Attention is All You Need (논문리뷰)

12181912 김현수

목차

Table of contents

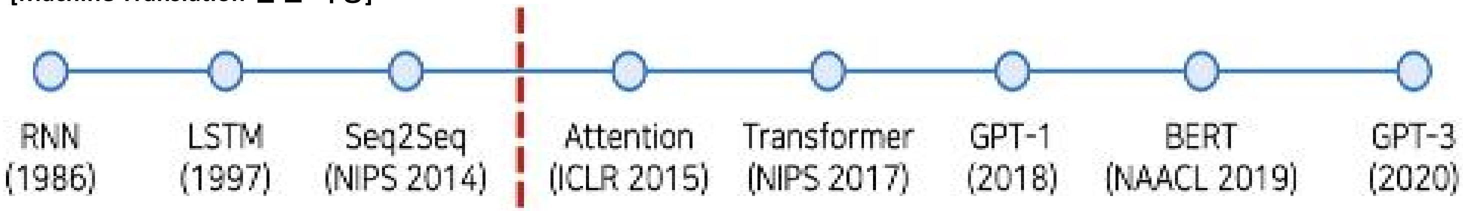
- Attention (Self-Attention, Multi-head Attention)
- Encoder
- Decoder
- Result

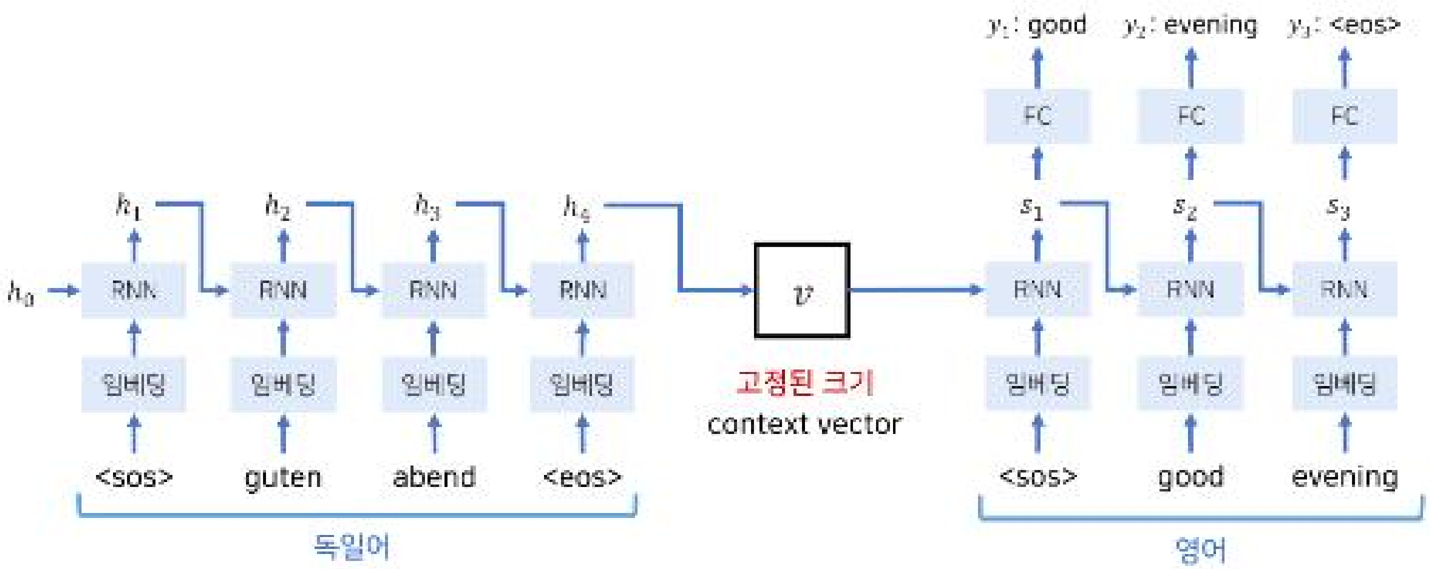
Attention

- Attention (Self-Attention, Multi-head Attention)
- Encoder
- Decoder
- Result

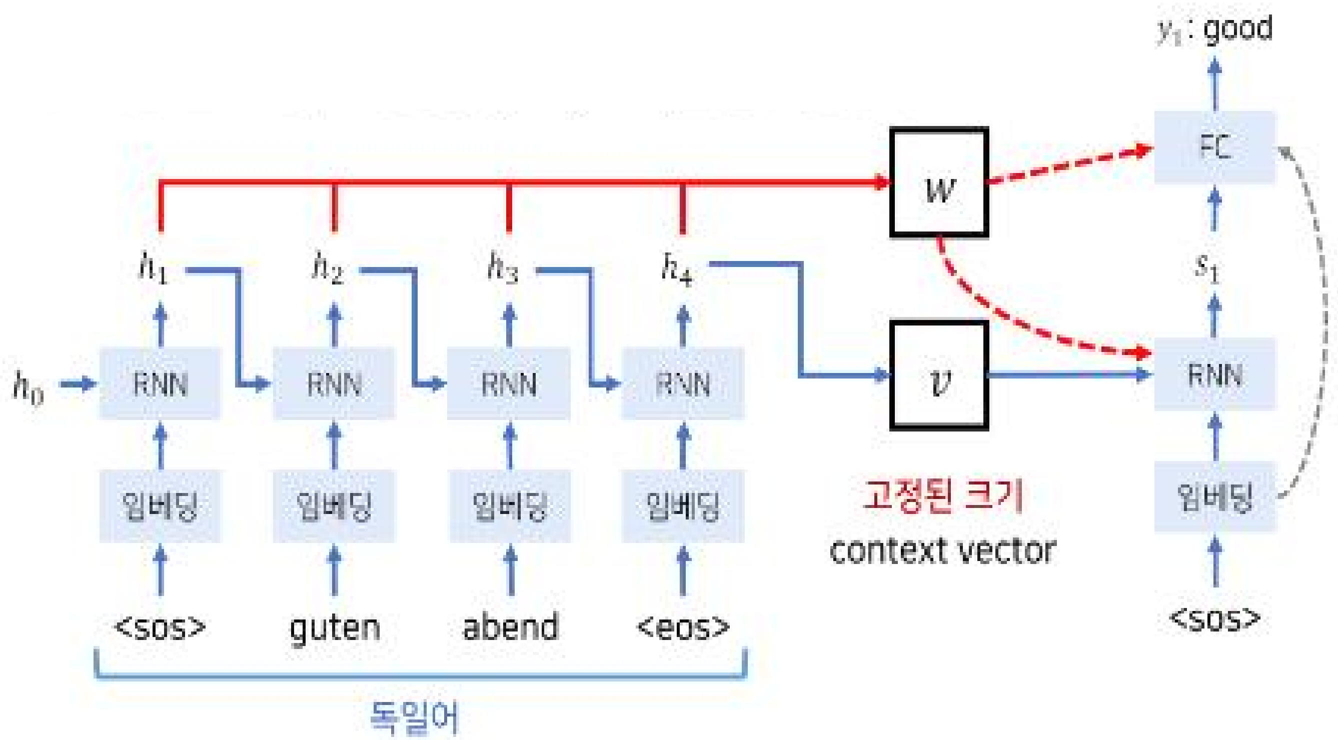
Attention?

[Machine Translation 발전 과정]



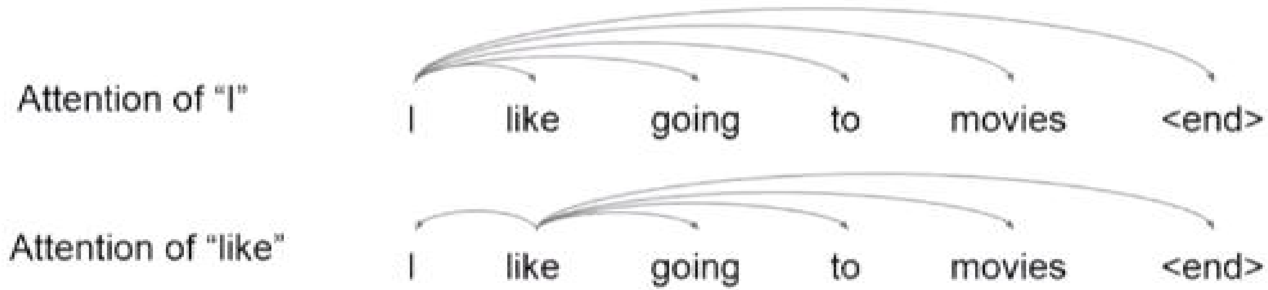


Attention?



y_1 : good
FC
w
h_1
h_2
h_3
h_4
s_1
h_0
RNN
RNN
RNN
RNN
v
RNN
임베딩
임베딩
임베딩
임베딩
고정된 크기
context vector
임베딩
<sos>
guten
abend
<eos>
<sos>
독일어

Self-Attention



- $Attention(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = Softmax\left(\frac{\boldsymbol{QK}^T}{\sqrt{d}}\right)\boldsymbol{V}$

0.5	0.1	0.0	0.2	0.2	0.0
0.2	0.6	0.0	0.0	0.1	0.0
...	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...

$Softmax\left(\frac{\boldsymbol{QK}^T}{\sqrt{d}}\right)$

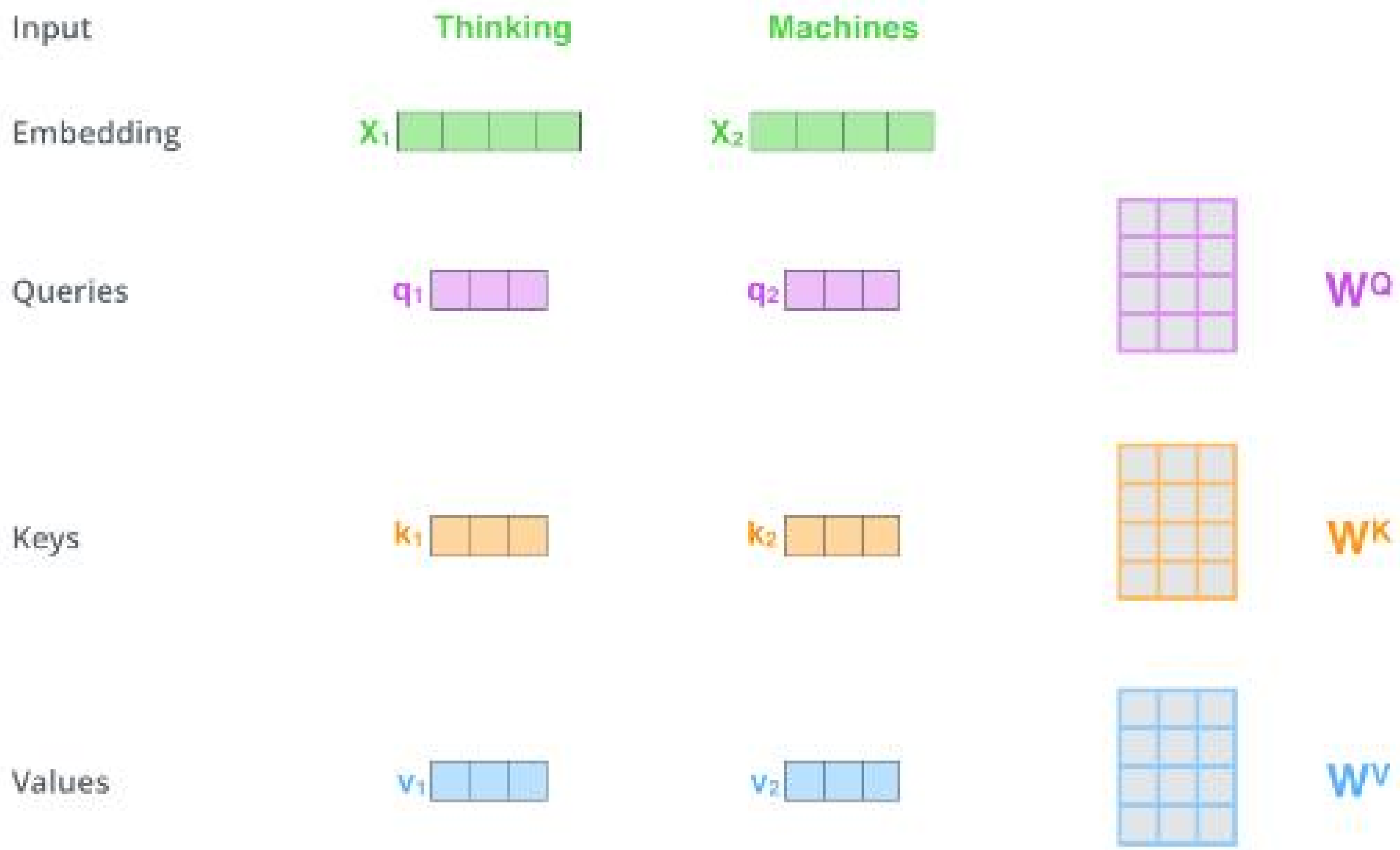
I
like
going
to
movies
<end>

$\boldsymbol{V}$

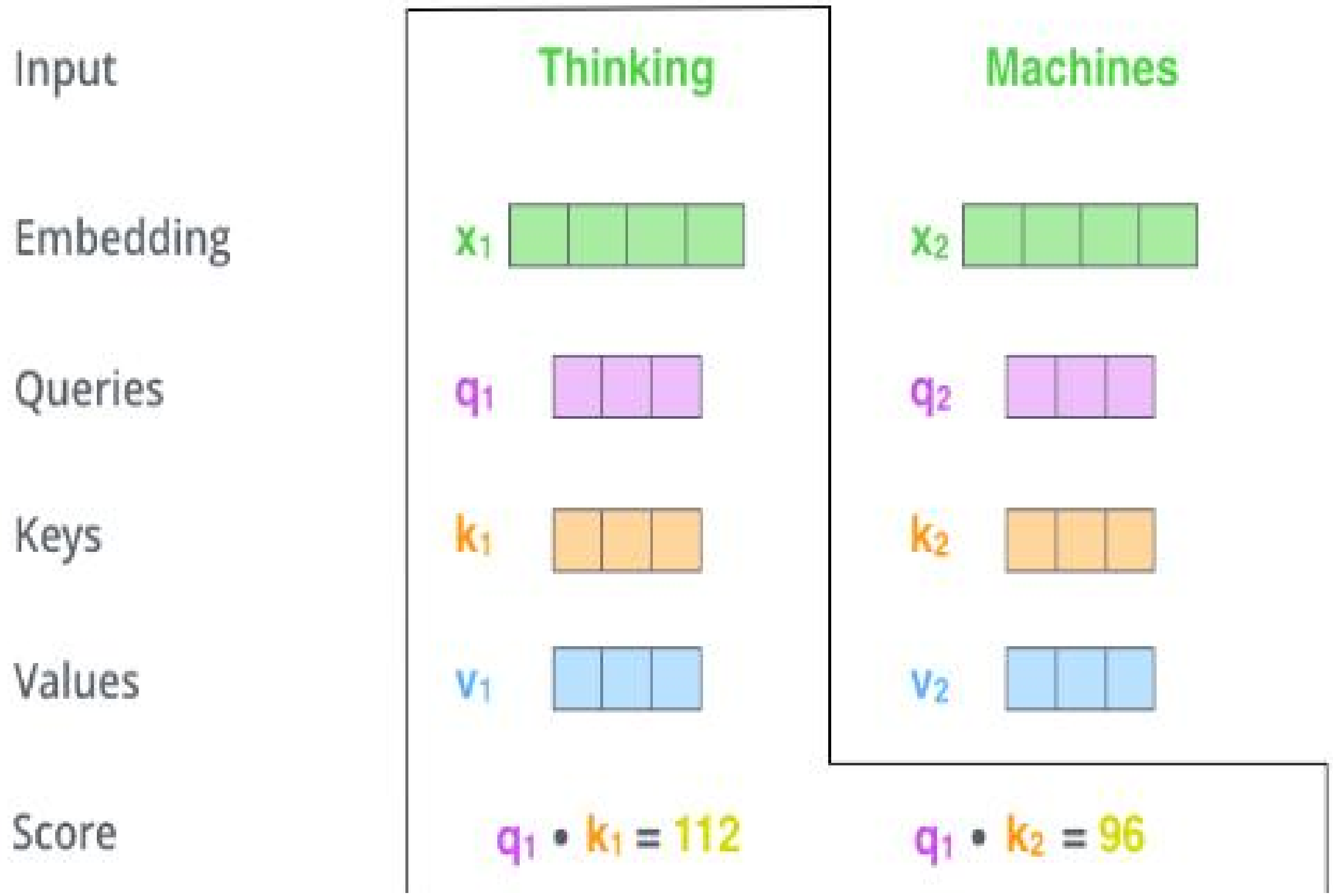
0.5*I + 0.1*like + 0.2*to + 0.2* movies
0.2*I + 0.6*like + 0.1*movies
...
...
...
...

$Attention(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})$

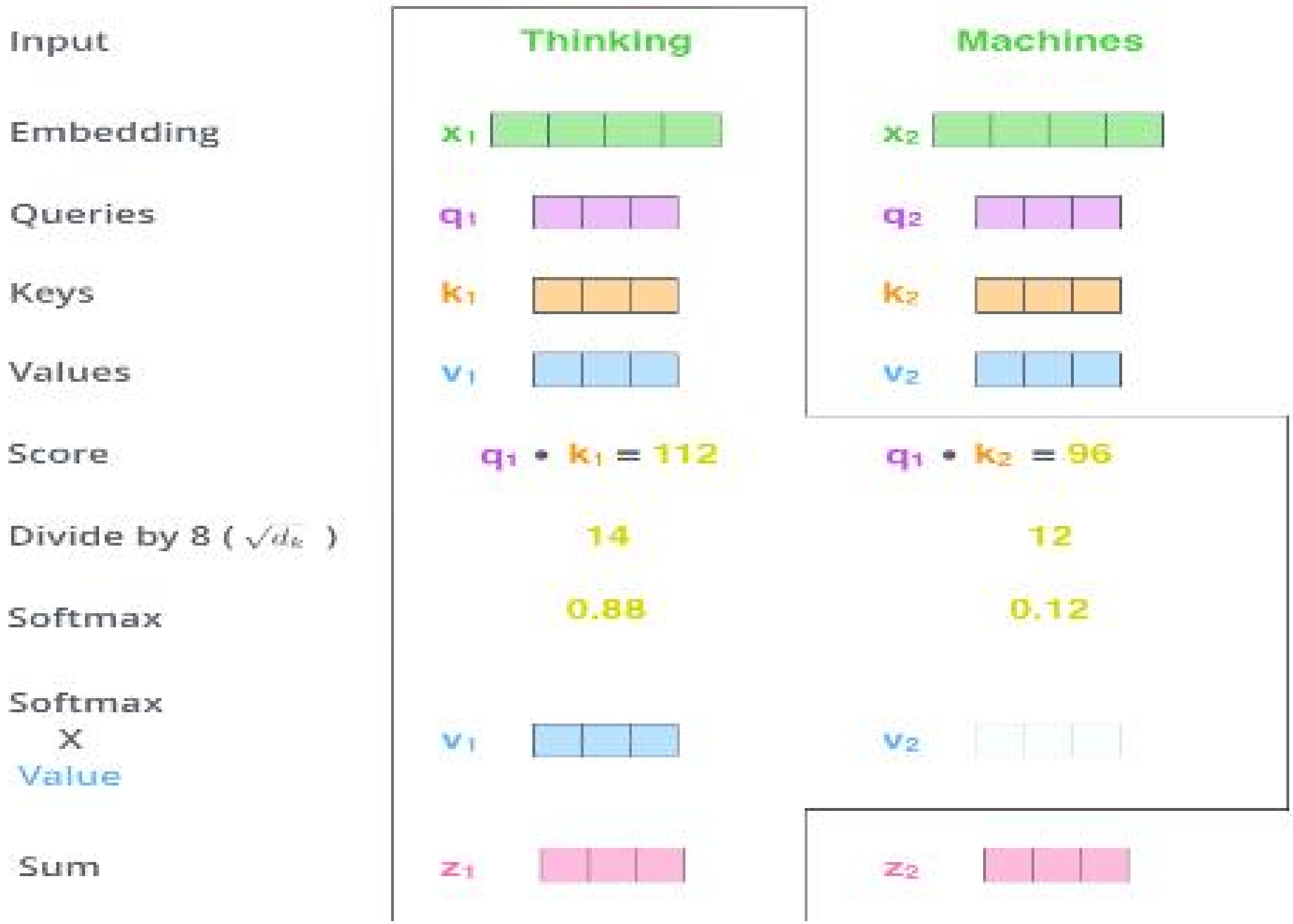
Self-Attention



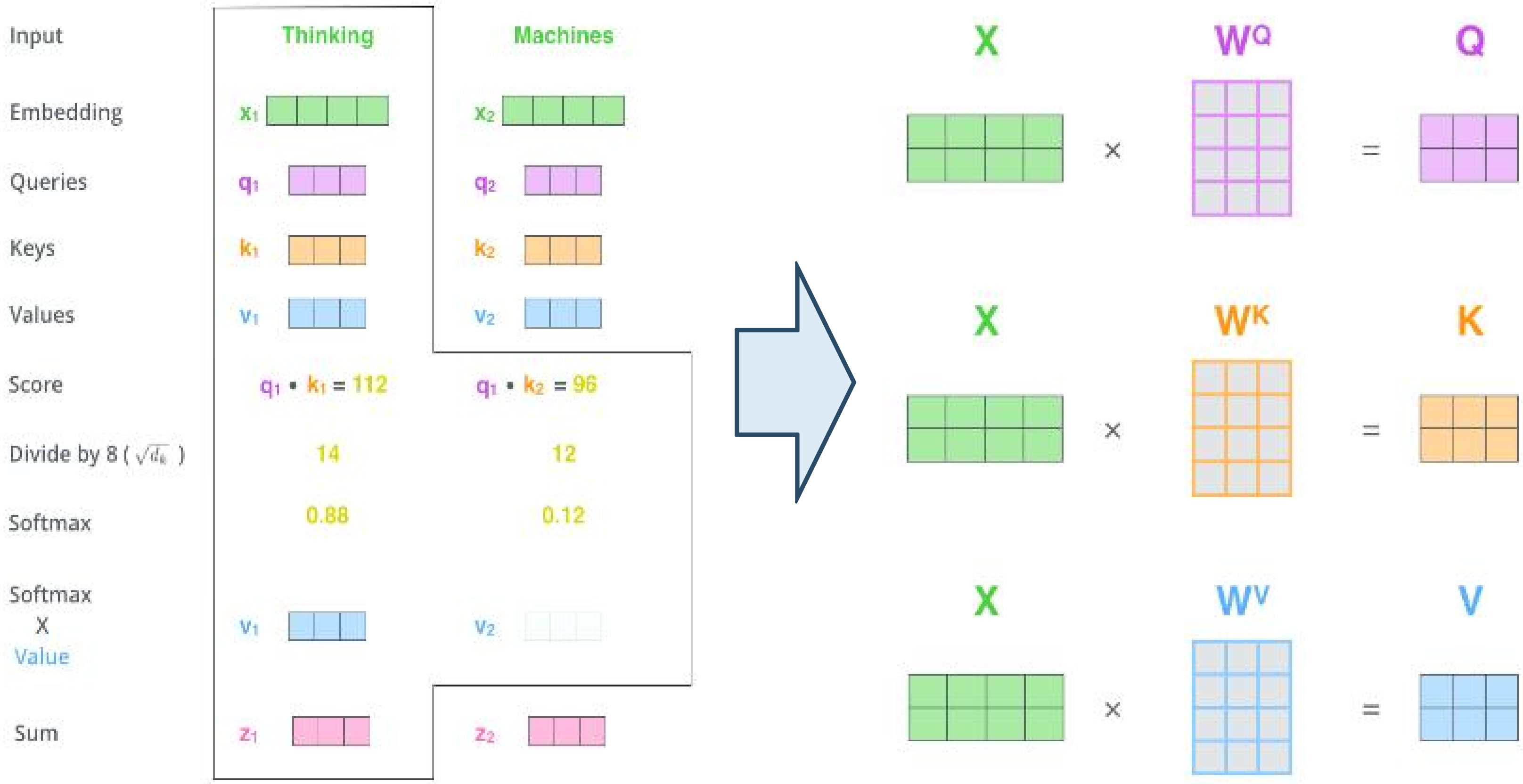
Self-Attention



Self-Attention



Self-Attention



Self-Attention

- $Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = Softmax\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}}\right)\boldsymbol{V}$

0.5	0.1	0.0	0.2	0.2	0.0
0.2	0.6	0.0	0.0	0.1	0.0
..	..	..	..	..	..
..	..	..	..	..	..
..	..	..	..	..	..
..	..	..	..	..	..

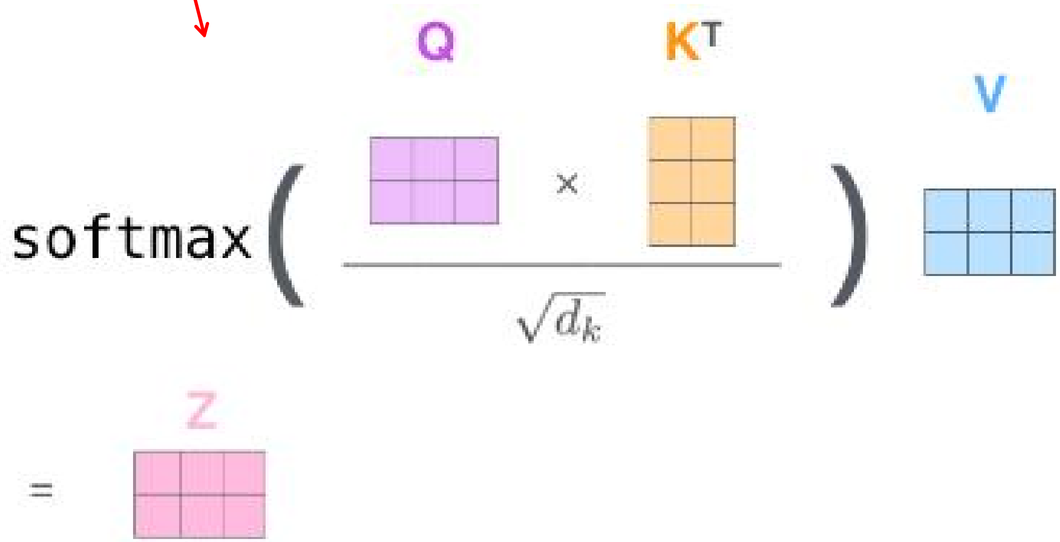
$Softmax\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}}\right)$

I
like
going
to
movies
<end>

$\boldsymbol{V}$

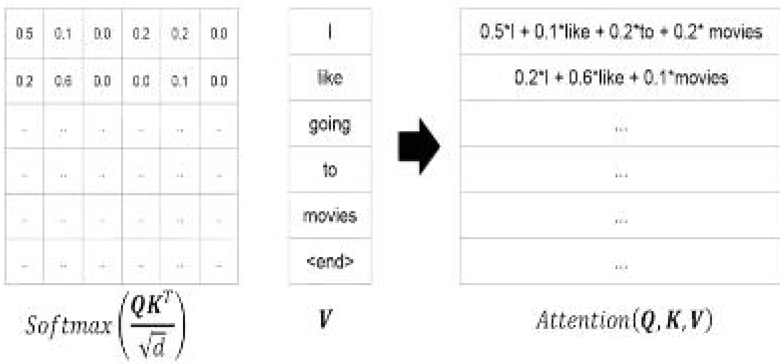
0.5*I + 0.1*like + 0.2*to + 0.2* movies
0.2*I + 0.6*like + 0.1*movies
...
...
...
...

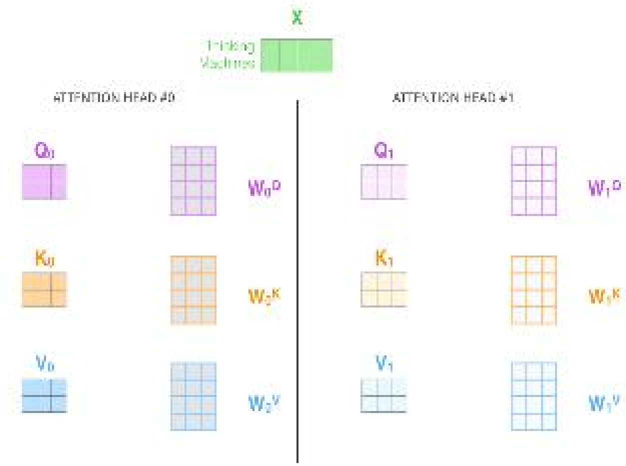
$Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$

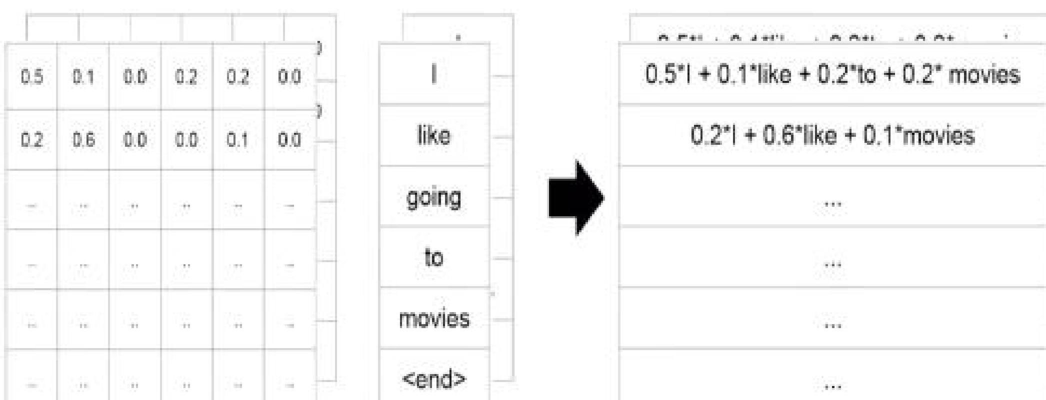


Multi-head Attention

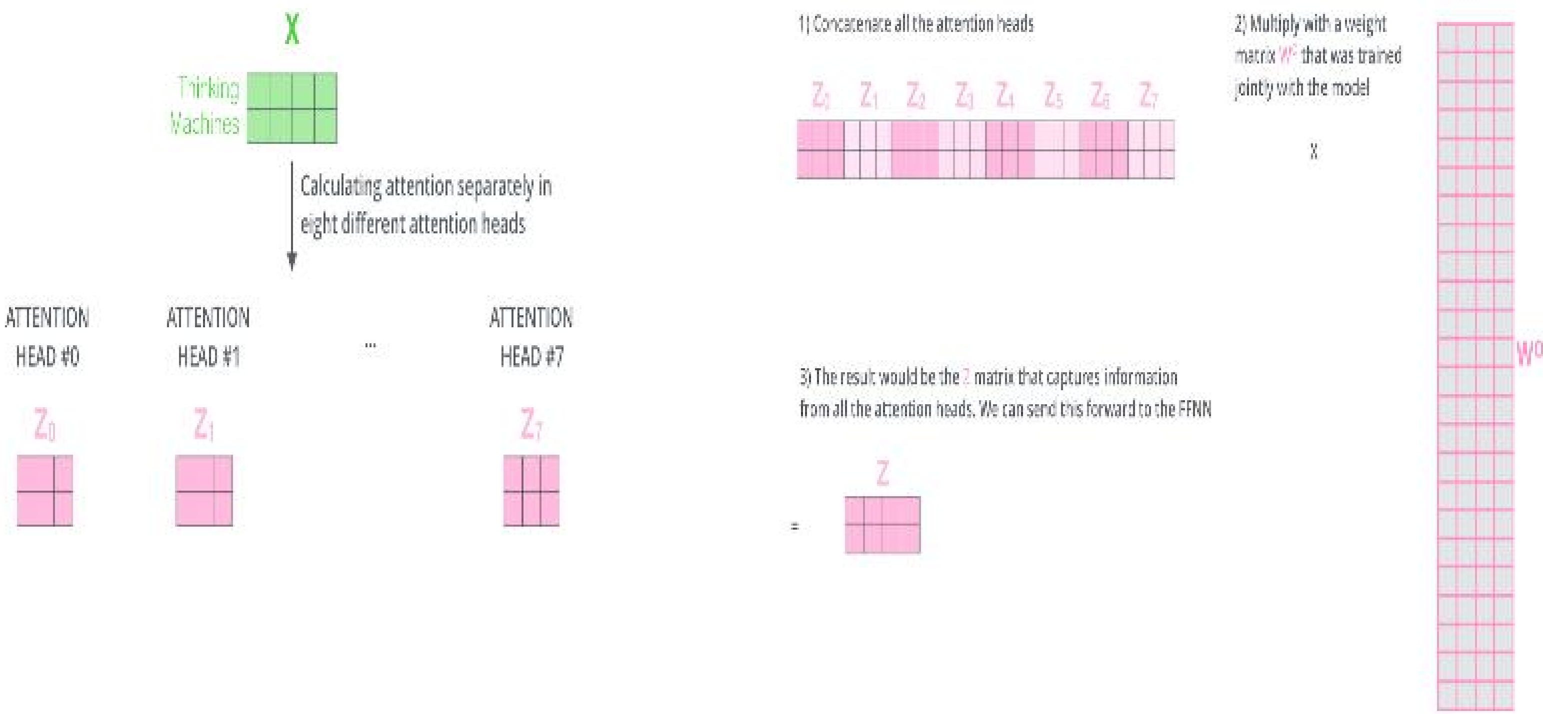
- $Attention(Q,K,V) = Softmax\left(\frac{QK^T}{\sqrt{d}}\right)V$



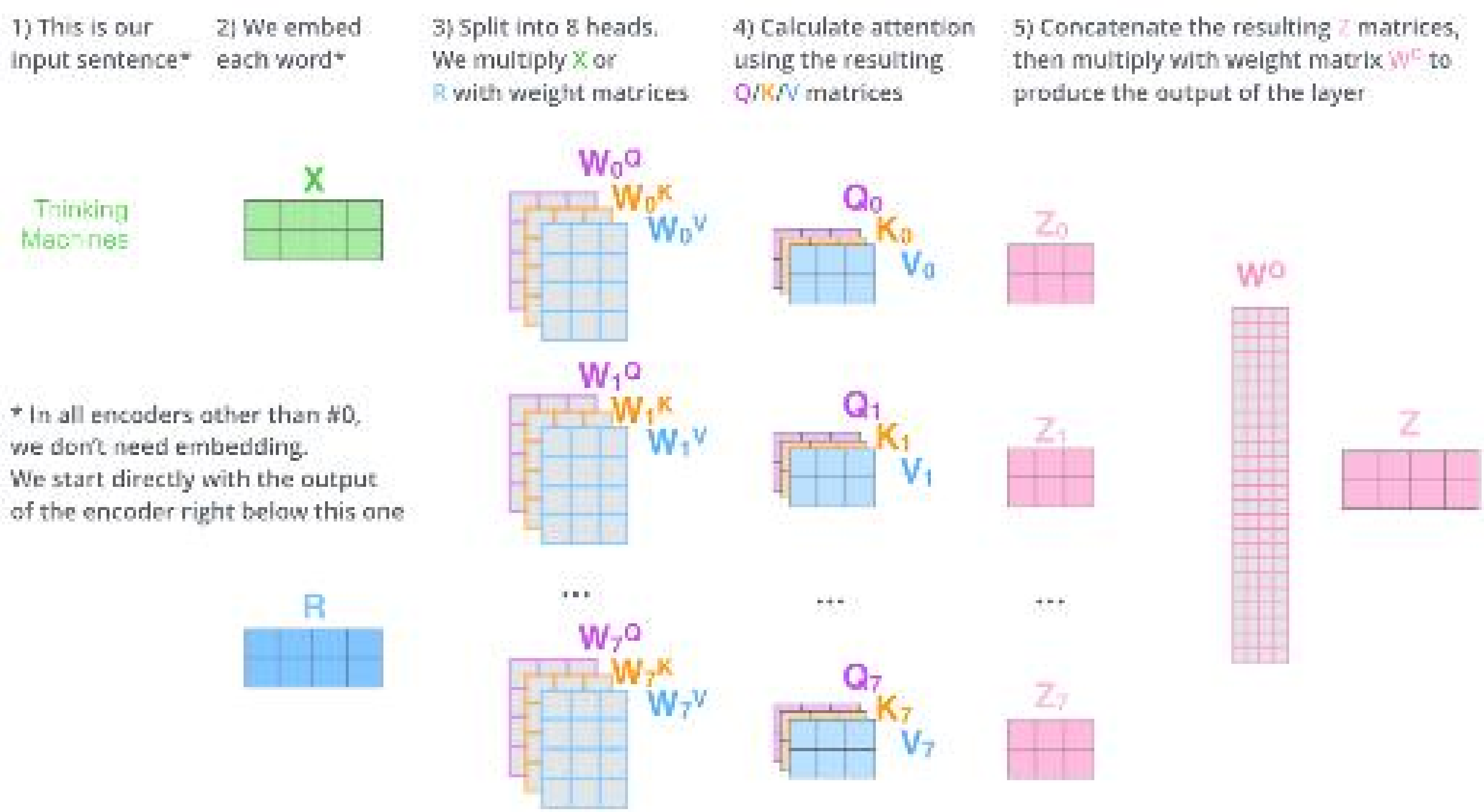




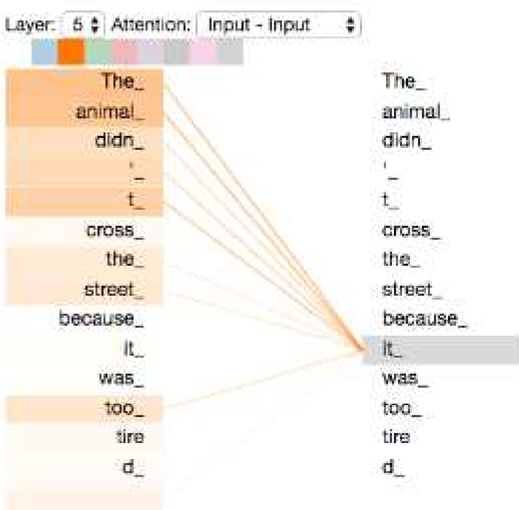
Multi-head Attention



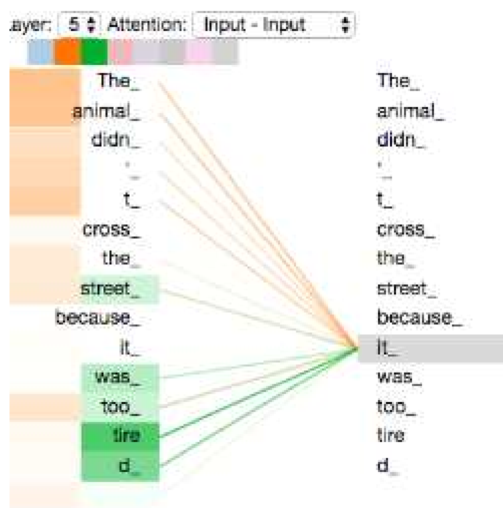
Multi-head Attention



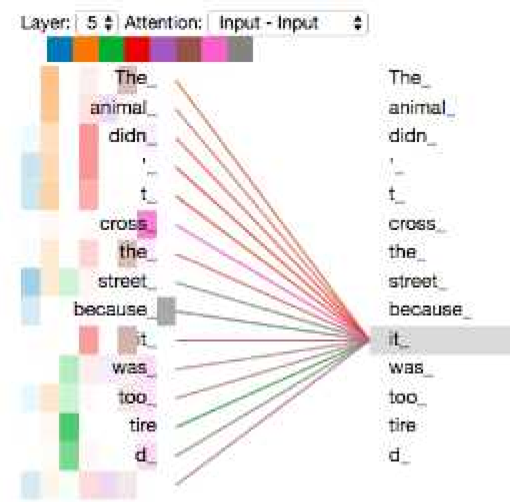
Visualize Attention



1 attention



2 attentions



8 attentions

Encoder

- Attention (Self-Attention, Multi-head Attention)
- Encoder
- Decoder
- Result

Encoder

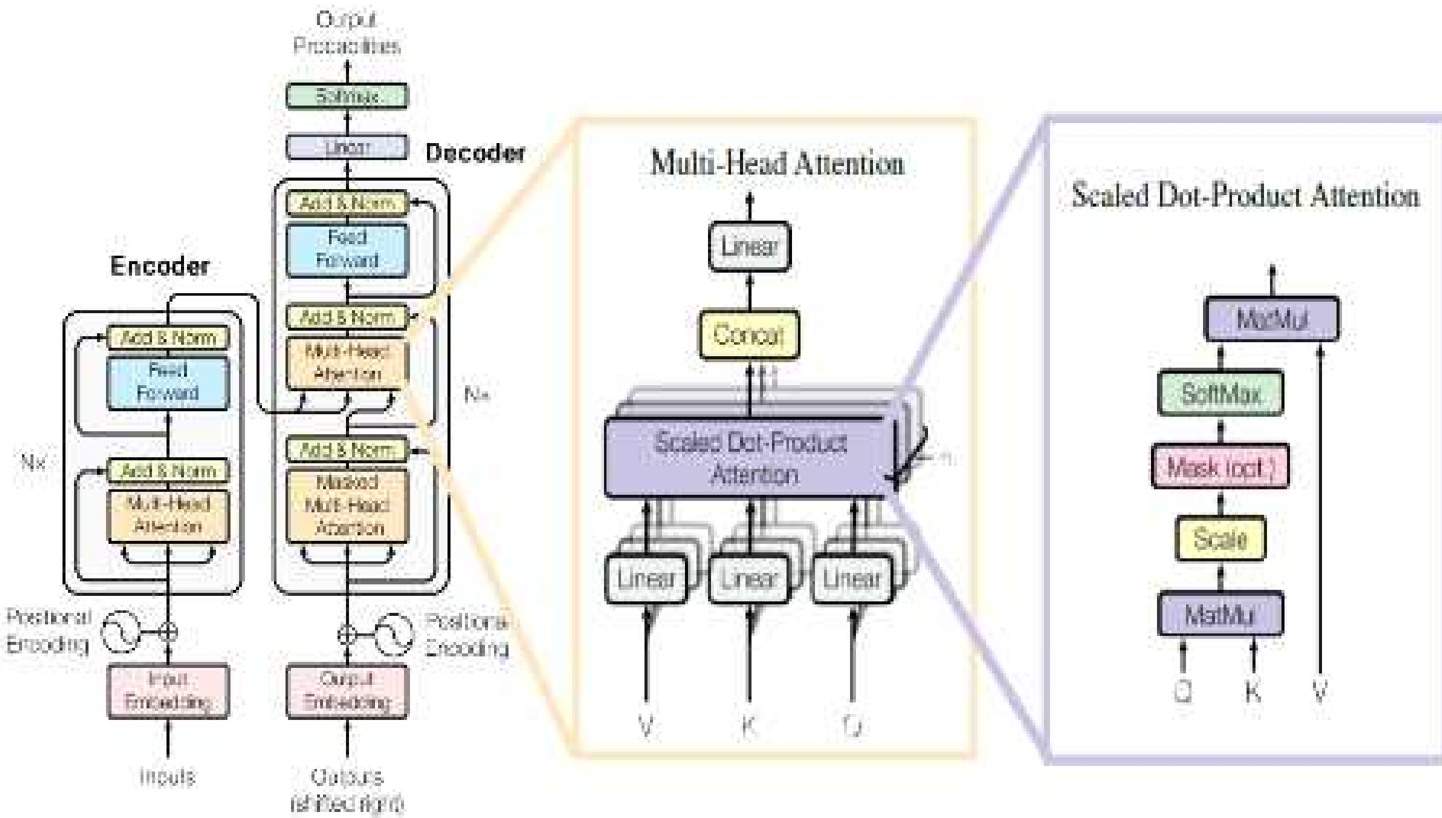
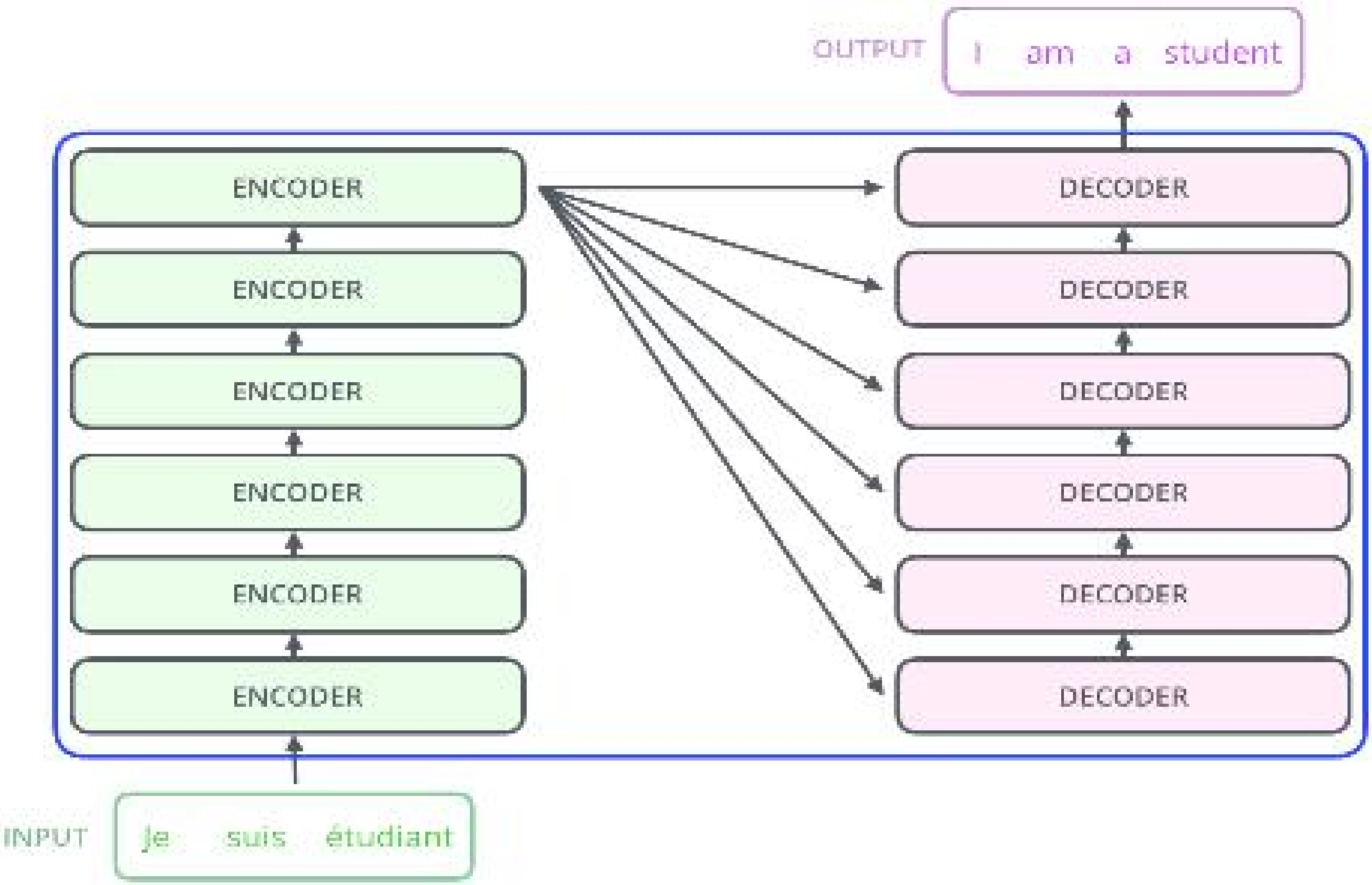


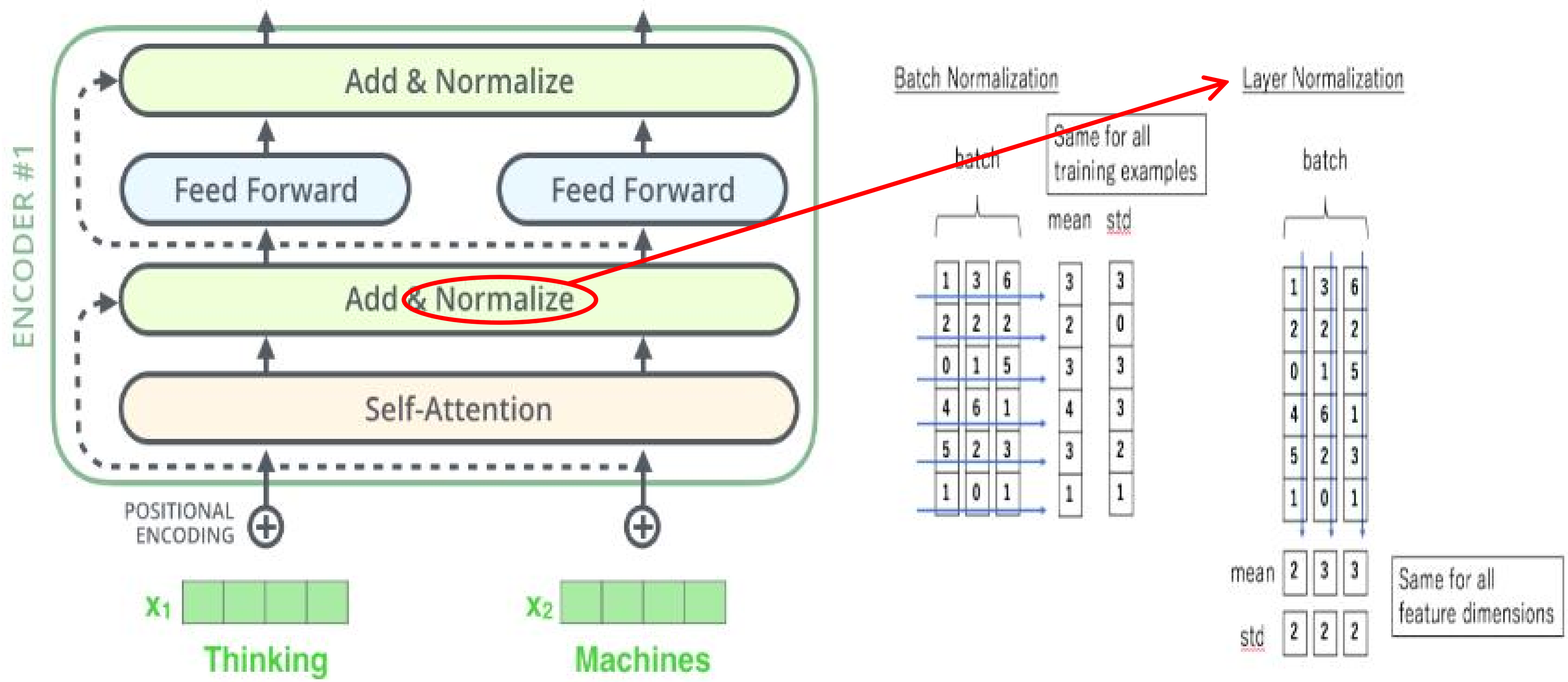
Figure 1: The Transformer - model architecture.

Encoder

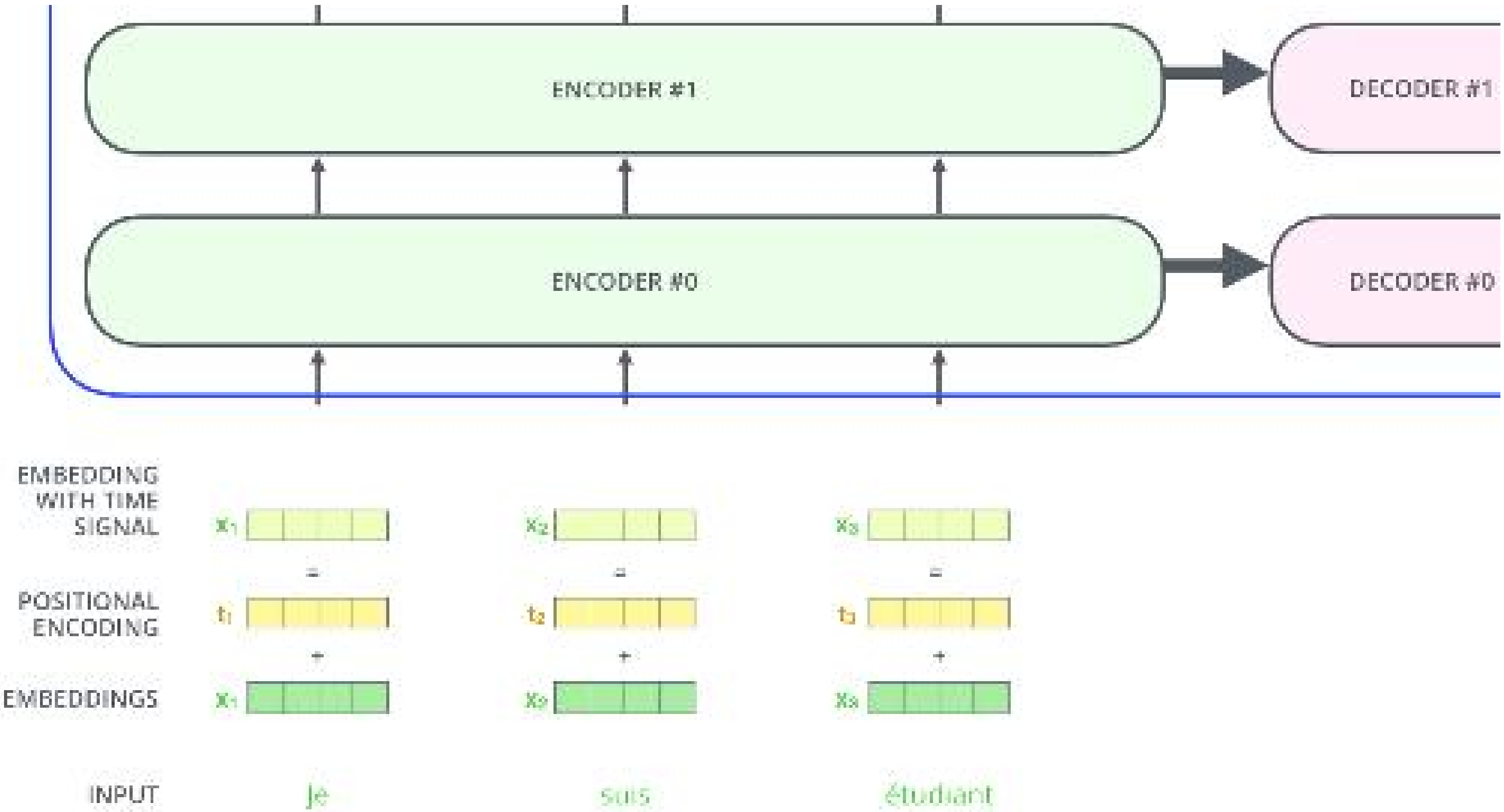


OUTPUT
I am a student
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
DECODER
DECODER
DECODER
DECODER
DECODER
DECODER
INPUT
je suis étudiant

Layer Normalization



Positional Encoding



Positional Encoding

<How to Positional Encoding>

1. 각 time-step(문장에서 단어의 위치)에 대해 고유한 인코딩을 출력해야 합니다.
2. 일정 time-step 사이의 거리는 길이가 다른 문장끼리 일정해야 한다.
3. test data에서 train data에서보다 더 긴 시퀀스가 들어왔을때도 처리할 수 있어야 한다. (더 긴 문장도 일반화가 가능해야함)

-> 상한(upper bound)이 필요하다.

4. 정확하게 위치를 결정할 수 있어야한다.(확률값 x)

- Training sample

Sentence	Dark	horses	are	faster	than	white	horses
Pos 1	1	2	3	4	5	6	7
Pos 2	0.14	0.28	0.42	0.56	0.70	0.84	1.00

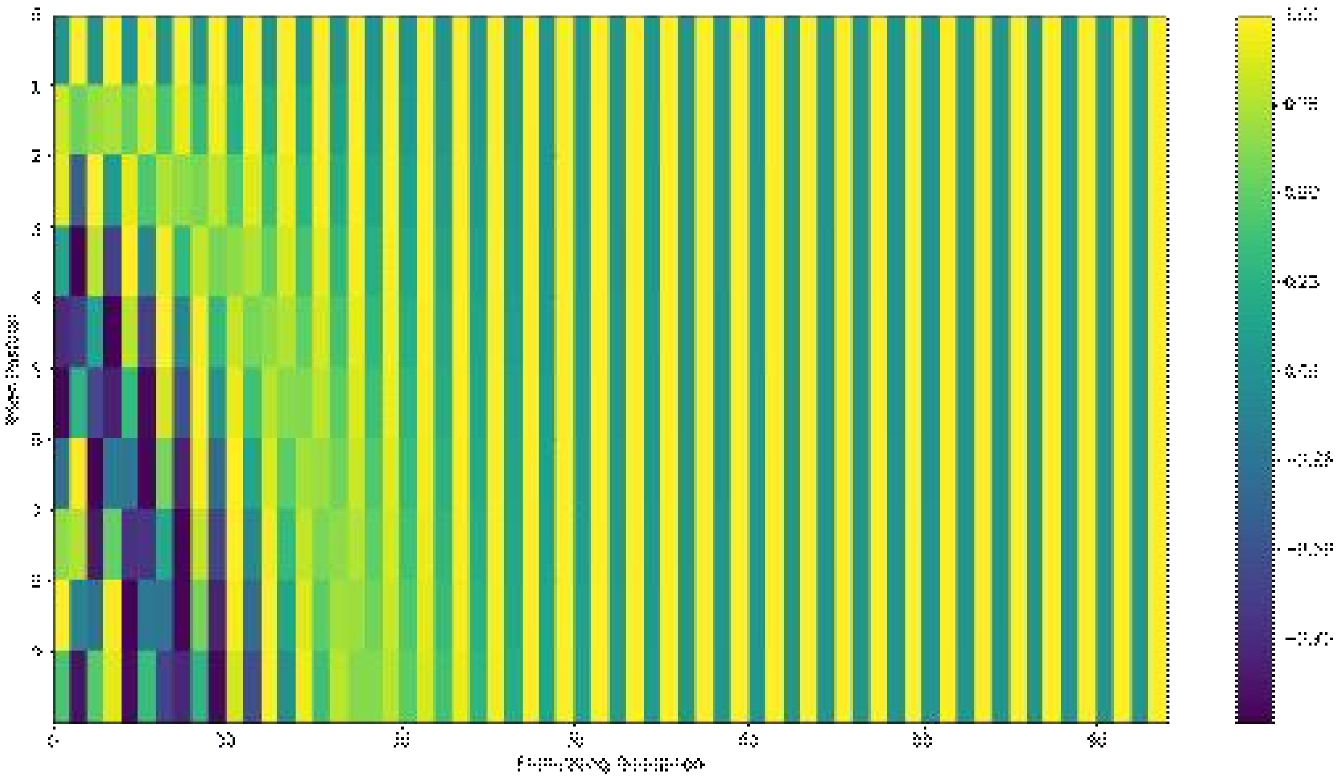
- Test sample

Sentence	Dark	horses	might	be	faster	than	white	horses
Pos 1	1	2	3	4	5	6	7	8
Pos 2	0.12	0.25	0.37	0.50	0.62	0.75	0.87	1.00

Positional Encoding

$$\vec{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k+1 \end{cases} \qquad \omega_k = \frac{1}{10000^{2k/d}}$$

$$\vec{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

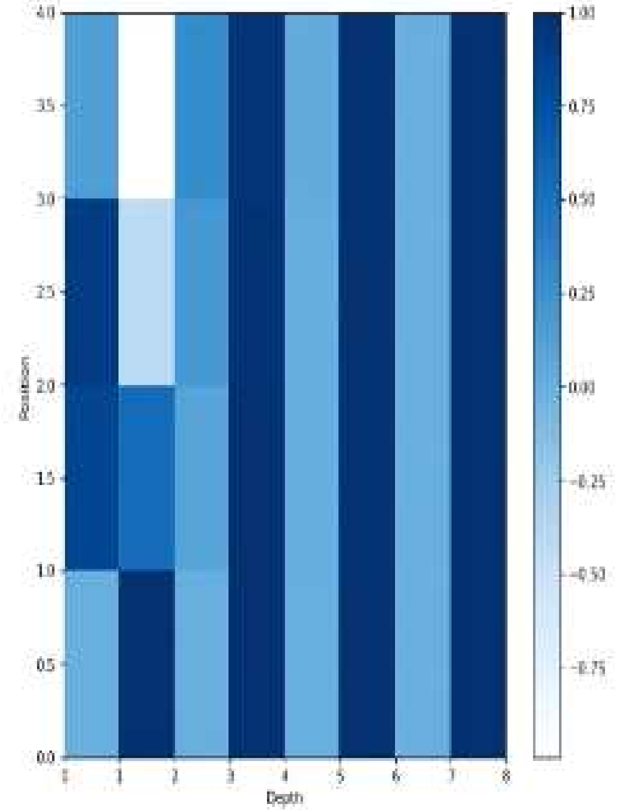


Positional Encoding

$$\vec{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

```python
import math
import matplotlib.pyplot as plt

n = 4 # 단어(word)의 개수
dim = 8 # 임베딩(embedding) 차원

def get_angles(pos, i, dim):
    angles = 1 / math.pow(10000, (2 * (i // 2)) / dim)
    return pos * angles

def get_positional_encoding(pos, i, dim):
    if i % 2 == 0: # 짝수인 경우 사인 함수
        return math.sin(get_angles(pos, i, dim))
    # 홀수인 경우 코사인 함수
    return math.cos(get_angles(pos, i, dim))

result = [[0] * dim for _ in range(n)]

for i in range(n):
    for j in range(dim):
        result[i][j] = get_positional_encoding(i, j, dim)
```

출력 결과: plt.pcolormesh(result, cmap='Blues')



Decoder

- Attention (Self-Attention, Multi-head Attention)
- Encoder
- Decoder
- Result

Decoder

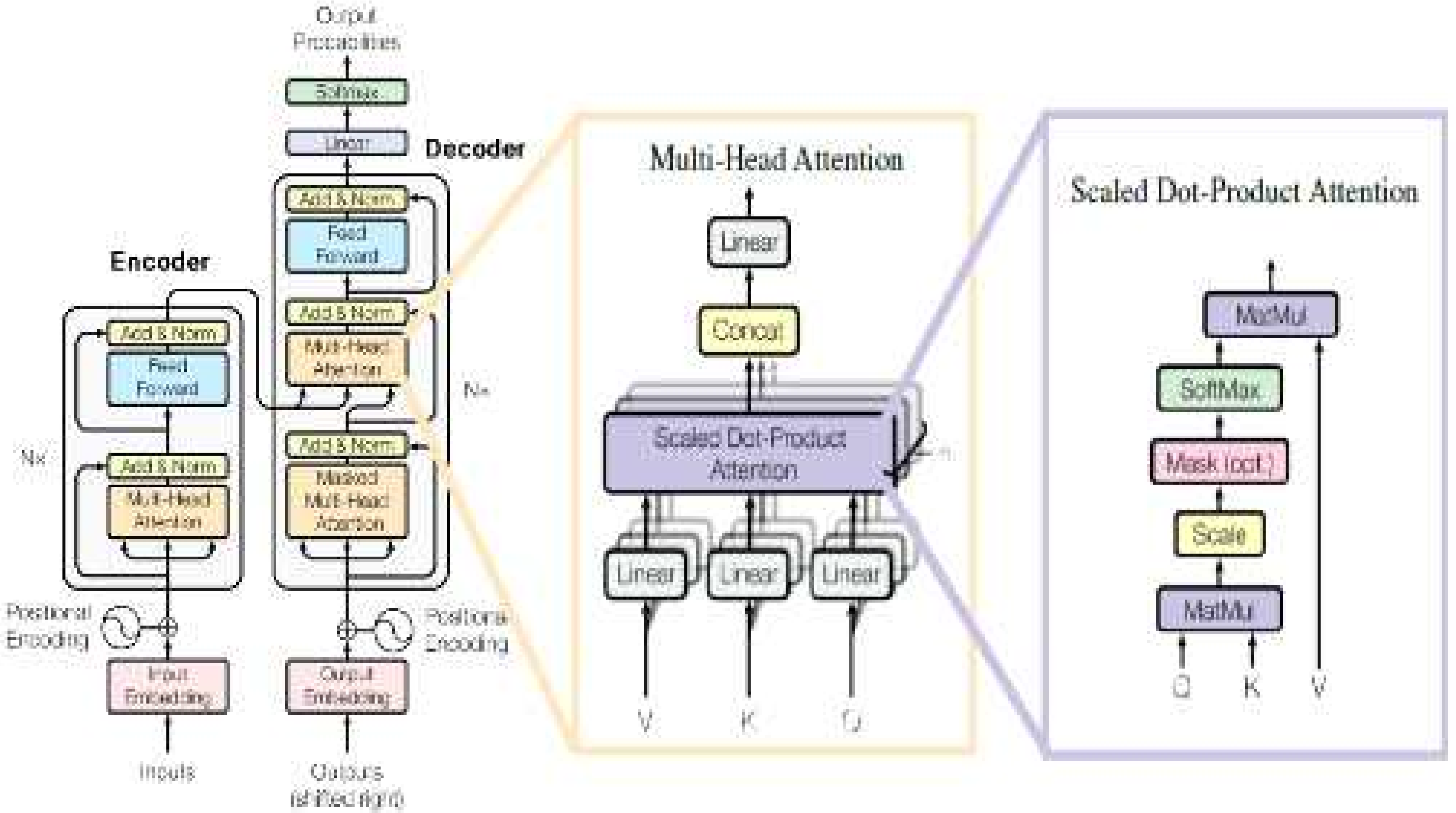
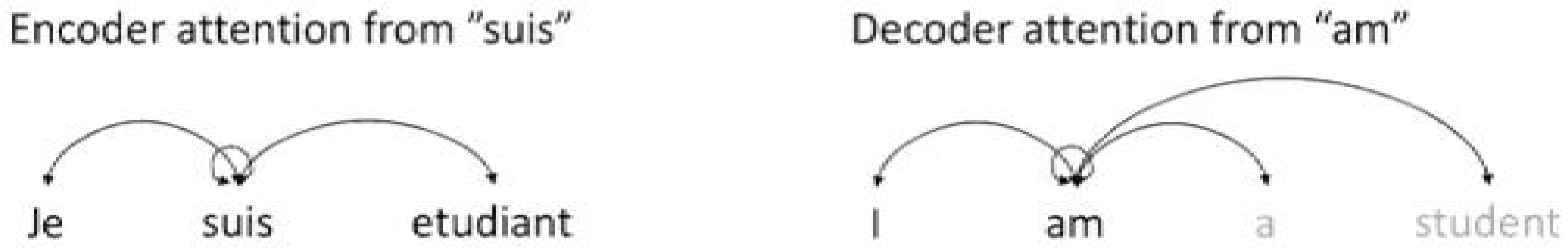
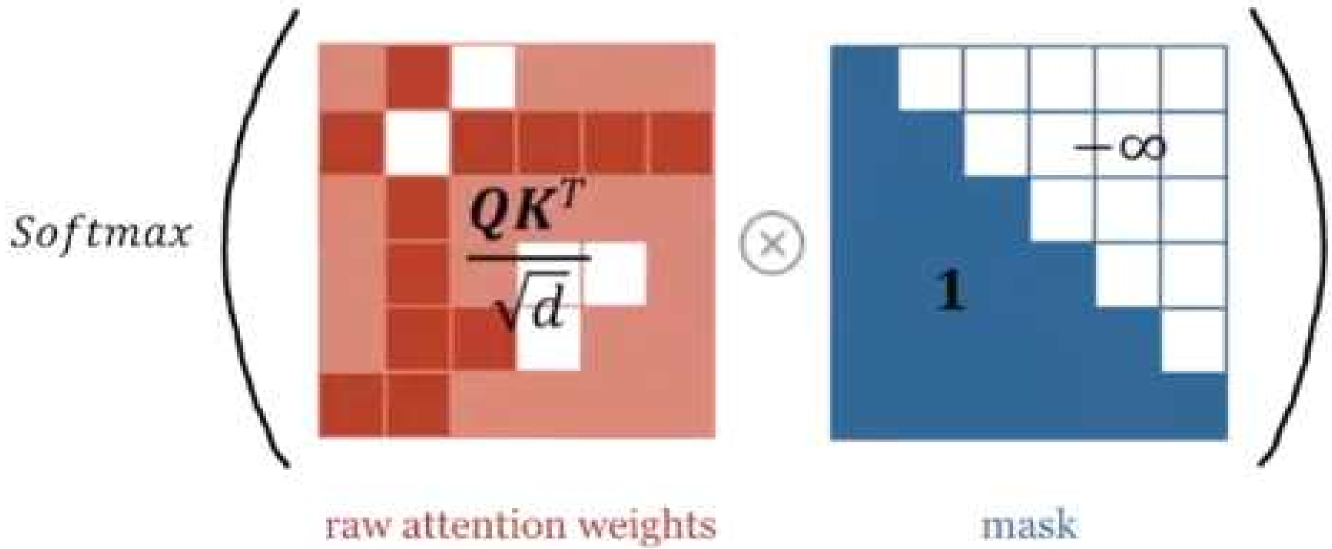


Figure 1: The Transformer - model architecture.

Masked Multi-Head Attention





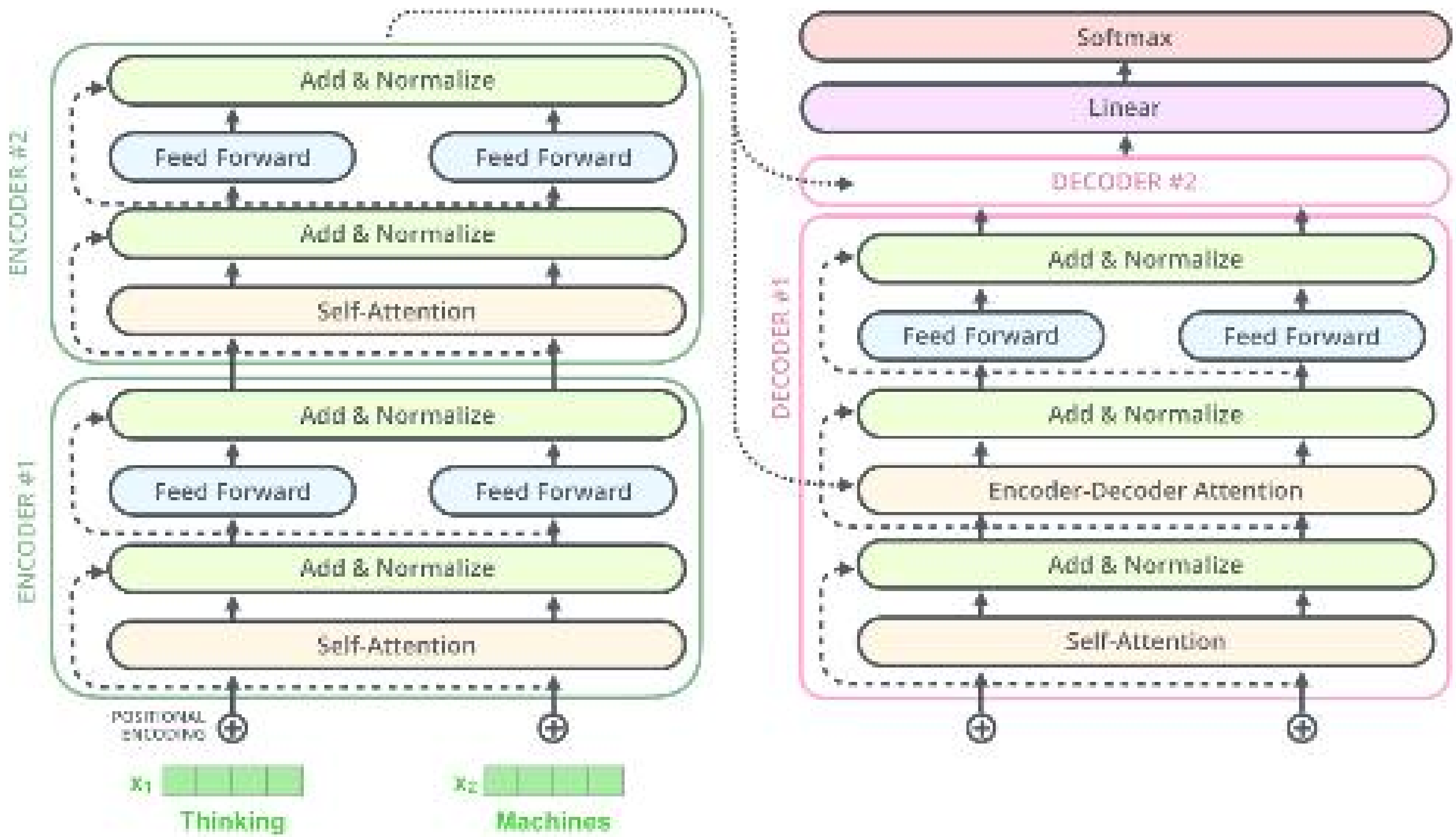
Masked Multi-Head Attention

1.0	0	0	0	0	0
0.3	0.7	0	0	0	0
0.2	0.3	0.5	0	0	0
0.1	0.2	0.1	0.6	0	0
..	..	..	..	..	0
..	..	..	..	..	..

I
like
going
to
movies
.

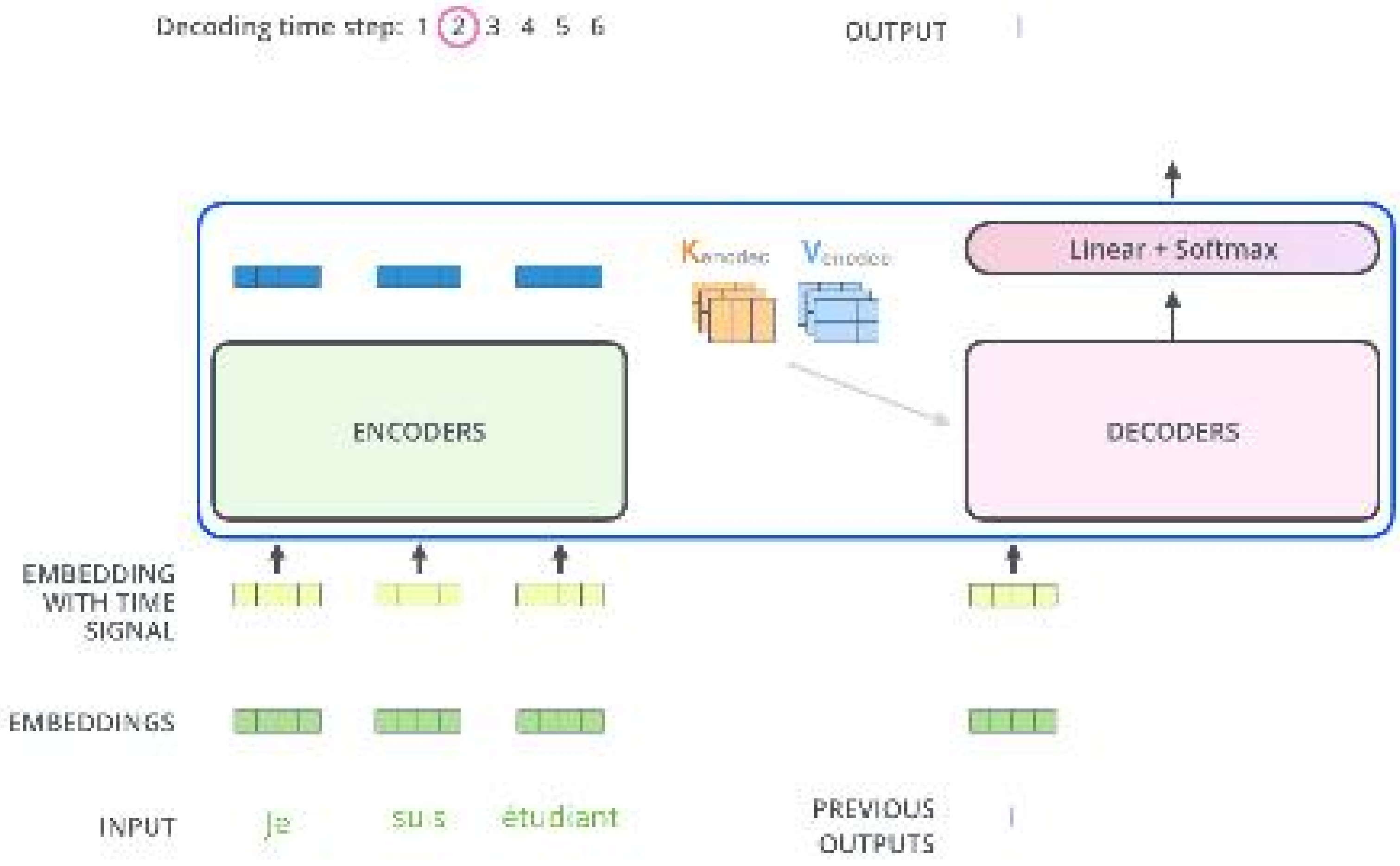
1.0*I
0.3*I + 0.7*like
0.2*I + 0.3*like + 0.5*going
0.1*I + 0.2*like + 0.1*going + 0.6*to
..
..

Decoder

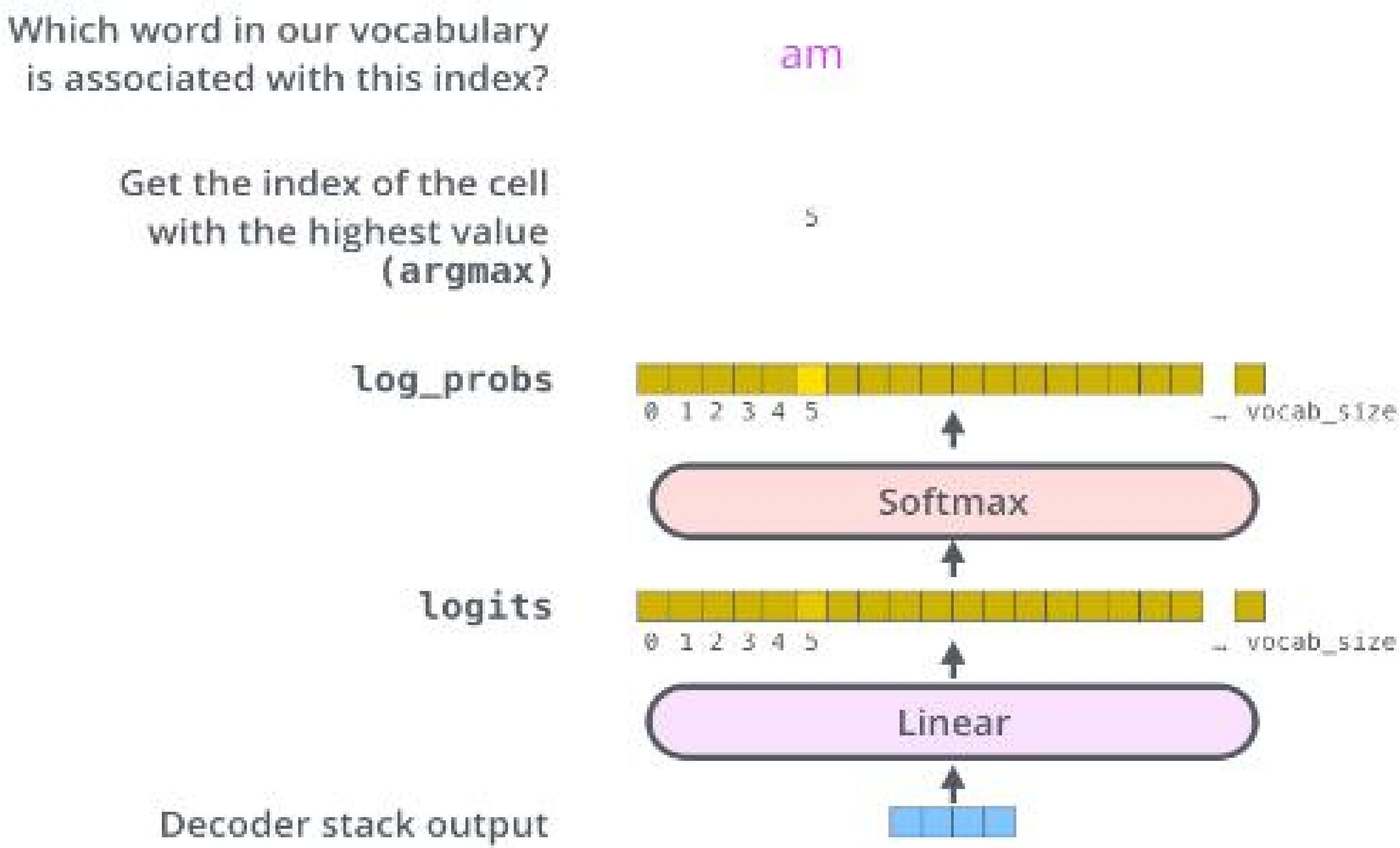


Softmax
Linear
DECODER #2
DECODER #1
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Encoder-Decoder Attention
Add & Normalize
Self-Attention
ENCODER #2
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Self-Attention
ENCODER #1
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Self-Attention
POSITIONAL ENCODING
x_1
Thinking
x_2
Machines

Decoder


Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Linear + Softmax
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
je
suis
étudiant
PREVIOUS OUTPUTS
I

Decoder


Which word in our vocabulary is associated with this index?
am
Get the index of the cell with the highest value
(argmax)
5
log_probs
0 1 2 3 4 5
… vocab_size
Softmax
logits
0 1 2 3 4 5
… vocab_size
Linear
Decoder stack output

Reference

- https://arxiv.org/pdf/1706.03762.pdf
- https://kazemnejad.com/blog/transformer_architecture_positional_encoding/#what-is-positional-encoding-and-why-do-we-need-it-in-the-first-place
- https://www.youtube.com/watch?v=AA621UofTUA
- https://www.youtube.com/watch?v=h8avp8yDKV4&list=PLLENHvsRRLjDHllrXj0B8sz5-4xVbisBL&index=24

Q & A

Thank you.