자연어처리 책 정리 발표 2주차

2022.03.25(금)

목차

1. n-gram
2. NNLM
3. Transformer

1. n-gram

주어진 문장에 대해 어떻게 확률을 구할 수 있을까?

어쩔, TV

1. n-gram

어쩔, TV

P(어쩔, TV)

1. n-gram

어쩔, TV

P(어쩔, TV)

P(어쩔, TV) = P(어쩔)P(TV|어쩔)

because $P(\text{TV} \mid \text{어쩔}) = \dfrac{P(\text{어쩔}, \text{TV})}{P(\text{어쩔})}$

1. n-gram

$$P(\text{어쩔}, TV, \cdots, \text{쿠쿠루삥뽕}) = P(\text{어쩔})P(TV\,|\,\text{어쩔})P(\text{어쩔}\,|\,\text{어쩔}, TV)\cdots P(\text{쿠쿠루삥뽕}\,|\,\text{어쩔}, TV, \text{어쩔}, \text{냉장고})$$

1. n-gram

$$\underline{P(\text{어쩔}, TV, \text{어쩔}, \text{냉장고})} = \underline{P(\text{냉장고} \mid \text{어쩔}, TV, \text{어쩔})P(\text{어쩔}, TV, \text{어쩔})}$$

1. n-gram

$$
\begin{aligned}
P(\text{어쩔}, TV, \text{어쩔}, \text{냉장고}) &= P(\text{냉장고} \mid \text{어쩔}, TV, \text{어쩔})\underline{P(\text{어쩔}, TV, \text{어쩔})} \\
&= P(\text{냉장고} \mid \text{어쩔}, TV, \text{어쩔})\underline{P(\text{어쩔} \mid \text{어쩔}, TV)P(\text{어쩔}, TV)}
\end{aligned}
$$

1. n-gram

$$
\begin{aligned}
P(\text{어쩔}, TV, \text{어쩔}, \text{냉장고}) &= P(\text{냉장고} \mid \text{어쩔}, TV, \text{어쩔})P(\text{어쩔}, TV, \text{어쩔}) \\
&= P(\text{냉장고} \mid \text{어쩔}, TV, \text{어쩔})P(\text{어쩔} \mid \text{어쩔}, TV)\underline{P(\text{어쩔}, TV)} \\
&= P(\text{냉장고} \mid \text{어쩔}, TV, \text{어쩔})P(\text{어쩔} \mid \text{어쩔}, TV)\underline{P(TV \mid \text{어쩔})P(\text{어쩔})}
\end{aligned}
$$

1. n-gram

$$P(w_1, w_2, \cdots, w_n) = \prod_{i=1}^{n} P(w_i \mid w_{<i})$$

1. n-gram

$$\log P(w_1, w_2, \cdots, w_n) = \sum_{i=1}^{n} \log P(w_i \mid w_{<i})$$

1. n-gram

$$P(\text{냉장고} \mid BOS, \text{어쩔}, TV, \text{어쩔}) \approx \frac{Count(BOS, \text{어쩔}, TV, \text{어쩔}, \text{냉장고})}{Count(BOS, \text{어쩔}, TV, \text{어쩔})}$$

1. n-gram

$$P(\text{냉장고} \mid BOS, \text{어쩔}, TV, \text{어쩔}) \approx \frac{Count(BOS, \text{어쩔}, TV, \text{어쩔}, \text{냉장고})}{Count(BOS, \text{어쩔}, TV, \text{어쩔})}$$

희소성 문제

마르코프 가정 (Markov assumption)

1. n-gram

$$P(x_i \mid x_1, x_2, \cdots, x_{i-1}) \approx P(x_i \mid x_{i-k}, \cdots, x_{i-1})$$

1. n-gram

$$P(w_1, w_2, \cdots, w_n) = \prod_{i=1}^{n} P(w_i \mid w_{<i})$$

1. n-gram

$$P(w_1, w_2, \cdots, w_n) = \prod_{i=1}^{n} P(w_i \mid w_{<i})$$

$$\approx \prod_{i=1}^{n} P(x_i \mid x_{i-k}, \cdots, x_{i-1})$$

1. n-gram

$$P(w_1, w_2, \cdots, w_n) = \prod_{i=1}^{n} P(w_i \mid w_{<i})$$

$$\approx \prod_{i=1}^{n} P(x_i \mid x_{i-k}, \cdots, x_{i-1})$$

$$\log P(w_1, w_2, \cdots, w_n) = \sum_{i=1}^{n} \log P(w_i \mid w_{i-k}, \cdots, x_{i-1})$$

1. n-gram

k	n-gram	명칭
0	1-gram	uni-gram
1	2-gram	bi-gram
2	3-gram	tri-gram

1. n-gram

k	n-gram	명칭
0	1-gram	uni-gram
1	2-gram	bi-gram
2	3-gram	tri-gram

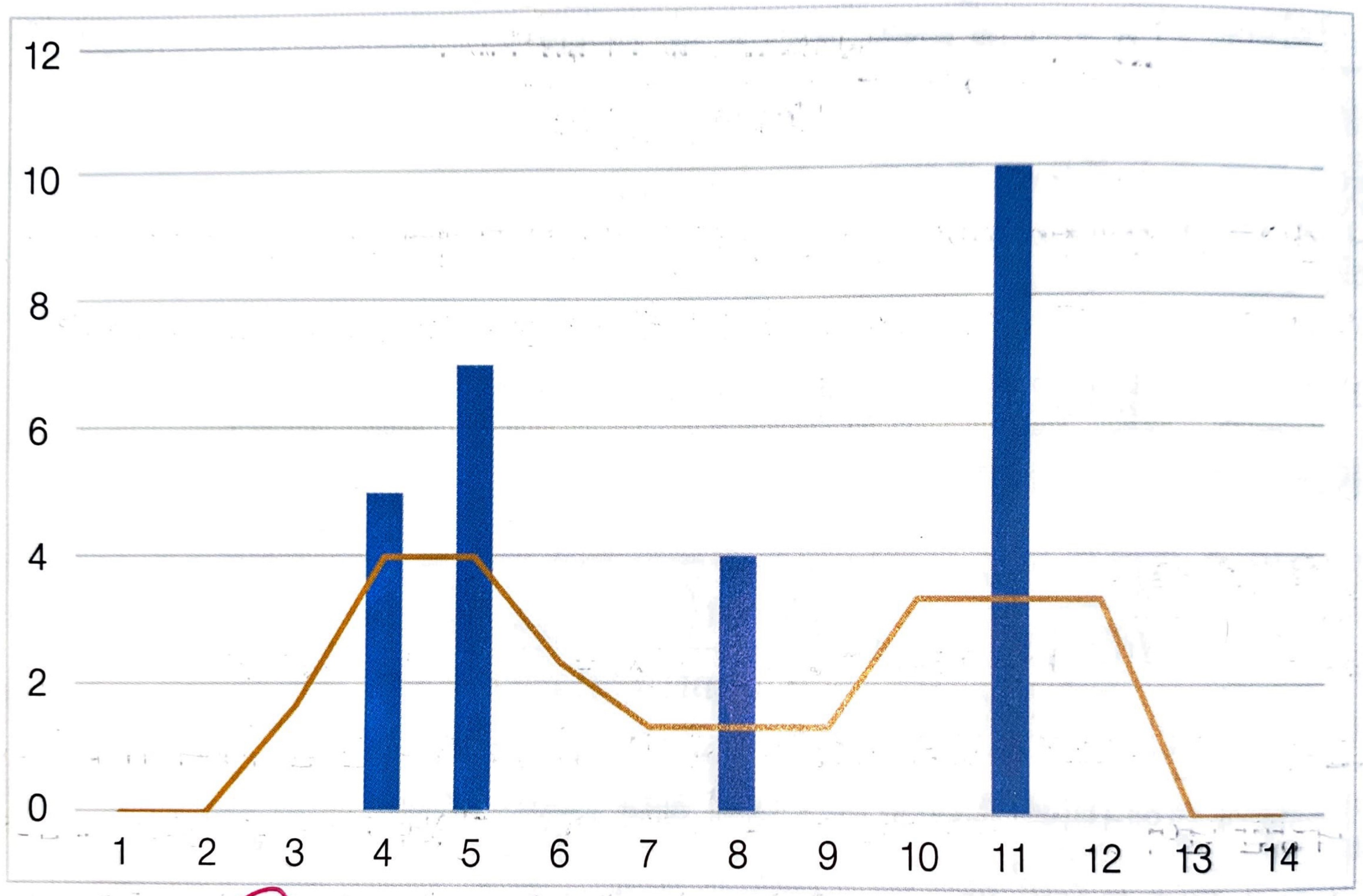
1. n-gram

$$P(x_i \mid x_{i-2}, x_{i-1}) = \frac{Count(x_{i-2}, x_{i-1}, x_i)}{Count(x_{i-2}, x_{i-1})}$$

일반화

출현 횟수를 단순히 확률값으로 추정할 경우 문제가 없을까?

스무딩과 디스카운팅



▶ 스무딩을 통해 분포의 모양을 좀 더 평탄하게 만들 수 있습니다.

문장 출현 빈도.

1. n-gram

$$P(w_i \mid w_{<i}) \approx \frac{Count(w_{<i}, w_i) \underline{+ 1}}{Count(w_{<i}) \underline{+ V}}$$

1. n-gram

$$P(w_i \mid w_{<i}) \approx \frac{Count(w_{<i}, w_i) + k}{Count(w_{<i}) + kV}$$

$$\approx \frac{Count(w_{<i}, w_i) + (m/V)}{Count(w_{<i}) + m}$$

1. n-gram

$$P(w_i \mid w_{<i}) \approx \frac{Count(w_{<i}, w_i) + m\underline{P(w_i)}}{Count(w_{<i}) + m}$$

Kneser-Ney(KN) 디스카운팅

machine learning

deep learning

learning ↑

↓ laptop

1. n-gram

$$Score_{continuation}(w) \propto |\{v : Count(v, w) > 0\}|$$

w 와 함께 나타난 v 들의 집합의 크기

1. n-gram

w 와 함께 나타난 v 들의 집합의 크기

$$Score_{continuation}(w) = \frac{|\{v : Count(v, w) > 0\}|}{\sum_{w' \in W} |\{v : Count(v, w') > 0)\}|}$$

전체 단어 집합에서 샘플링한 w' 이 v 와 함께 나타난 집합의 크기 합

1. n-gram

출현 빈도를 확률로 근사한 방법과 유사

다양한 단어 뒤에 나타나는 단어의 점수

$$P_{KN}(w_i \mid w_{i-1}) = \frac{\max(Count(w_{i-1}, w_i) - d, 0)}{Count(w_{i-1})} + \lambda(w_{i-1}) \times Score_{continuation}(w_i),$$

$$where\ \lambda(w_{i-1}) = \frac{d}{\sum_v Count(w_{i-1})} \times |\{w : c(w_{i-1}, v) > 0)\}|$$

인터폴레이션(보간)
interpolation

1. n-gram

$$\tilde{P}(w_n \mid w_{n-k}, \cdots, w_{n-1}) = \underline{\lambda} P_1(w_n \mid w_{n-k}, \cdots, w_{n-1}) + \underline{(1-\lambda)} P_2(w_n \mid w_{n-k}, \cdots, w_{n-1})$$

$$where \ 0 < \lambda < 1$$

1. n-gram

일반 영역

- P(진정제 | 준비, 된) = 0.00001
- P(사나이 | 준비, 된) = 0.01

1. n-gram

일반 영역

- P(진정제 | 준비, 된) = 0.00001
- P(사나이 | 준비, 된) = 0.01

특화 영역

- P(진정제 | 준비, 된) = 0.09
- P(약 | 준비, 된) = 0.04

1. n-gram

일반 영역

- P(진정제 | 준비, 된) = 0.00001
- P(사나이 | 준비, 된) = 0.01

특화 영역

- P(진정제 | 준비, 된) = 0.09
- P(약 | 준비, 된) = 0.04

인터폴레이션 결과

- P(진정제 | 준비, 된) = $\underbrace{0.5}_{\lambda}$ * 0.09 + $\underbrace{(1 - 0.5)}_{1-\lambda}$ * 0.00001 = 0.045005

백오프(back-off)

1. n-gram

$$\begin{aligned}\tilde{P}(w_n \mid w_{n-k}, \cdots, w_{n-1}) = &\lambda_1 P(w_n \mid w_{n-k}, \cdots, w_{n-1}) \\ &+\lambda_2 P_2(w_n \mid w_{n-k}, \cdots, w_{n-1}) \\ &+\cdots \\ &+\lambda_k P(w_n),\end{aligned}$$

$$\textit{where} \ \sum_i \lambda_i = 1$$

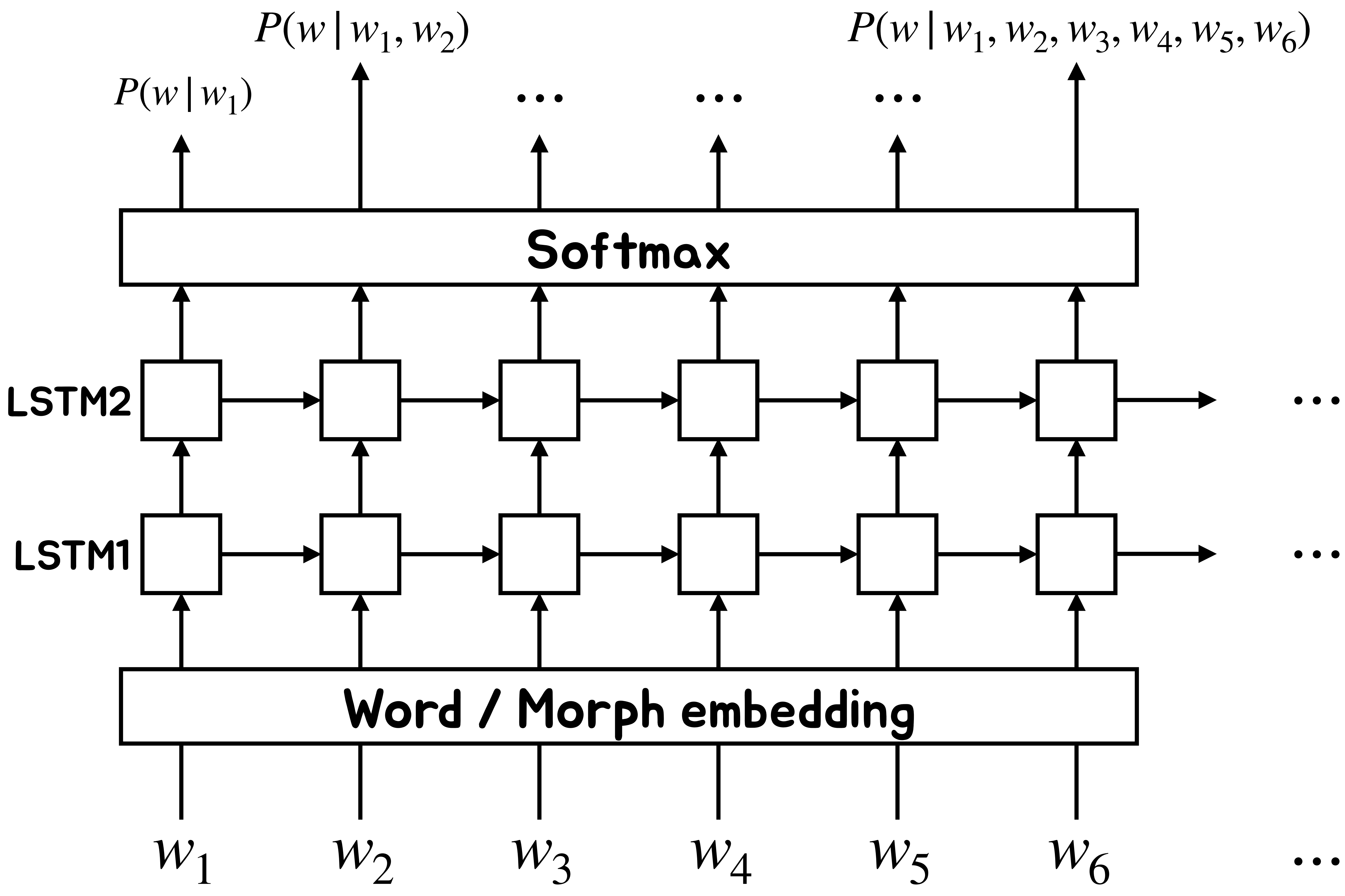
2. NNLM

고양이는 좋은 반려동물입니다.

고양이는 좋은 반려동물입니다.

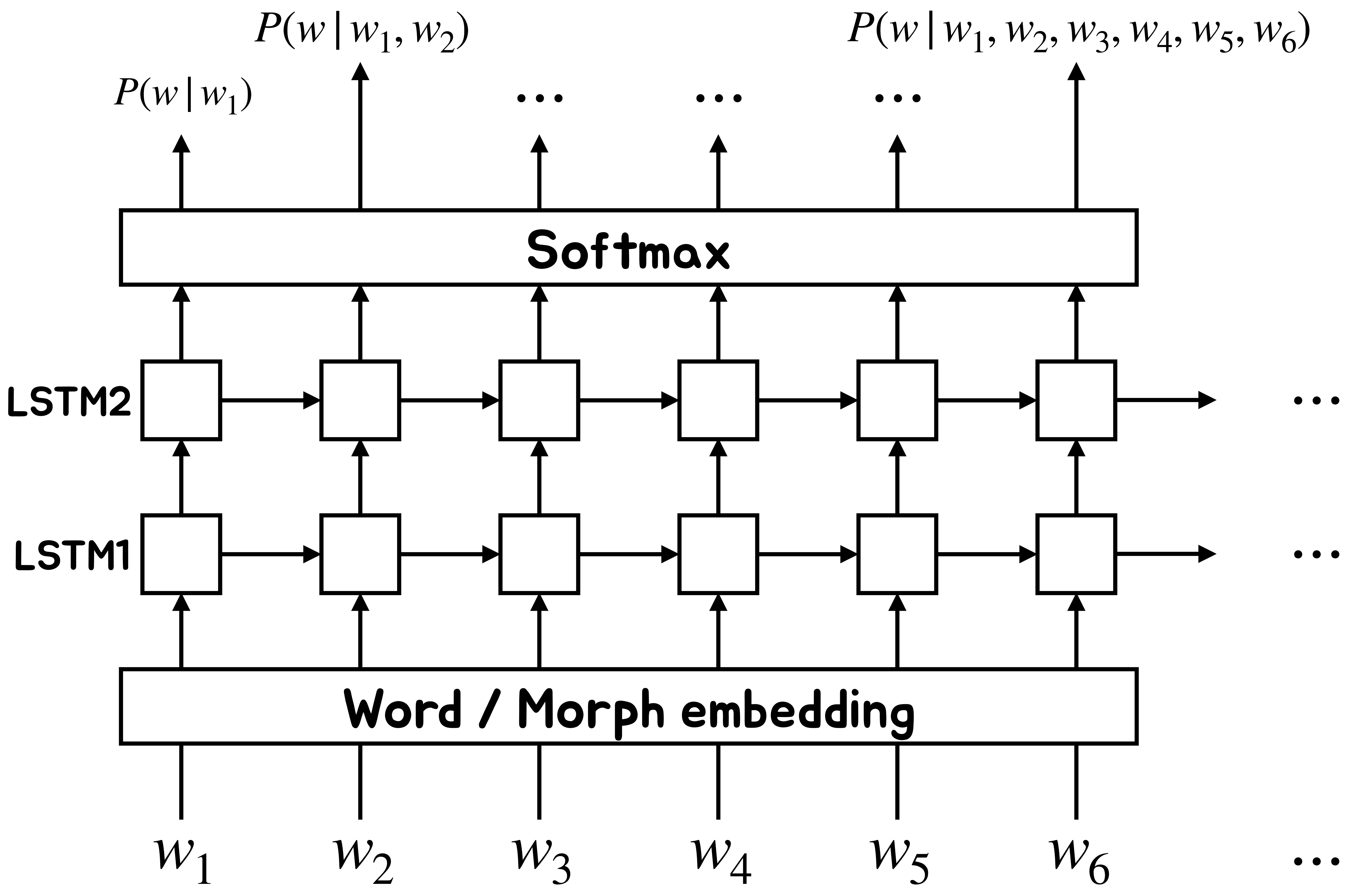
P(반려동물 | 강아지는, 좋은)
P(반려동물 | 자동차는, 좋은)

2. NNLM



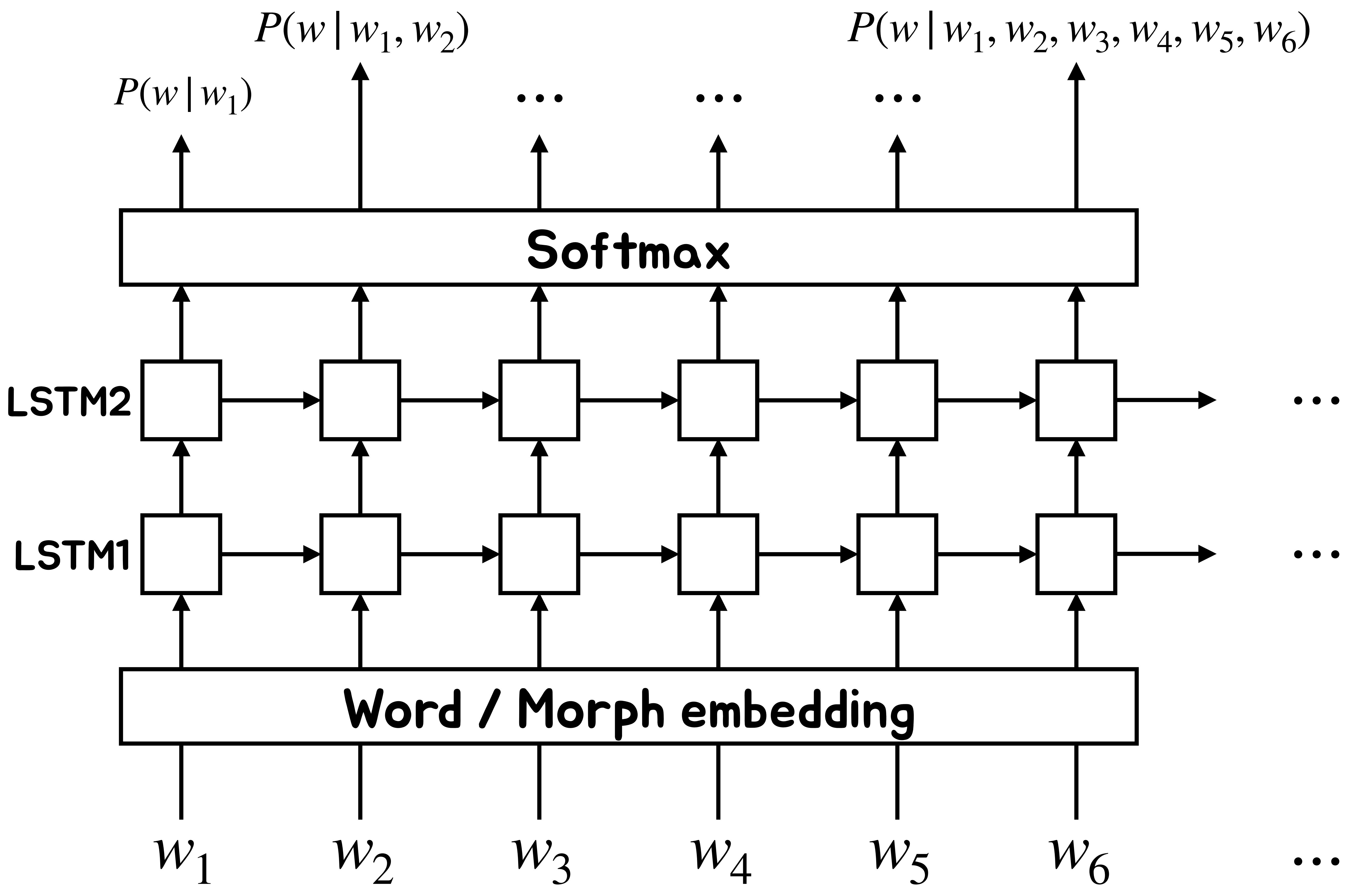
$P(w|w_1, w_2)$
$P(w|w_1, w_2, w_3, w_4, w_5, w_6)$
$P(w|w_1)$
…
…
…
Softmax
LSTM2
…
LSTM1
…
Word / Morph embedding
w_1
w_2
w_3
w_4
w_5
w_6
…

2. NNLM



$$P(w_1, w_2, \cdots, w_k) = \prod_{i=1}^{k} P(w_i \mid w_{<i})$$

2. NNLM



$$\log P(w_1, w_2, \cdots, w_k) = \sum_{i=1}^{k} \log P(w_i | w_{<i})$$

2. NNLM

$$x_{1:n} = \{x_0, x_1, \cdots, x_n, x_{n+1}\}$$

$$where\ x_0 = BOS\ and\ x_{n+1} = EOS$$

$$\hat{x}_{i+1} = softmax(linear_{hidden\_size \rightarrow |V|}(RNN(emb(x_i))))$$

$$\hat{x}_{1:n}[1:] = softmax(linear_{hidden\_size \rightarrow |V|}(RNN(emb(x_{1:n}[:-1])))),$$

$$linear_{d_1 \rightarrow d_2}(x) = Wx + b\ where\ W \in \mathbb{R}^{d_1 \times d_2}\ and\ b \in \mathbb{R}^{d_2},$$

$$and\ hidden\_size\ is\ dimension\ of\ hidden\ state\ and\ |V|\ is\ size\ of\ vocabulary.$$

2. NNLM

$$x_{1:n} = \{x_0, x_1, \cdots, x_n, x_{n+1}\}$$

$$where\ x_0 = BOS\ and\ x_{n+1} = EOS$$

$$\hat{x}_{i+1} = softmax(linear_{hidden\_size \rightarrow |V|}(RNN(emb(x_i))))$$

$$\hat{x}_{1:n}[1:] = softmax(linear_{hidden\_size \rightarrow |V|}(RNN(emb(x_{1:n}[:-1])))),$$

$$linear_{d_1 \rightarrow d_2}(x) = Wx + b\ where\ W \in \mathbb{R}^{d_1 \times d_2}\ and\ b \in \mathbb{R}^{d_2},$$

$$and\ hidden\_size\ is\ dimension\ of\ hidden\ state\ and\ |V|\ is\ size\ of\ vocabulary.$$

2. NNLM

$$x_{1:n} = \{x_0, x_1, \cdots, x_n, x_{n+1}\}$$

$$where\ x_0 = BOS\ and\ x_{n+1} = EOS$$

$$\hat{x}_{i+1} = softmax(linear_{hidden\_size \rightarrow |V|}(RNN(emb(x_i))))$$

$$\hat{x}_{1:n}[1:] = softmax(linear_{hidden\_size \rightarrow |V|}(RNN(emb(x_{1:n}[:-1])))),$$

$$linear_{d_1 \rightarrow d_2}(x) = Wx + b\ where\ W \in \mathbb{R}^{d_1 \times d_2}\ and\ b \in \mathbb{R}^{d_2},$$

$$and\ hidden\_size\ is\ dimension\ of\ hidden\ state\ and\ |V|\ is\ size\ of\ vocabulary.$$

2. NNLM

$$x_{1:n}[:-1] = \{x_0, x_1, \cdots, x_n\}$$

$$x_{emb} = emb(x_{1:n}[1:-1])$$

EOS 제외한 BOS + 입력문장(n)의 길이

$$where\ |x_{1:n}[:-1]| = (batch\_size, \underline{n+1})$$

$$and\ |x_{emb}| = (batch\_size, n+1, word\_vec\_dim)$$

2. NNLM

$$\boxed{h_{0:n} = RNN(x_{emb})}$$

$$Where\ |h_{0:n}| = (batch\_size, n+1, hidden\_size)$$

2. NNLM

$$\hat{x}_{1:n} = softmax\left(linear_{hidden\_size \rightarrow |V|}(h_{0:n})\right)$$

$$Where\ |\hat{x}_{1:n}| = (batch\_size, n+1, |V|)$$

$$and\ x_{1:n}[1:] = \{x_1, x_2, \cdots, x_{n+1}\}$$

2. NNLM

$$\mathscr{L}(\hat{x}_{1:n}, x_{1:n}[1:]) = -\frac{1}{n}\sum_{i=1}^{m}\sum_{j=1}^{n+1} x_j^i \log \hat{x}_j^i$$

$$Where\ x_j^i\ is\ one-hot\, vector$$

n-gram vs NNLM

n-gram 성능 NNLM

n-gram vs NNLM

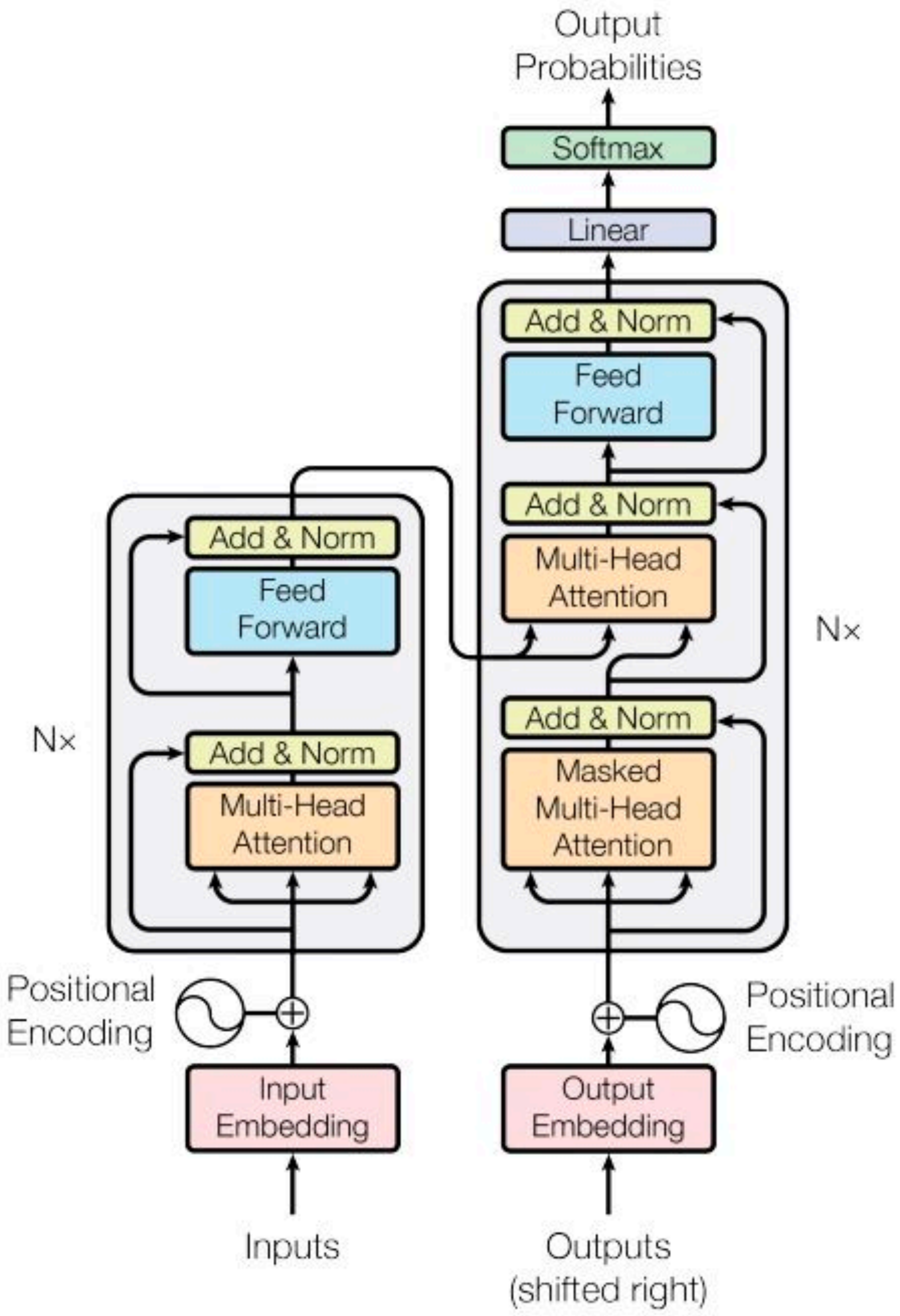
n-gram ↓ 성능 NNLM ↑

n-gram ↓ 연산량 NNLM ↑

3. Transformer

Attention is all you need

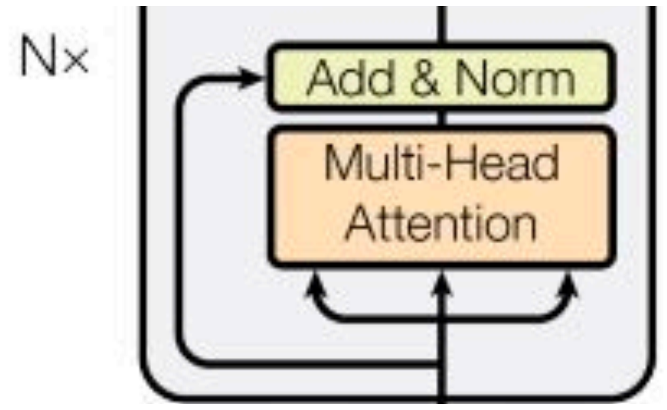
3. Transforemr



Sub Module

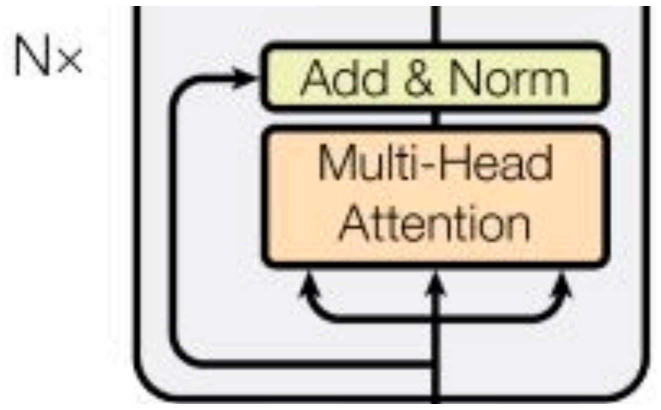
Sub Module

self-attention

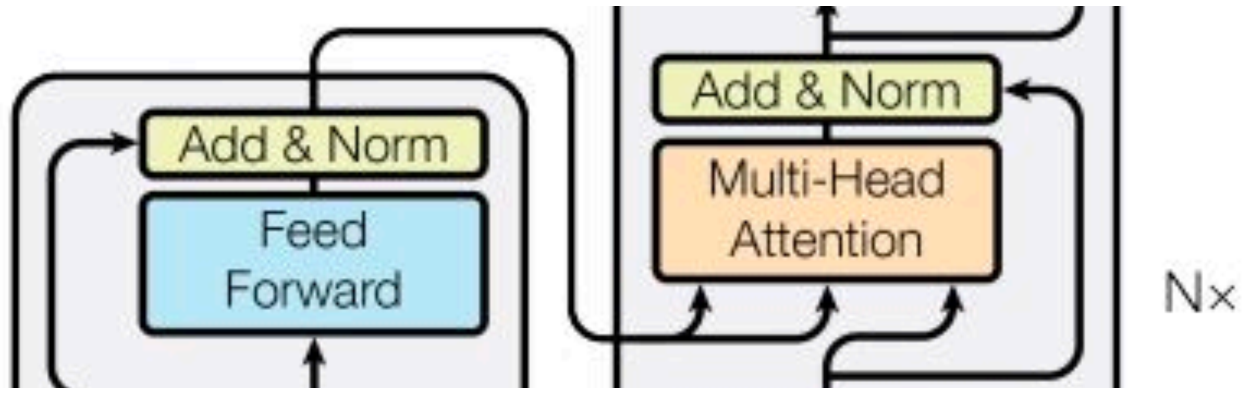


Sub Module

self-attention

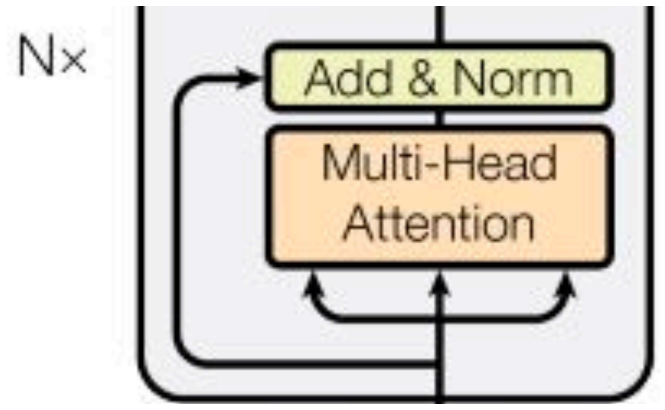


attention

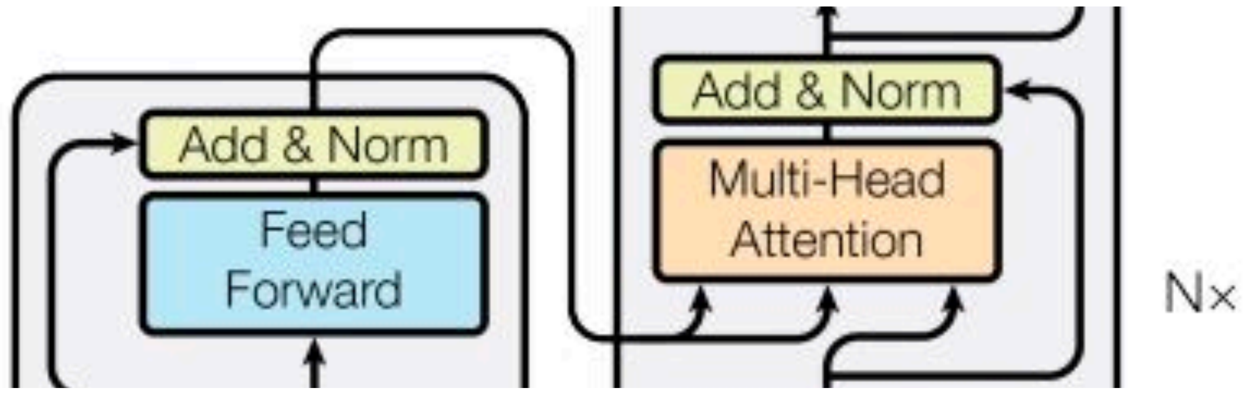


Sub Module

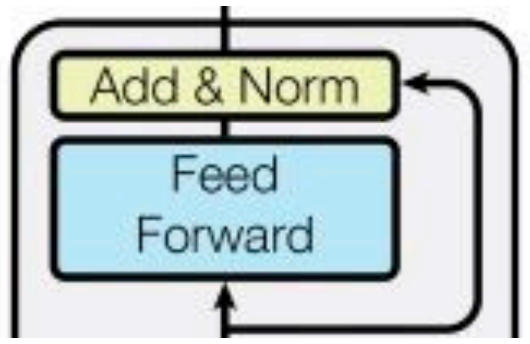
self-attention



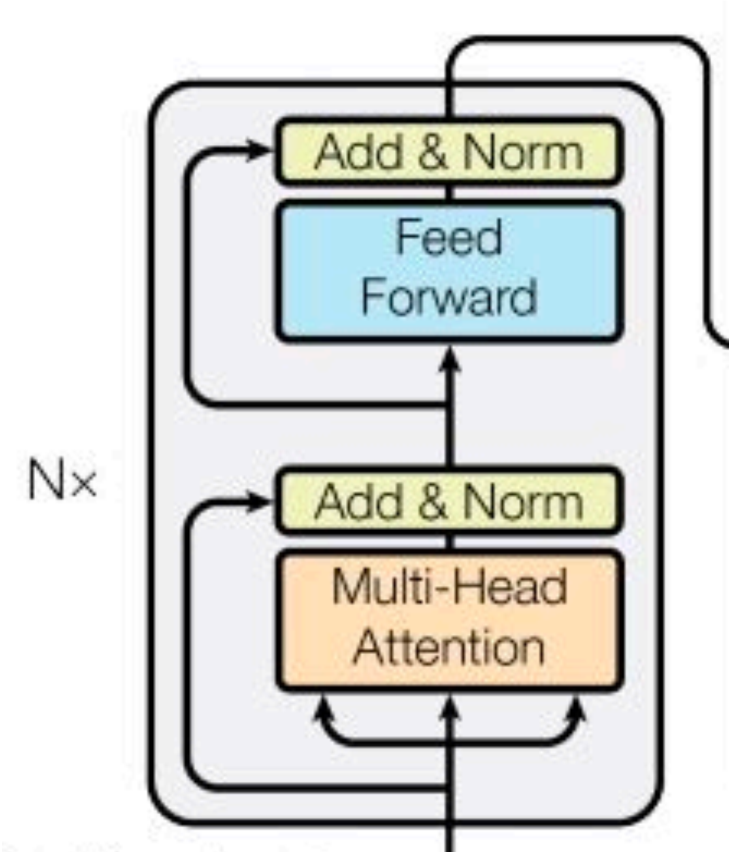
attention



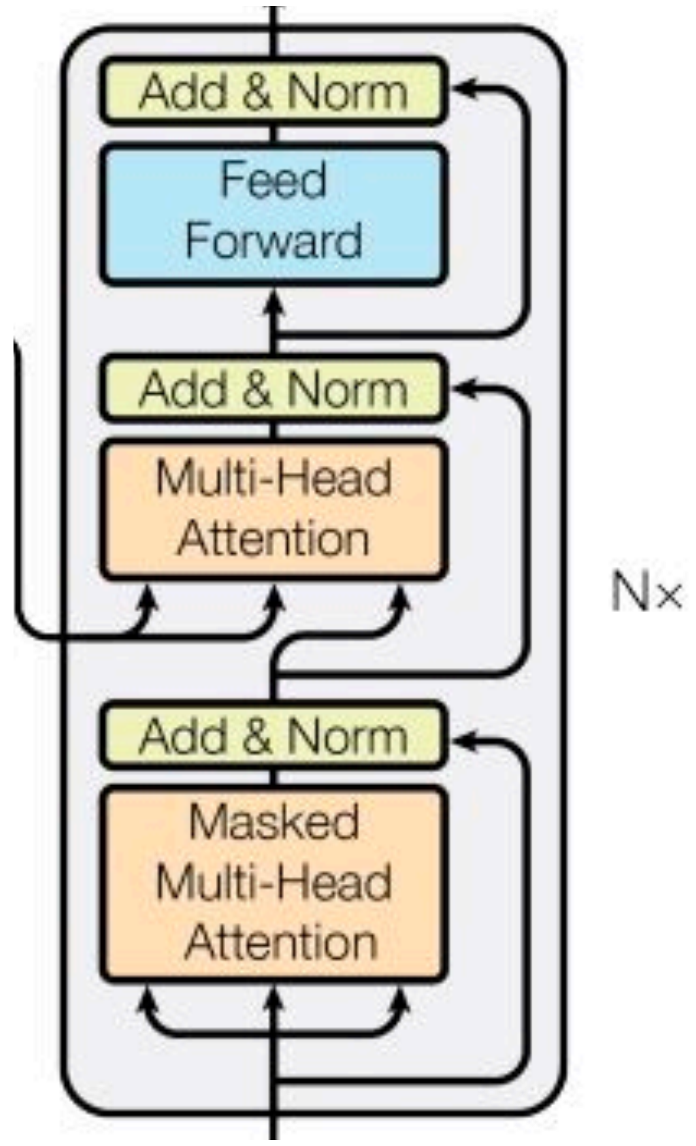
Feed Forward Layer



Encoder

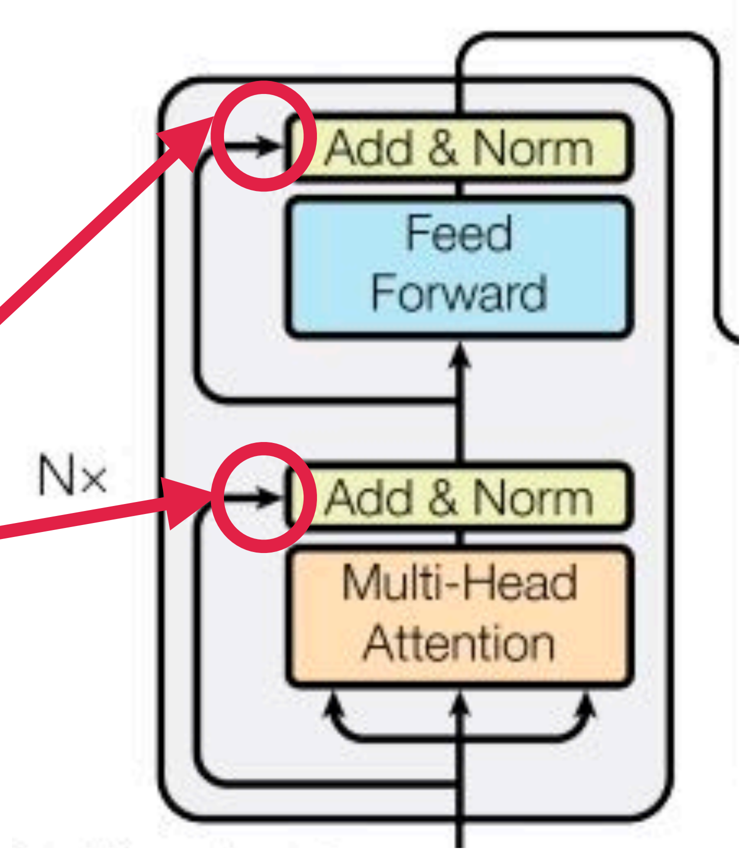


Decoder


Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx

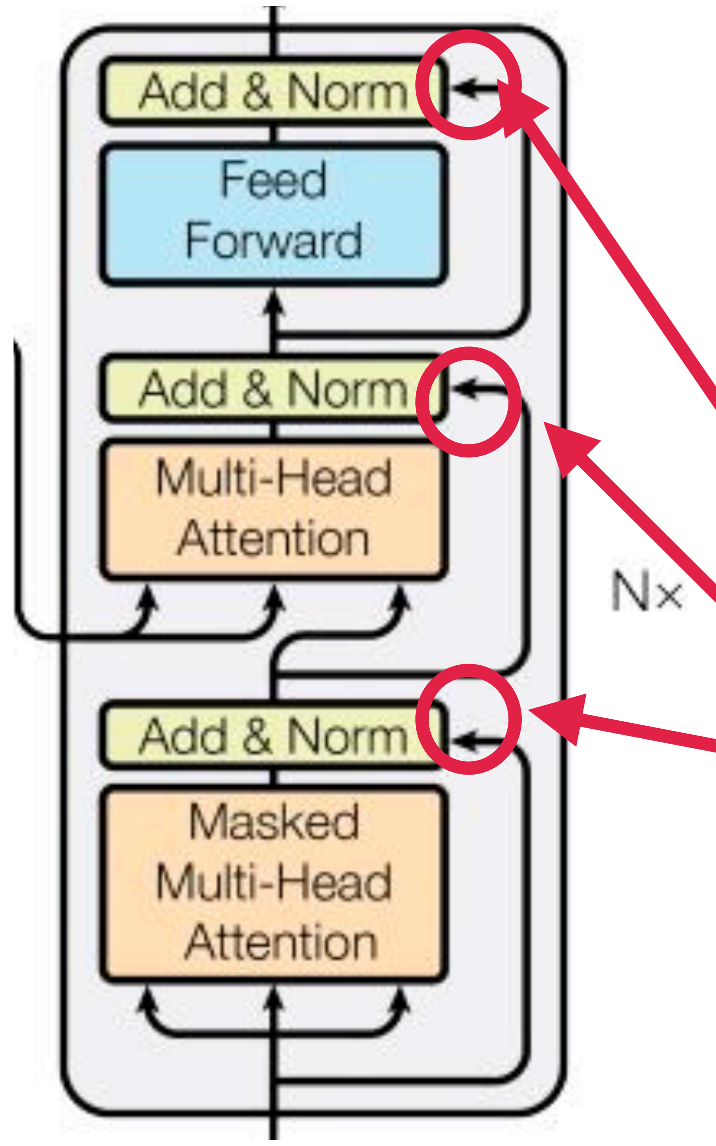
3. Transforemr

Encoder



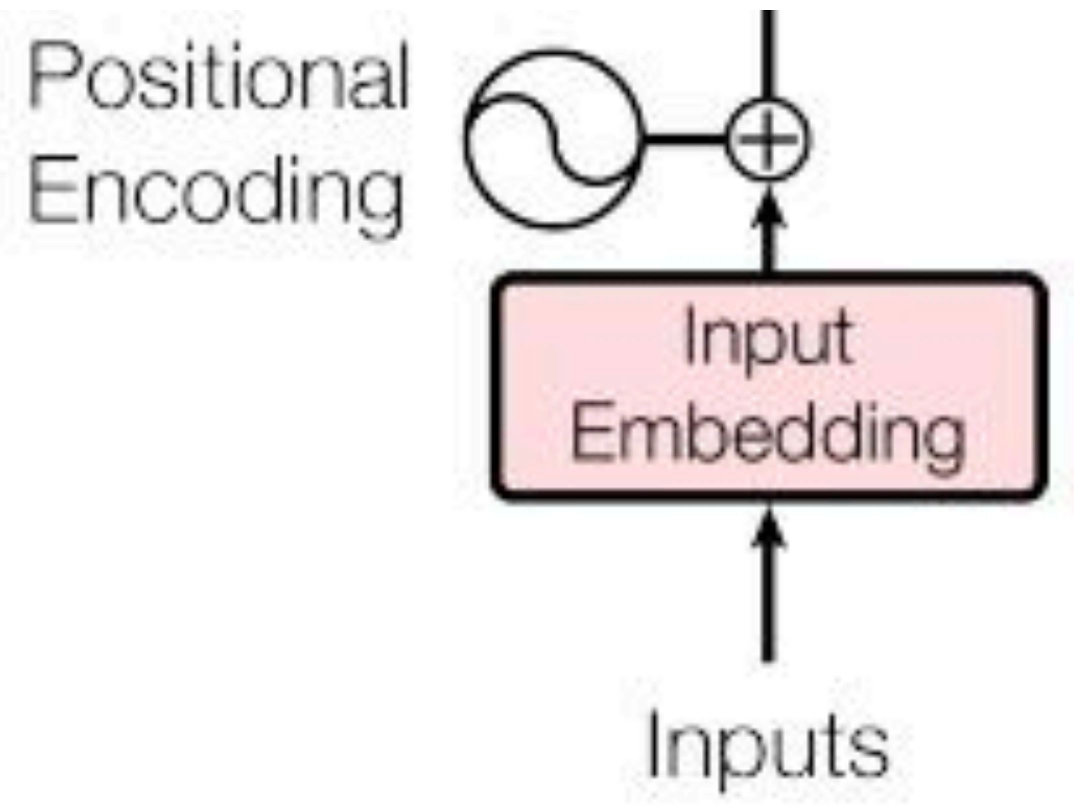
residual connection
layer-normalization

Decoder



residual connection
layer-normalization

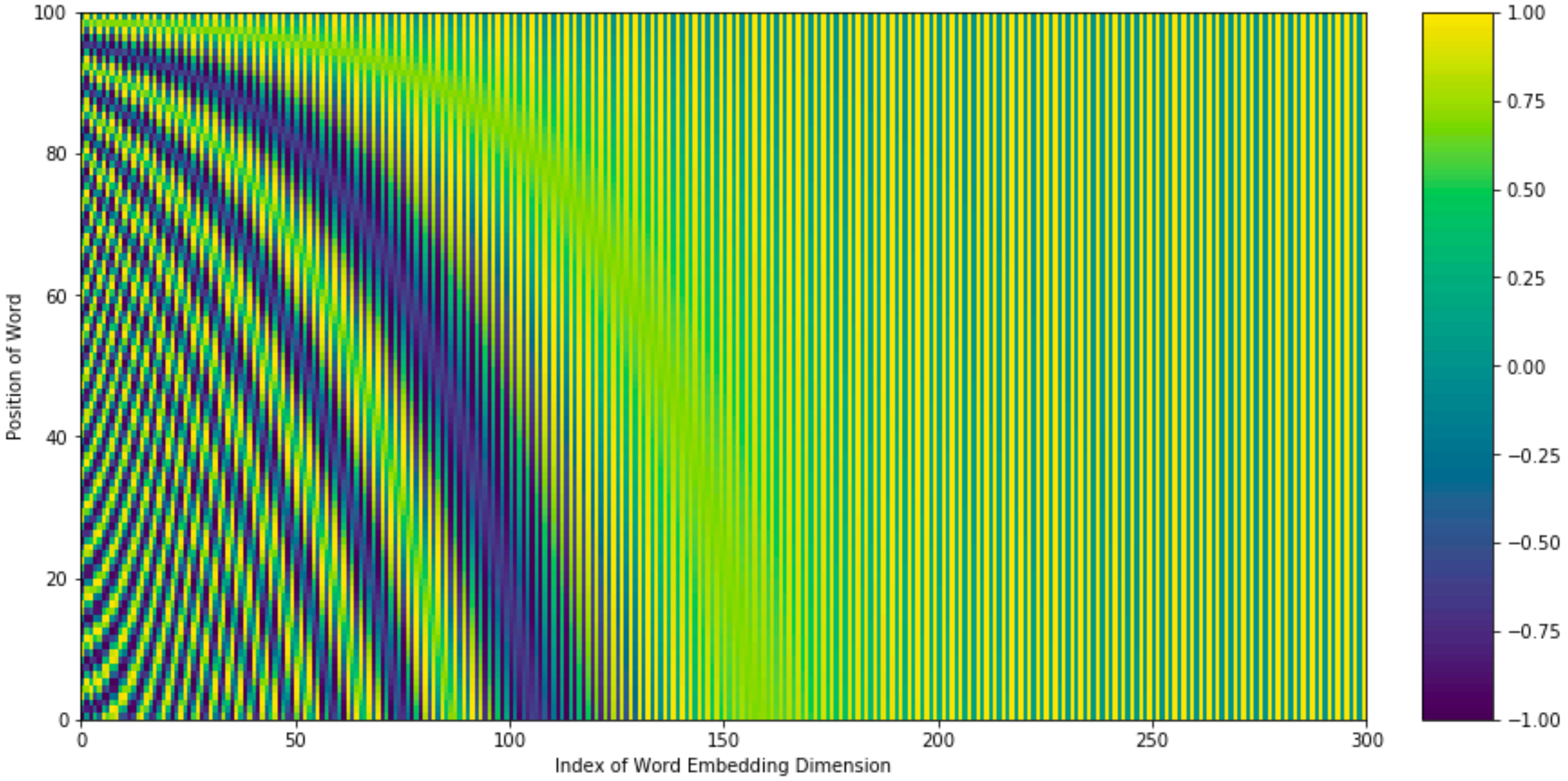
Positional Encoding



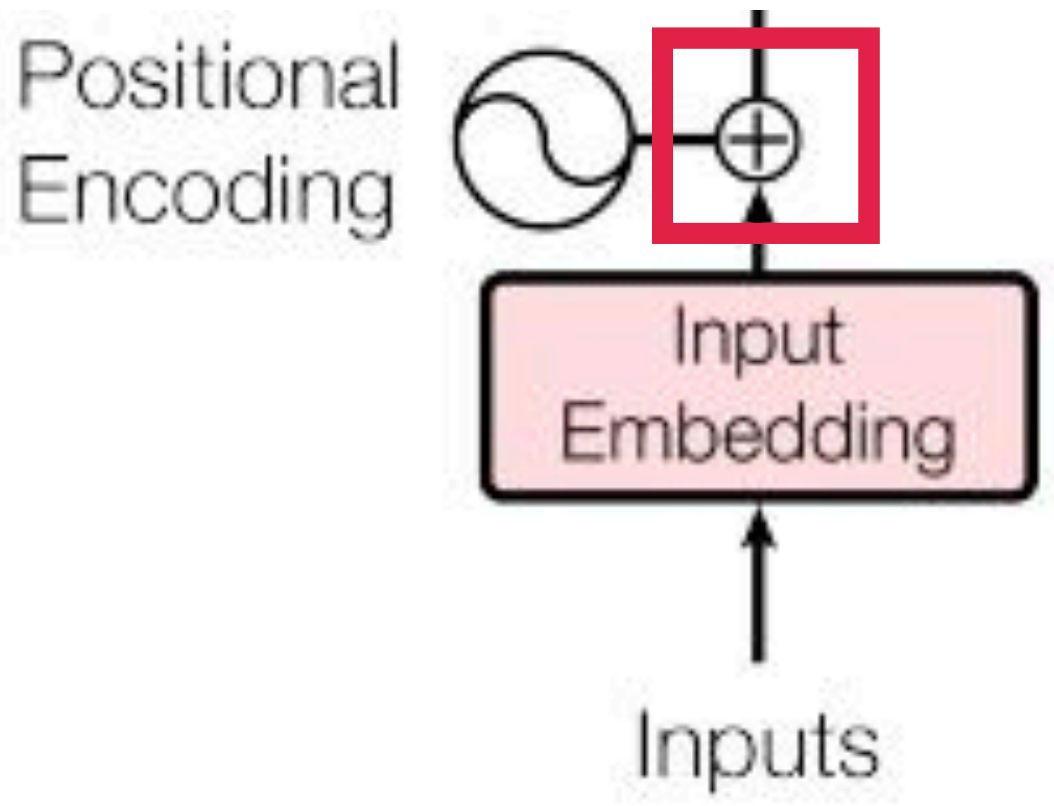
3. Transforemr

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

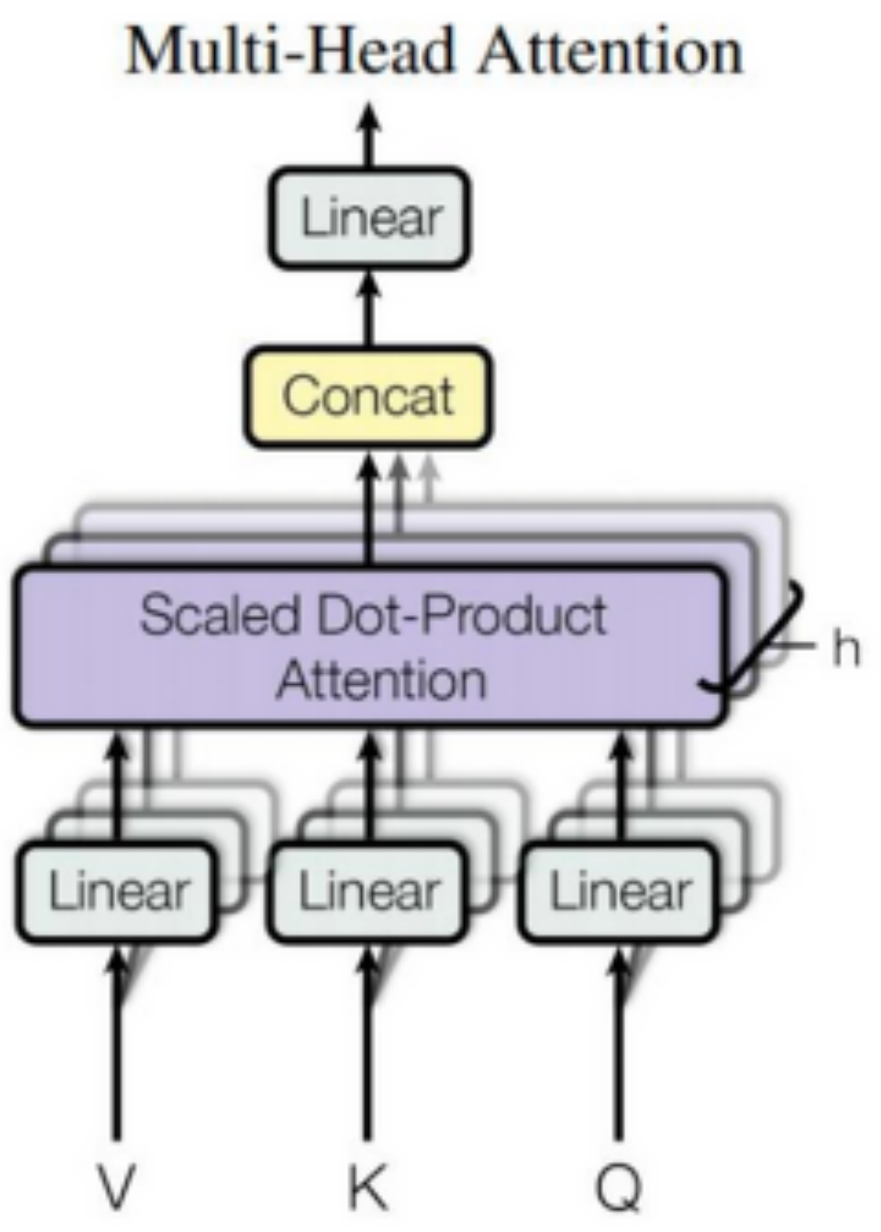
$$PE(pos, 2i+1) = \cos(pos/10000^{2i/d_{model}})$$



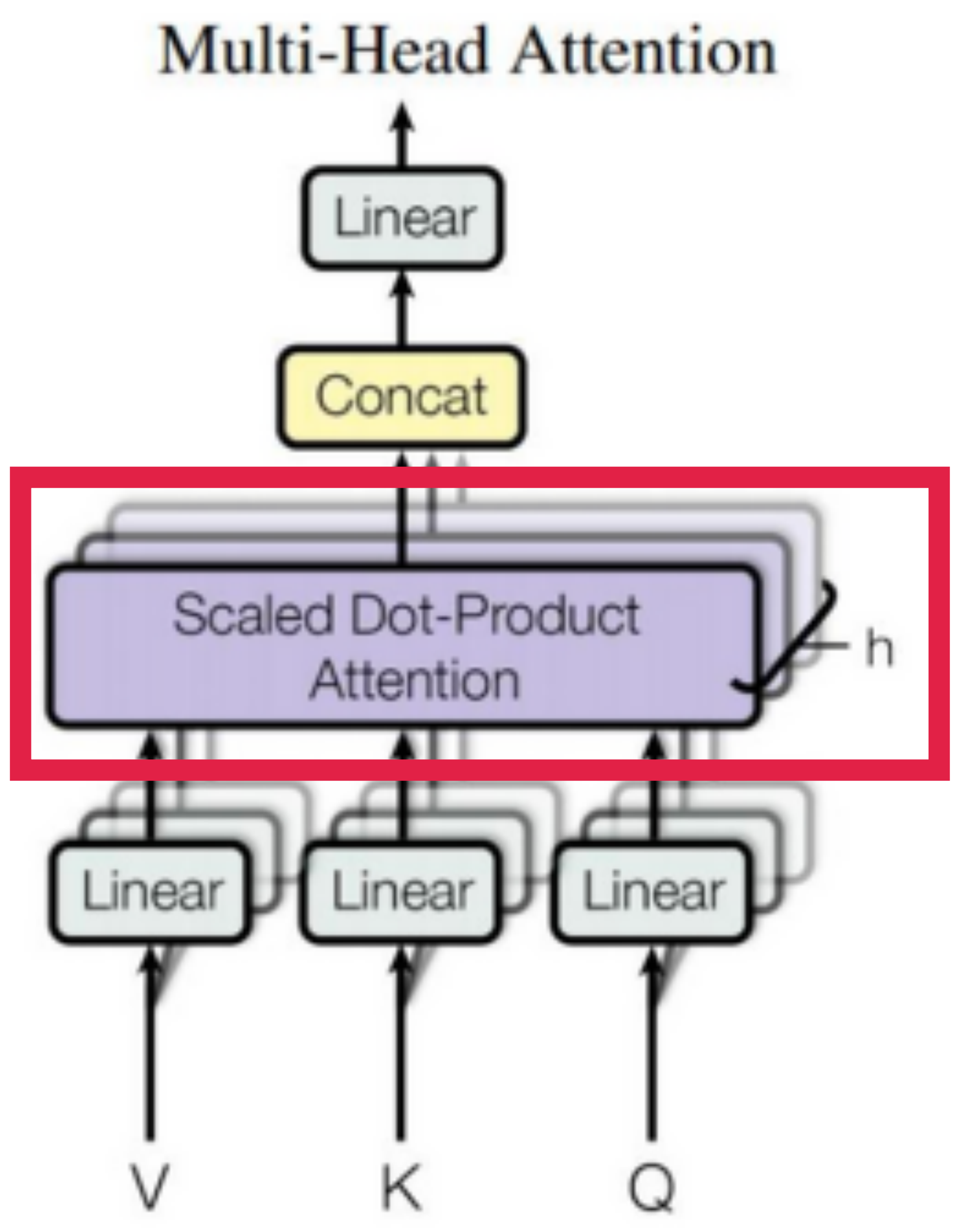
Positional Encoding



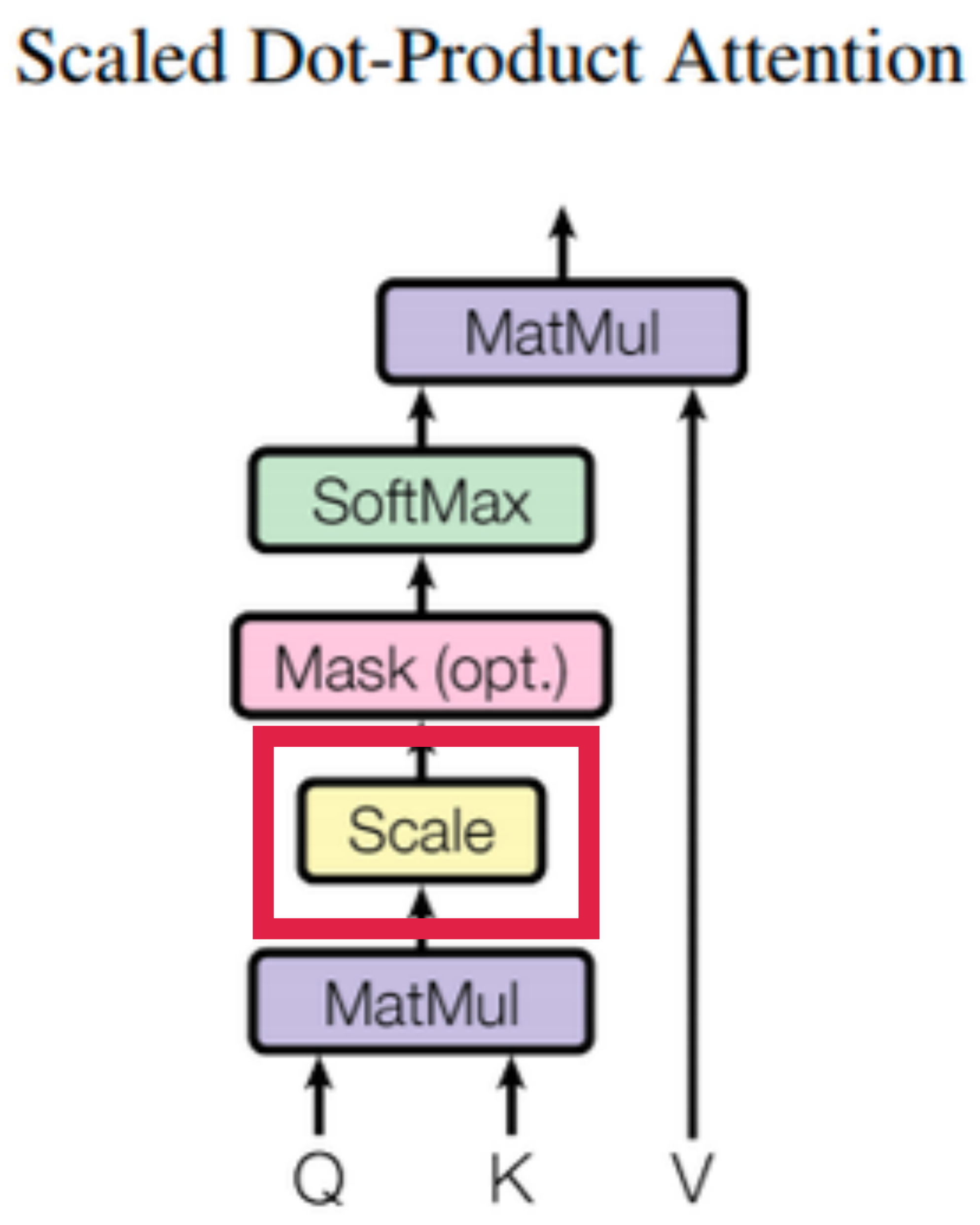
Multi-Head Attention



Multi-Head Attention

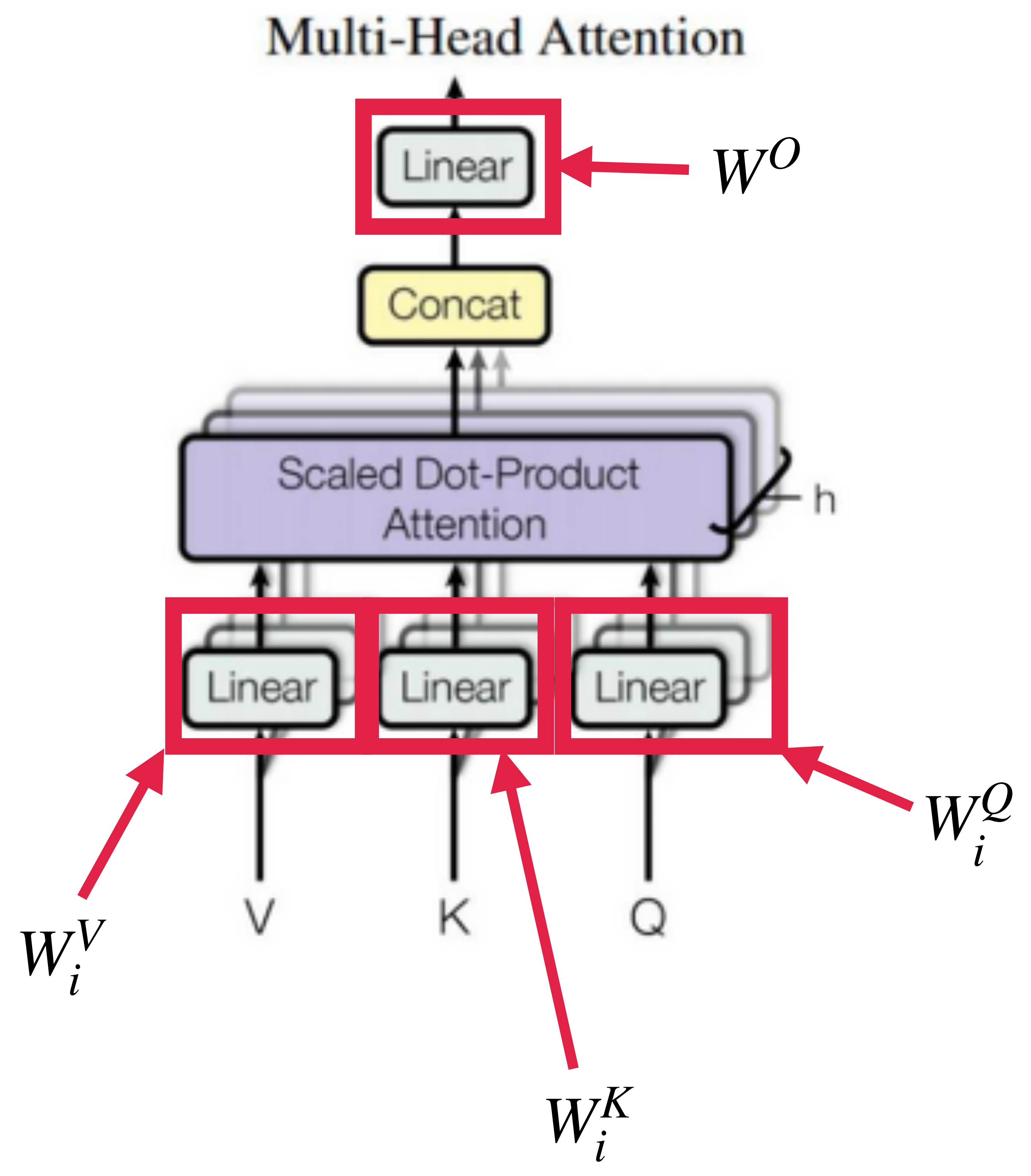


Scaled Dot-Product Attention



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

3. Transforemr



concat 을 의미

$$MultiHead(Q, K, V) = [head_1; head_2; \cdots; head_h]W^O$$

$$where\ \ head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

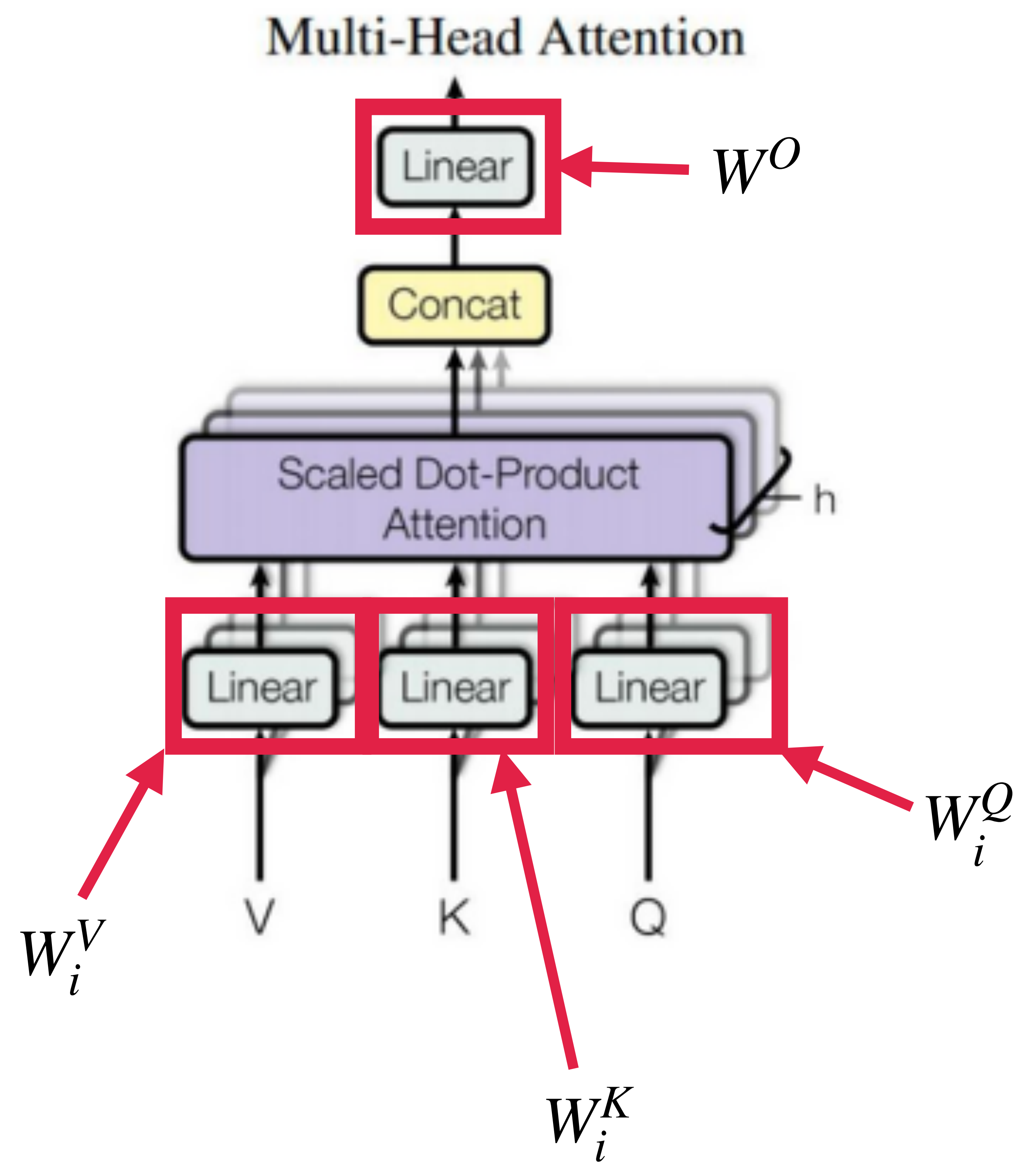
3. Transforemr

$$|Q| = (batch\_size, \underline{m,} hidden\_size)$$

$$|K| = |V| = (batch\_size, \underline{n,} hidden\_size)$$

$$where\ n\ is\ length\ of\ source\ sentence,\ and\ m\ is\ length\ of\ target\ sentence.$$

3. Transforemr



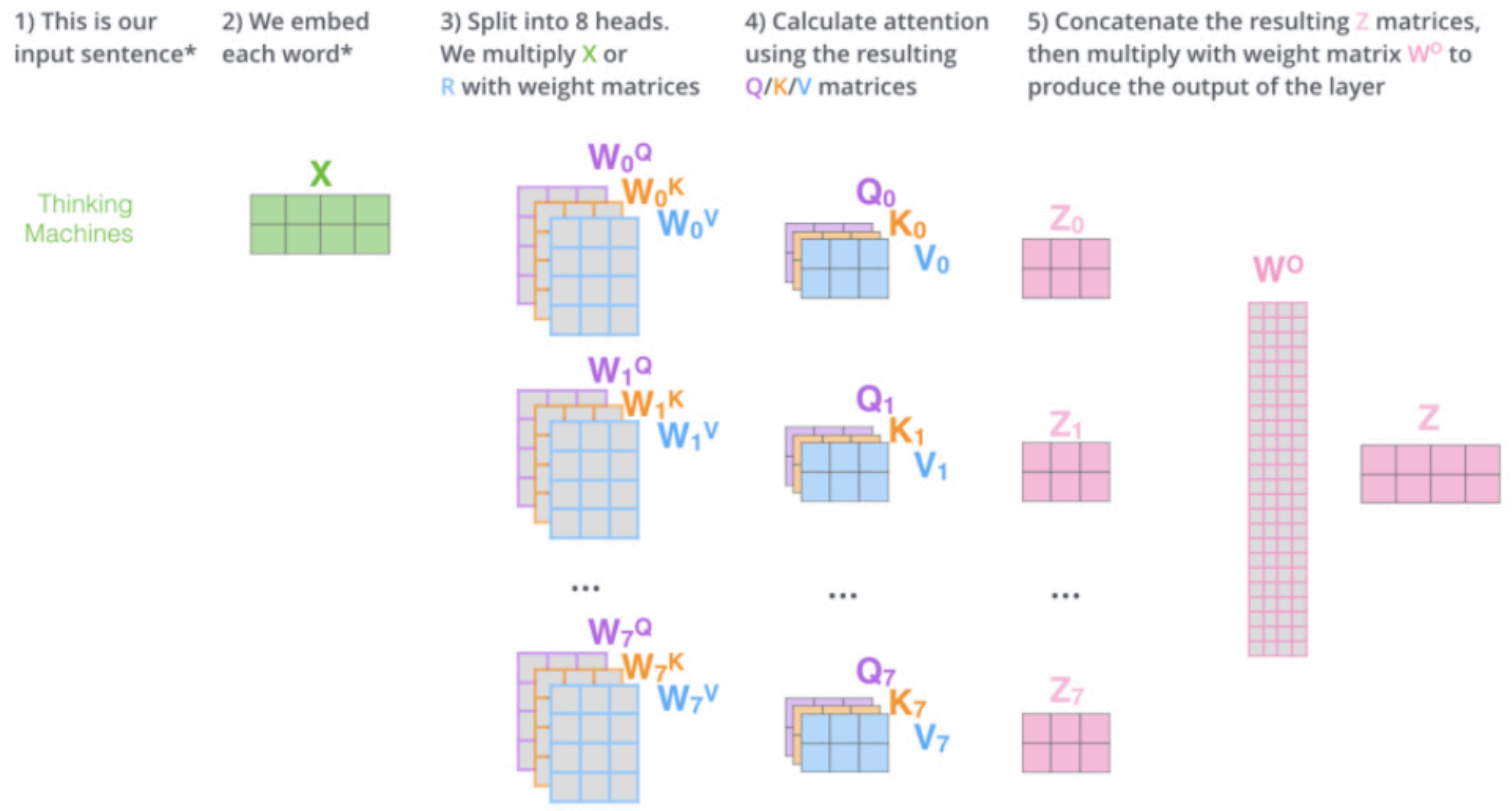
$$|W_i^Q| = |W_i^K| = |W_i^v| = (hidden\_size, head\_size)$$

$$|W^O| = (head\_size \times h, hidden\_size)$$

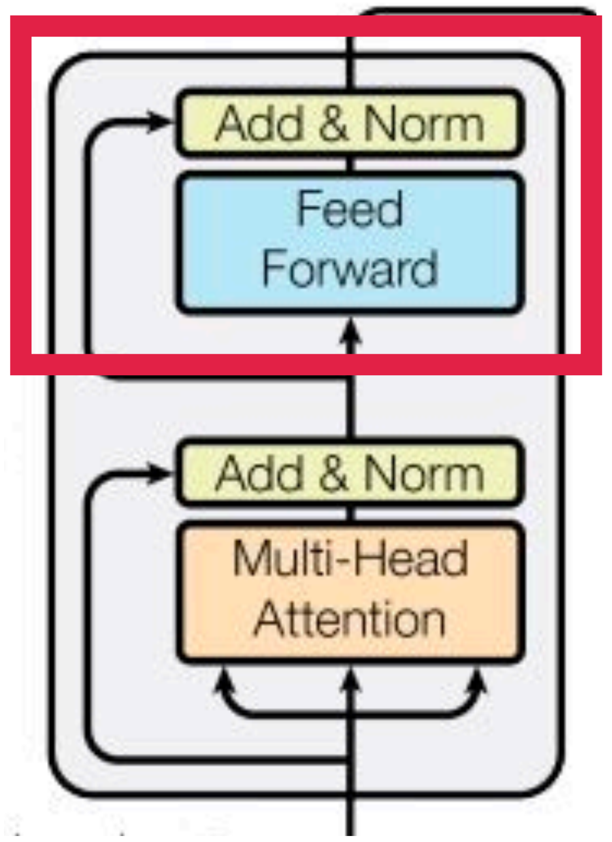
$$where\ hidden\_size = 512,\ h = 8\ and\ hidden\_size = head \times h$$

3. Transforemr

Multi-Head Attention

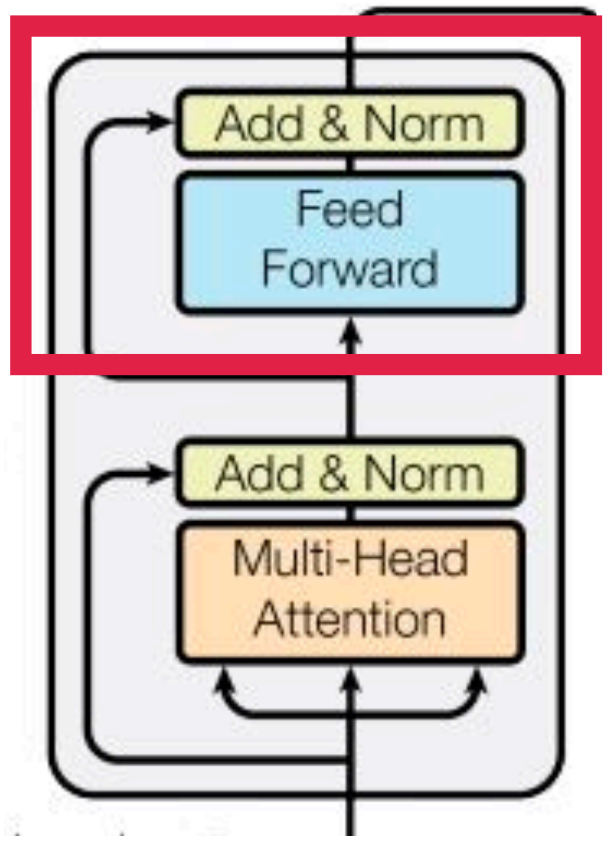


Feed Forward Layer



3. Transforemr

Feed Forward Layer



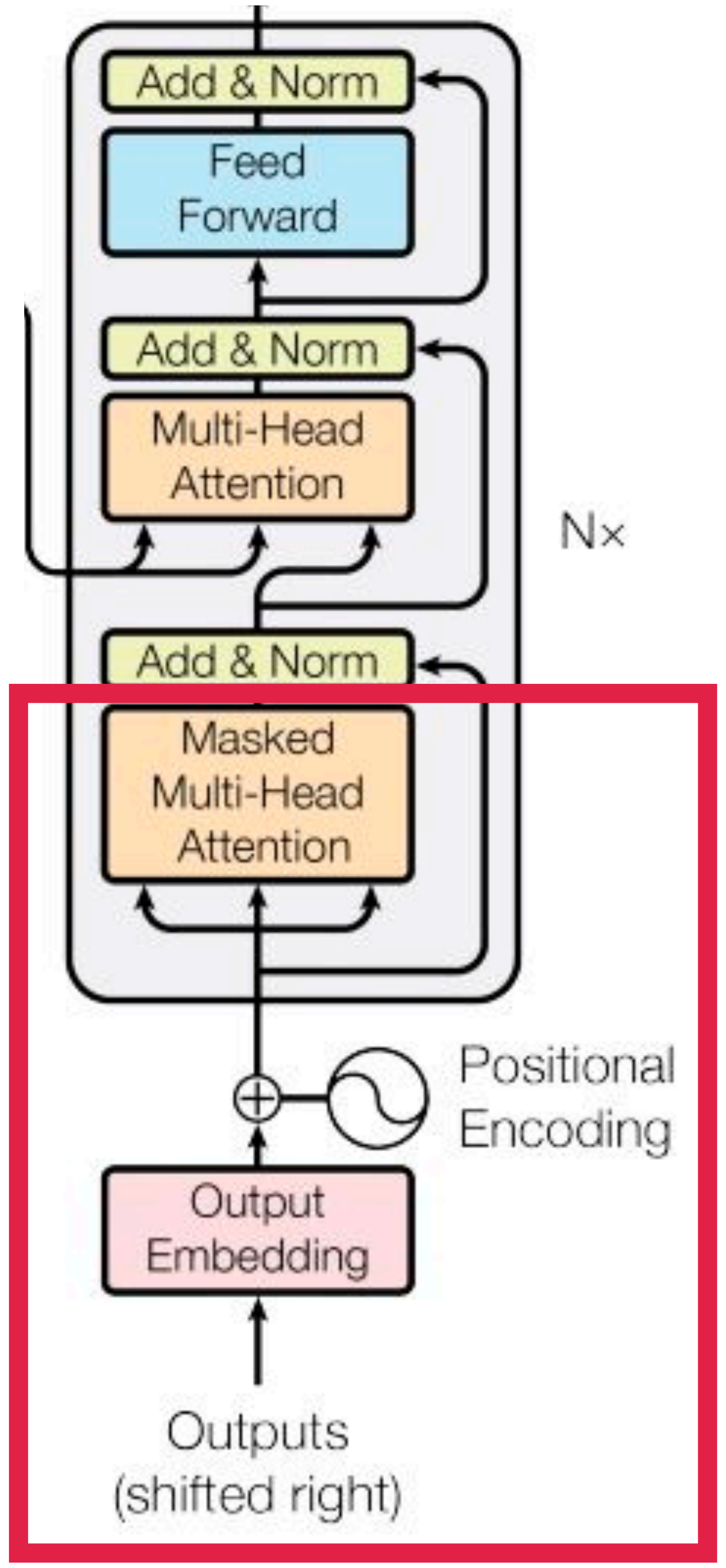
$$FFN(x) = ReLU(xW_1 + b_1)W_2 + b_2$$

$$where\ |x| = (batch\_size, n, hidden\_size)$$

$$and\ w_1 \in \mathbb{R}^{hidden\_size \times d_{ff}},\ W_2 \in \mathbb{R}^{d_{ff} \times hidden\_size}\ and\ d_{ff} = 2048$$

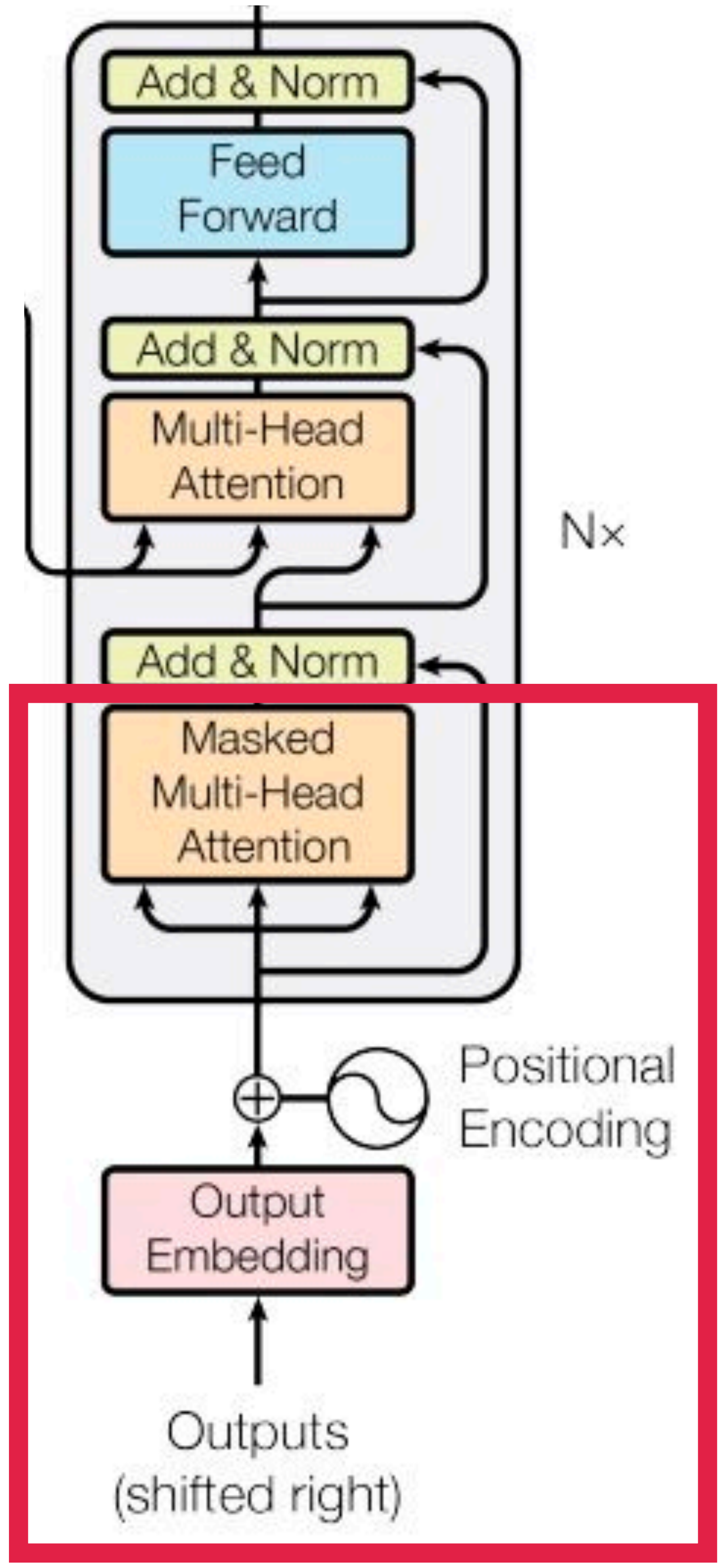
3. Transforemr

Masked Multi-Head Attention



3. Transforemr

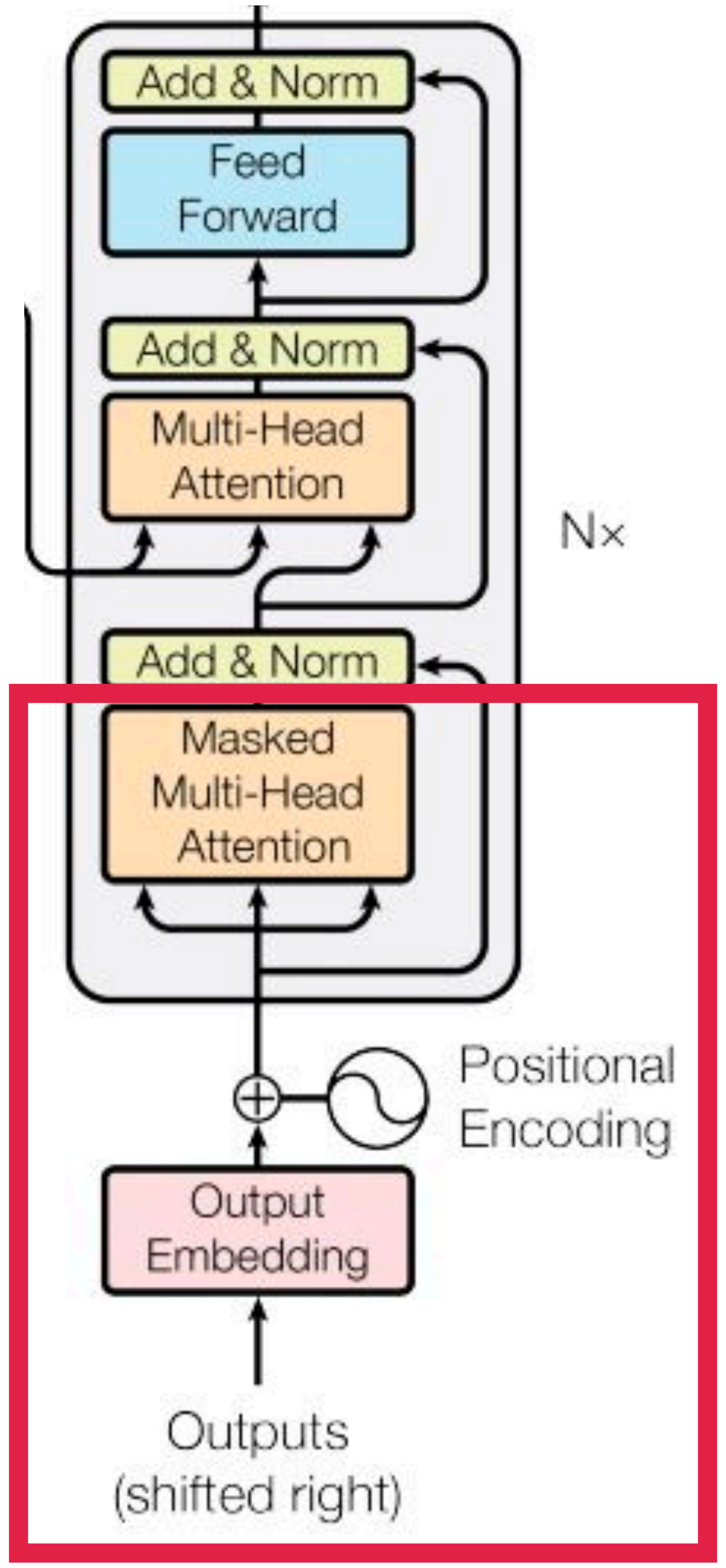
Masked Multi-Head Attention

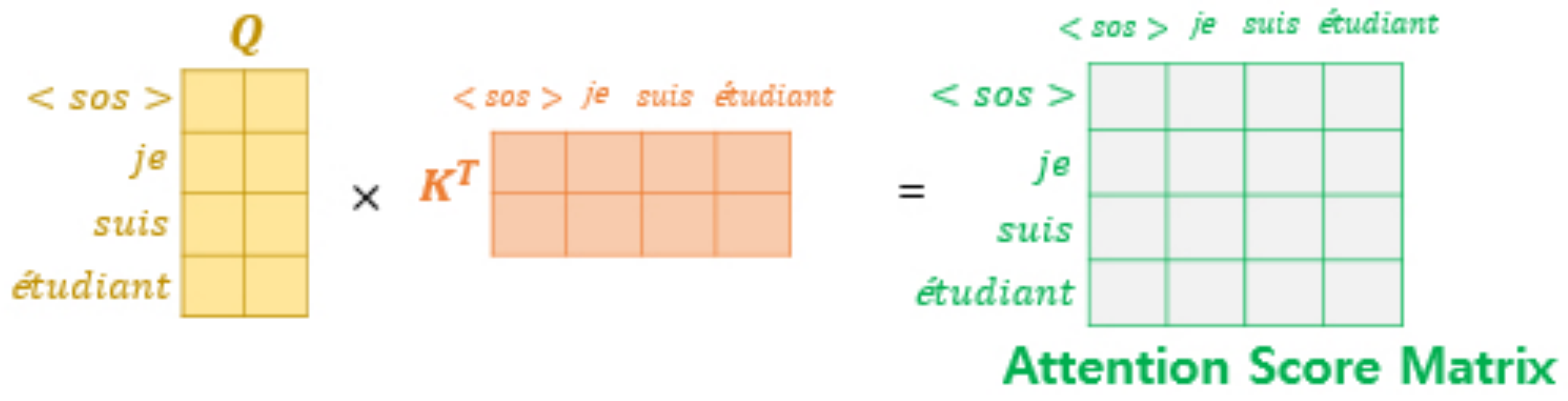


문제 발생!!

3. Transforemr

Masked Multi-Head Attention

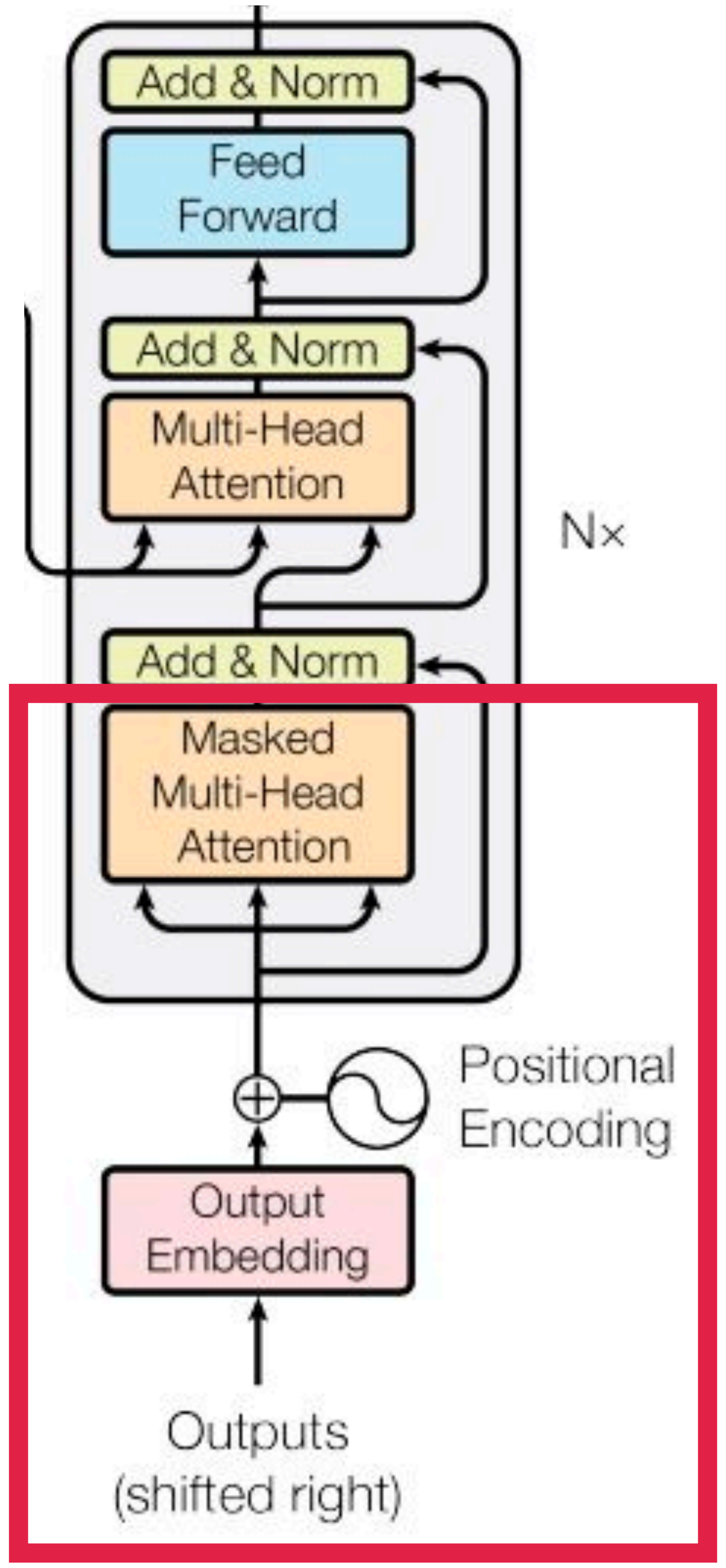




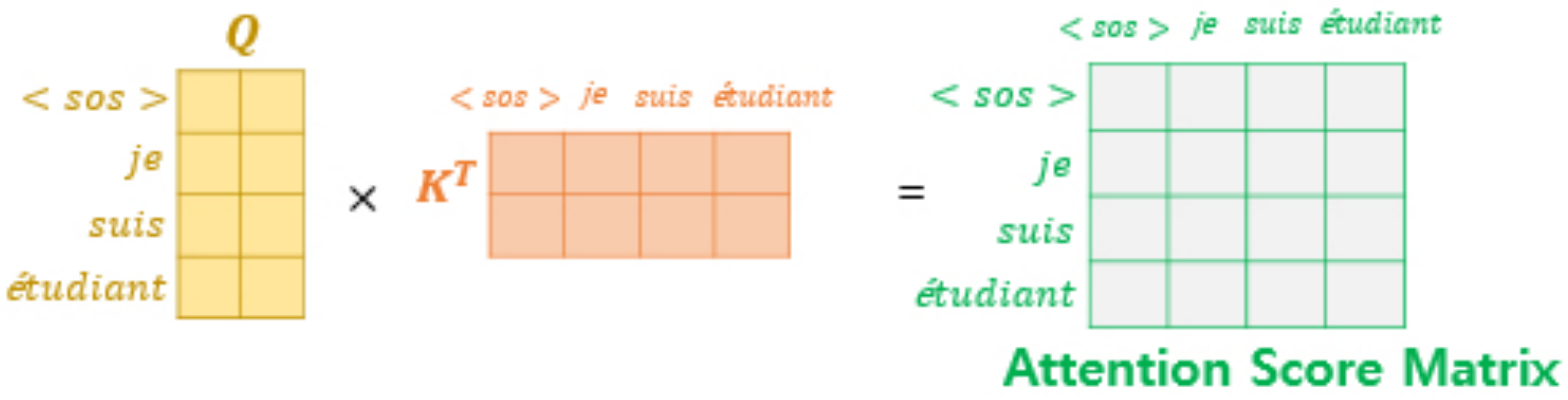
문제 발생!!

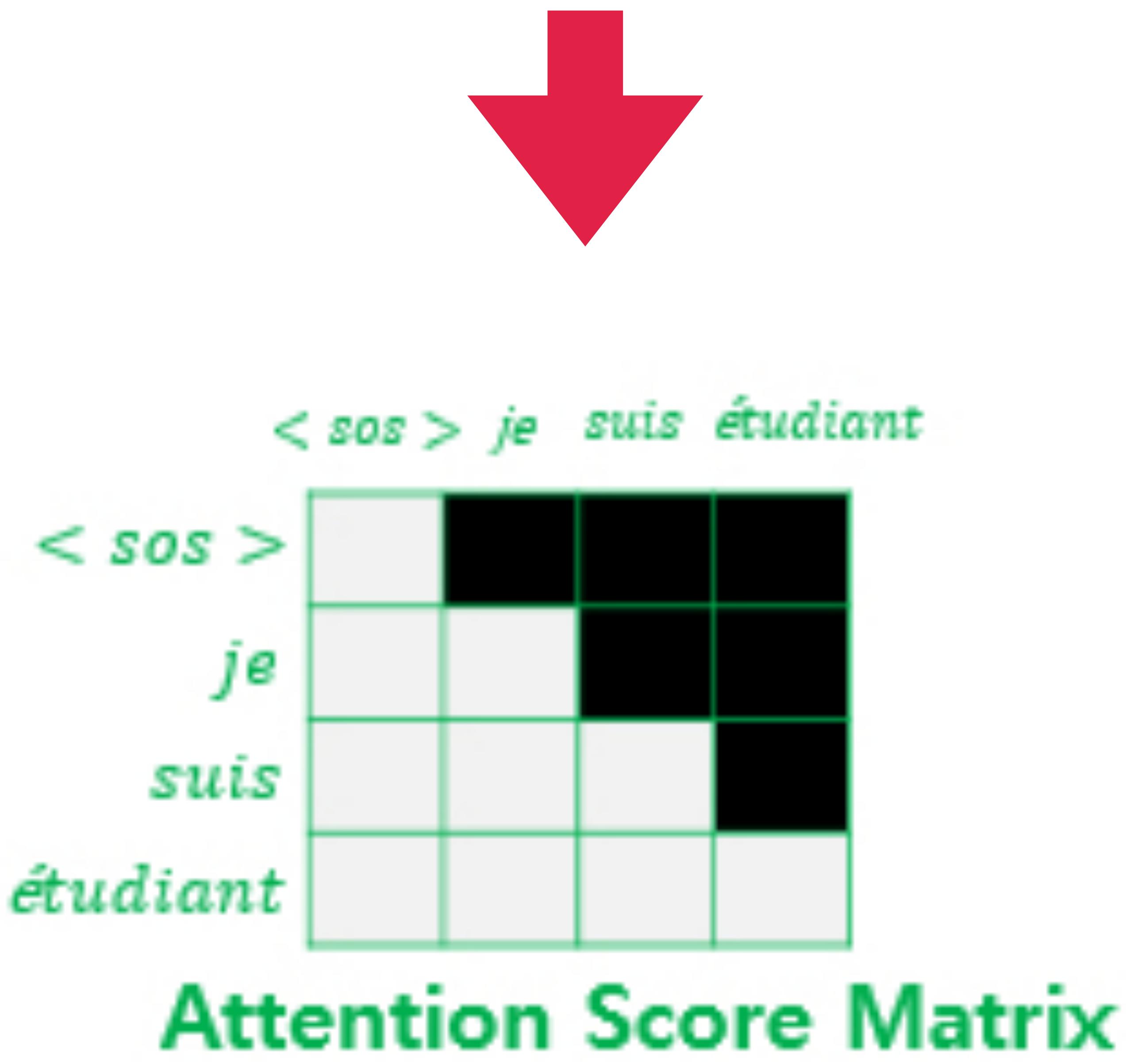
3. Transforemr

Masked Multi-Head Attention

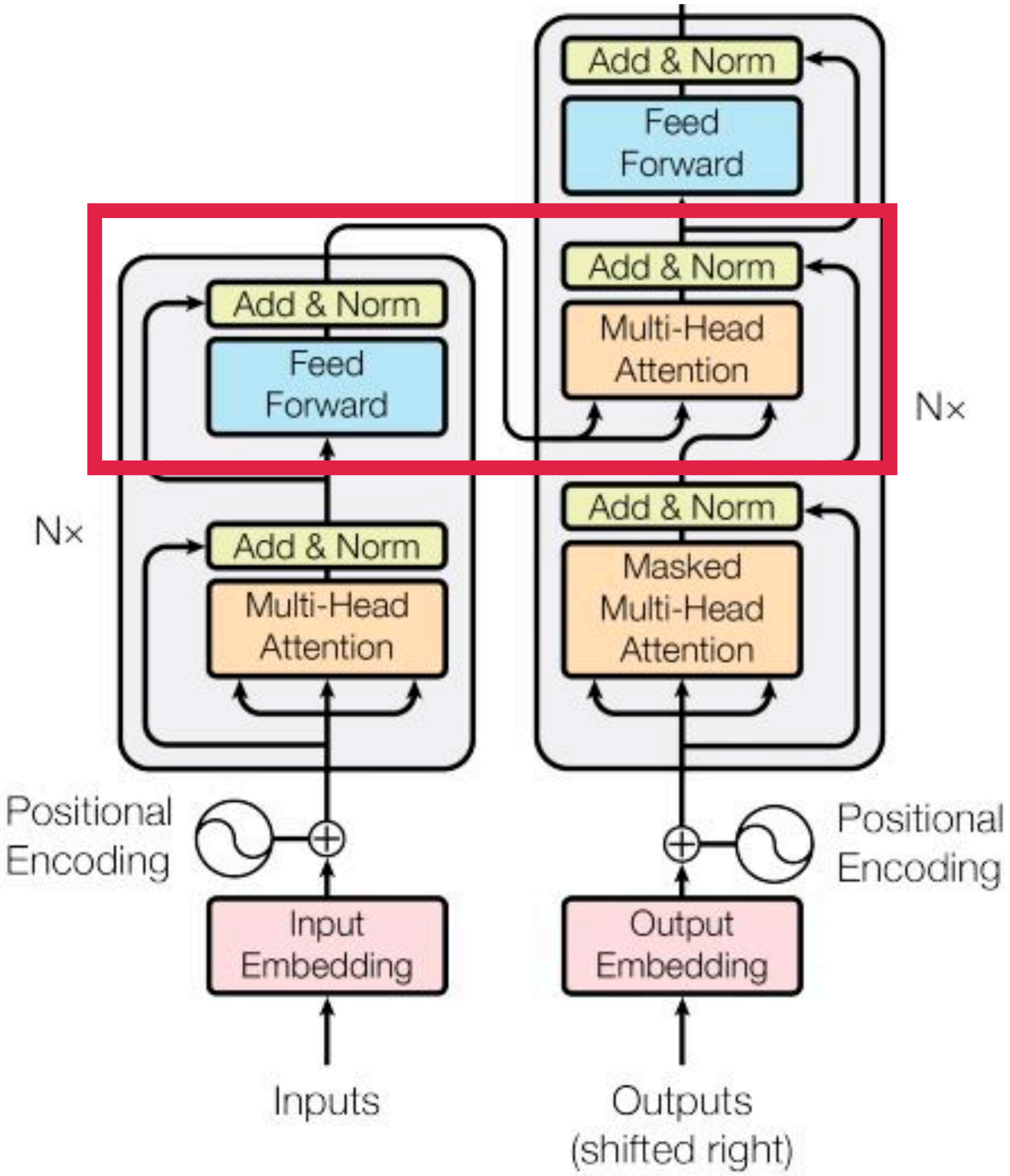


문제 발생!!



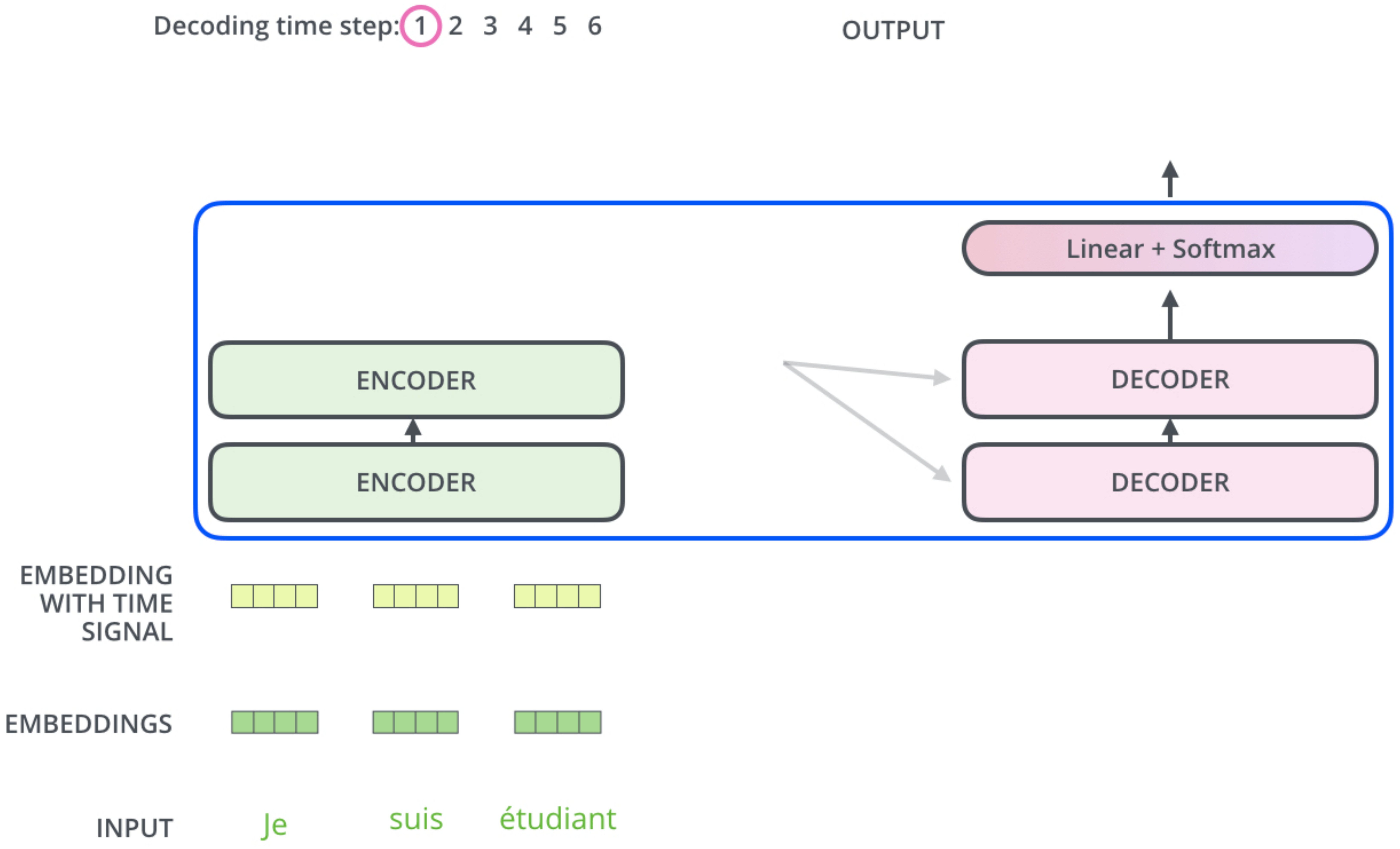


Encoder-Decoder Attention



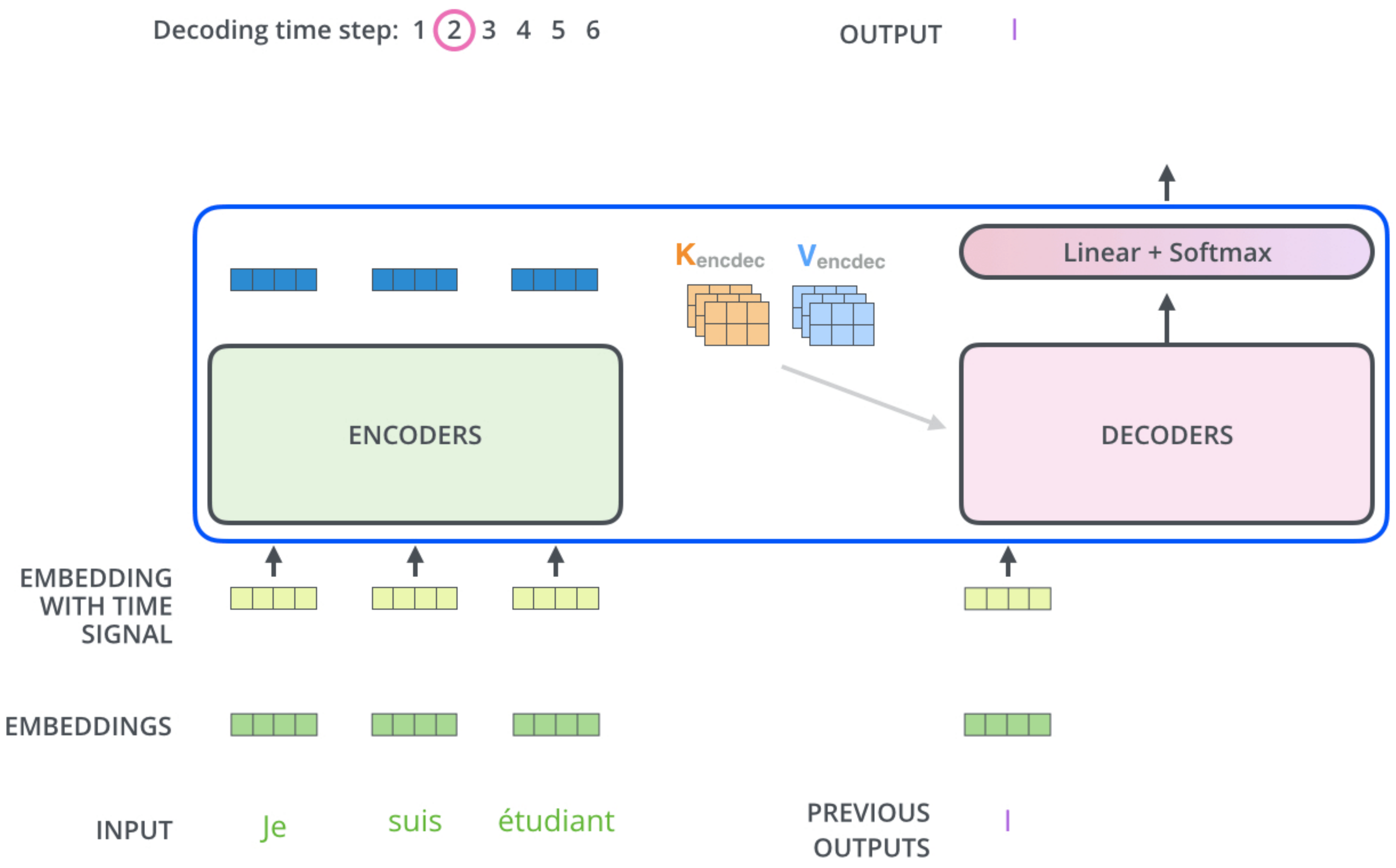
3. Transforemr

Encoder-Decoder Attention

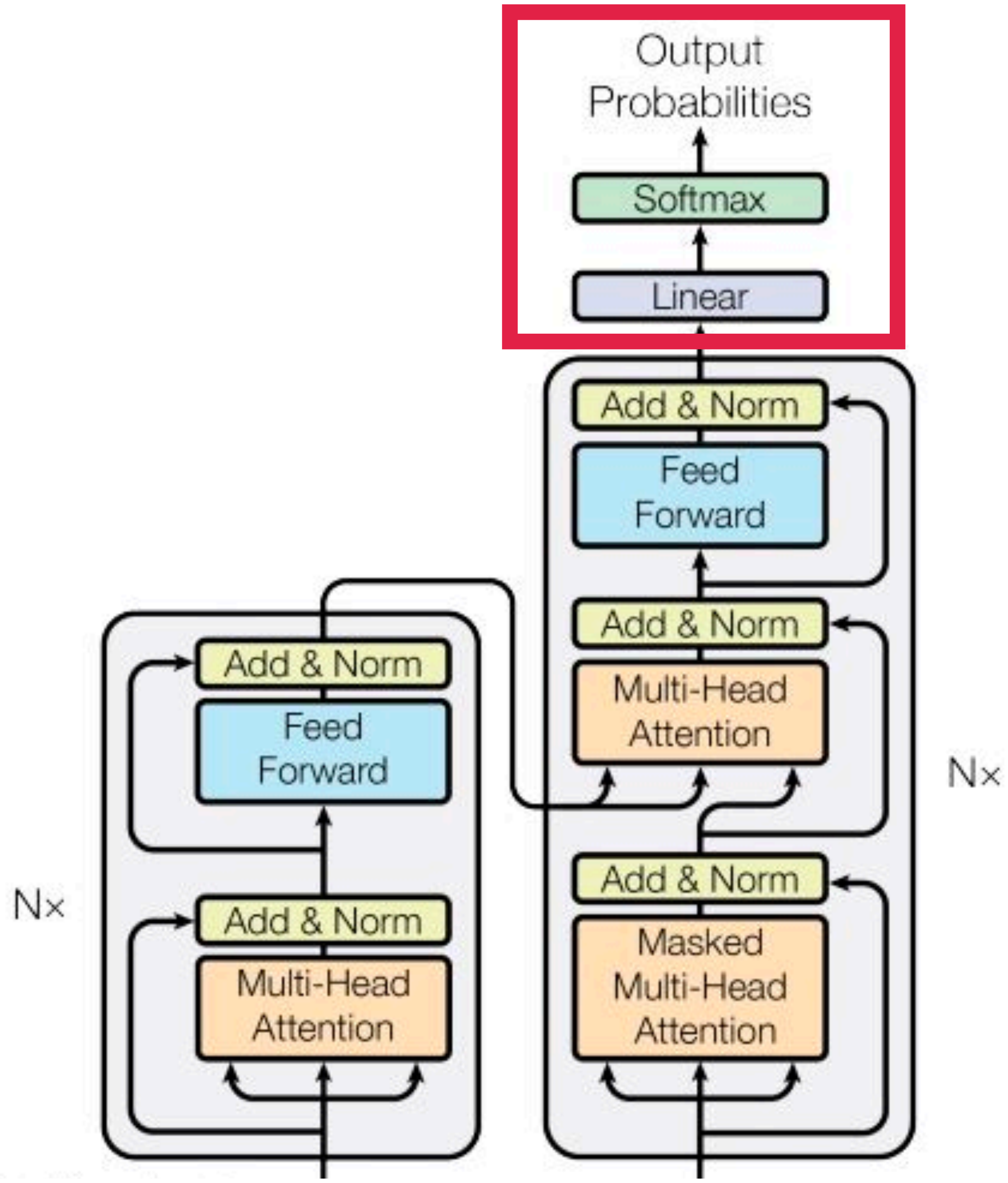


3. Transforemr

Encoder-Decoder Attention



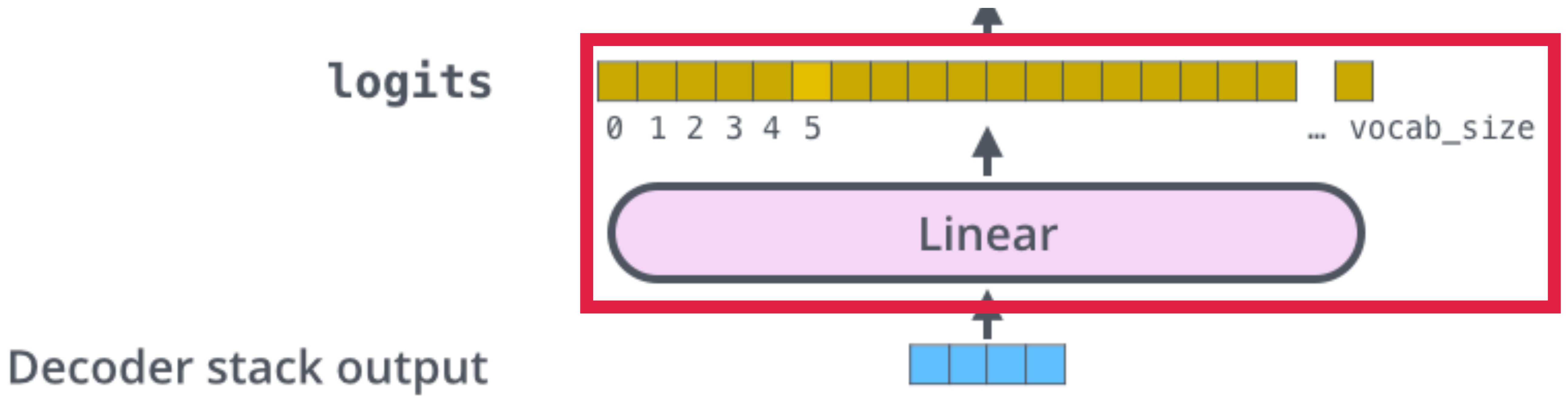
Linear Layer and Softmax Layer



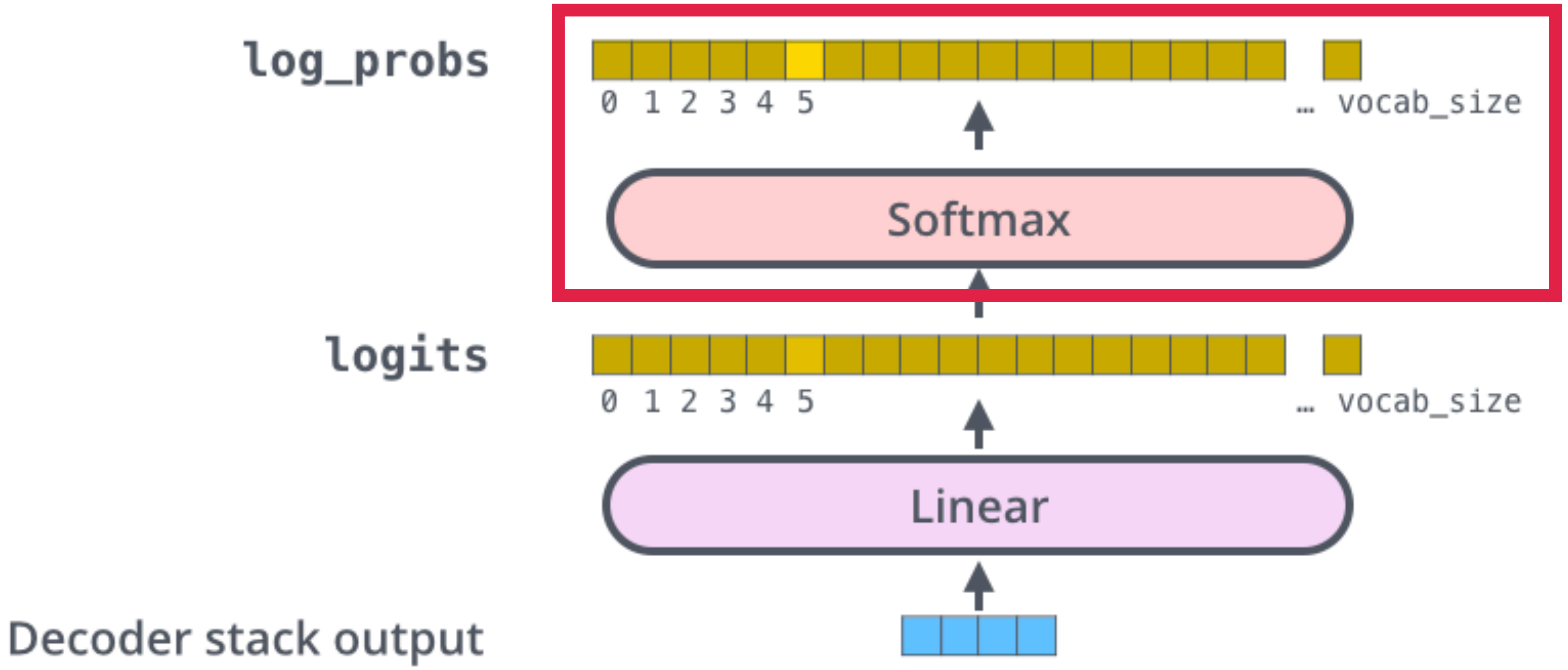
3. Transforemr

Linear Layer and Softmax Layer

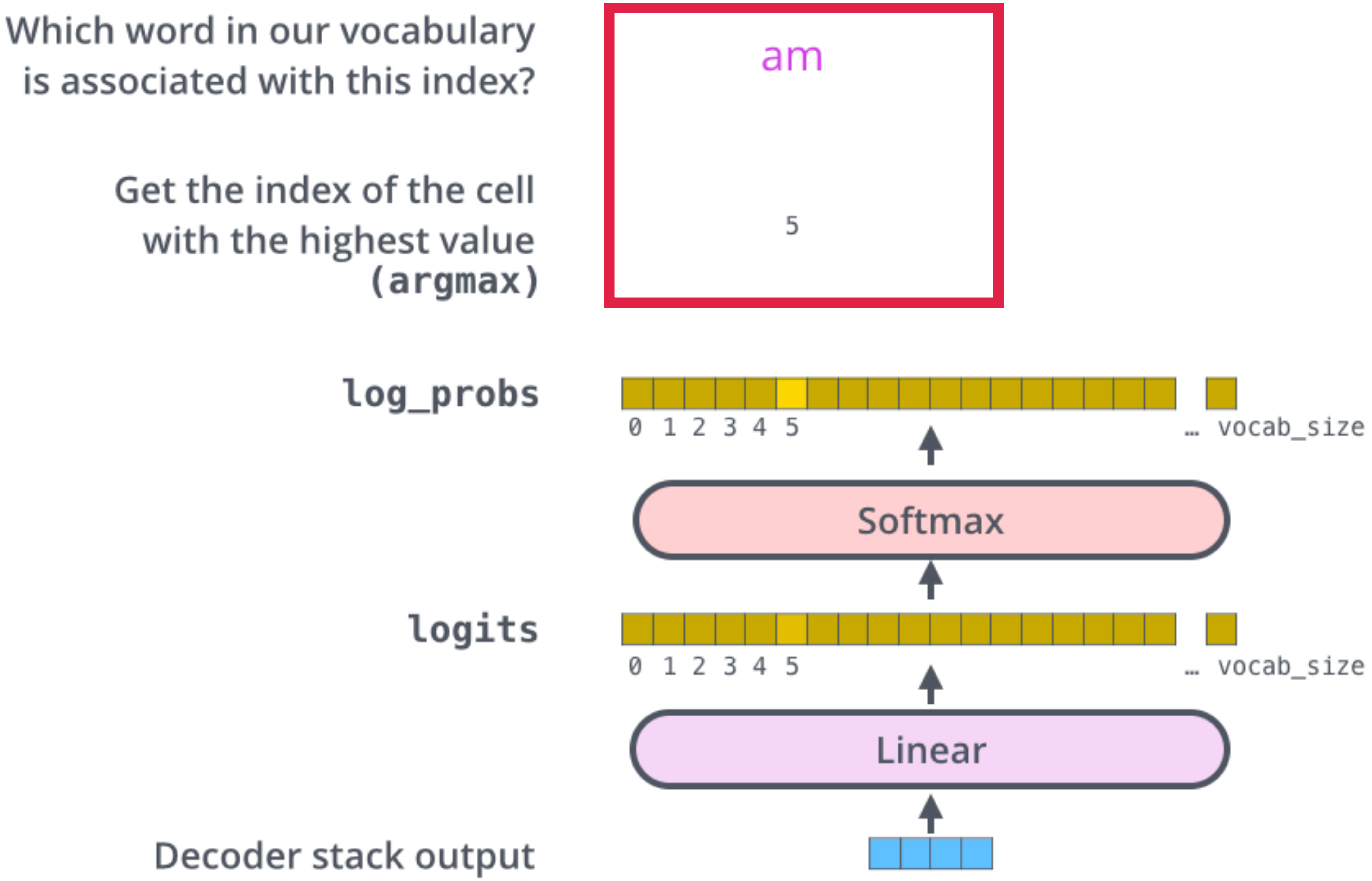
Linear Layer and Softmax Layer



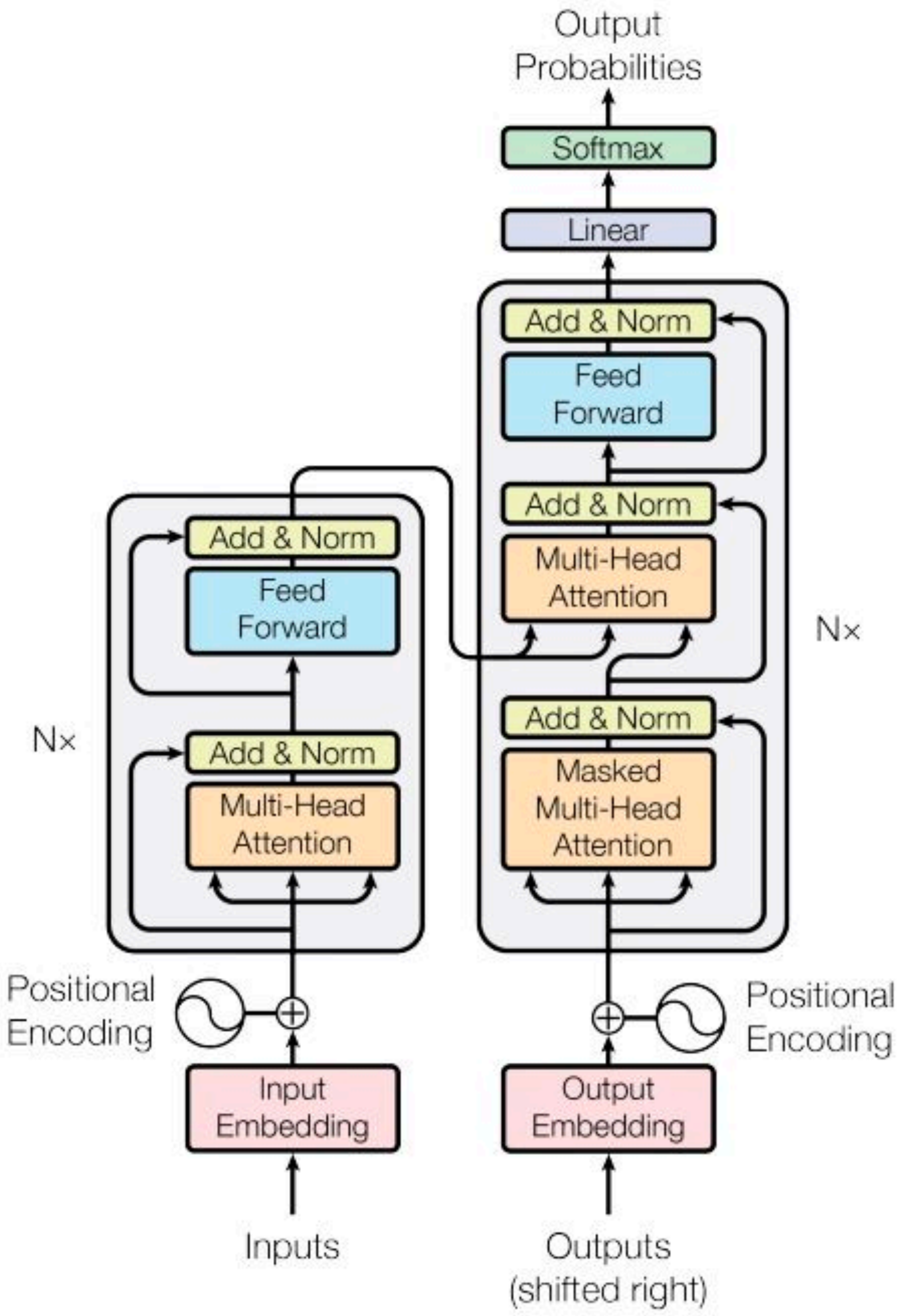
Linear Layer and Softmax Layer



Linear Layer and Softmax Layer



3. Transforemr



Q&A