

Attention is All You Need

HAI 여름방학 논문리딩 스터디

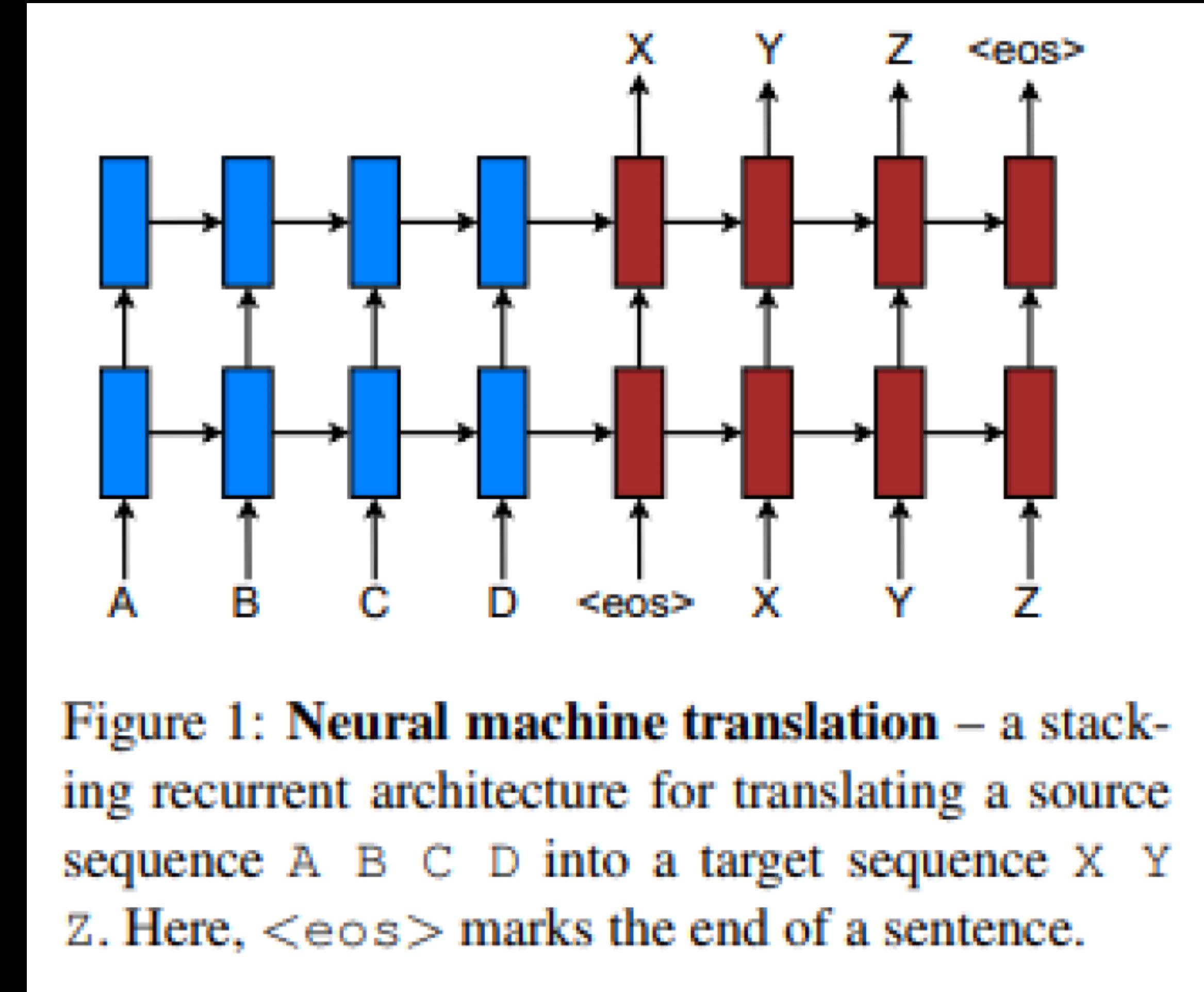
김성환 | 2021.07.25

Before Transformer

Problem Definition

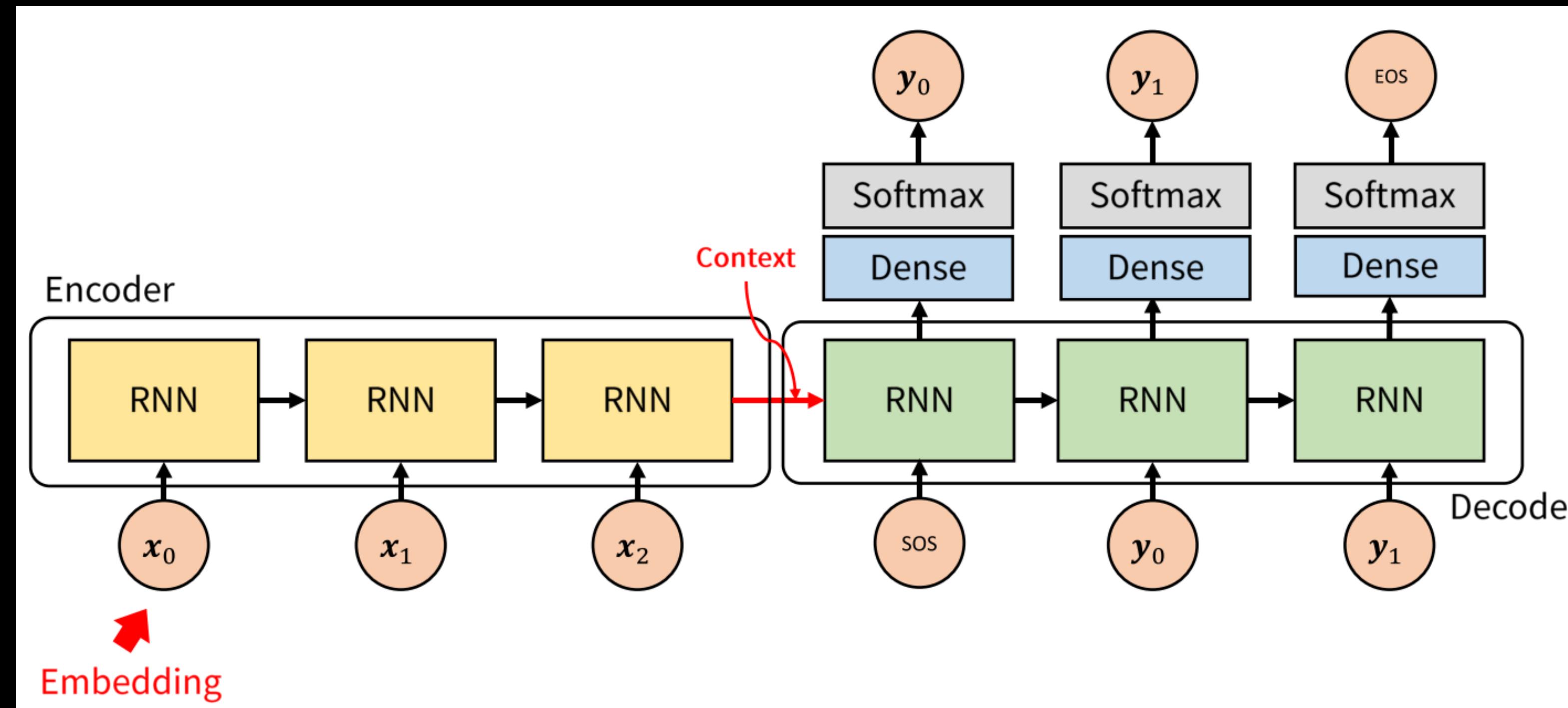
Before Transformer

- Neural Translation
 - A언어로 작성된 문장을 B언어로 작성된 문장으로 변환
 - 기존에는 번역을 언어학적으로 접근 했다면, ML을 이용해서 언어학적인 사전지식 없이도 번역을 가능하게 함.
 - 주로 RNN/LSTM/GRU와 같은 Recurrent Layer들이 사용되었음.



Sequence to Sequence

Before Transformer



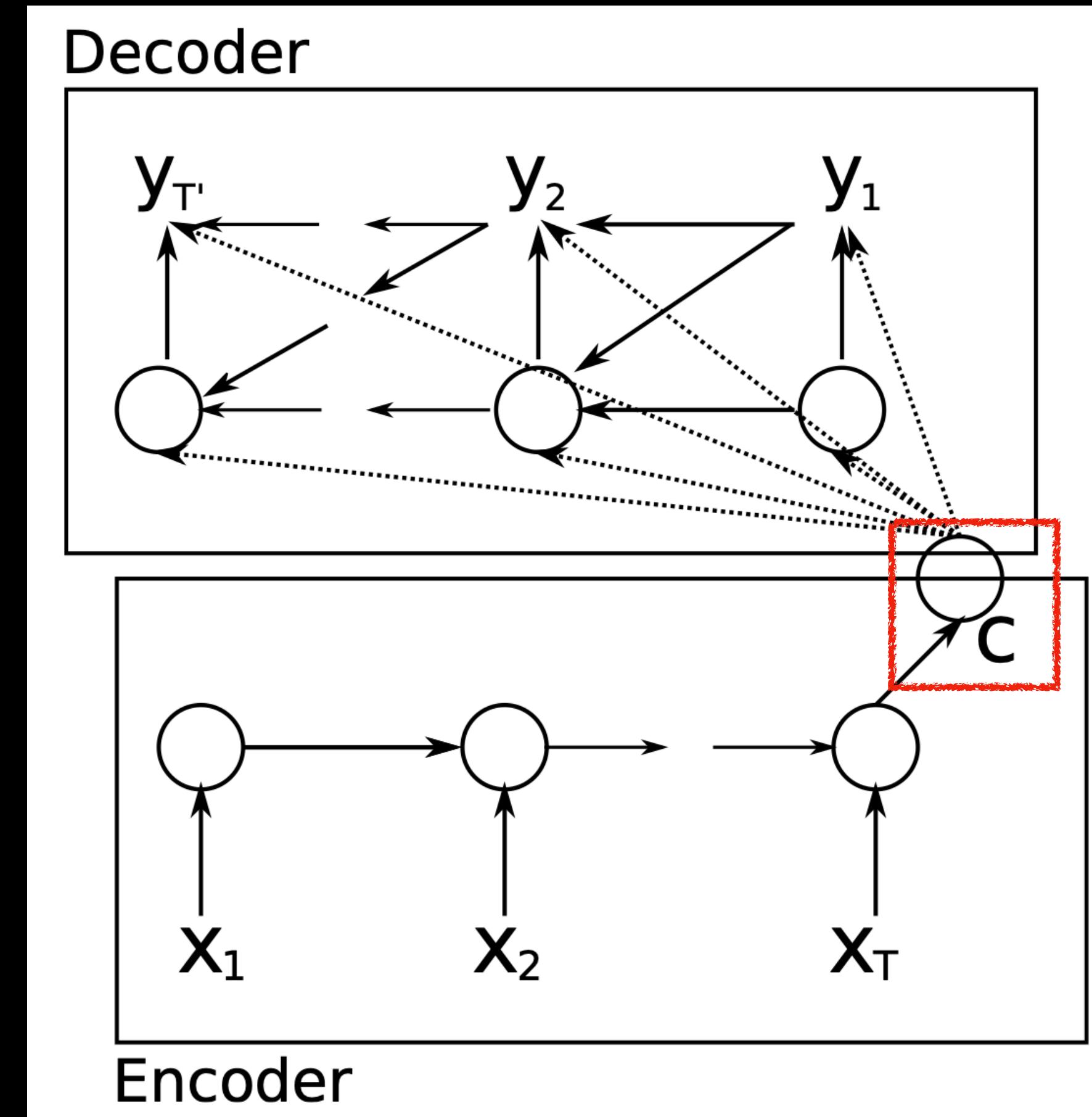
Encoder and Decoder Architecture (Seq2Seq)

Encoder를 통해 한정된 크기의 Context Vector를 학습,
Decoder는 이 Context를 통해 번역을 수행.

Sequence to Sequence

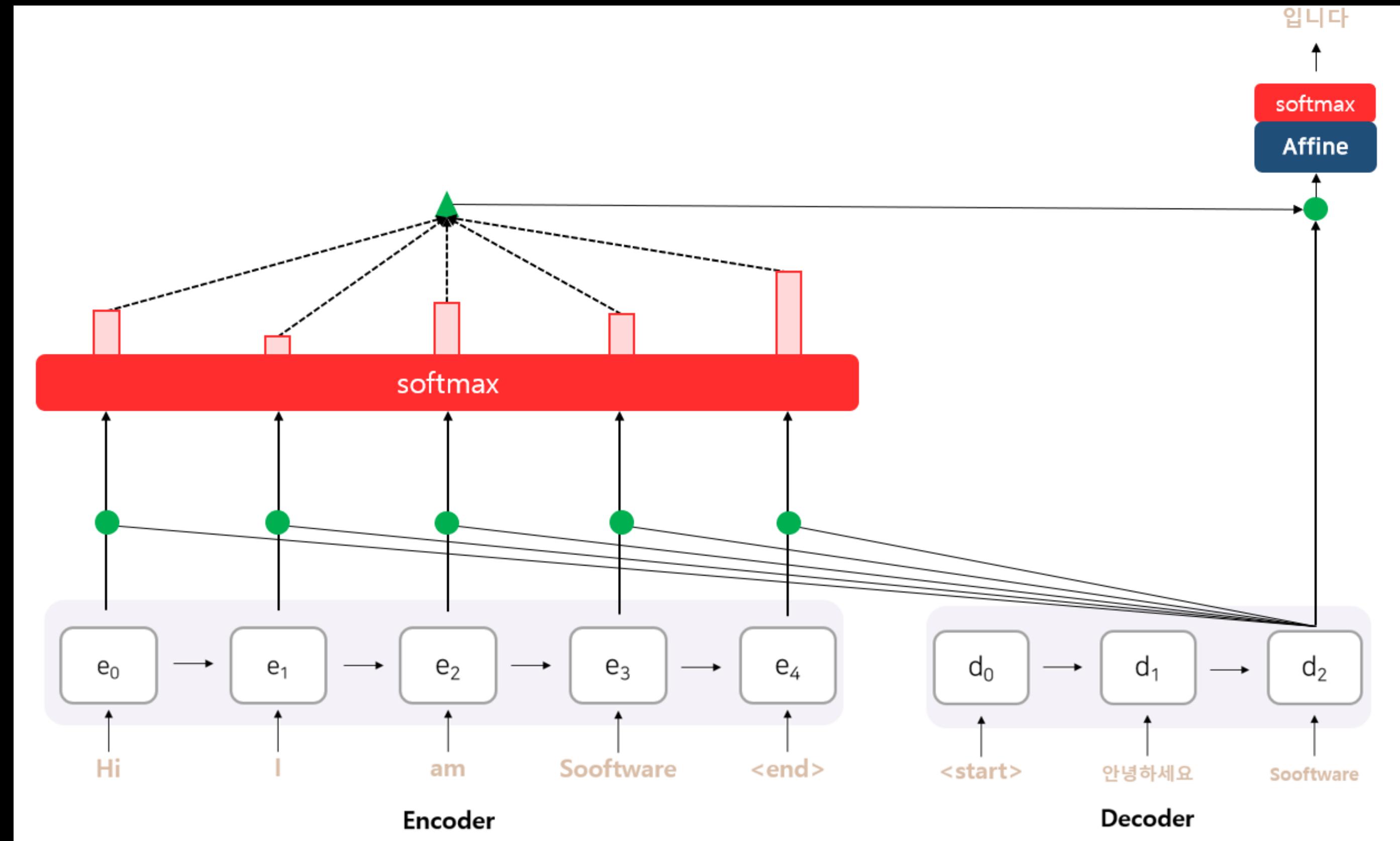
Before Transformer

- Sequence to Sequence 모델은 주어진 입력을 한정된 크기의 Context Vector로 압축.
- 하지만, 이 과정에서 정보가 손실되는 병목현상(Bottleneck)이 발생
- Source 문장의 모든 정보를 한정된 크기의 Context Vector가 표현하기에는 어려움이 있음.
- 또한, 이전 정보를 참고해야 하는 Auto Regressive 모델에서는 연산의 병렬화가 힘들어 GPU가속을 받기 어려움.



Attention Machanism

Before Transformer

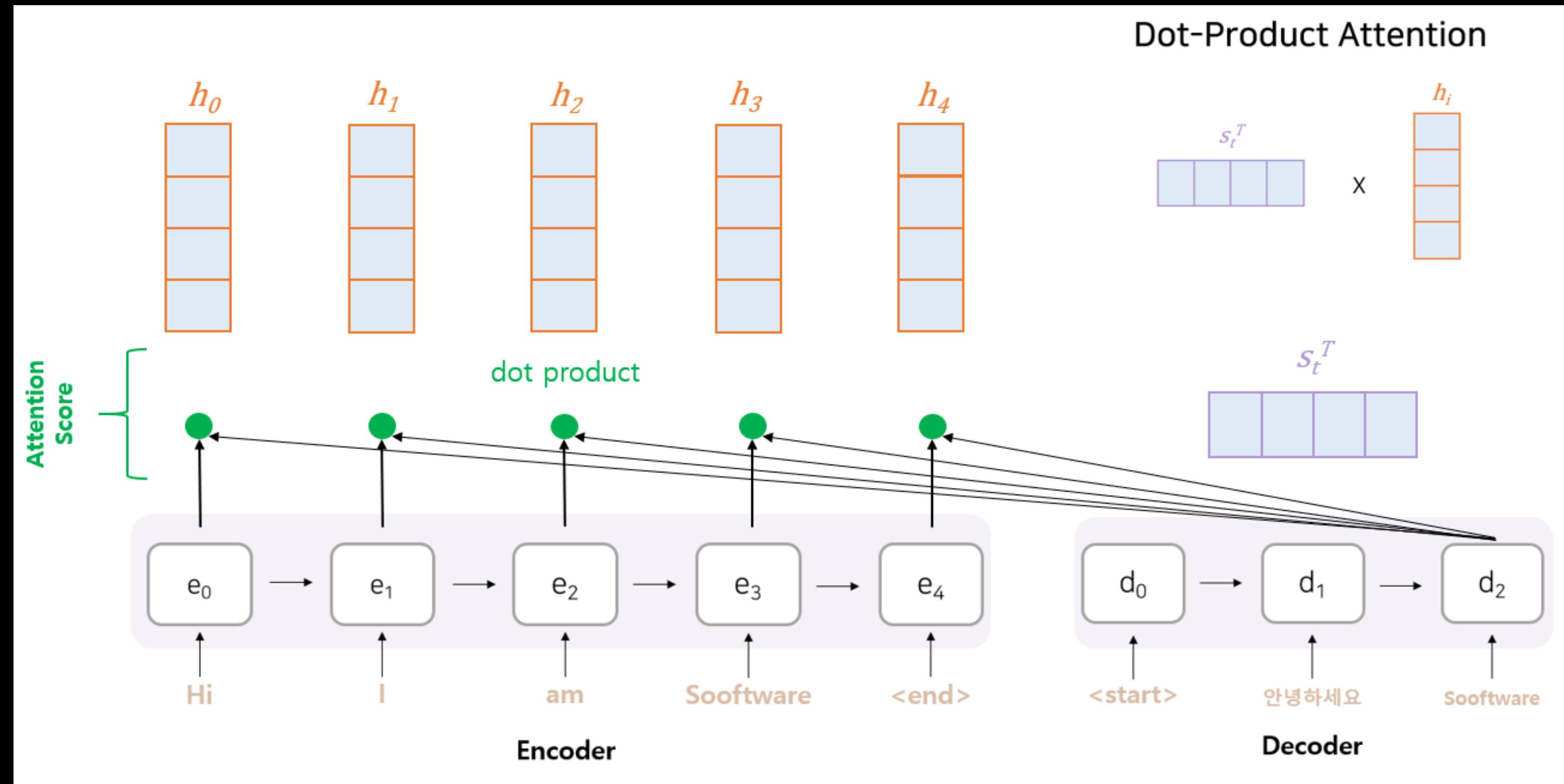


Attention Machanism

Encoder의 모든 정보를 참고하여 Context를 Weighted Sum으로 만들어냄.

Attention Mechanism

Before Transformer

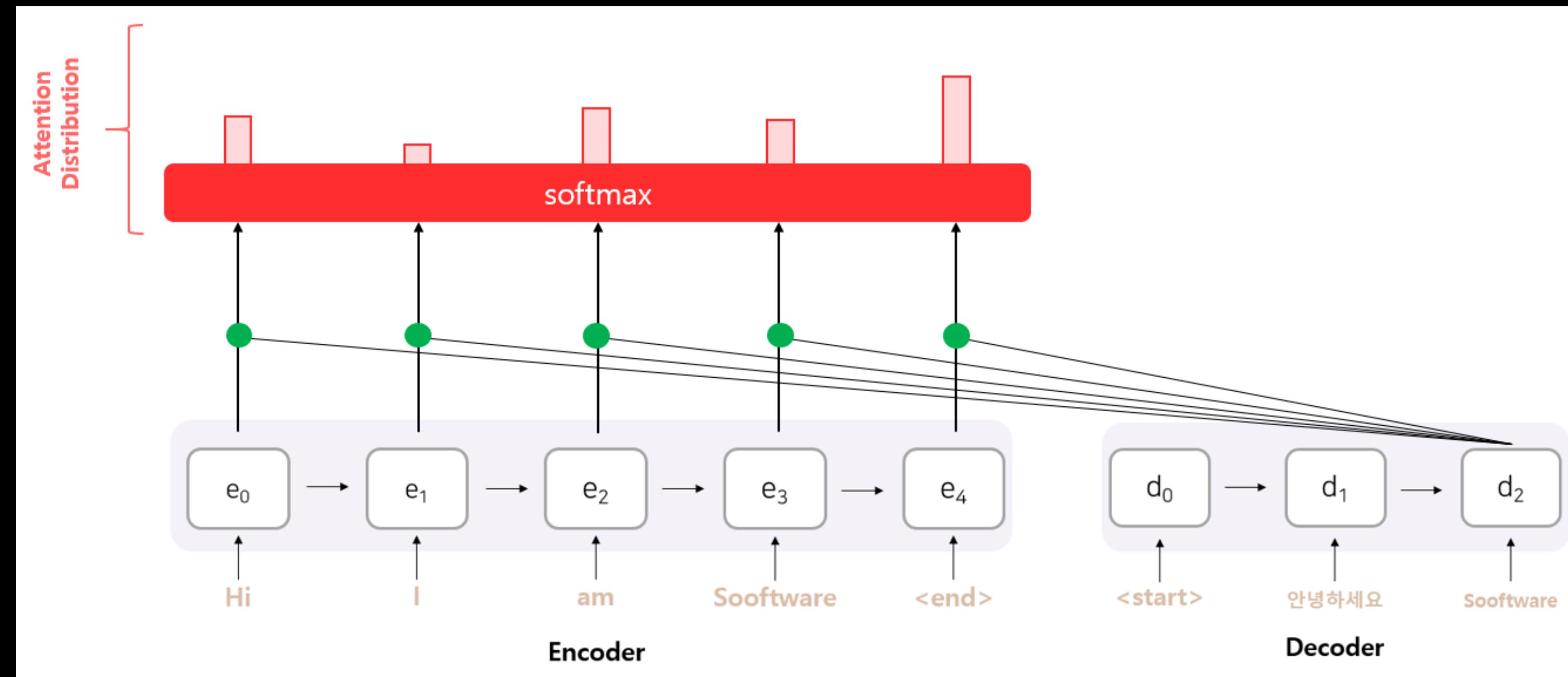


Attention Part 1 : Get Attention Score

현재 Decoder의 Hidden State(i)와 Encoder의 모든 Hidden State(j)를
내적하여 Attention Score(e)를 구한다.

Attention Mechanism

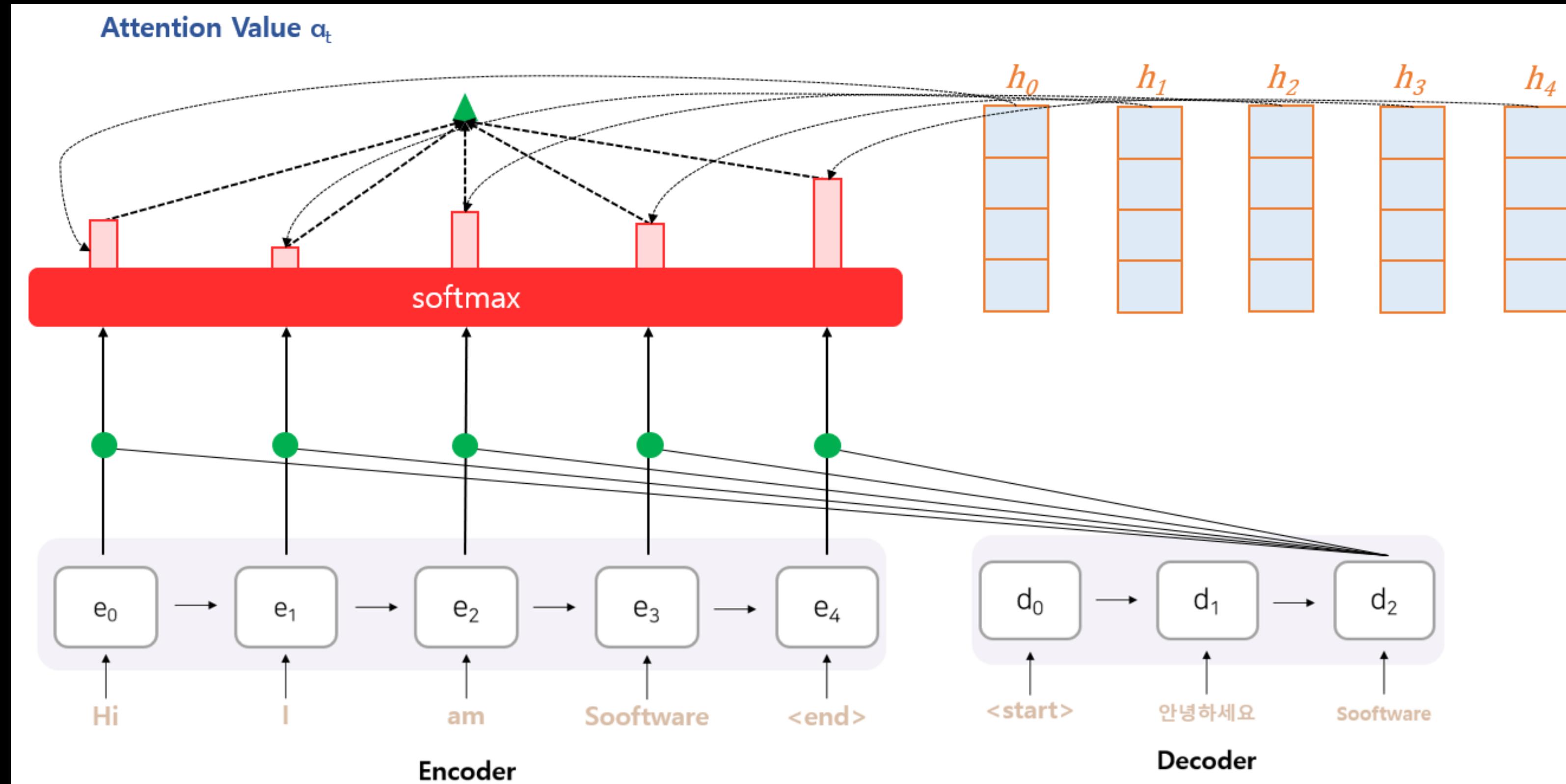
Before Transformer



Attention Part 2 : Get Attention Distribution
Softmax를 통하여 정규화하여 확률의 형태로 만든다.

Attention Mechanism

Before Transformer

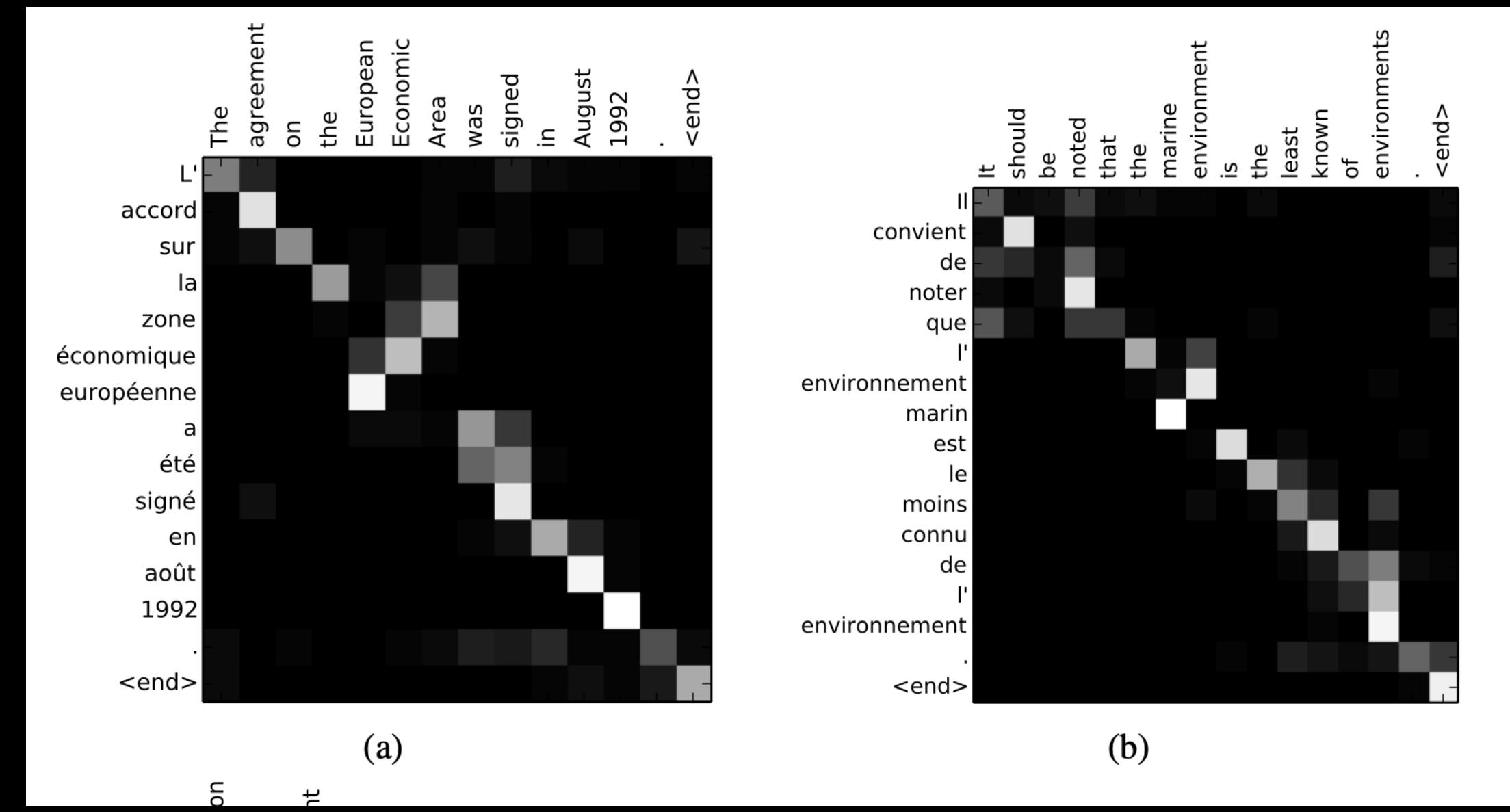


Attention Part 3 : Weighted Sum

Attention Distribution에 따라 Encoder의 Hidden State들을 가중치를 곱하여 합한다.

Attention Machanism

Before Transformer



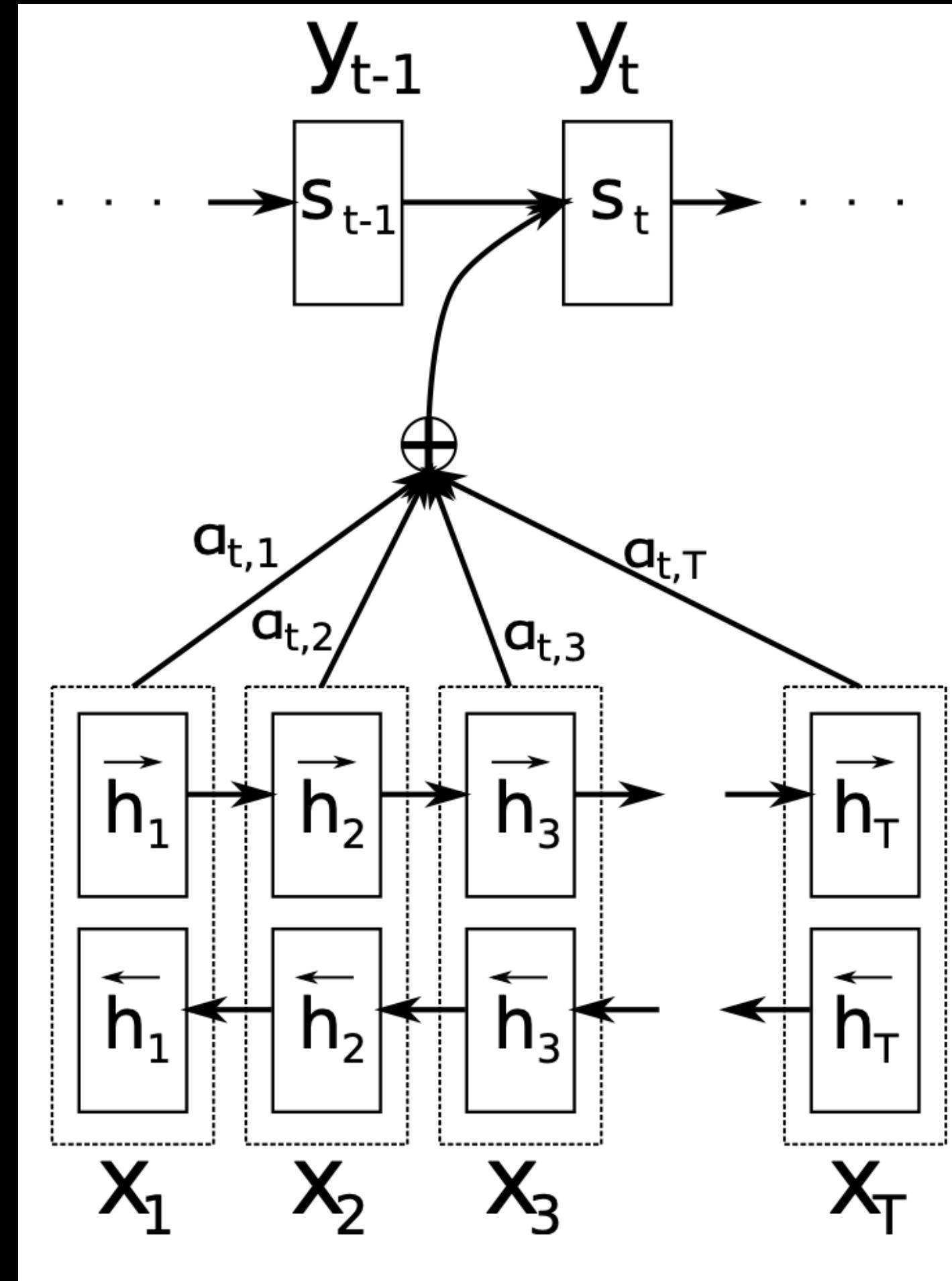
Result of Attention Machanism

모델이 입력 문장의 어떤 부분을 참고하여 Output을 내놓는지 알 수 있음.

Attention Machanism

Before Transformer

- 현재 사용된 Attention Machanism은 LSTM/GRU와 같은 Recurrent Network과 함께 사용됨.
- Encoder Network의 Hidden State를 바탕으로 Attention Machanism을 통해 Weighted Sum으로 모든 정보를 참고할 수 있었음.
- 하지만, 여전히 RNN의 사용으로 순차적인 계산으로 인해 병렬화가 힘들다는 단점이 존재함.



Transformer

Overview Transformer

- Positional Encoding을 통해 RNN을 사용하지 않아도 순서에 대한 정보를 모델에게 전달할 수 있다.
- RNN/CNN를 사용하지 않고, Attention Mechanism만을 통해 Machine Translation 분야에서 SOTA 성능을 냈었음.
- Self-Attention, 즉 문장 자신에게 Attention Score를 구하여 문장의 Context를 구해내고, 이를 여러 Layer에 걸쳐서 반복.

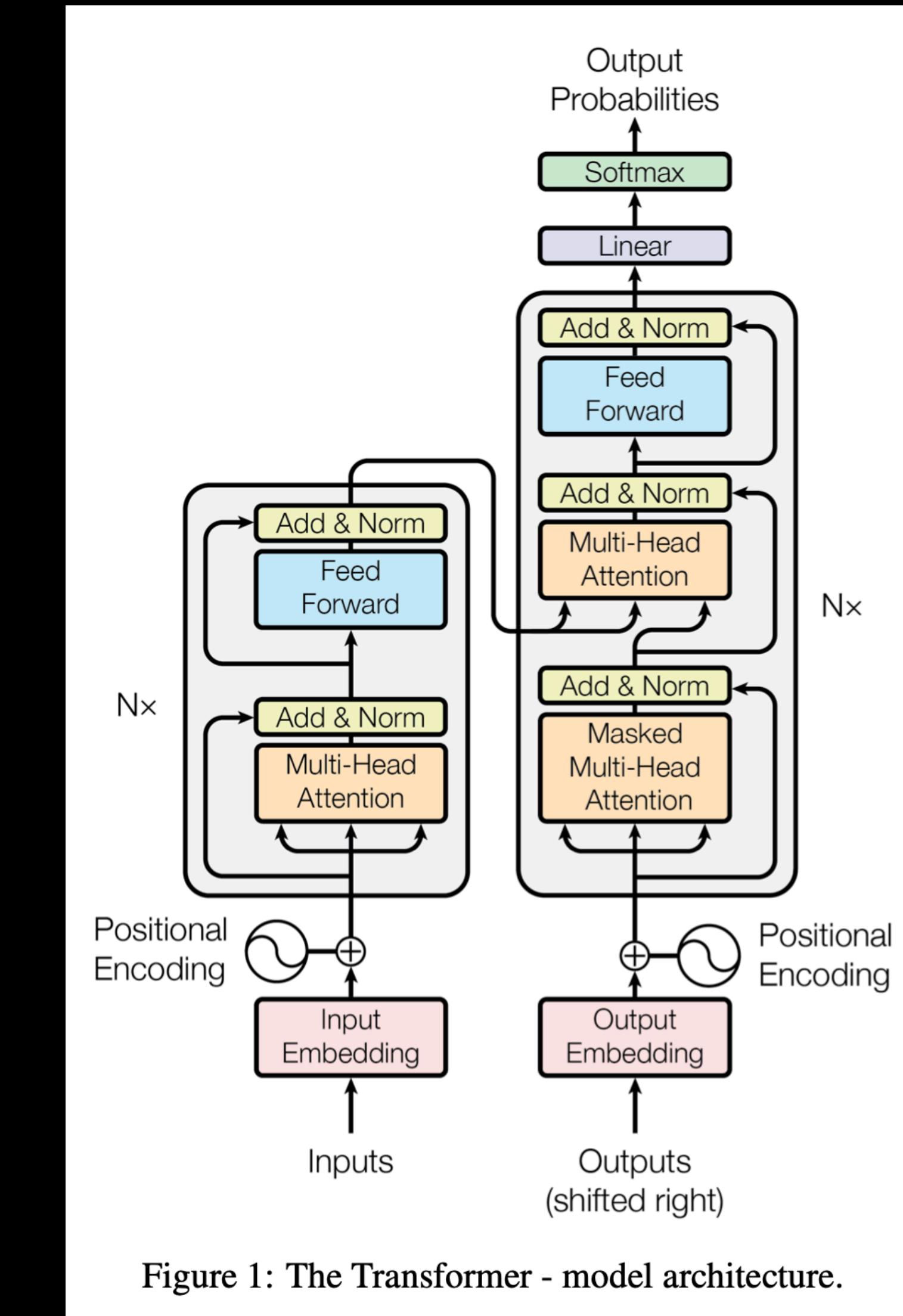


Figure 1: The Transformer - model architecture.

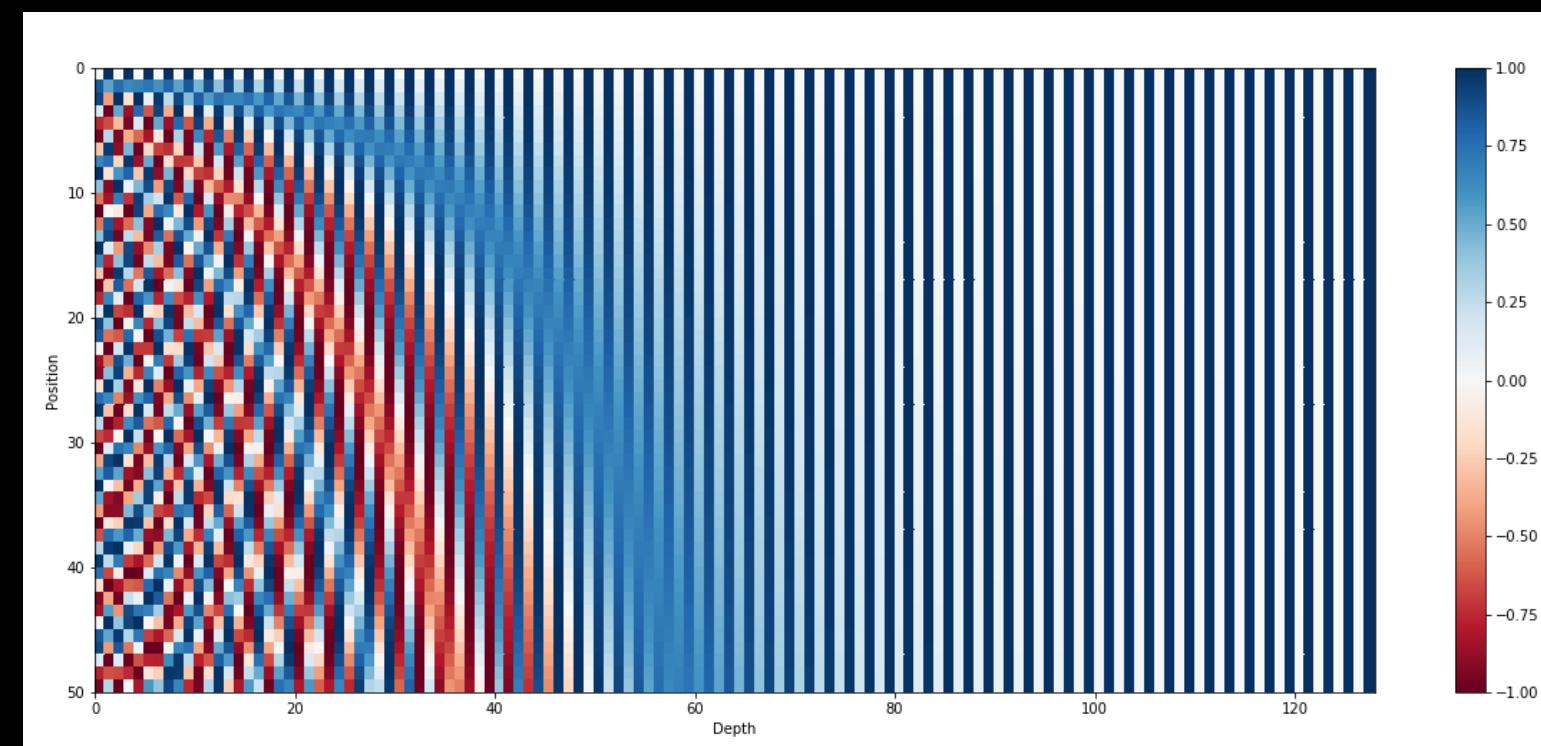
Positional Encoding

Transformer

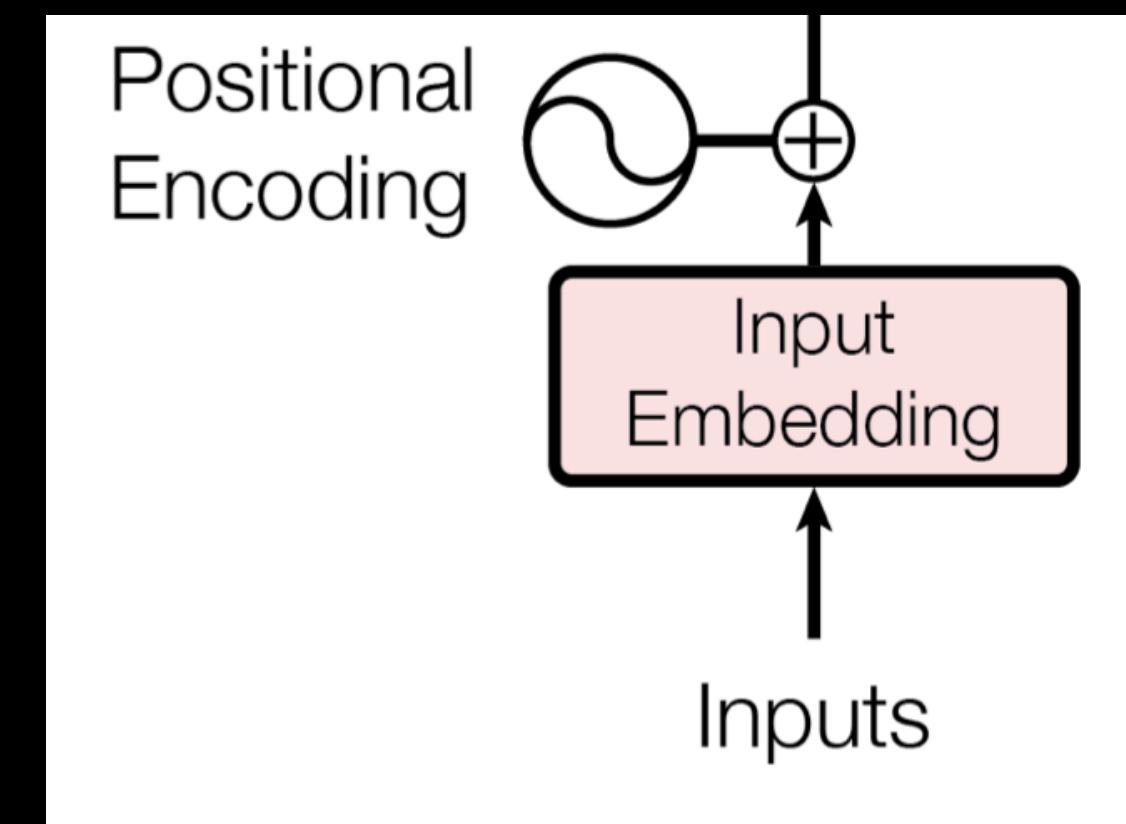
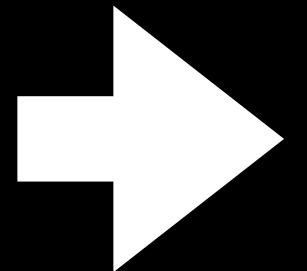
$$\vec{p}_t^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k \cdot t), & \text{if } i = 2k \\ \cos(\omega_k \cdot t), & \text{if } i = 2k + 1 \end{cases}$$

[1]

$$\omega_k = \frac{1}{10000^{2k/d}}$$



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



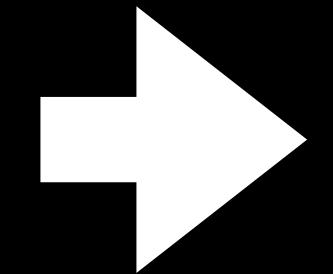
Positional Encoding in Transformer

모델에게 순서에 대한 정보를 sinusoidal function을 통해 Embed

Positional Encoding

Transformer

0 :	0 0 0 0	8 :	1 0 0 0
1 :	0 0 0 1	9 :	1 0 0 1
2 :	0 0 1 0	10 :	1 0 1 0
3 :	0 0 1 1	11 :	1 0 1 1
4 :	0 1 0 0	12 :	1 1 0 0
5 :	0 1 0 1	13 :	1 1 0 1
6 :	0 1 1 0	14 :	1 1 1 0
7 :	0 1 1 1	15 :	1 1 1 1



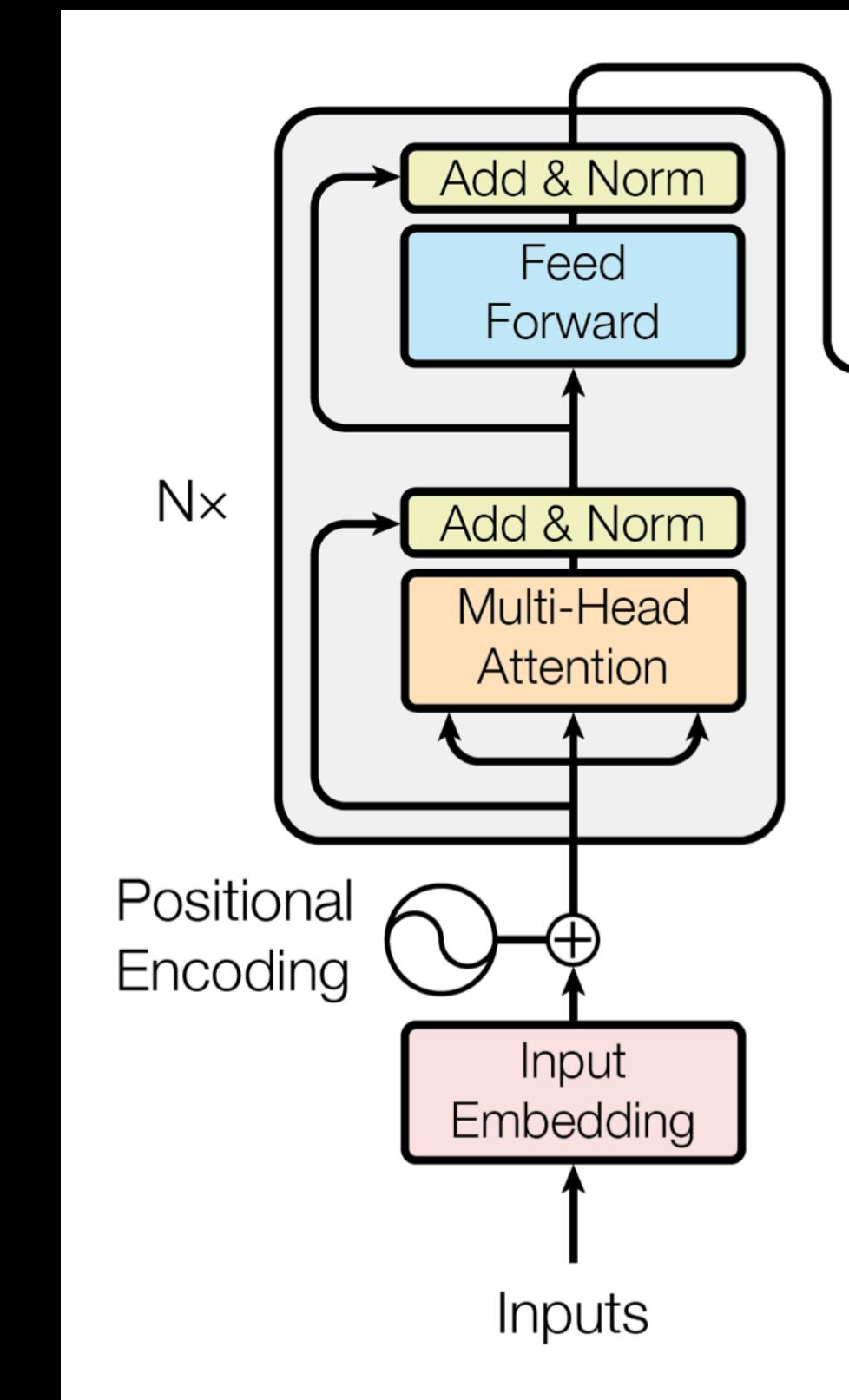
$$\vec{p}_t = \begin{bmatrix} \sin(\omega_1 \cdot t) \\ \cos(\omega_1 \cdot t) \\ \sin(\omega_2 \cdot t) \\ \cos(\omega_2 \cdot t) \\ \vdots \\ \sin(\omega_{d/2} \cdot t) \\ \cos(\omega_{d/2} \cdot t) \end{bmatrix}_{d \times 1}$$

Why Sinusoidal in Positional Encoding

Binary로 순서를 표현할 때 각 bit의 주기가 자리수가 커질수록 증가하는 것처럼, 이를 Sinusoidal Function을 통해 저비용으로 표현해낼 수 있다.

Encoder in Transformer

Transformer

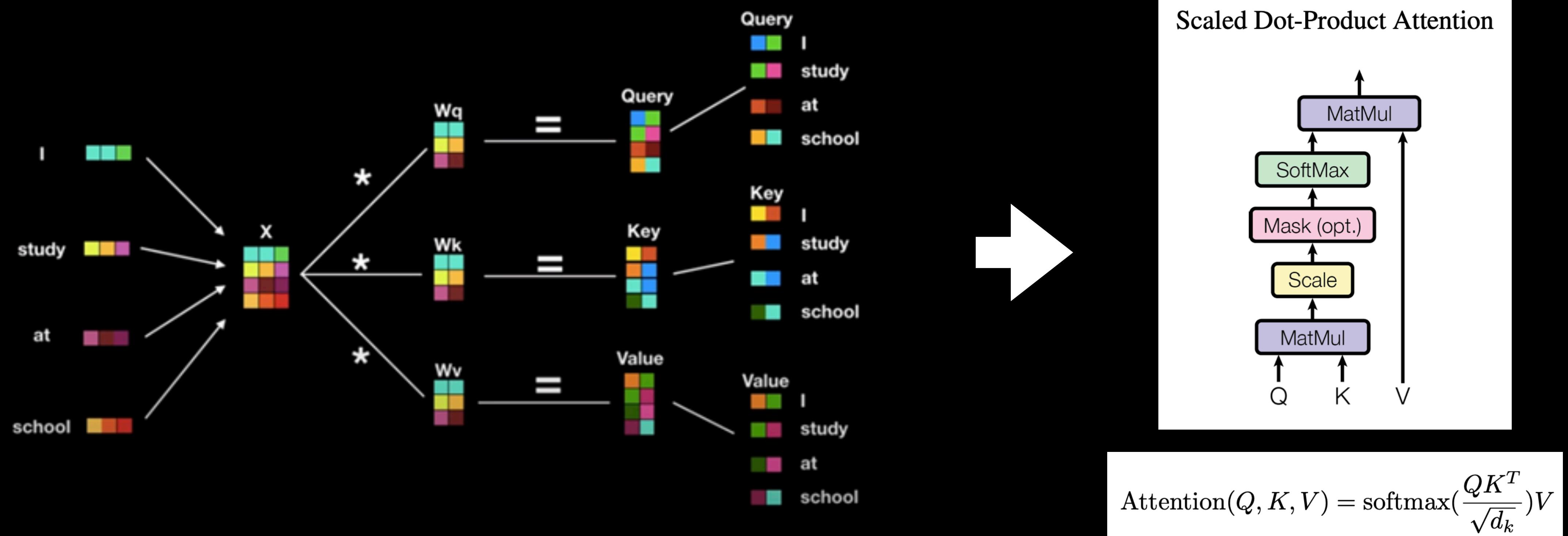


Encoder

Embedding한 값을 Multi-Head Attention, Residual Connection, 그리고 FFNet으로
Hidden State를 만들고, 이를 여러번 반복하여 최종 Hidden State를 구해낸다.

Attention in Transformer

Transformer

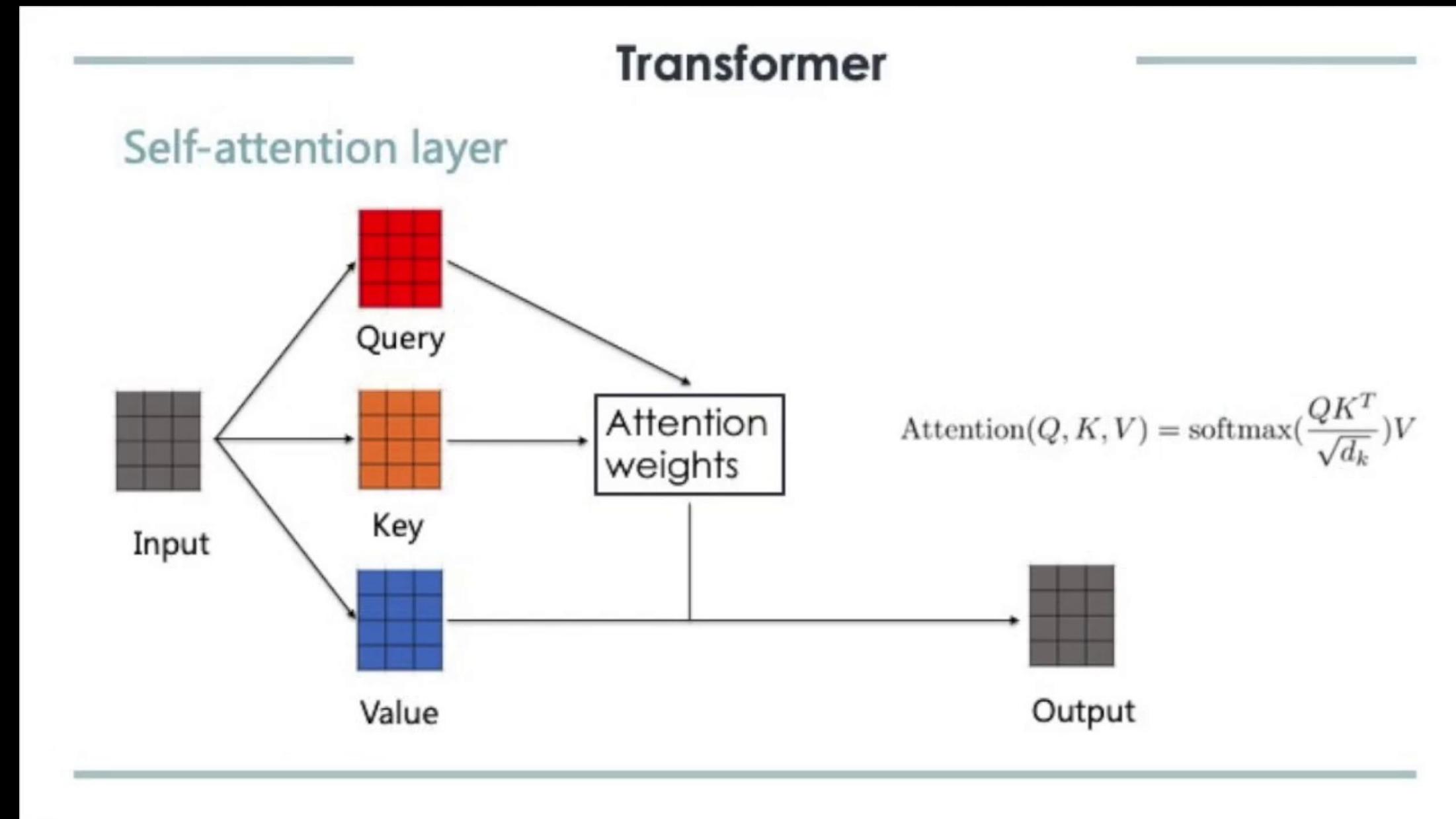


Attention in Transformer

Query, Key, Value값을 받으며, Query와 Key간 Attention Score를 구하고, 이를 Value에 대해 Weighted Sum을 해준다.

Attention in Transformer

Transformer



Query, Key, Value (문장 B의 정보로 문장 A의 Attention을 구할 때)

Query(Q) : Attention을 구할 Target 문장의 정보로, B로부터 영향받는 A의 변수

Key(K) : Attention을 구할 때 참고하는 Source 문장의 정보로, A에게 영향을 주는 B의 변수

Value(V) : 문장 B가 영향을 주는 정도, 즉 영향에 대한 가중치

Attention in Transformer

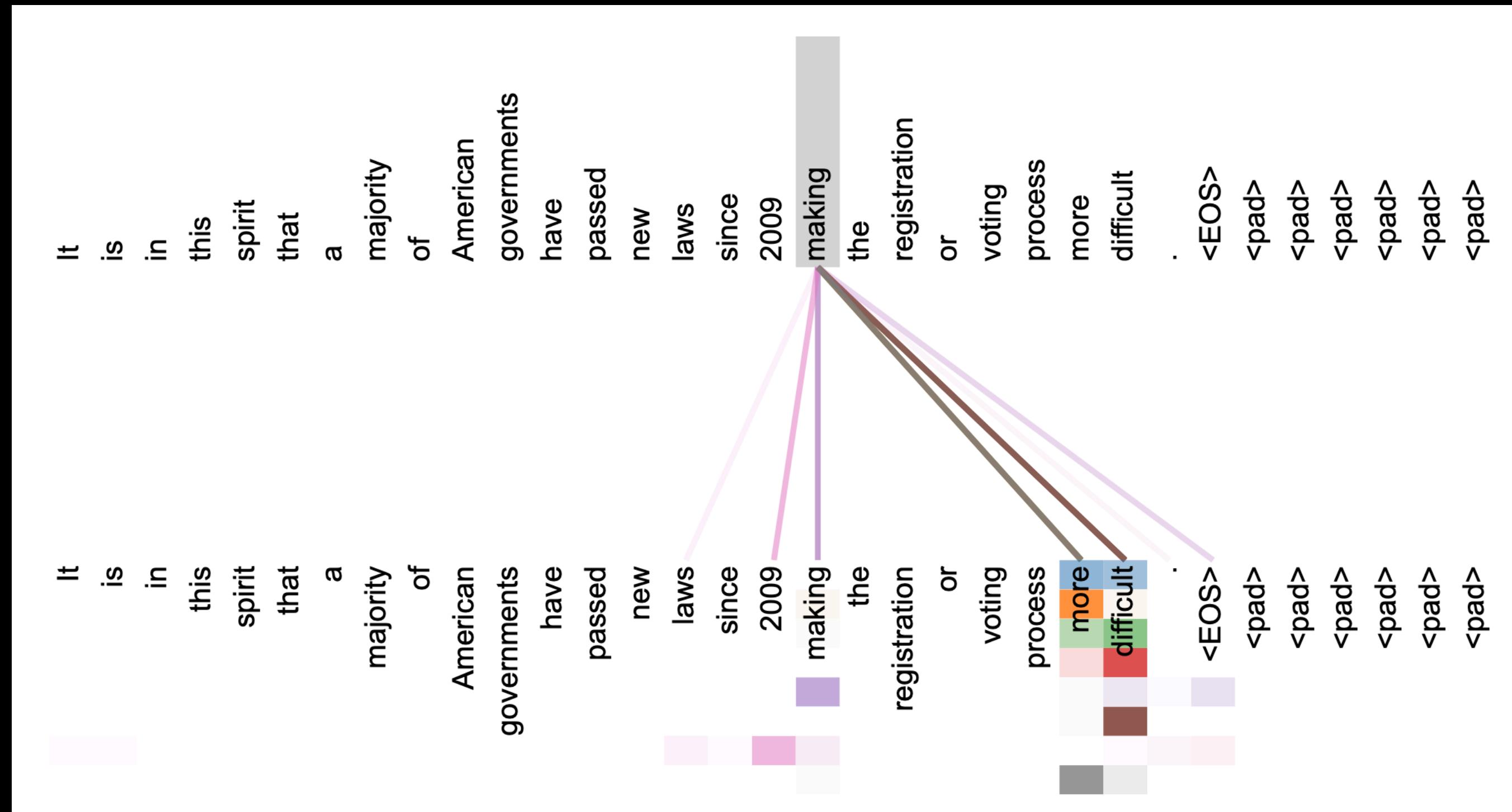
Transformer

	Query * Key ^T	Score	Softmax	Value	Softmax * Value	\sum Softmax * Value (Attention layer output)
I	I * I	= 130	0.92			
	I * study	= 50	0.05			
	I * at	= 20	0.02			
	I * school	= 10	0.01			
study	study * I	= 30	0.02			
	study * study	= 110	0.70			
	study * at	= 20	0.03			
	study * school	= 70	0.25			
at	at * I	= 30	0.03			
	at * study	= 50	0.10			
	at * at	= 90	0.80			
	at * school	= 40	0.07			

Attention in Transformer

입력문장의 모든 단어에 대해서 Attention Output을 만든다.

Attention in Transformer

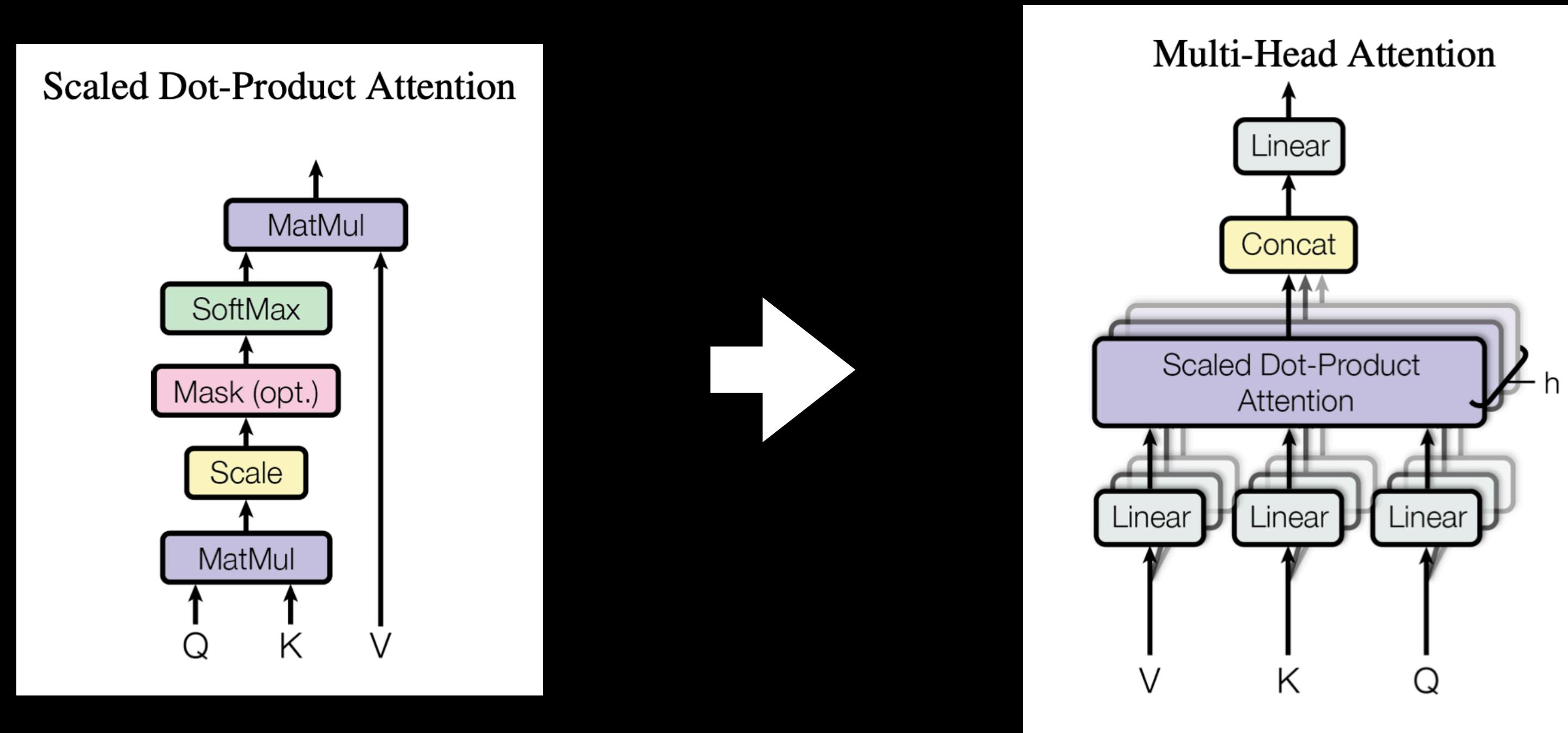


Self-Attention

입력문장 자신에게 Attention 연산을 진행하여 Context를 만들어낸다.
즉, Query Key Value 모두가 입력문장의 정보가 된다.

Attention in Transformer

Transformer



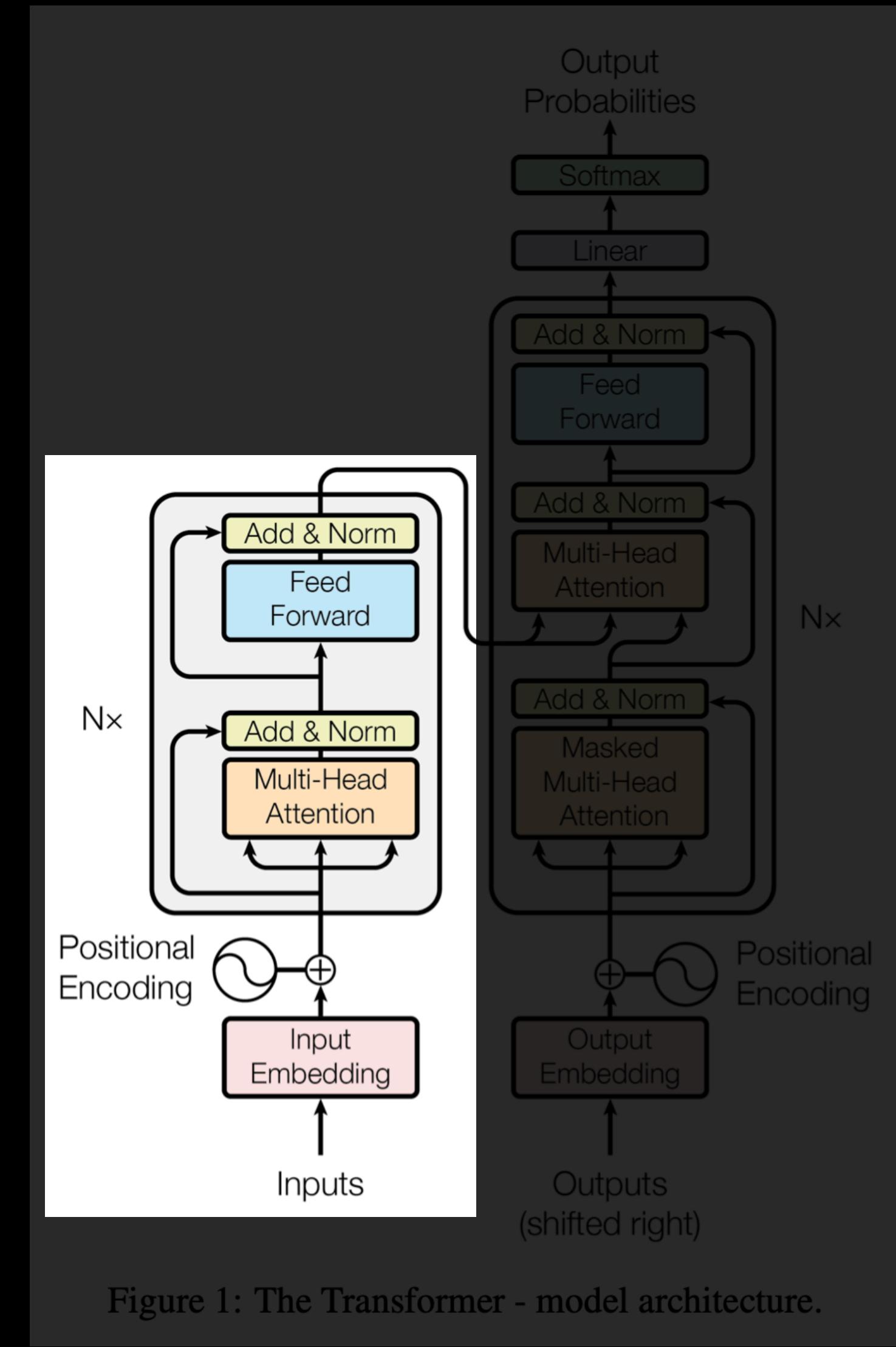
Multi-Head Attention

같은 문장에 대해 여러개의 Attention을 구하여 Ensemble처럼 다양한 각도로 문장에서 정보를 추출하도록 한다.

Encoder in Transformer

Transformer

- Input Embedding에 Positional Encoding을 Embed시킨 뒤, 이를 Multi-Head Attention으로 문장의 Context를 추출한다.
- Residual Connection을 통해 깊은 Network라도 쉽게 수렴할 수 있도록 한다. (Target을 직접 학습하는 것이 아닌, Input으로부터 달라지는 부분을 학습)
- Feed Forward Network를 뒤에 붙여서 Context를 학습 가능한 Space로 Projection한다.
- 이 과정을 N번 반복하여 최종적인 Encoder의 Hidden State를 추출한다.



Decoder in Transformer

Transformer

- Encoder의 마지막 Output을 바탕으로 모든 Decoder Layer에 활용
- 전단계의 단어를 Embedding시키고, Masked Multi-Head Attention을 통해 Decoder에서 미래에 나올 단어들을 무시시키면서 Self-Attention을 구할 수 있도록 한다.
- 그 다음, Masked Multi-Head Attention의 Output을 Query로, Encoder의 Output을 Key와 Value로 Attention을 수행한다.
- 똑같이 Feed Forward Network로 Projection을 진행하며, 이 전체 과정을 N번 반복한다.
- 최종 Output에서는 Softmax로 확률을 구한다.

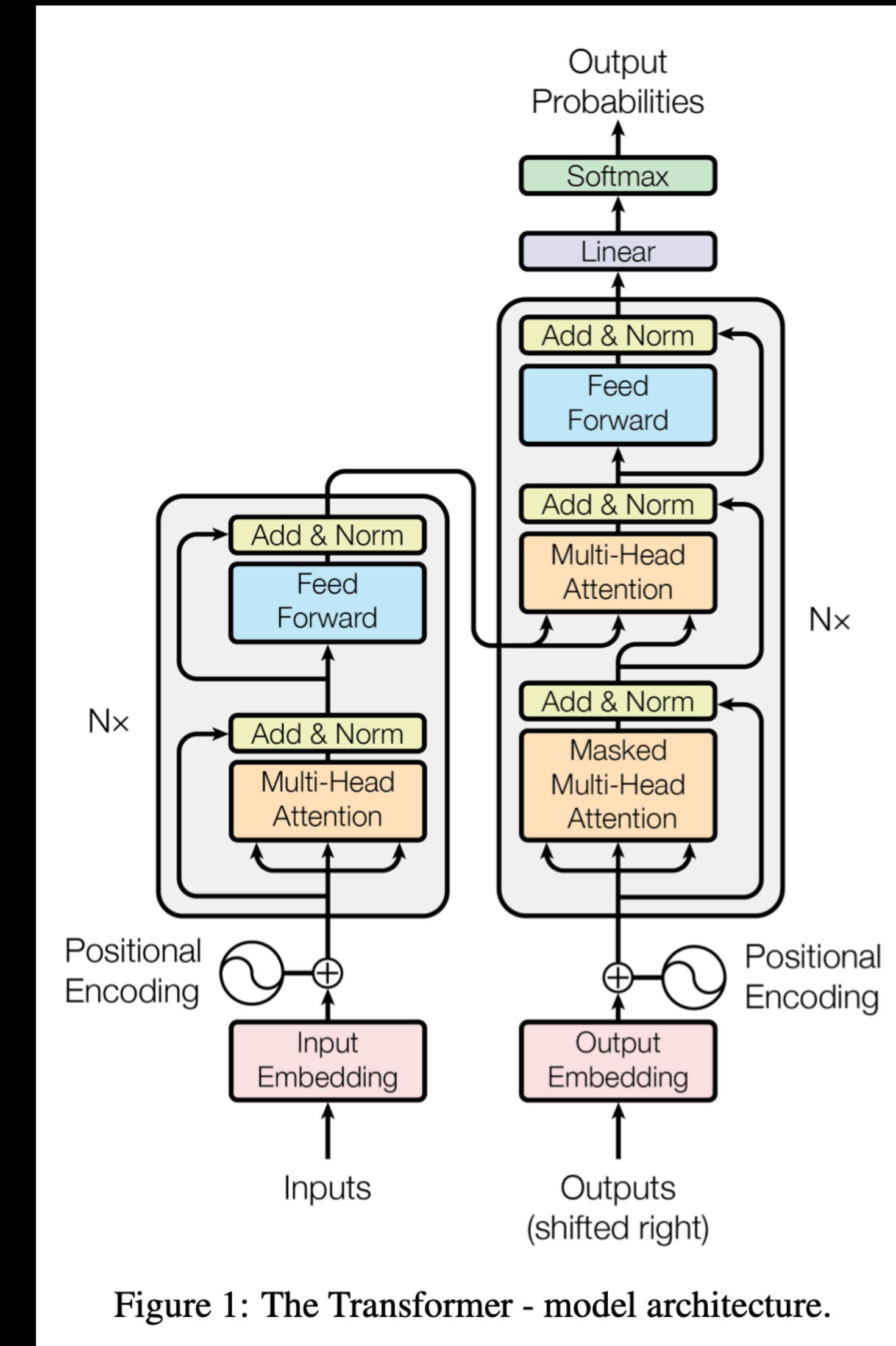
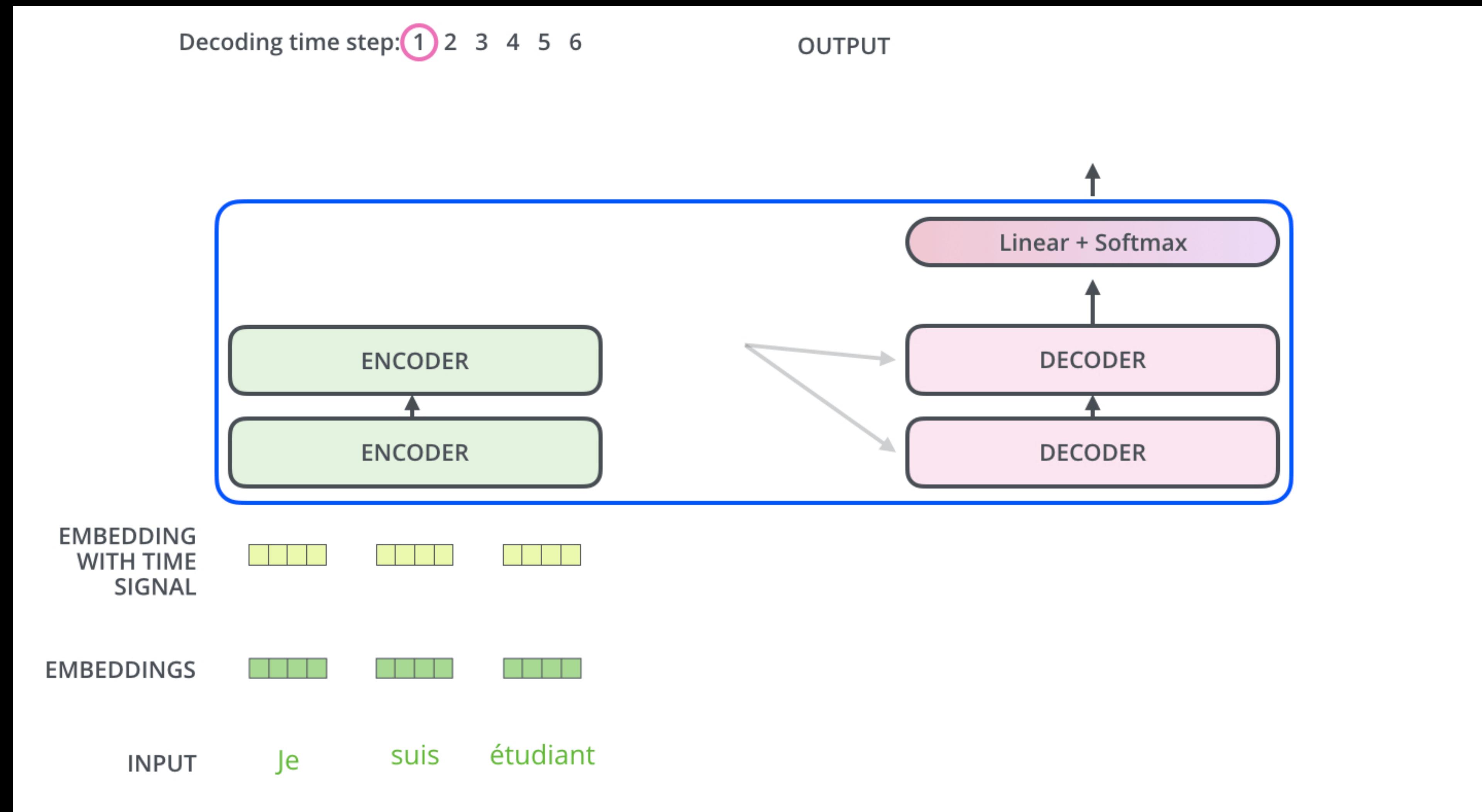


Figure 1: The Transformer - model architecture.

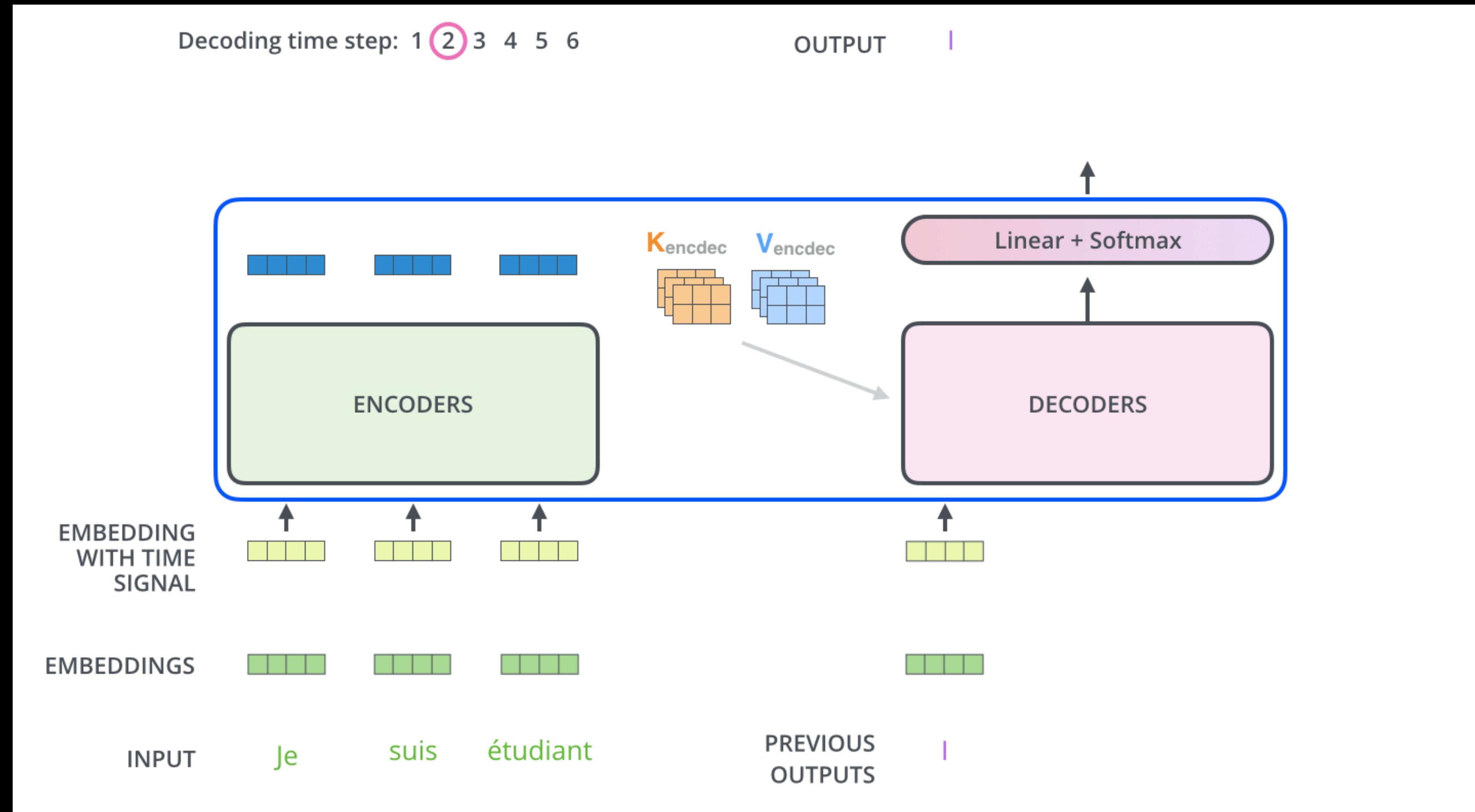
Process of Transformer

Transformer



Process of Transformer

Transformer



Result

Training Result

5.1 Training Data and Batching

We trained on the standard WMT 2014 English-German dataset consisting of about 4.5 million sentence pairs. Sentences were encoded using byte-pair encoding [3], which has a shared source-target vocabulary of about 37000 tokens. For English-French, we used the significantly larger WMT 2014 English-French dataset consisting of 36M sentences and split tokens into a 32000 word-piece vocabulary [38]. Sentence pairs were batched together by approximate sequence length. Each training batch contained a set of sentence pairs containing approximately 25000 source tokens and 25000 target tokens.

5.2 Hardware and Schedule

We trained our models on one machine with 8 NVIDIA P100 GPUs. For our base models using the hyperparameters described throughout the paper, each training step took about 0.4 seconds. We trained the base models for a total of 100,000 steps or 12 hours. For our big models,(described on the bottom line of table 3), step time was 1.0 seconds. The big models were trained for 300,000 steps (3.5 days).

Training Data, and Hardware

WMT 2014 영어-독일어 Dataset 사용. (450만 문장쌍)

Training Result

Result

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	41.29	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$3.3 \cdot 10^{18}$	
Transformer (big)	28.4	41.8	$2.3 \cdot 10^{19}$	

Training Result (BLEU Score)

당시에 SOTA를 달성했으며, 연산량도 다른 모델에 비해 적었음

Training Result

Result

BLEU

BLEU(Bilingual Evaluation Understudy) score란 성과지표로 데이터의 X가 순서정보를 가진 단어들(문장)로 이루어져 있고, y 또한 단어들의 시리즈(문장)로 이루어진 경우에 사용되며, 번역을 하는 모델에 주로 사용된다. 3가지 요소를 살펴보자.

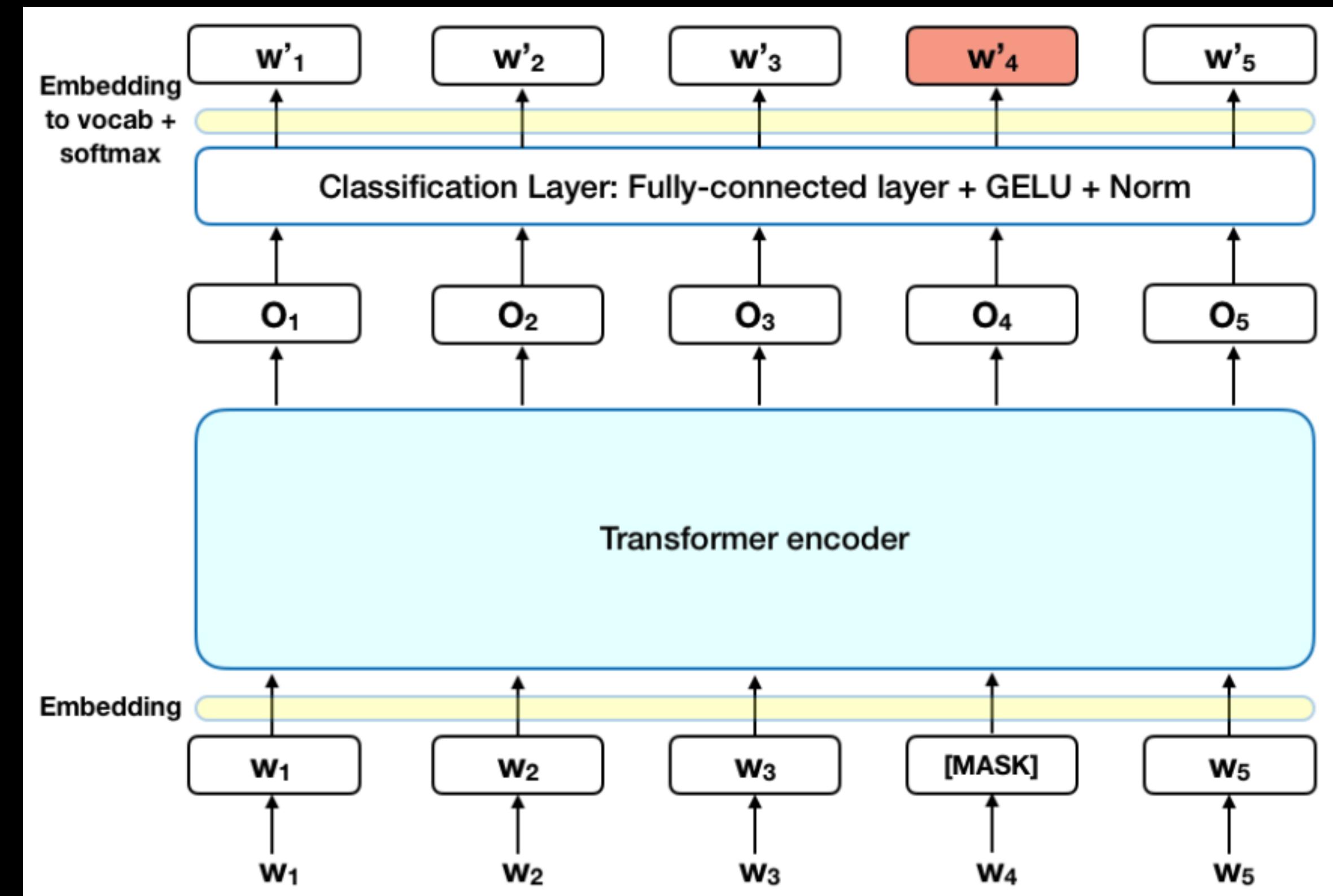
- n-gram을 통한 순서쌍들이 얼마나 겹치는지 측정(precision)
- 문장길이에 대한 과적합 보정 (Brevity Penalty)
- 같은 단어가 연속적으로 나올때 과적합 되는 것을 보정(Clipping)

$$BLEU = \min(1, \frac{output\ length(\text{예측 문장})}{reference\ length(\text{실제 문장})})(\prod_{i=1}^4 precision_i)^{\frac{1}{4}}$$

BLEU Score
Machine Translation의 Metric

Usage of Transformer

Result



Transformer는 현재 NLP에서는 BERT, GPT등 여러 모델에 활용중이며,
NLP분야 뿐만이 아닌 비전, 오디오 등 다양한 분야에 사용중.

Thank You