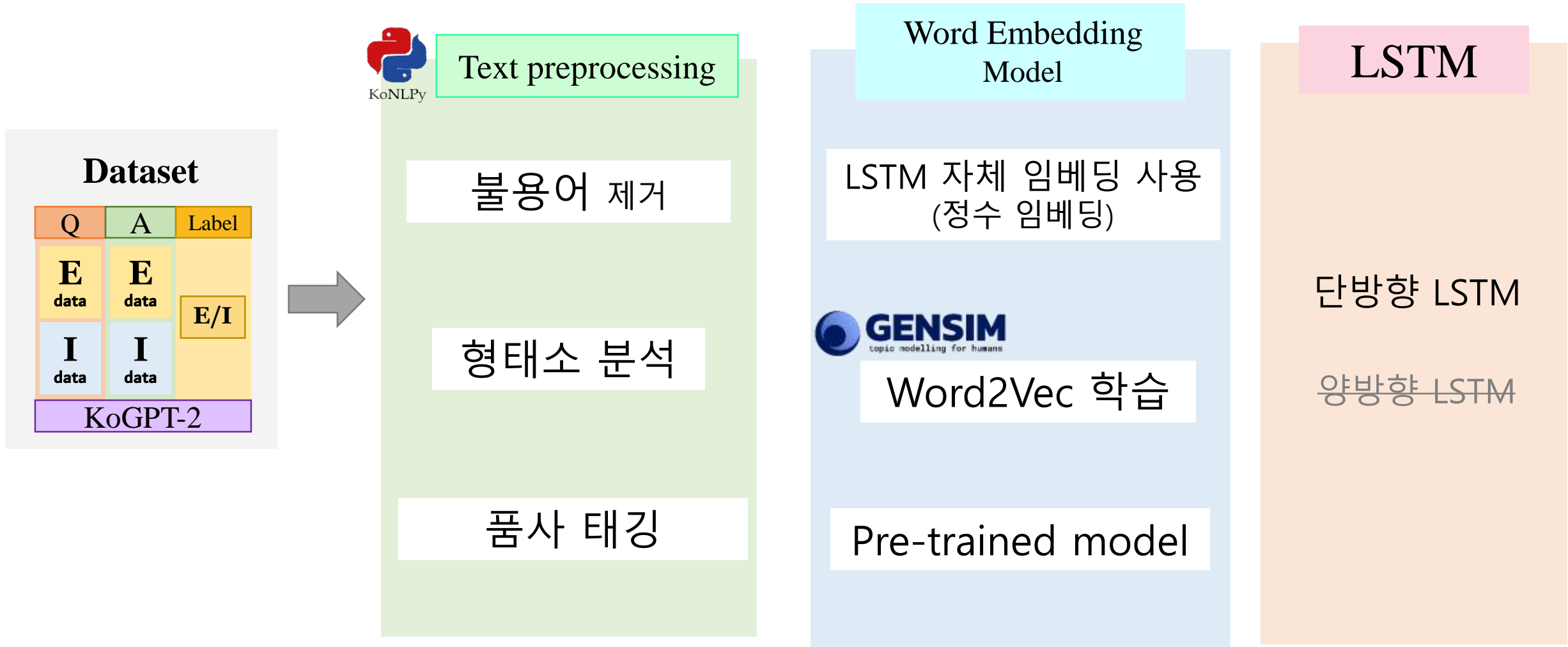
Identifying  
a single MBTI type

User's Answer

Output

: User's MBTI

# LSTM 학습



# LSTM 학습



## Text preprocessing

불용어 제거

형태소 분석

품사 태깅

은, 는, 을, 를, 이, 가 ...

-> 명사나 어간까지 제거되는 부작용

ex) 이발소, 가고있는, 은사님

# LSTM 학습



## Text preprocessing

불용어 제거

형태소 분석

품사 태깅

## KoNLPy 형태소 분석기

- Kkma
- Okt
- hannanum
- Mecab (Not supported on MS Windows)



kkma

mecab

okt

hannanum

# LSTM 학습



Text preprocessing

불용어 제거

형태소 분석

품사 태깅

KoNLPy 품사 태깅

Tag		Tag	
okt		Mecab	
Noun	명사 (Nouns, Pronouns, Company Names, Proper Noun, Person Names, Numerals, Standalone, Dependent)	NNG	일반 명사
		NNP	고유 명사
		NNB	의존 명사
		NNBC	단위를 나타내는 명사
		NR	수사
		NP	대명사
		VV	동사
		VA	형용사
Verb	동사	VX	보조 용언
Adjective	형용사	VCP	긍정 지정사
		VCN	부정 지정사
Determiner	관형사 (ex: 새, 헌, 참, 첫, 이, 그, 저)	MM	관형사
Adverb	부사 (ex: 잘, 매우, 빨리, 반드시, 과연)	MAG	일반 부사
Conjunction	접속사	MAJ	접속 부사
Exclamation	감탄사 (ex: 헐, 어머니, 얼씨구)	IC	감탄사

Tag		Tag	
okt		Mecab	
Josa	조사 (ex: 의, 에, 에서)	JKS	주격 조사
		JKC	보격 조사
		JKG	관형격 조사
		JKO	목적격 조사
		JKB	부사격 조사
		JKV	호격 조사
		JKQ	인용격 조사
		JC	접속 조사
		JX	보조사
PreEomi	선어말어미 (ex: 었)	EP	선어말어미
Eomi	어미 (ex: 다, 요, 여, 하댕ㅋㅋ)	EF	종결 어미
		EC	연결 어미
		ETN	명사형 전성 어미
		ETM	관형형 전성 어미
		XPN	체언 접두사
Suffix	접미사	XSN	명사파생 접미사
		XSV	동사 파생 접미사
		XSA	형용사 파생 접미사

### 3. “아이폰 기다리다 지쳐 애플공홈에서 언락폰질러버렸다 6+ 128기가실버 ㅋ”

어간 구분 탁월

포함되지 않은 단어

-> 일반명사 및 고유명사로 구분

Mecab	Twitter
아이폰 / NNP	아이폰 / Noun
기다리 / VV	기다리 / Verb
다 / EC	다 / Eomi
지쳐 / VV+EC	지쳐 / Verb
애플 / NNP	애플 / Noun
공 / NNG	공홈 / Noun
홈 / NNG	에서 / Josa
에서 / JKB	언락폰 / Noun
언락 / NNG	질 / Verb
폰 / NNG	러 / Eomi
질러버렸다 / VV+EC+VX+EP	버렸다 / Verb
다 / EC	다 / Eomi

# LSTM 학습



## Text preprocessing

불용어 제거

형태소 분석

품사 태깅

## KoNLPy 품사 태깅

> 전부 사용할 것인가?

> 특정 단어만 사용할 것인가?  
(명사, 어간, 동사, 형용사)

대조와 같은 문맥 이해 어렵다는 한계

ex) 나는 친구를 만나는 걸 좋아하지만,

그래도 집에 혼자만의 시간을 보내는 게 더 좋다.

# LSTM 학습

## Word Embedding Model

### LSTM 자체 임베딩 사용 (정수 임베딩)

A	label
대인관계 능력 향상과 어울림의 즐거움을 얻는 것 같다	1
옴티와 같이 사람들이 모인 장소에 가서 많은 이들과 대화를 나눌 때 가장 어렵다	0
혼자서 에어팟을 끼고 독백을 즐기며 혼밥에 혼자 사진 찍는 것이 멋이라고 생각한다	0

입력문장 : < 대인관계 능력 향상과 어울림의 즐거움을 얻는 것 같다. >

[[81.77594]]% 정도로 E입니다.

입력문장 : < 옴티와 같이 사람들이 모인 장소에 가서 많은 이들과 대화를 나눌 때 가장 어렵다 >

[[60.11038]]% 정도로 E입니다.

입력문장 : < 혼자서 에어팟을 끼고 독백을 즐기며 혼밥에 혼자 사진 찍는 것이 멋이라고 생각한다 >

[[90.534744]]% 정도로 I입니다.

# LSTM 학습

Word Embedding  
Model

LSTM 자체 임베딩 사용  
(정수 임베딩)

Word2Vec 학습

Pre-trained model



Dataset

Q	A	Label
<b>E</b> data	<b>E</b> data	<b>E/I</b>
<b>I</b> data	<b>I</b> data	
KoGPT-2		

**overfitting**

# LSTM 학습

Word Embedding  
Model

LSTM 자체 임베딩 사용  
(정수 임베딩)

Word2Vec 학습

Pre-trained model

word2vec-ko

FastText-KR

Ko-BERT

한국어 ELMO



# FastText

**subword**(내부단어)의 존재

$n = 3$

글자단위	안녕하세요	=	<안녕	안녕하	녕하세	하세요	세요>				
자모단위	안녕하세요	=	<ㅇ	ㅏ	ㅇ	ㅏ	ㄴ	ㅏ	ㄴ	ㄴ	...

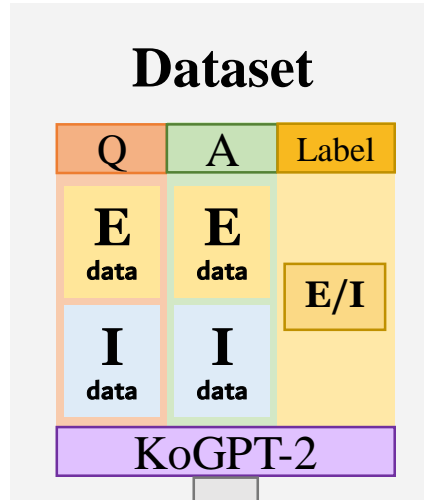


OOV (Out of Vocabulary) 에 대한 대응 ↑

안녕하삼, 안녕함, 안녕하롱, 안뇽, 안녕안녕

# Pre-trained FastText -

KR



Pre-trained  
FastText - KR

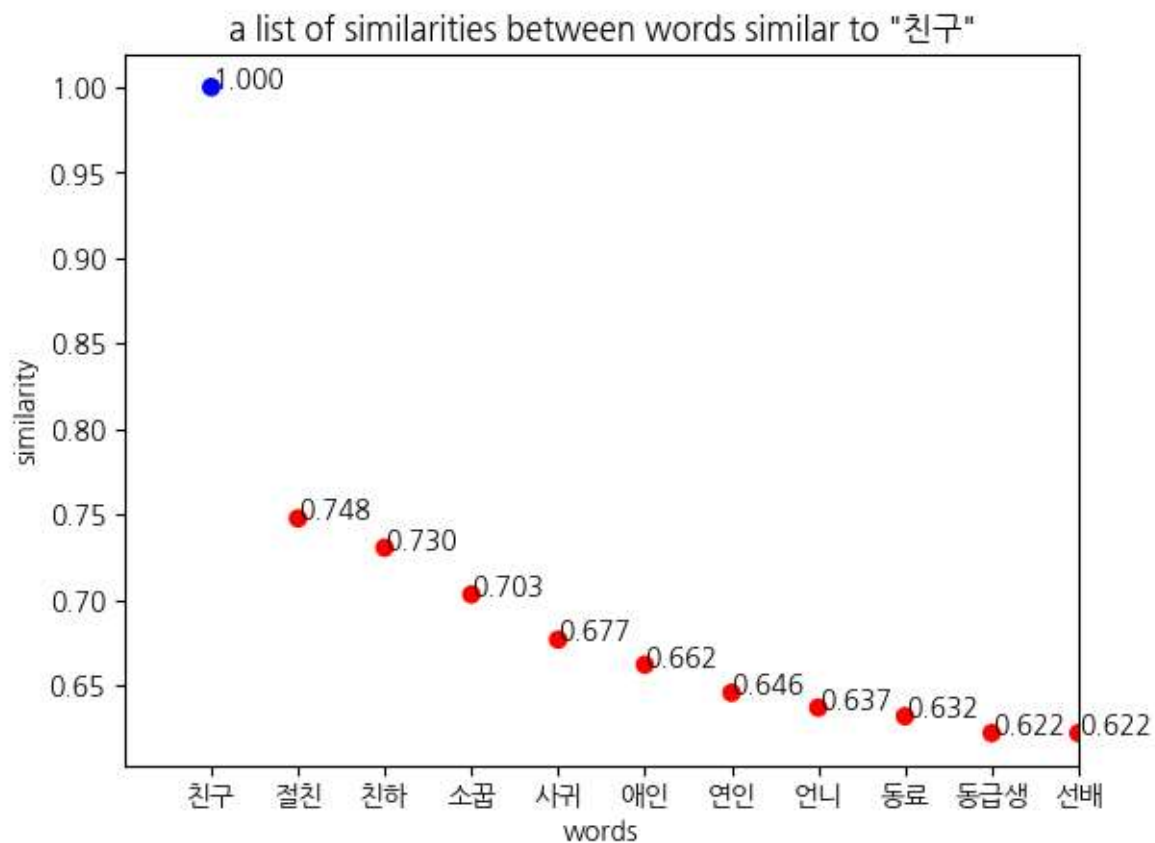


**fine-tuning**

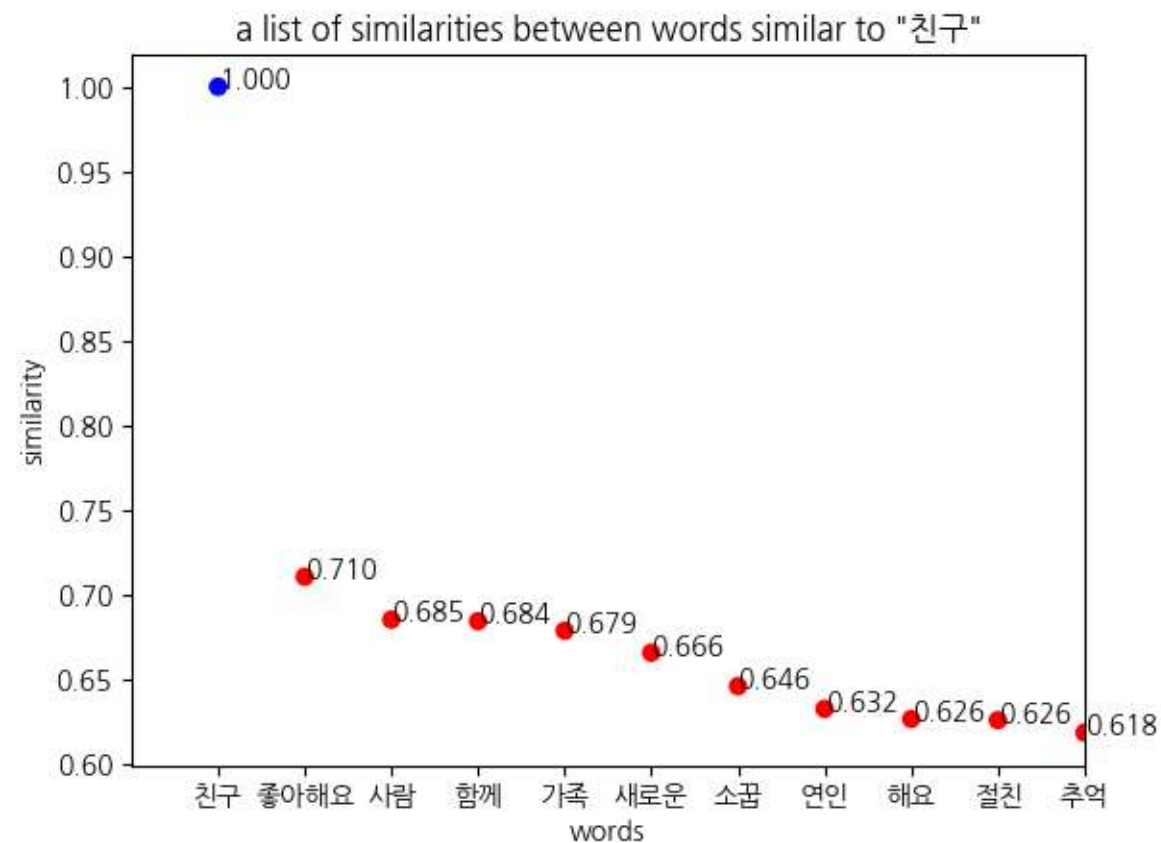
- overfitting  
방지
- 1) Learning rate 조절
  - 2) Data Augmentation
  - 3) Dropout

new\_FastText  
model

## Pre-trained FastText - KR



## new\_FastText model



# LSTM 학습

## LSTM

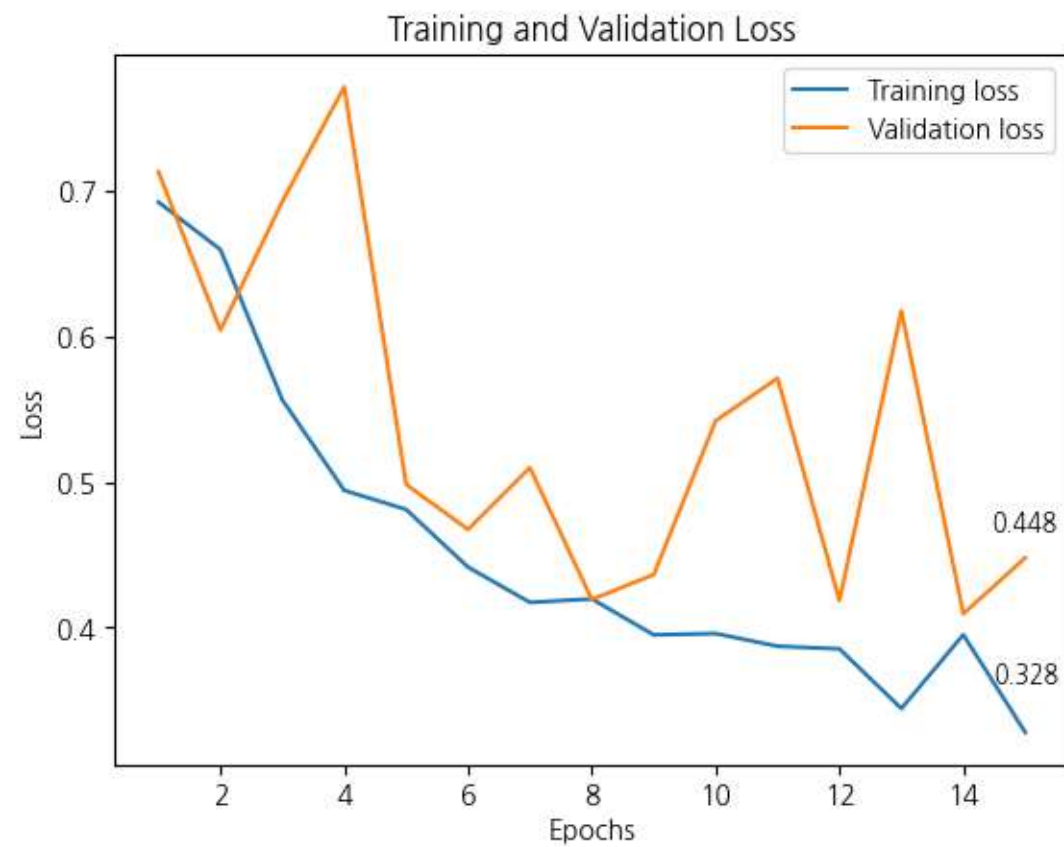
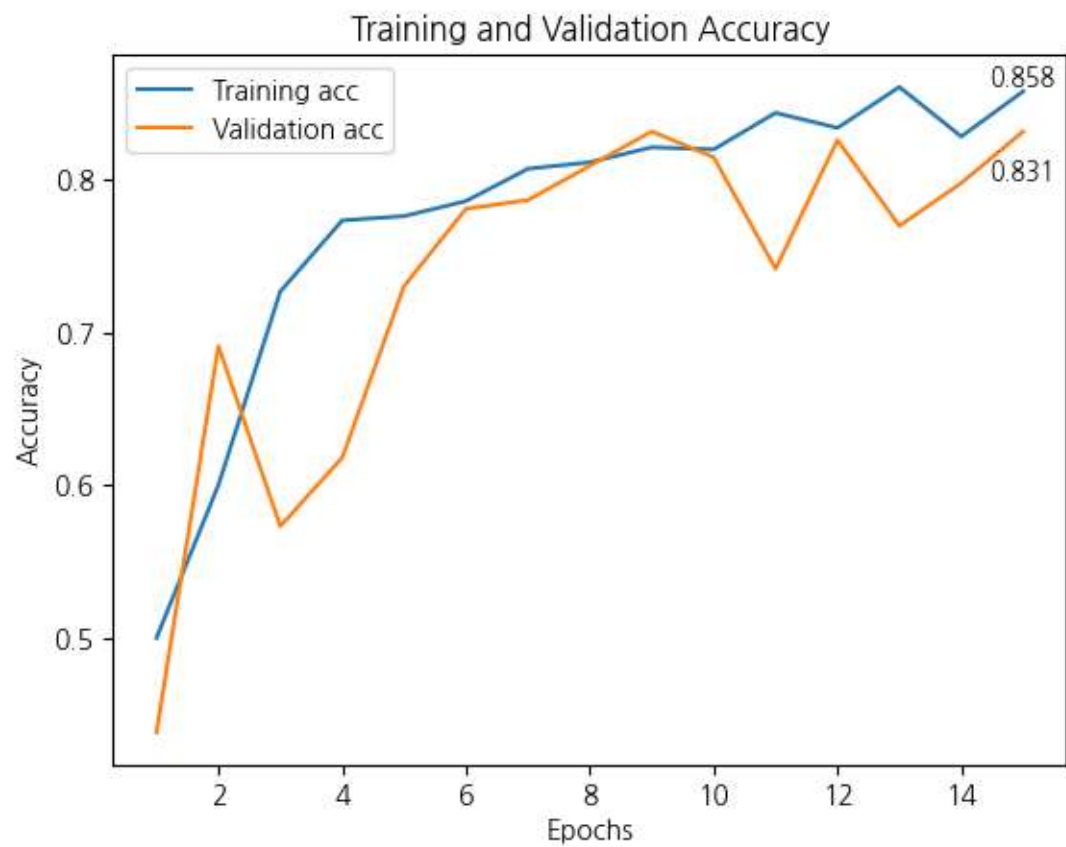
입력문장 : < 혼자 있는 걸 선호한다.. >  
[[98.932755]]% 정도로 1입니다.

입력문장 : < 보통 혼자 시간을 보내곤 하지? >  
[[98.95439]]% 정도로 1입니다.

입력문장 : < 조용한 카페나 독서실이 나에게 가장 적합한 장소이다. >  
[[79.28607]]% 정도로 0입니다.

입력문장 : < 나는 친구들이 많은게 너무너무 좋다. >  
[[83.62886]]% 정도로 1입니다.

오류



엠티

0/0

## LSTM model

입력문장 : < 엠티와 같이 사람들이 모인 장소에 가서 많은 이들과 대화를 나눌 때 가장 어렵다 >

[[60.11038]]% 정도로 E입니다.

## new\_FastText model

pre-trained Embedding Model  
+ LSTM

입력문장 : < 엠티와 같이 사람들이 모인 장소에 가서 많은 이들과 대화를 나눌 때 가장 어렵다. >  
[[61.712944]]% 정도로 I입니다.

대인관계

0/0

LSTM model

입력문장 : < 대인관계 능력 향상과 어울림의 즐거움을 얻는 것 같다. >

[[81.77594]]% 정도로 E입니다.

new\_FastText model

pre-trained Embedding Model  
+ LSTM

입력문장 : < 대인관계 능력 향상과 어울림의 즐거움을 얻는 것 같다. >

[[66.93203]]% 정도로 E입니다.

혼밥|

0/0

## LSTM model

입력문장 : < 혼자서 에어팟을 끼고 독백을 즐기며 혼밥에 혼자 사진 찍는 것이 멋이라고 생각한다 >

[[90.534744]]% 정도로 1입니다.

## new\_FastText model

pre-trained Embedding Model  
+ LSTM

입력문장 : < 혼자서 에어팟을 끼고 독백을 즐기며 혼밥에 혼자 사진 찍는 것이 멋이라고 생각한다 >

[[83.14197]]% 정도로 1입니다.



# 해결할 점

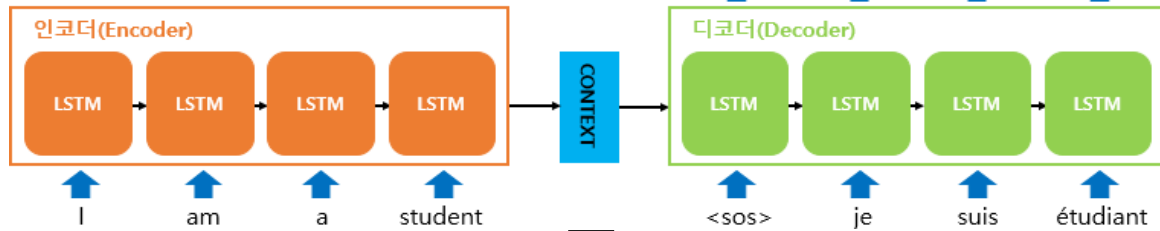
1. 과적합 문제 해결
2. 품사 태깅 문제 - 비교하기
3. 다른 pre-trained 임베딩 모델 사용
  - > Bi - LSTM



# Transformer - Encoder

23.08.29 유하영

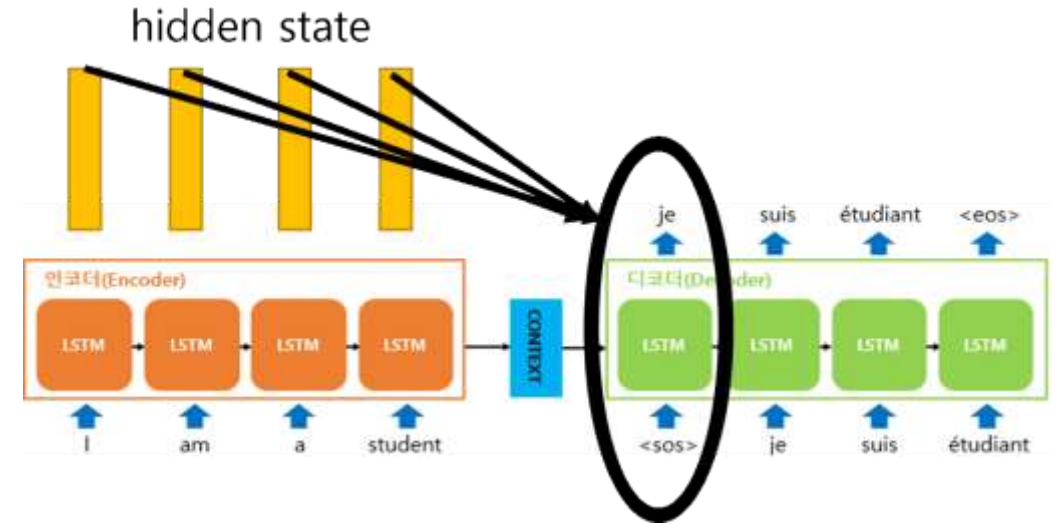
# seq2seq



Encoder의 마지막 hidden state만  
참조해 출력

-> 정보 손실

# Attention



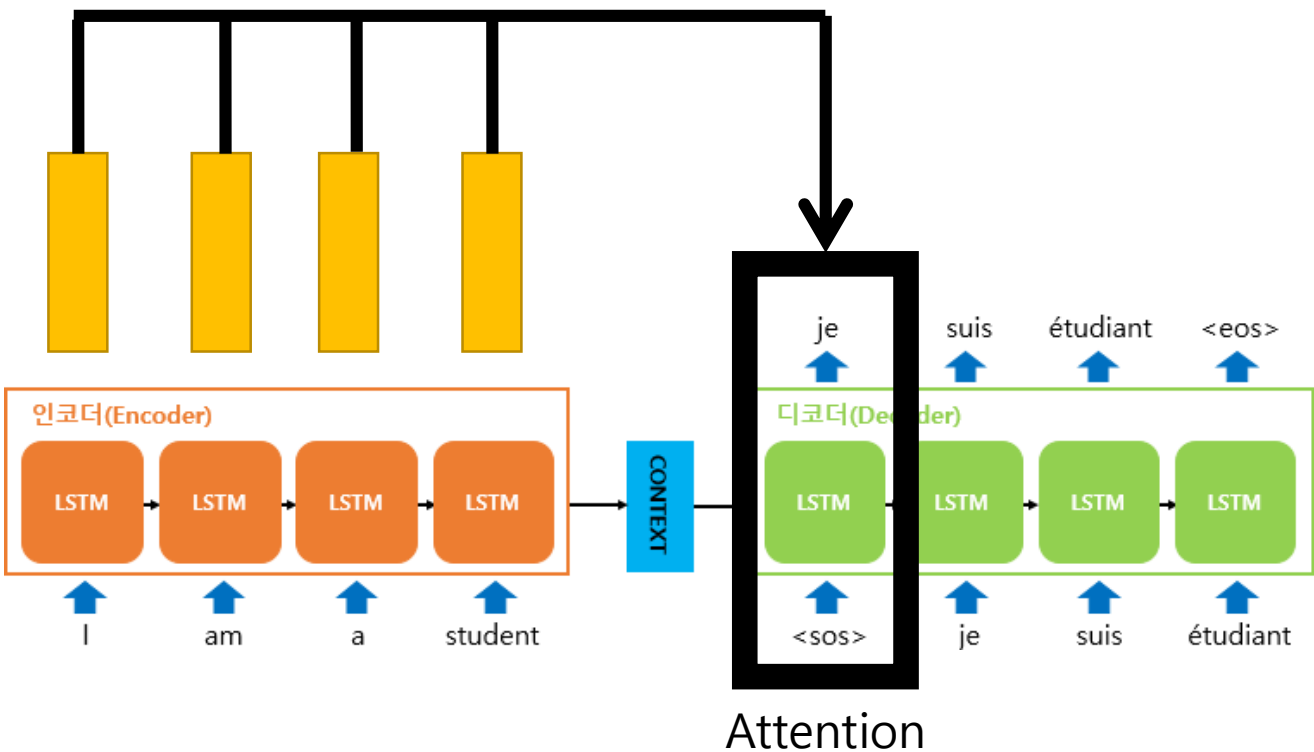
Decoder의 매 timestep마다  
Encoder의 모든 hidden state 값들을

**Attention** 한다

# Attention mechanism

## 1) Attention score

dot product




## 2) Attention distribution

$$\text{softmax()} \\ = [0.4 \ 0.4 \ 0.1 \ 0.1]$$

## 3) Attention value

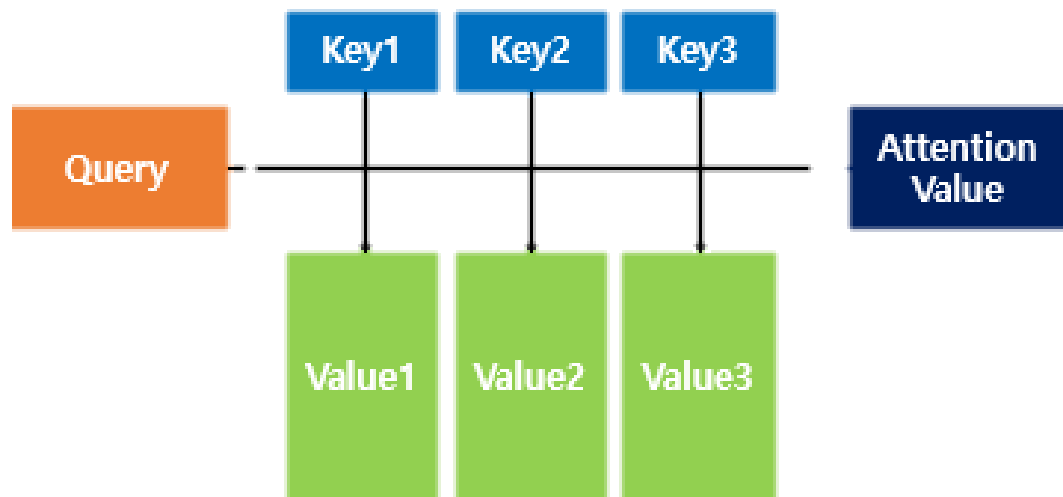
$$\Sigma \left[ \begin{array}{cc} \text{word's hidden state} & \text{attention value} \end{array} \right]$$

word's hidden state	attention value
je	0.4
suis	0.4
étudiant	0.1
<eos>	0.1

 : word's hidden state

**Attention(Q,K,V) = Attention Value**

\* 어텐션 메커니즘 구성요소를  
간결하게 나타냄



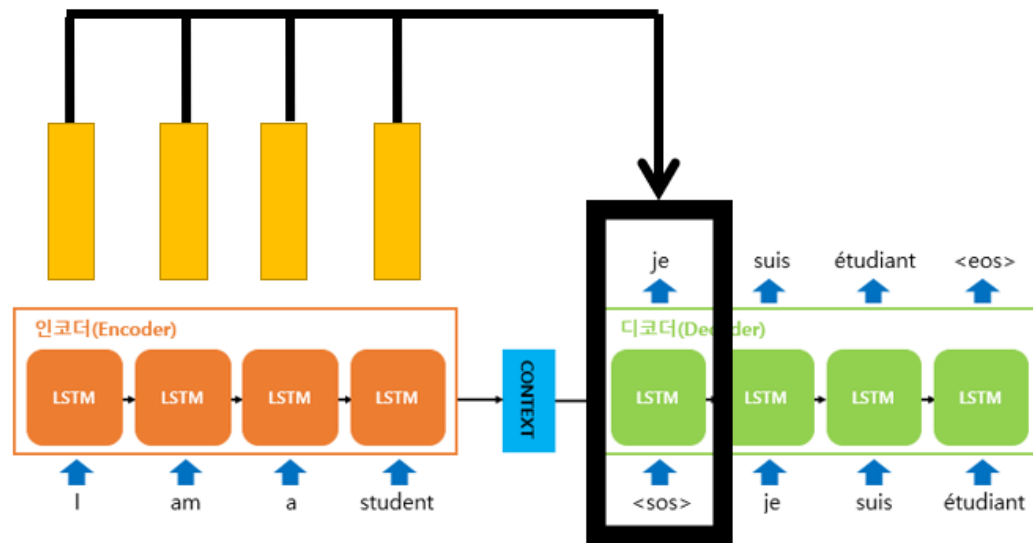
**value**

집중할 위치의 **정보**  
->가중치로 최종 값 계산

**key**

집중할 **위치** 결정 -> 가중치(연관성)를 계산

: 인코더의 은닉상태



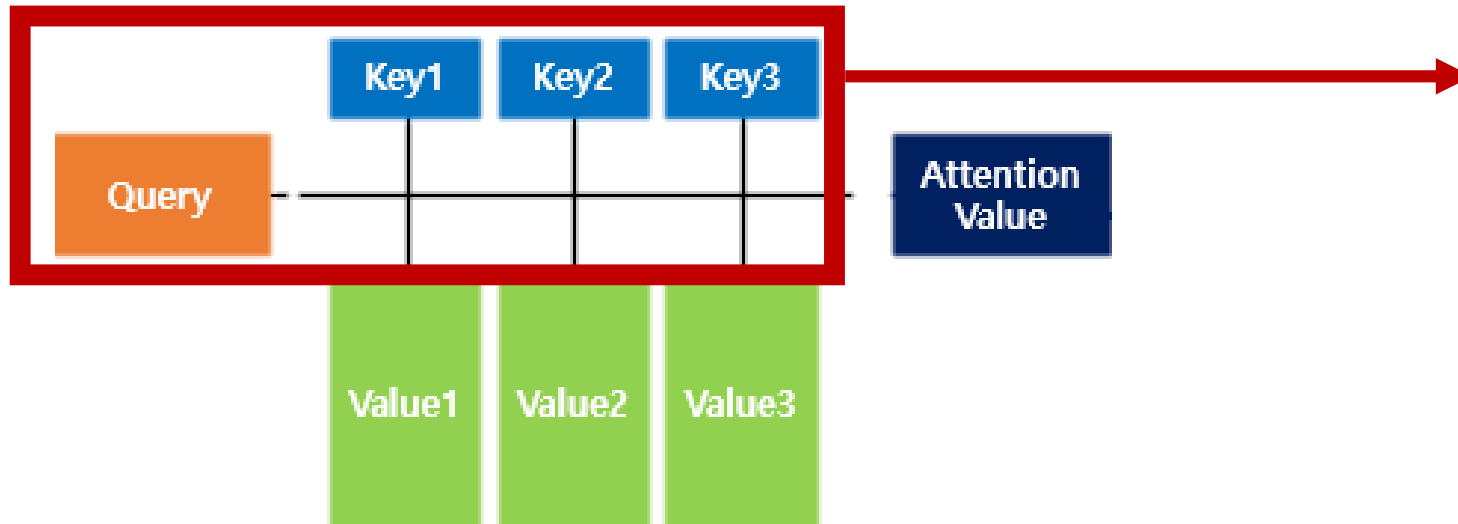
Attention

**Query**

: 디코더의 은닉상태

**Attention(Q,K,V) = Attention Value**

\* 어텐션 메커니즘 구성요소를  
간결하게 나타냄

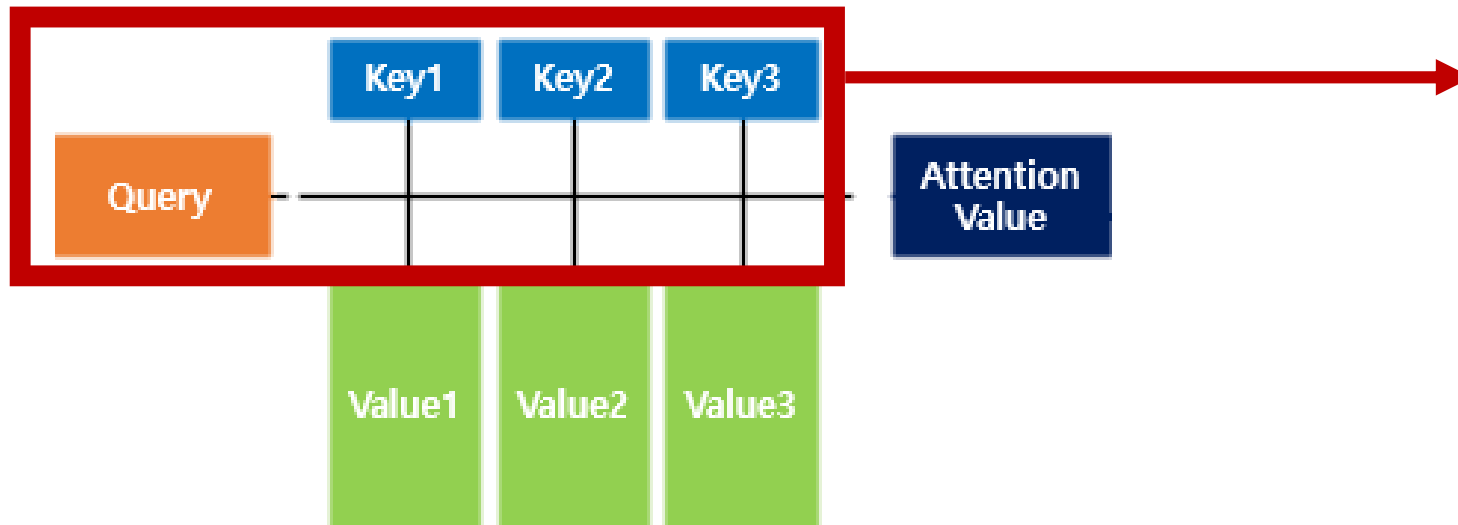


### 1) Attention score

Query와 Key 사이의  
유사도 계산

**Attention(Q,K,V) = Attention Value**

\* 어텐션 메커니즘 구성요소를  
간결하게 나타냄

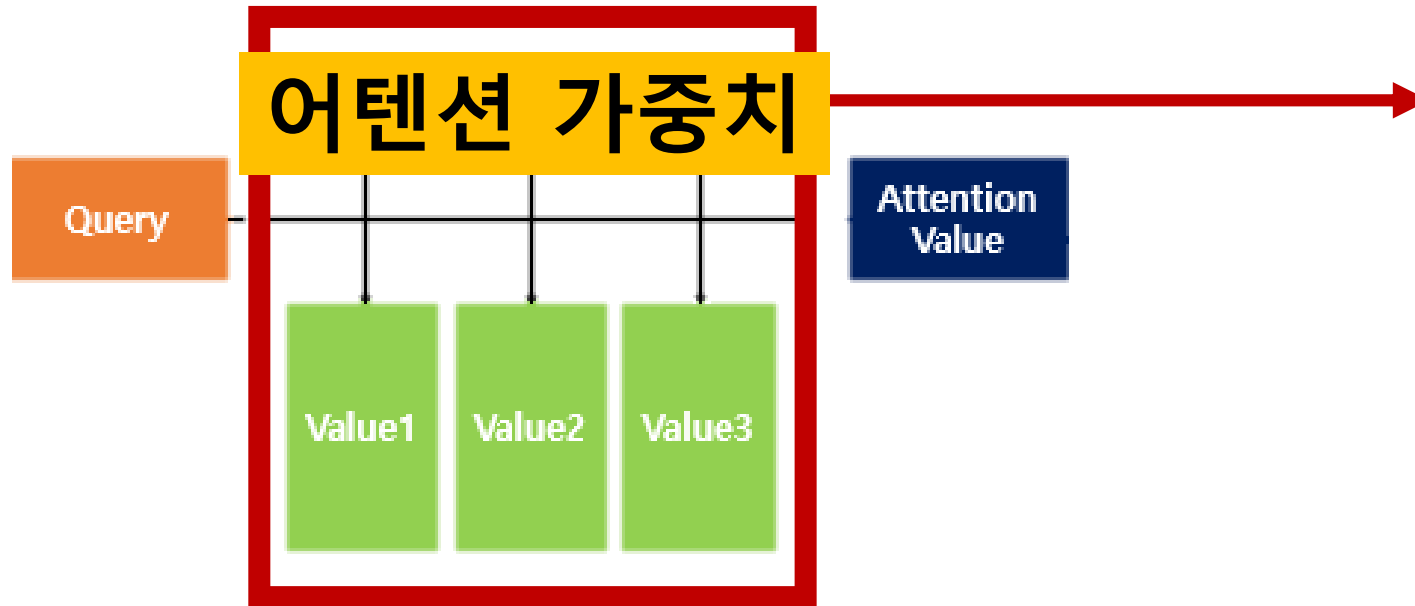


## 2) Attention distribution

softmax 함수를 적용하여  
**어텐션 가중치**를 얻음

**Attention(Q,K,V) = Attention Value**

\* 어텐션 메커니즘 구성요소를  
간결하게 나타냄



### 3) Attention value

어텐션 가중치를  
**value**에 가중합하여  
Attention value를 얻는다



## Attention

하나의 벡터로 압축하는 과정에서  
여전히 **정보 손실**



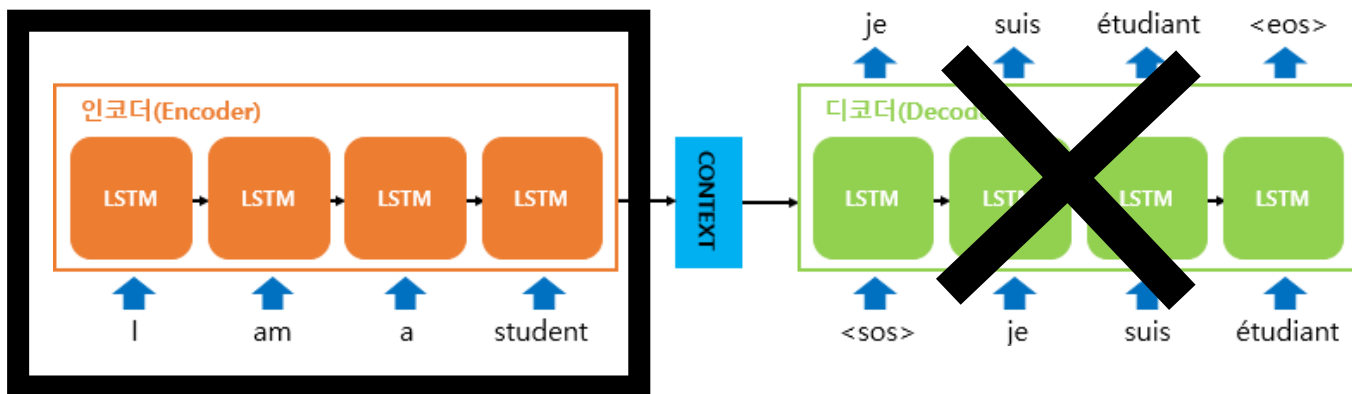
## self - Attention

인코더 문장 내에서  
단어들 간의 관계를 고려하여  
**어텐션 계산**

RNN없이 **Attention**만으로  
**Encoder와 Decoder**을 만들자

# self - Attention

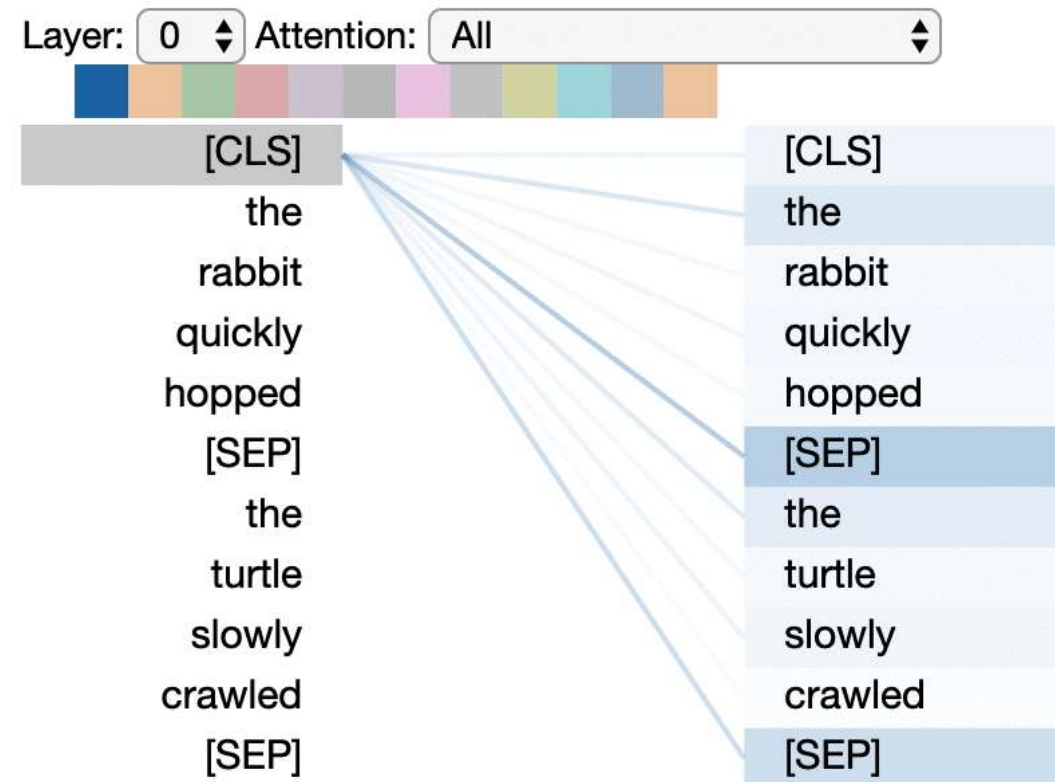
: 시퀀스 내의 모든 단어 간의 관계를  
동시에 파악



query  
key  
value



모두 Encoder의  
은닉상태



# Transformer

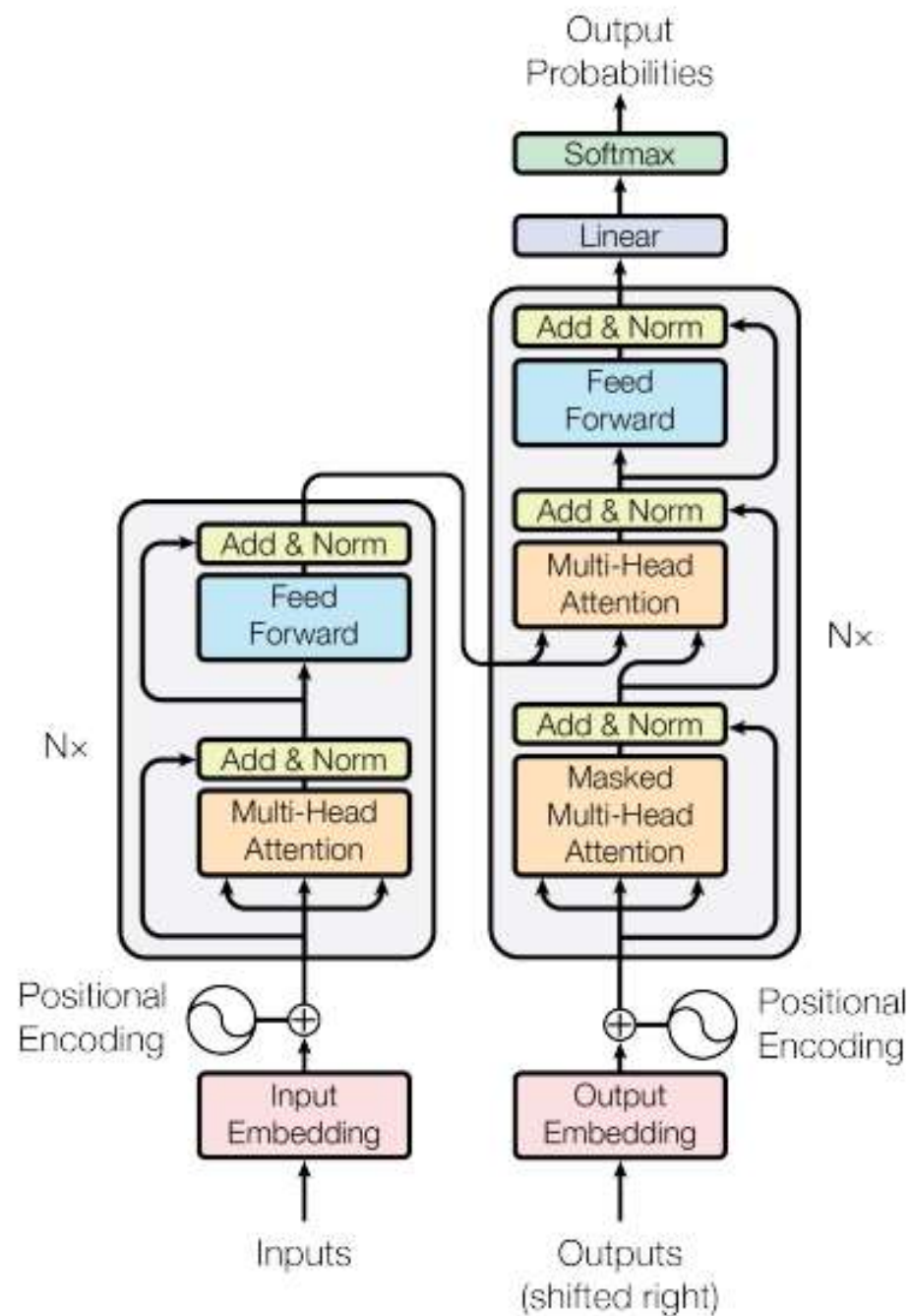
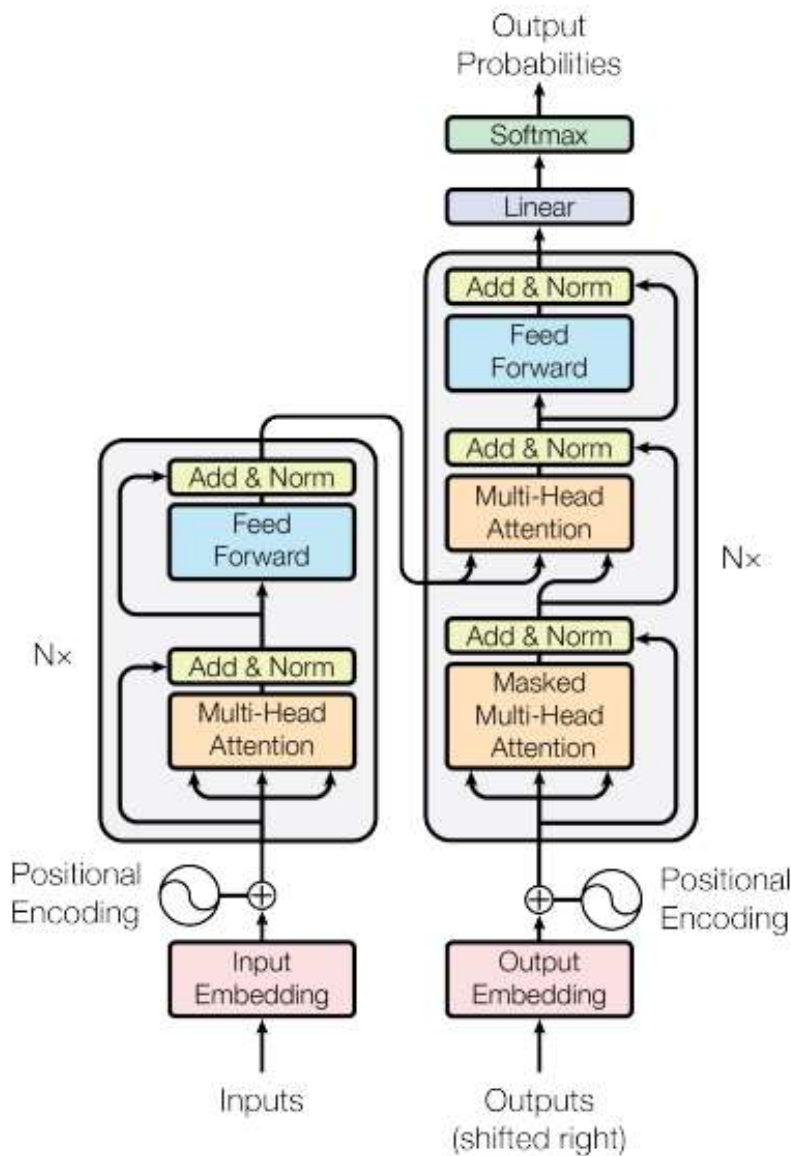


Figure 1: The Transformer - model architecture.

# Encoder

벡터 업데이트  
복잡한 패턴 인식  
문서분류, 감정분석



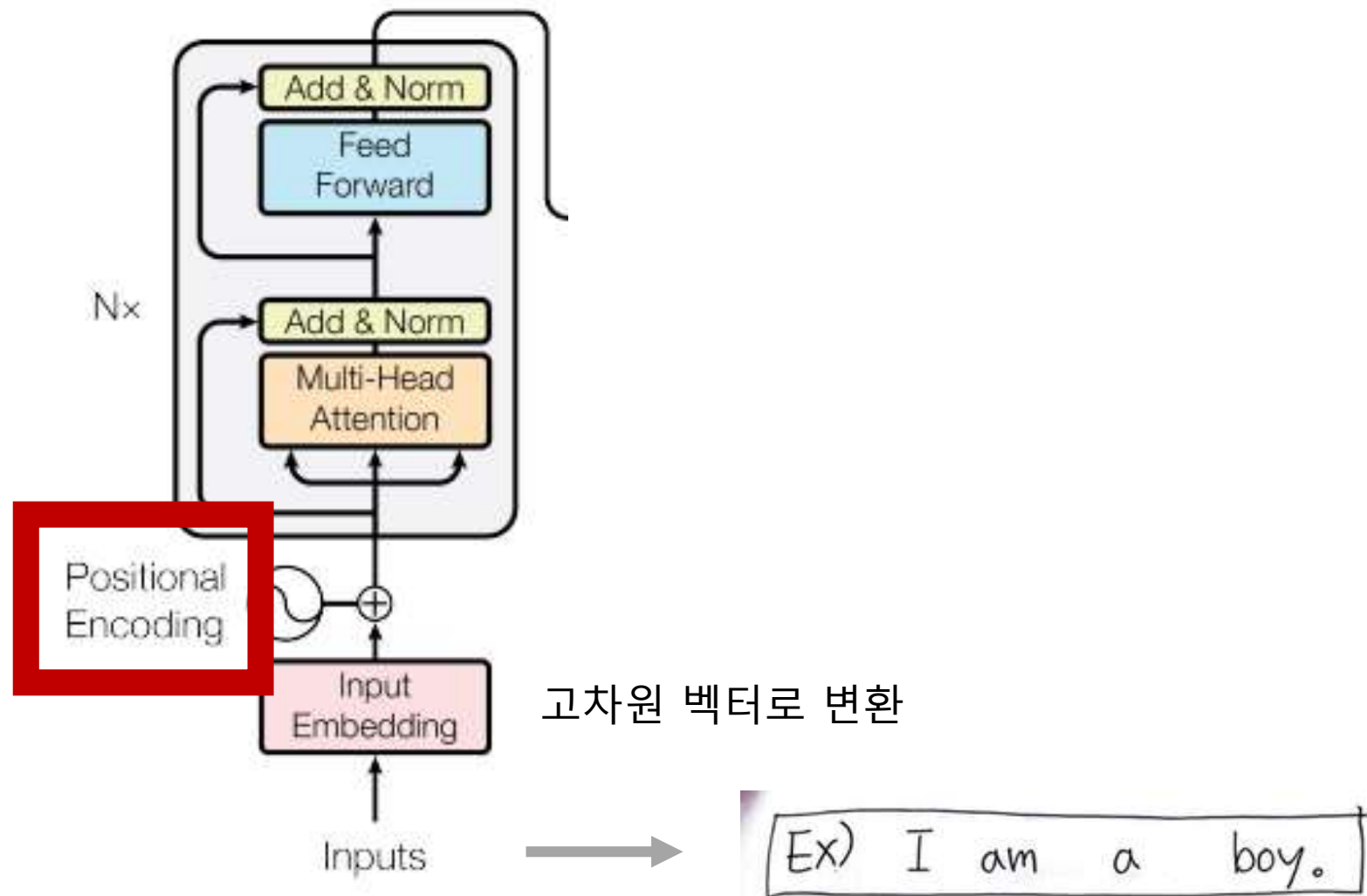
# Decoder

미래의 토큰을 볼 수  
없도록 masked  
토큰 예측->생성  
텍스트 생성, 언어 모델

Figure 1: The Transformer - model architecture.

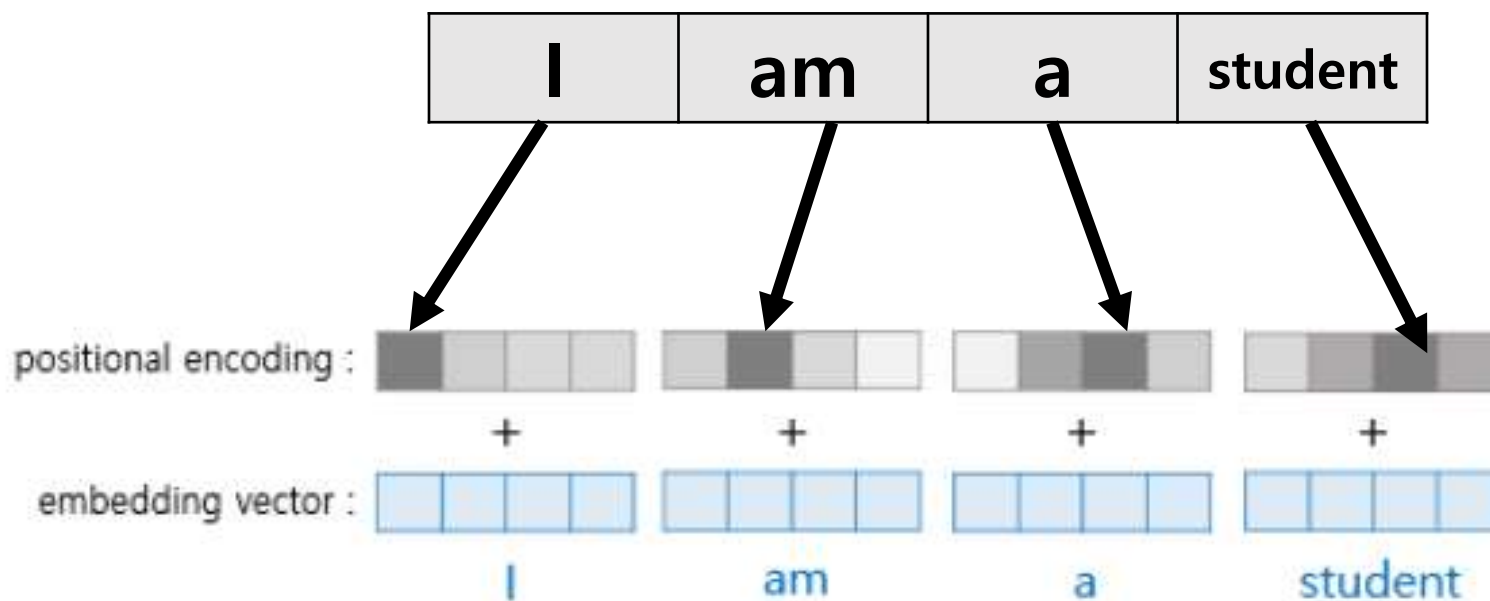
기계 번역, 문서요약, 질의응답

# Transformer -Encoder



# Positional Encoding

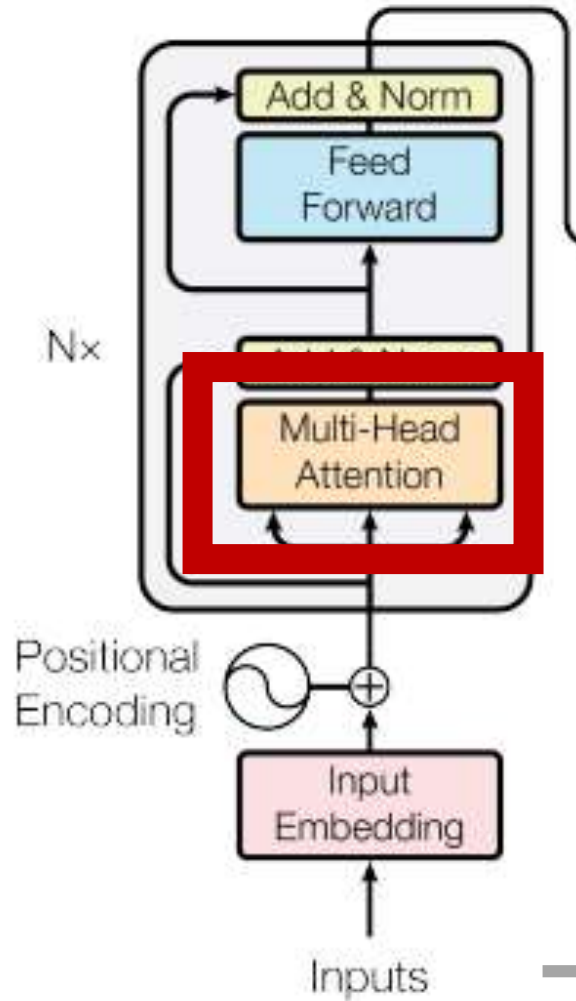
: 각 단어의 임베딩 벡터에 위치 정보를 더하는 것



같은 단어라도 문장 내 위치에 따라서  
임베딩 벡터값이 달라짐

-> 순서 정보를 고려하기 위해 PE 반영

# Transformer -Encoder



모든 단어 벡터와의 관계를  
계산하여 새로운 문맥벡터를 형성

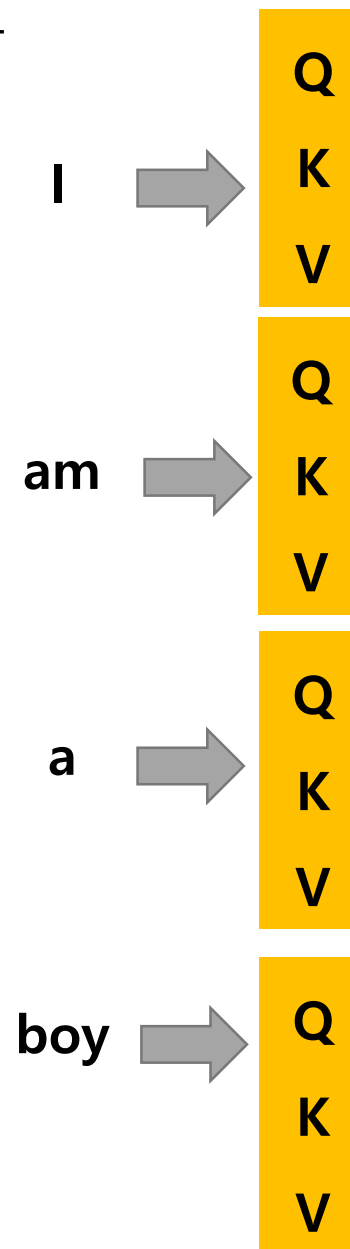
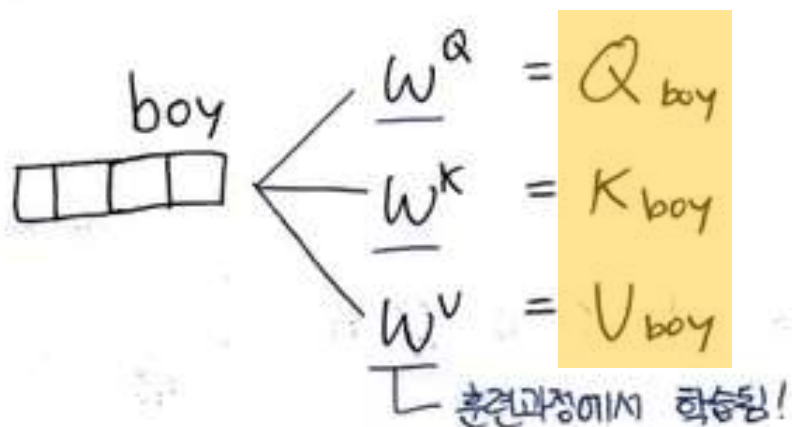
Ex) I am a boy.

# 선형 연산

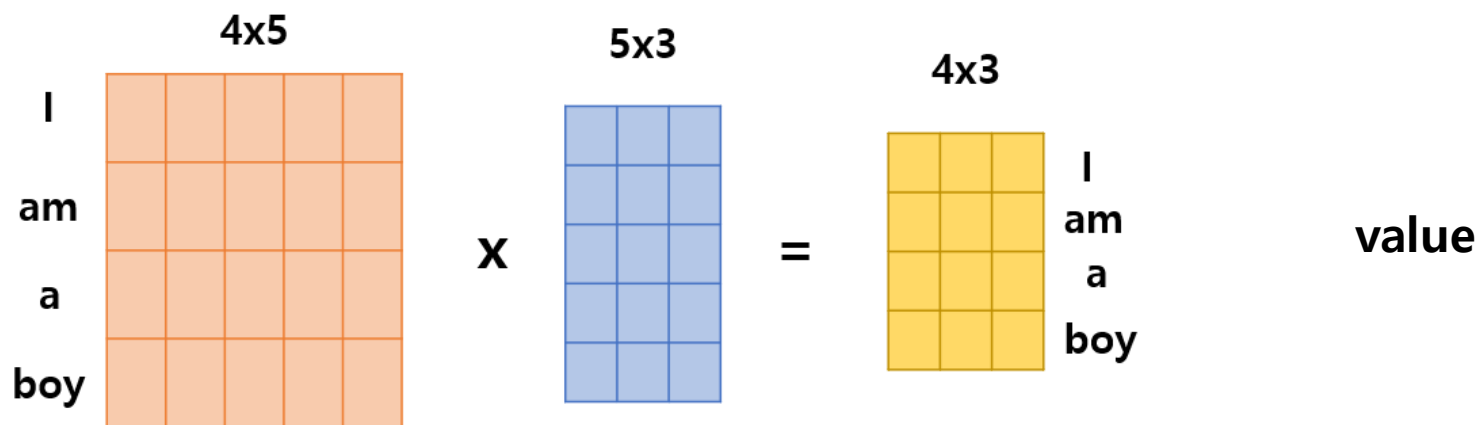
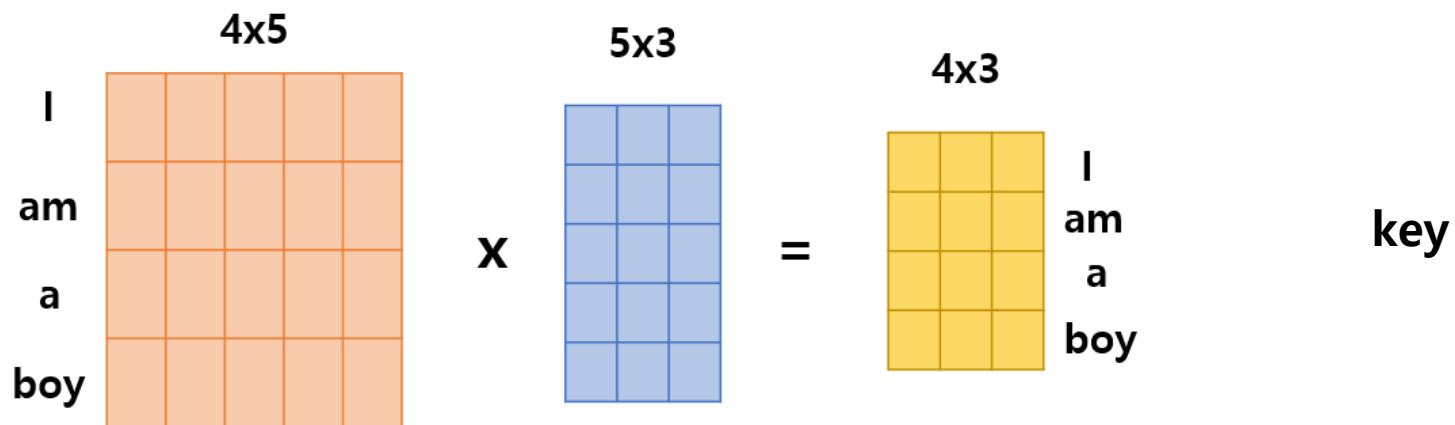
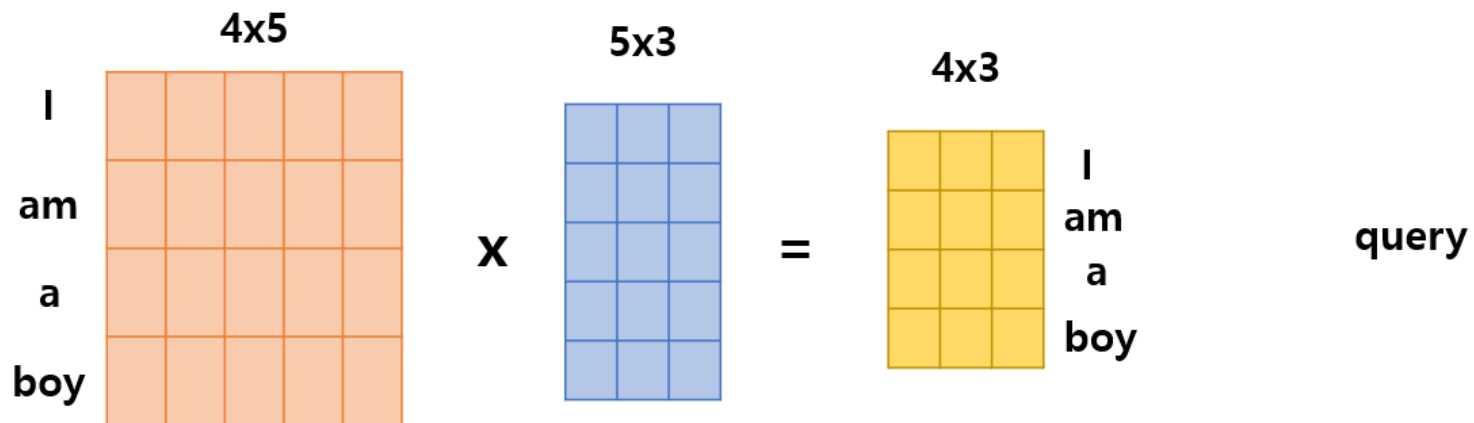
: 차원을 줄여서 병렬 연산에 적합한 구조를 만듦  
(입력 문장 -> 임베딩 벡터 -> Q, K, V 벡터로 변환)

[Ex) I am a boy.]

1) 각 단어들로 부터 Q, K, V 벡터 만들기







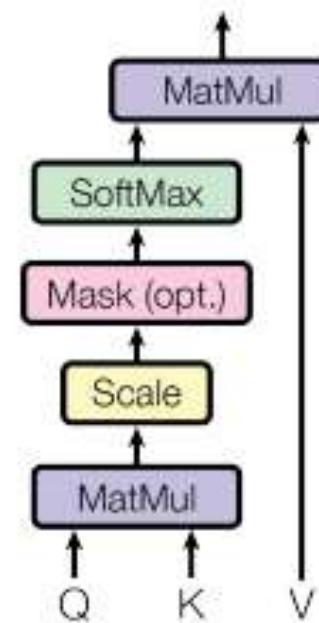
단어 행렬

가중치 행렬 qkv

행렬 연산을 통한  
일괄 계산 가능

# Scaled Dot-Product Attention (self-attention)

Scaled Dot-Product Attention



# Scaled Dot-Product Attention

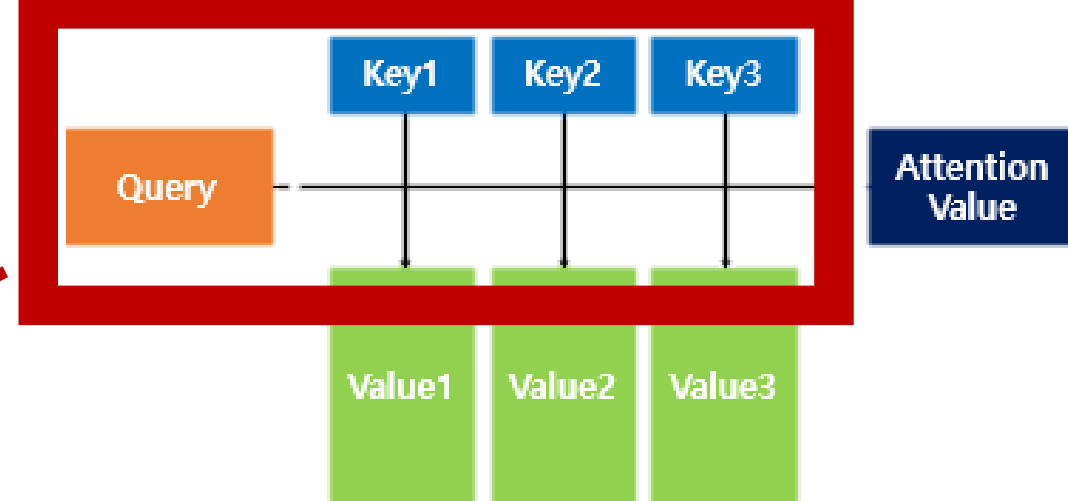
Ex) I am a boy.



## 1) Attention score

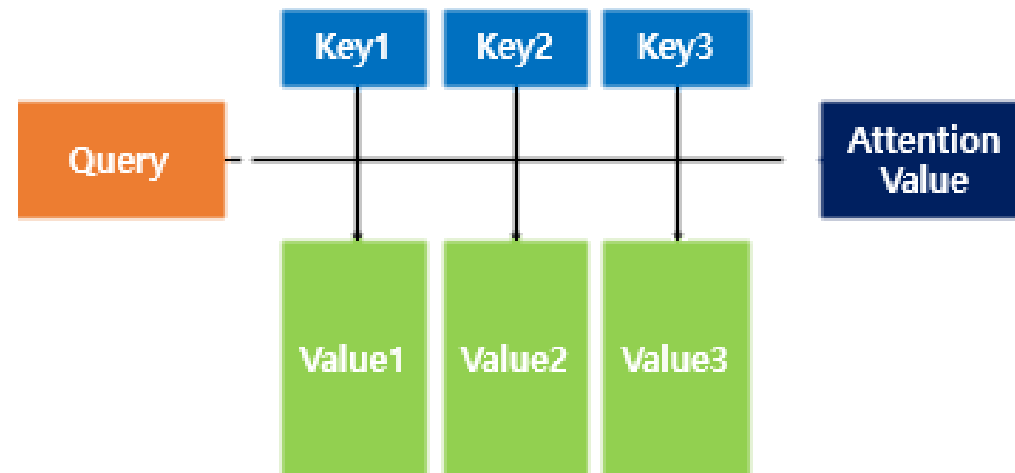
$$\begin{array}{lcl} Q_I & \times & K_I^T = \odot \\ \text{query} & \times & K_{am}^T = \odot \\ & \times & K_a^T = \odot \\ & \times & K_{boy}^T = \odot \end{array}$$

key



# Scaled Dot-Product Attention

Ex) I am a boy.



## 1) Attention score

$$\begin{aligned}
 Q_I &\times K_I^T = \odot \rightarrow \odot / \sqrt{d_k} = \boxed{\phantom{00}} \\
 &\times K_{am}^T = \odot \rightarrow \odot / \sqrt{d_k} = \boxed{\phantom{00}} \\
 &\times K_a^T = \odot \rightarrow \odot / \sqrt{d_k} = \boxed{\phantom{00}} \\
 &\times K_{boy}^T = \odot \rightarrow \odot / \sqrt{d_k} = \boxed{\phantom{00}}
 \end{aligned}$$

↓  
키의 차원

3) Scaled →

dot product 계산시 문장의 길이가 길어질수록 결과값 커짐

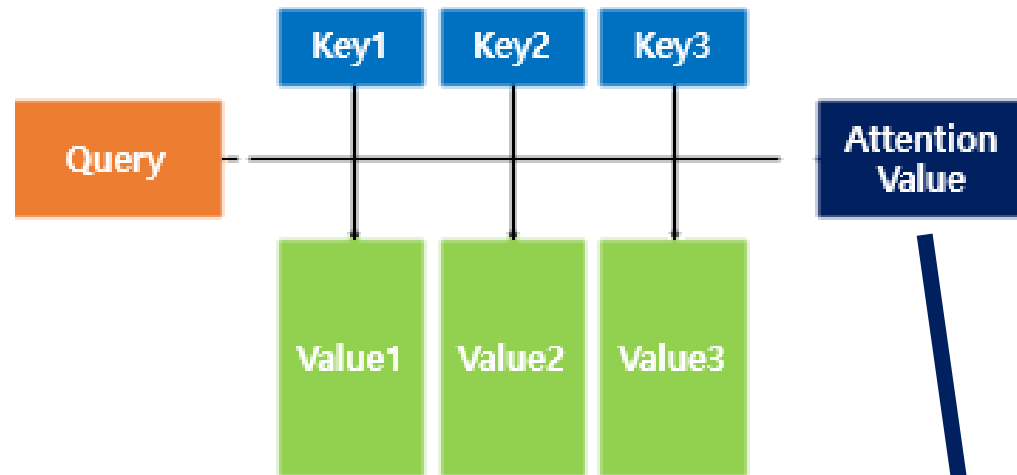
-> softmax에 넣으면 큰 값은 과도하게 살아남고, 나머지 값은 사라짐

## 2) Attention distribution

Softmax —

# Scaled Dot-Product Attention

Ex) I am a boy.



2) Attention score

$$Q_I \times K_I^T = \odot \rightarrow \odot / \sqrt{d_k} = \boxed{\phantom{00}}$$

$$\times K_{am}^T = \odot \rightarrow \odot / \sqrt{d_k} = \boxed{\phantom{00}}$$

$$\times K_a^T = \odot \rightarrow \odot / \sqrt{d_k} = \boxed{\phantom{00}}$$

$$\times K_{boy}^T = \odot \rightarrow \odot / \sqrt{d_k} = \boxed{\phantom{00}}$$

↓  
키의 차원

3) Scaled →

dot product 계산시 문장의 길이가 길어  
수록 결과값 커짐

-> softmax에 넣으면 큰 값은 과도하게  
살아남고, 나머지 값은 사라짐

4) Attention Distribution

Softmax

$U_I$   
 $U_{am}$   
 $U_a$   
 $U_{boy}$

value

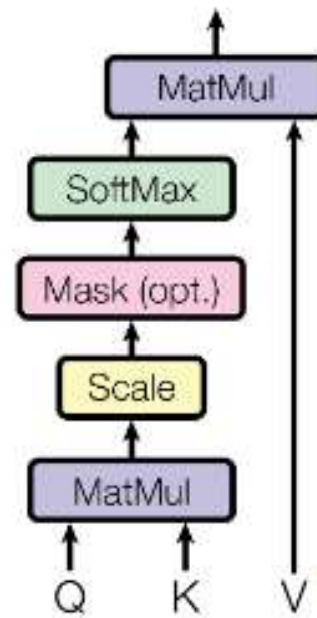
5) Attention Value  
=  

--	--

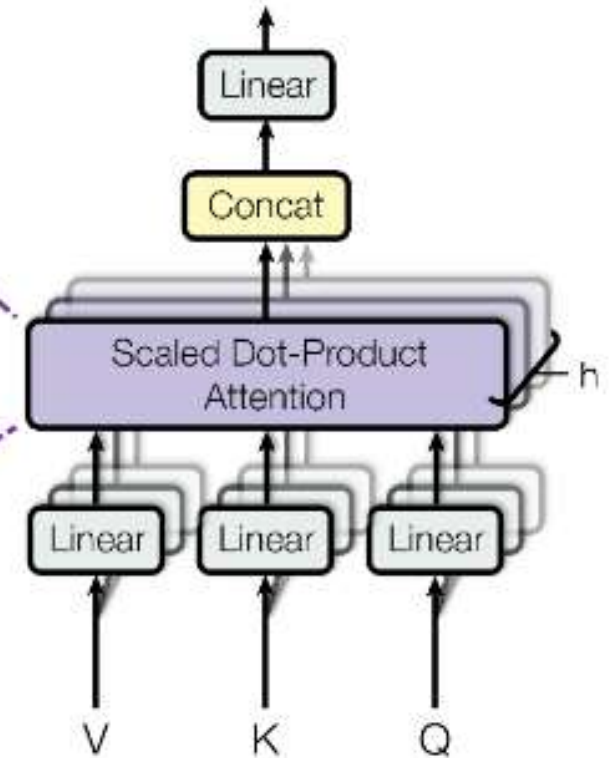
3) Attention value

# Multi-head Attention

Scaled Dot-Product Attention



Multi-Head Attention



# Multi-head Attention

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

: Self-Attention을 병렬로  $h$ 번 학습

1-head

Which do you like better, coffee or tea?

- 문장 타입에 집중하는 어텐션

1-head

Which do you like better, coffee or tea?

- 명사에 집중하는 어텐션

1-head

Which do you like better, coffee or tea?

- 관계에 집중하는 어텐션

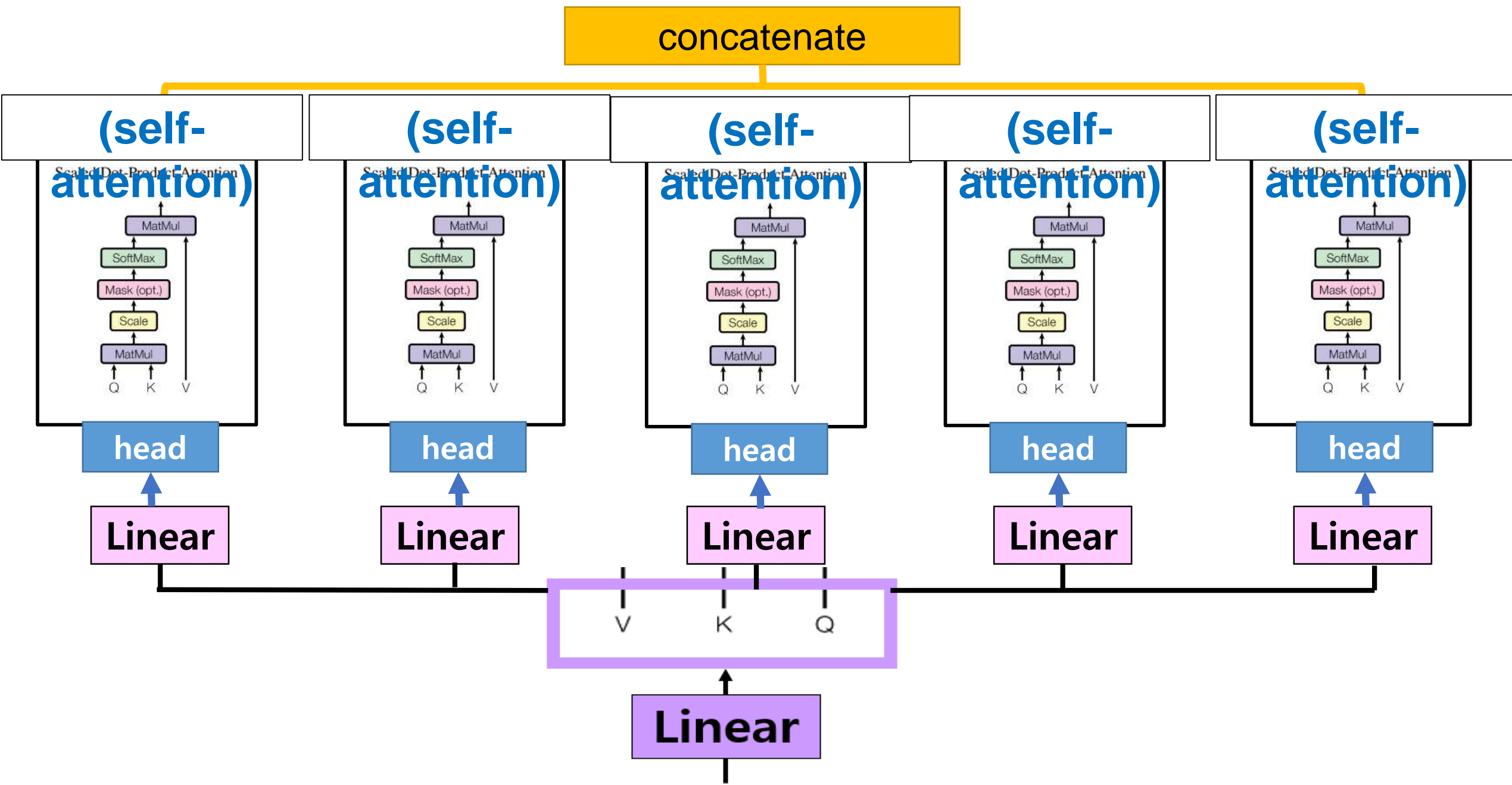
1-head

Which do you like better, coffee or tea?

- 강조에 집중하는 어텐션

다양한 시각에서  
자동으로  
정보 수집 및 포착

-> 표현력 풍부  
복잡한 관계 파악 용이



Which do you like better, coffee or tea?



which

do

you

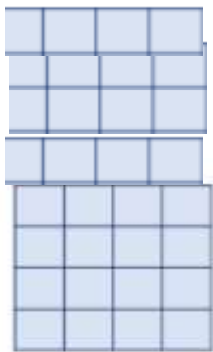
like

better

coffee

or

tea



Multi-head attention matrix

Linear

- concat으로 증가한 차원 복원
- 정보 통합

concatenate

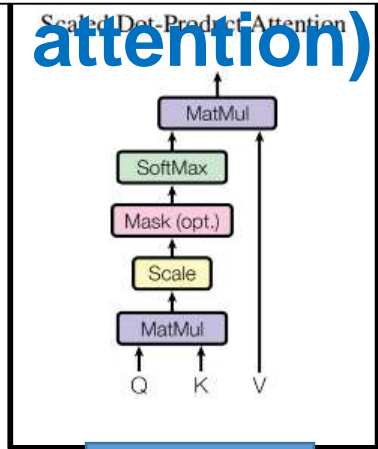
(self-attention)

(self-attention)

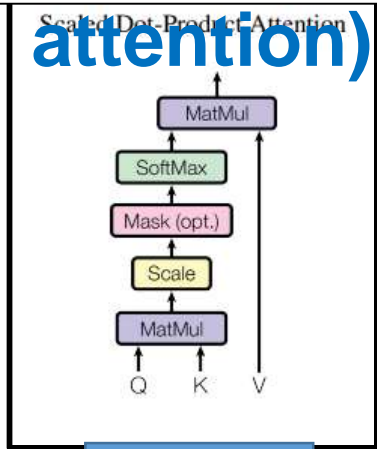
(self-attention)

(self-attention)

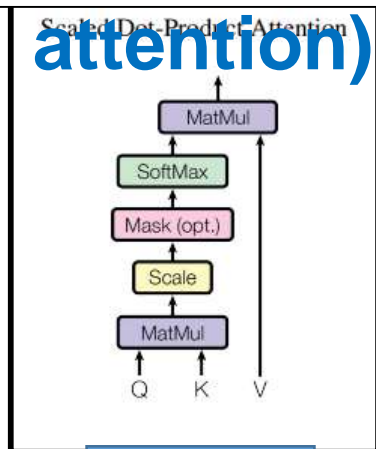
(self-attention)



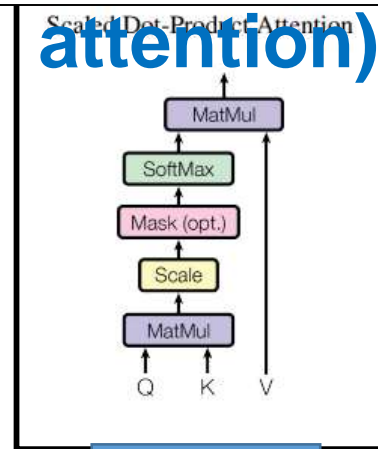
head



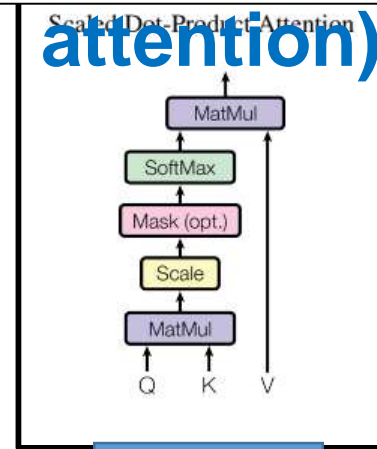
head



head

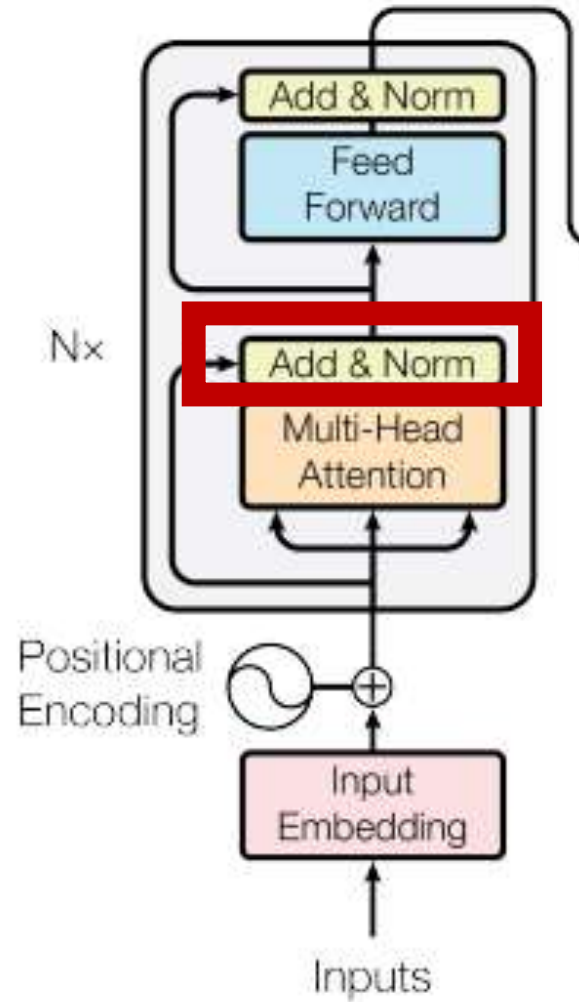


head



head

# Transformer -Encoder



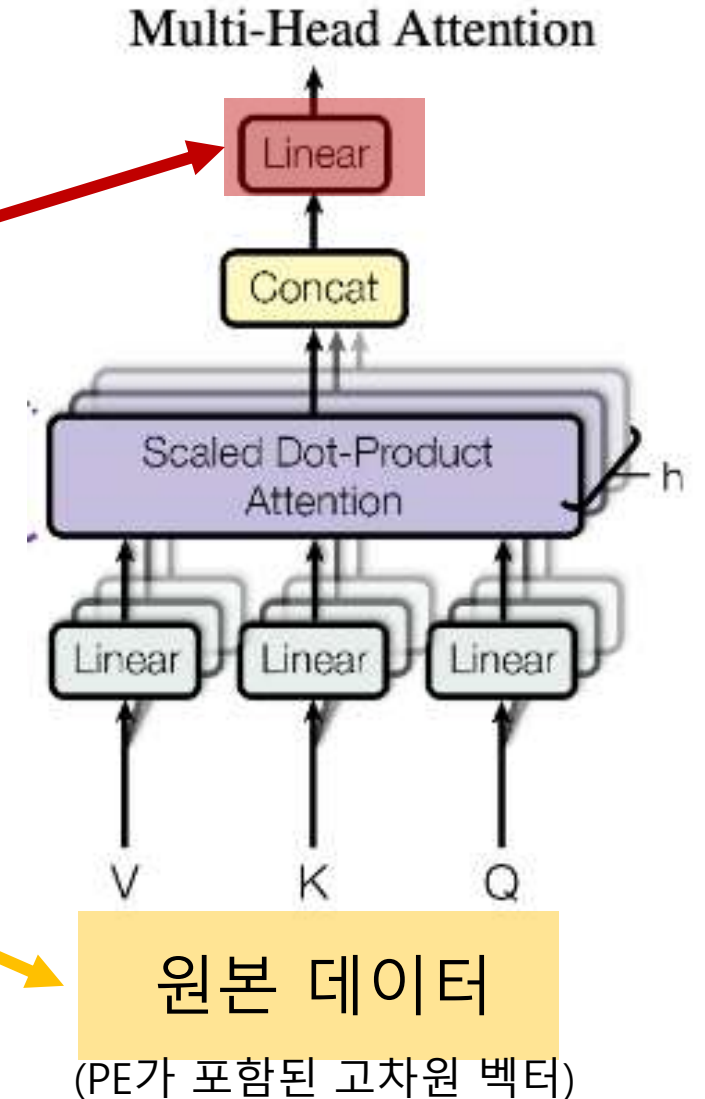
# Add & Norm

## Add

잔차 연결(Residual Connection)

$$H(x) = x + \text{Multi-head Attention}(x)$$

-> 학습 용이, 효율적, 빠른 학습



# Add & Norm

## Norm

층 정규화(Layer Normalization)

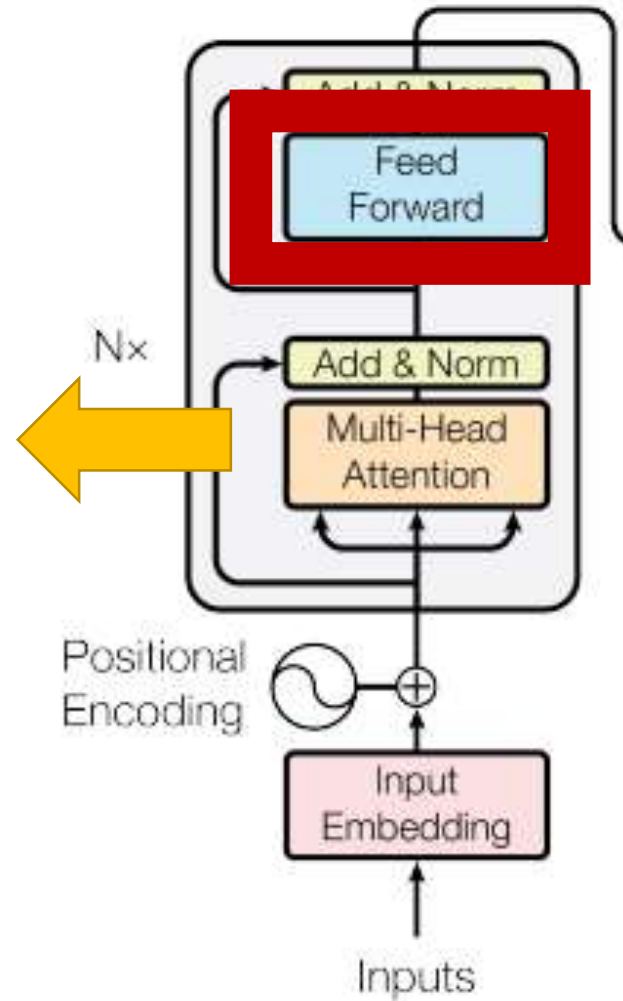
: 통계적 분포 조절(안정화)

keras.

LayerNormalization()

# Transformer

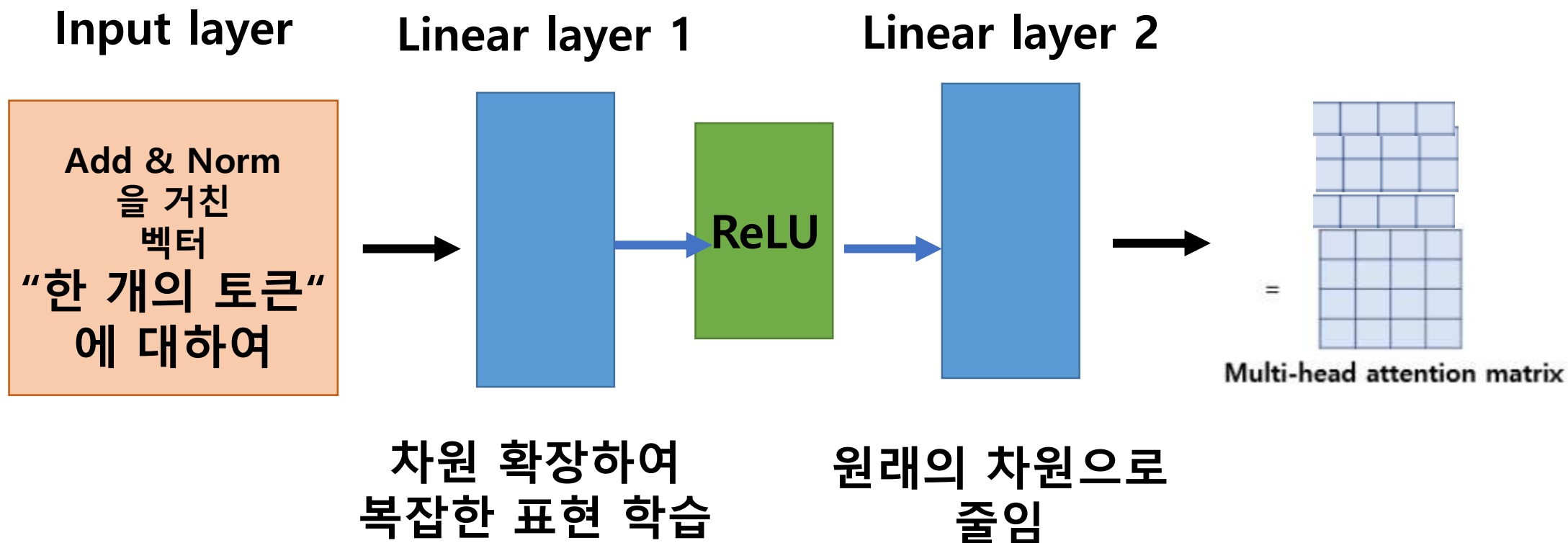
시퀀스 내,  
토큰 사이의 관계 파악



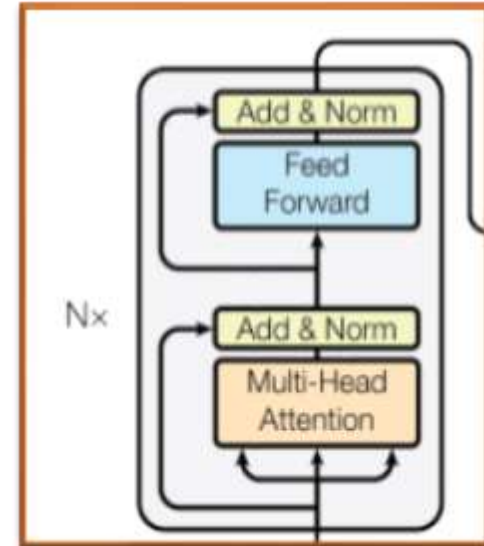
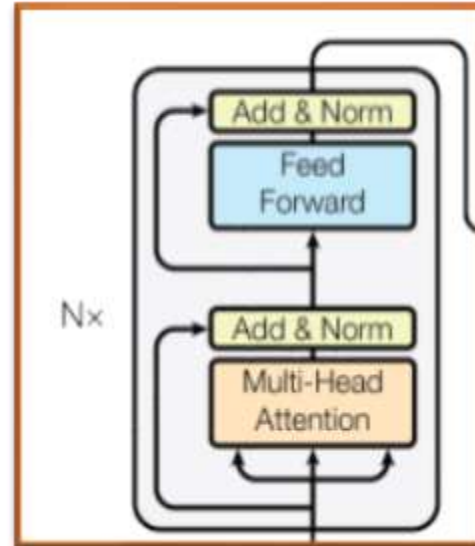
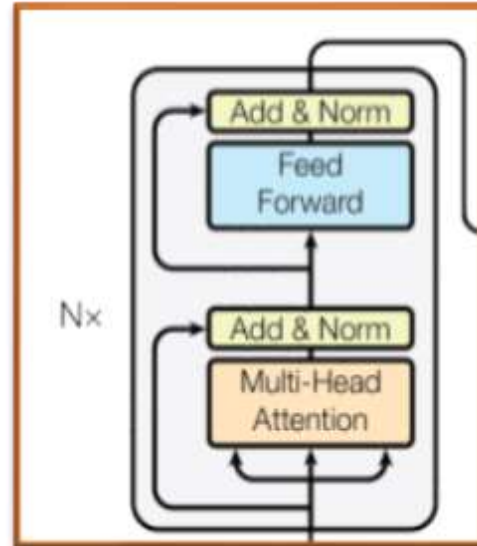
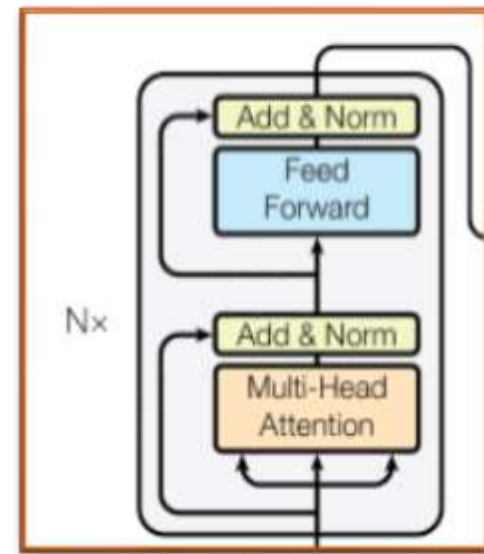
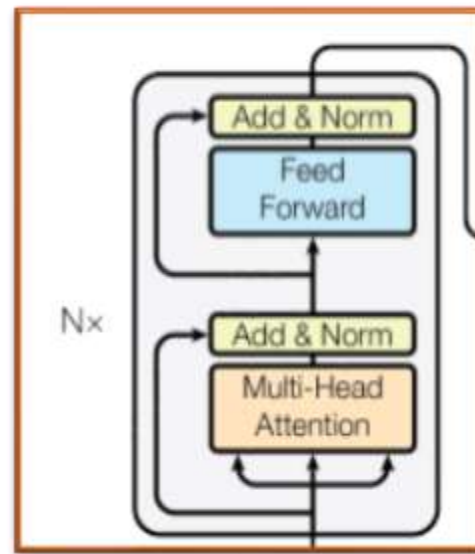
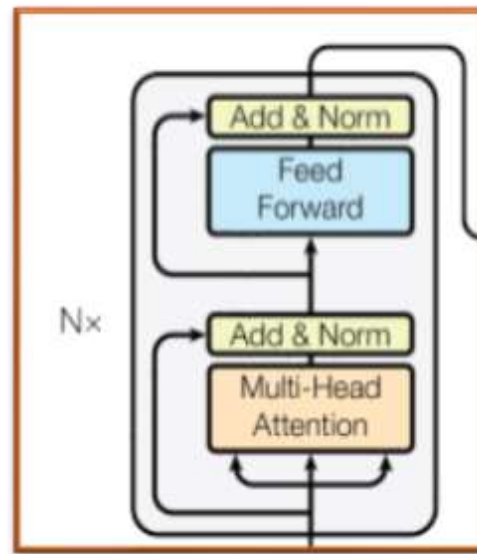
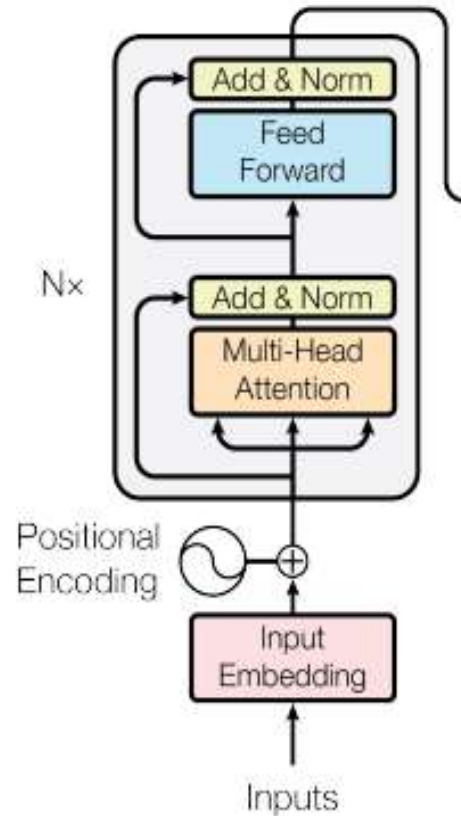
토큰의 표현을 발전시켜  
**더욱 정교한 표현을 얻음**  
(복잡한 특징이나  
패턴 추가로 학습)

# Position-wise FFNN

: 전체 임베딩 시퀀스를 하나의 벡터로 처리하지 않고  
각 토큰마다 독립적으로 처리 (병렬 처리)



# Transformer -Encoder



...