KOREA UNIVERSITY

DSBA CS224n 2021 Study

[Lecture 09]
Self- Attention and Transformers

DSBA
Data Science & Business Analytics

고려대학교 산업경영공학과

Data Science & Business Analytics Lab

발표자 : 노건호

00 발표 목차
Contents

1. From recurrence (RNN) to attention-based NLP models
2. Introducing the Transformer model
3. Great results with Transformers
4. Drawbacks and variants of Transformers

01 From recurrence (RNN) to attention-based NLP models

Sub-content

1. Issues with recurrent models

2. self-attention

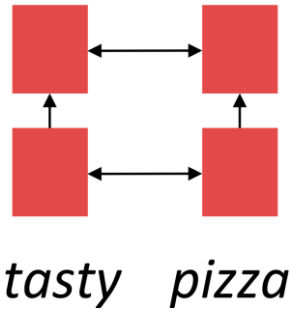
3. Barriers and solutions for Self-Attention as a building block

01 From recurrence (RNN) to attention-based NLP models

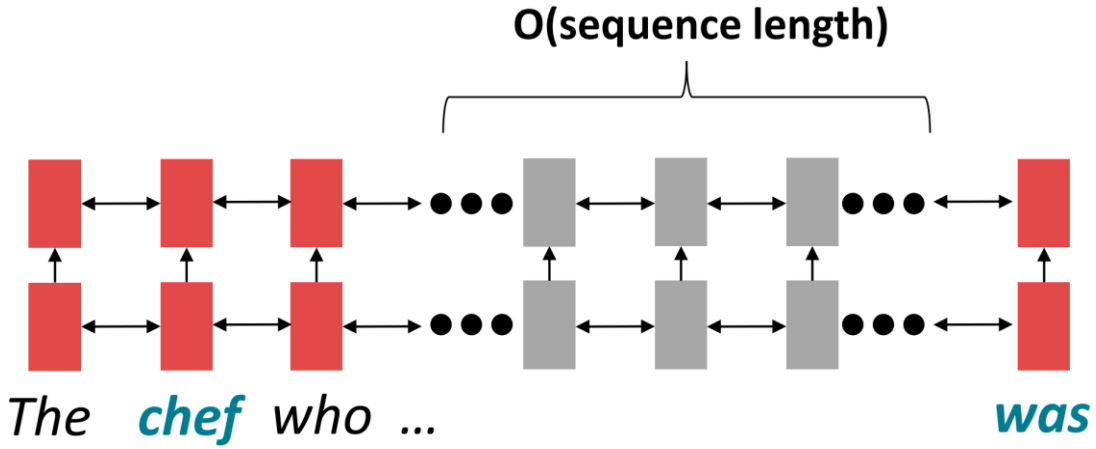
1. Issues with recurrent models

문제 1: Linear interaction distance

- RNN은 왼쪽부터 오른쪽으로 순차적으로 연산



가까운 단어들은 서로 상호작용이 용이하다.



아래처럼 주어인 chef와 동사 was 사이가 매우 멀다.

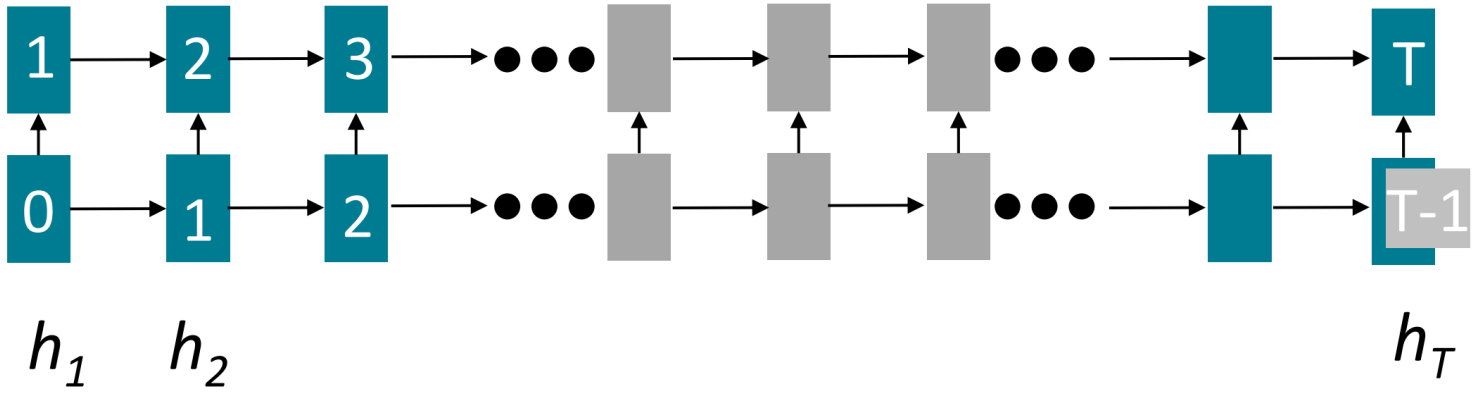
RNN은 멀리 떨어진 단어 쌍이 상호작용하기 위해선 O(sequence length) steps 만큼 소요되며, Gradient 문제로 멀리 떨어진 dependency는 학습하기 어렵다.

01 From recurrence (RNN) to attention-based NLP models

1. Issues with recurrent models

문제 2: Lack of parallelizability

- Future hidden state는 반드시 그 직전 hidden state의 계산이 완료되어야 처리가 가능하다.



#: 해당 state가 연산되기 까지 완료 되어야하는 최소 연산의 수

즉 병렬연산이 불가능하다.

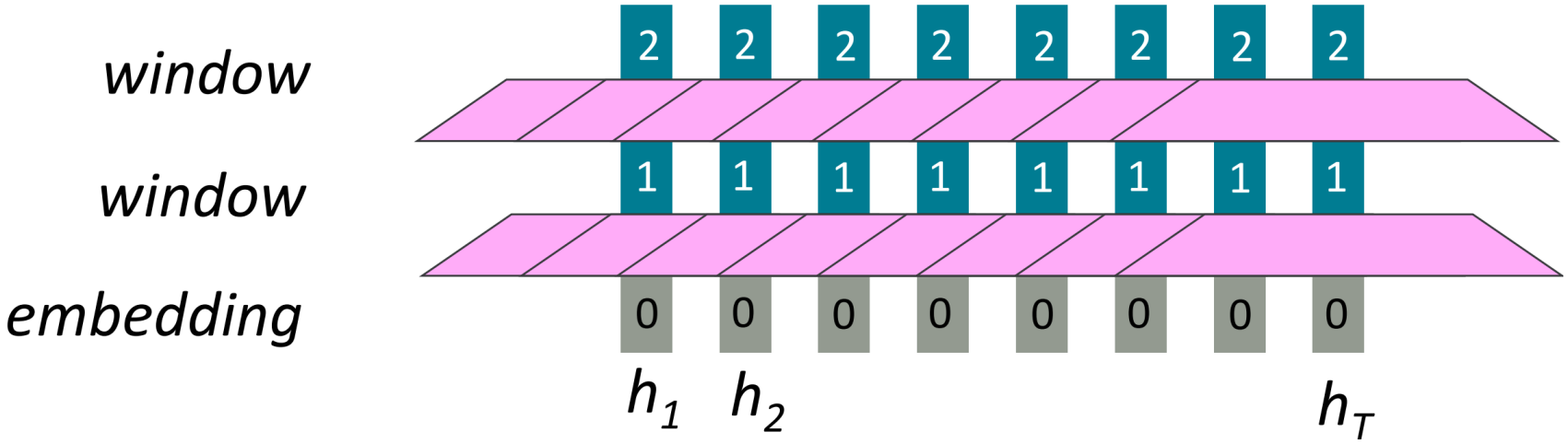
Forward and backward passes 모두 O(sequence length)의 병렬연산이 불가능한 과정을 거친다.

01 From recurrence (RNN) to attention-based NLP models

1. Issues with recurrent models – How about word window?

How about word windows?

- Word window 모델은 local context를 통합



sequence length가 증가하여도 병렬처리 불가한 연산이 증가하지 않는다!

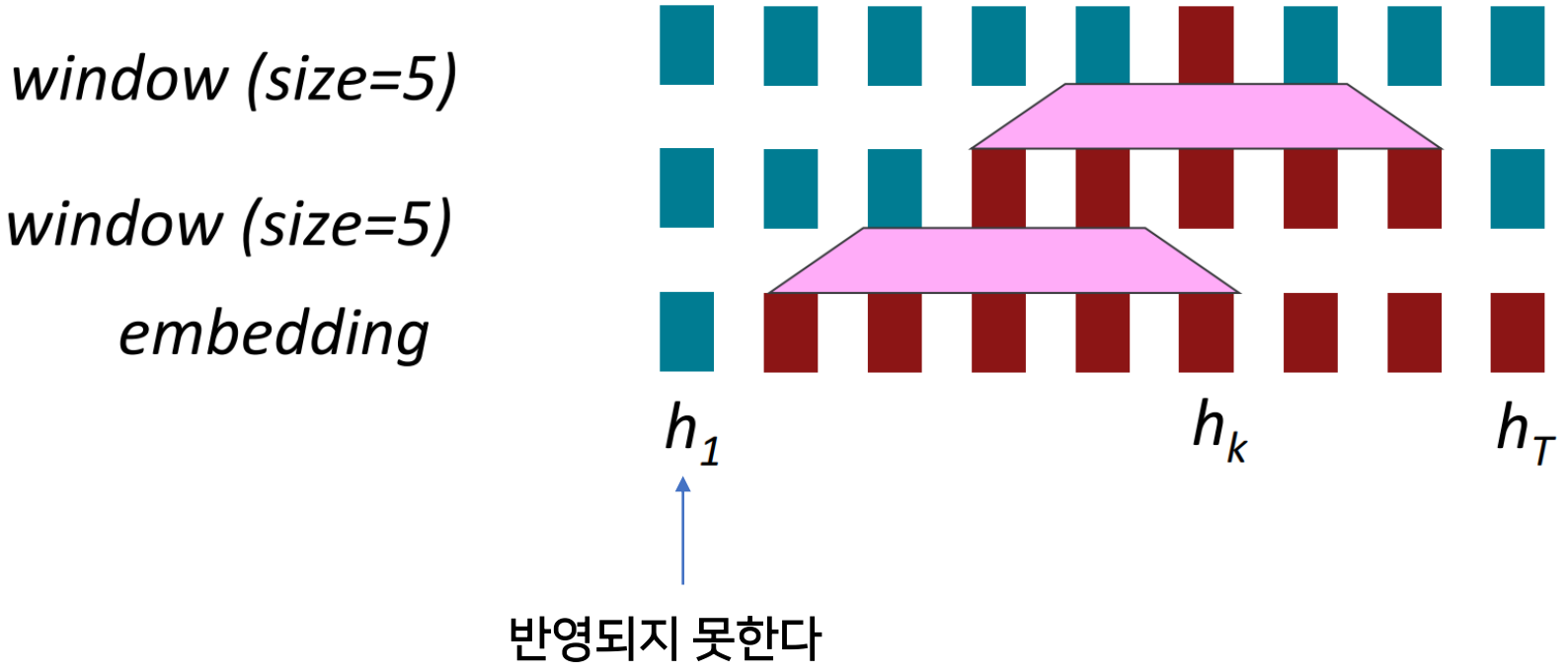
01 From recurrence (RNN) to attention-based NLP models

1. Issues with recurrent models – How about word window?

How about word windows?

- Long-distance dependency 문제

  Local contexts를 통합하는 word window의 특성상 멀리 떨어져 있는 dependency를 반영하기 위해선 더 깊게 window를 쌓아야 한다.

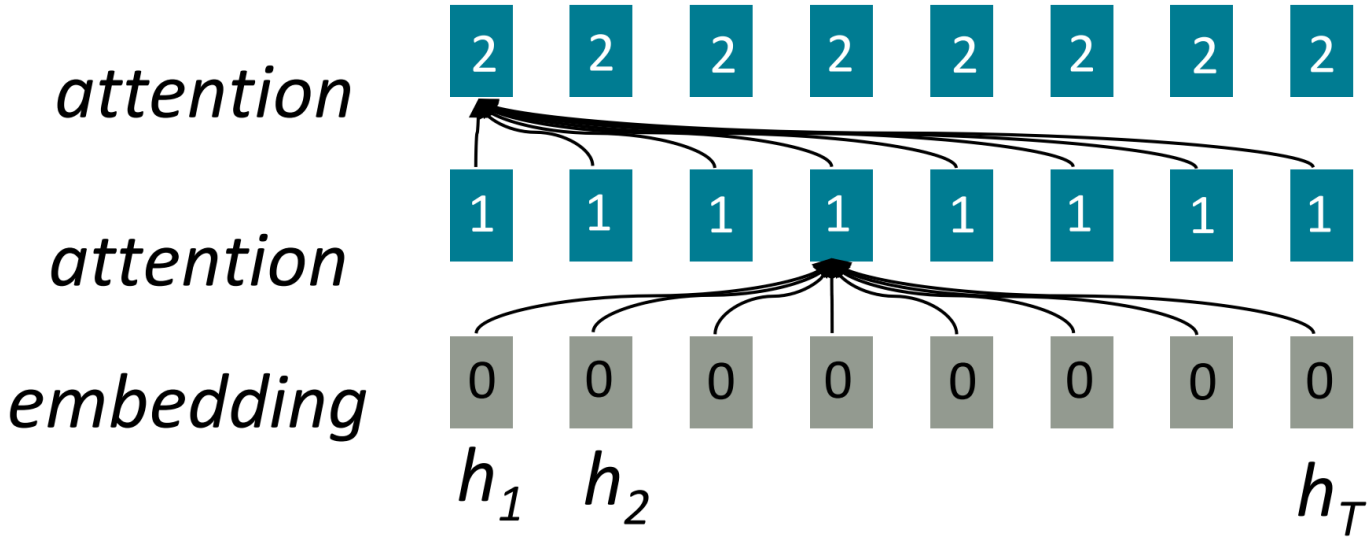


01 From recurrence (RNN) to attention-based NLP models

1) Issues with recurrent models

How about attention?

- Self Attention: encoder decode사이가 아닌 한 문장 안에서의 Attention



sequence length가 증가하여도 병렬처리 불가한 연산이 증가하지 않는다!

모든 단어가 연결되어 어떤 단어 사이에도 O(1) 연산만으로 상호작용이 가능하다

01 From recurrence (RNN) to attention-based NLP models

2) Self-Attention

Self-Attention은 병렬연산이 가능하고 Long distance dependency 문제가 발생하지 않는다.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

병렬연산 가능

Long distance dependency 문제 해결

01 From recurrence (RNN) to attention-based NLP models

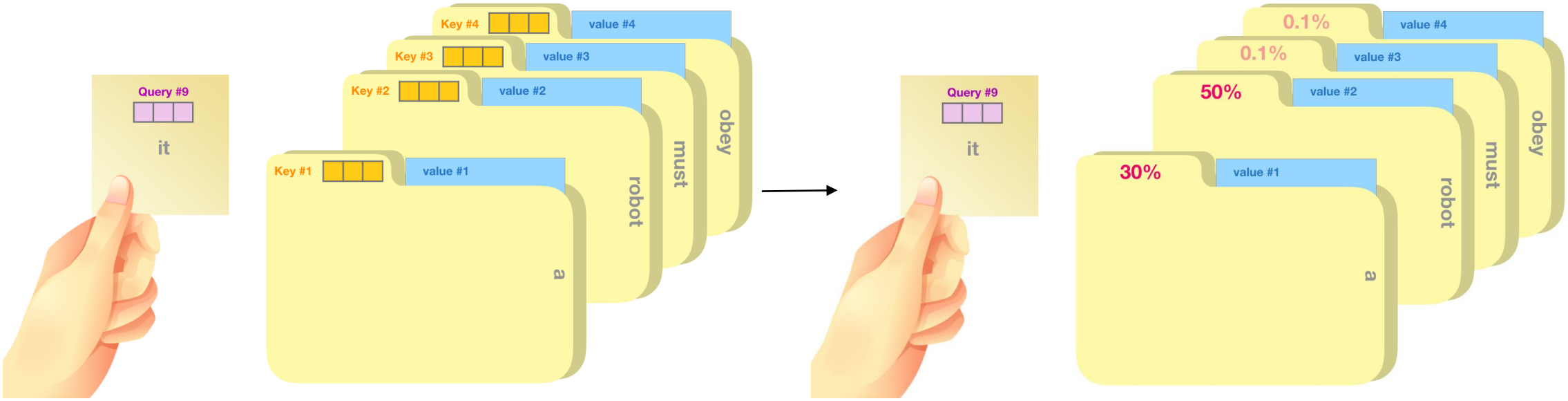
2) Self-Attention

Alamar (Transformer)

•Query: The query is a representation of the current word used to score against all the other words (using their keys). We only care about the query of the token we're currently processing.

•Key: Key vectors are like labels for all the words in the segment. They're what we match against in our search for relevant words.

•Value: Value vectors are actual word representations, once we've scored how relevant each word is, these are the values we add up to represent the current word.



01 From recurrence (RNN) to attention-based NLP models

2) Self-Attention

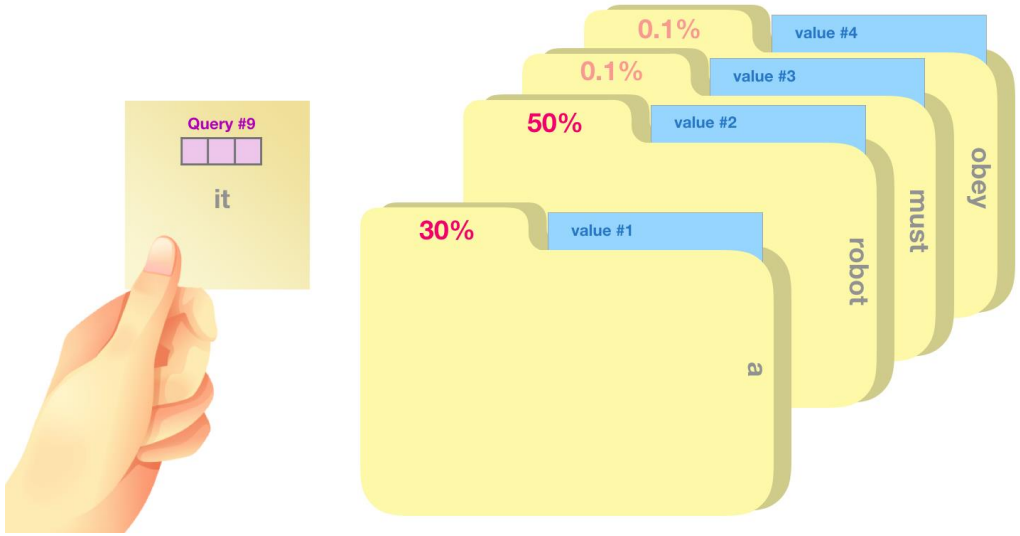
	Self-Attention	Attention
	$v_i = k_i = q_i = x_i$	
Attention score :	$e_{ij} = q_i^\top k_j$	$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$
Attention weight :	$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$	$\alpha^t = \mathrm{softmax}(\boldsymbol{e}^t) \in \mathbb{R}^N$
Attention output :	$\mathrm{output}_i = \sum_j \alpha_{ij} v_j$	$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$

01 From recurrence (RNN) to attention-based NLP models

2) Self-Attention

Alamar (Transformer)

- A robot must obey the orders given it



Word	Value vector	Score	Value X Score
<s>		0.001	
a		0.3	
robot		0.5	
must		0.002	
obey		0.001	
the		0.0003	
orders		0.005	
given		0.002	
it		0.19	
		Sum:	

From recurrence (RNN) to attention-based NLP models

3) Barriers and solutions for Self-Attention as a building block

1. Doesn’t have an inherent notion of order!

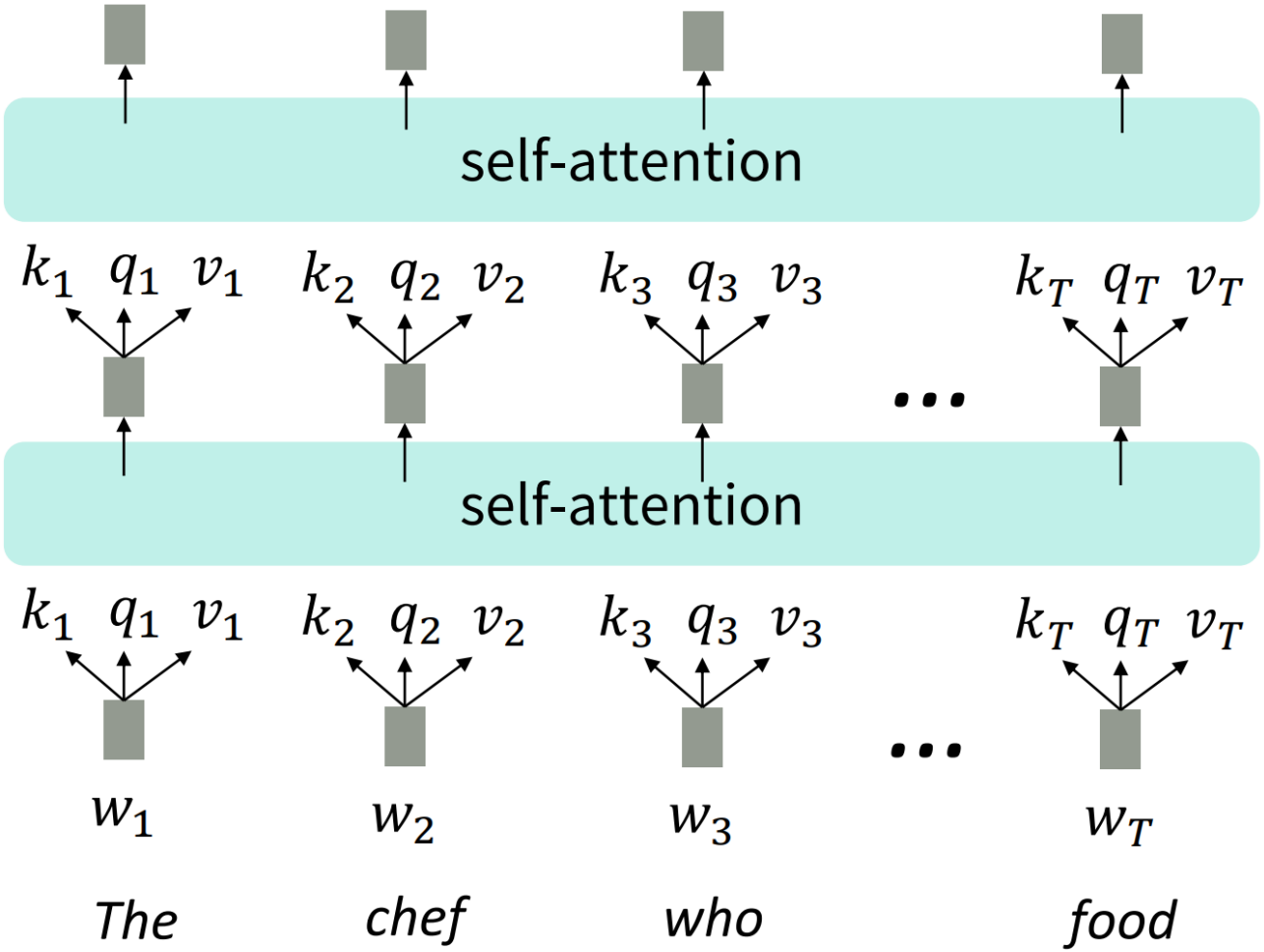
2. No nonlinearities for deep learning magic! It’s all just weighted averages

3. Need to ensure we don’t “look at the future” when predicting a sequence

01 From recurrence (RNN) to attention-based NLP models

3) Barriers and solutions for Self-Attention as a building block

문제 1: sequence order



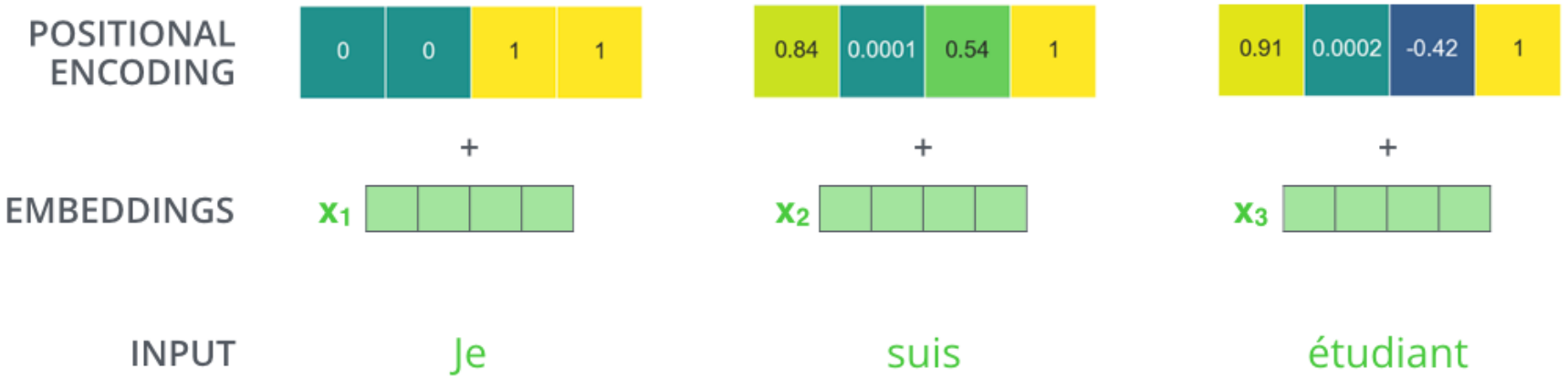
01 From recurrence (RNN) to attention-based NLP models

3) Barriers and solutions for Self-Attention as a building block

문제 1: sequence order

Position Vectors $p_i \in \mathbb{R}^d$, for $i \in \{1,2,\ldots,T\}$

$$v_i = \tilde{v}_i + p_i$$
$$q_i = \tilde{q}_i + p_i$$
$$k_i = \tilde{k}_i + p_i$$



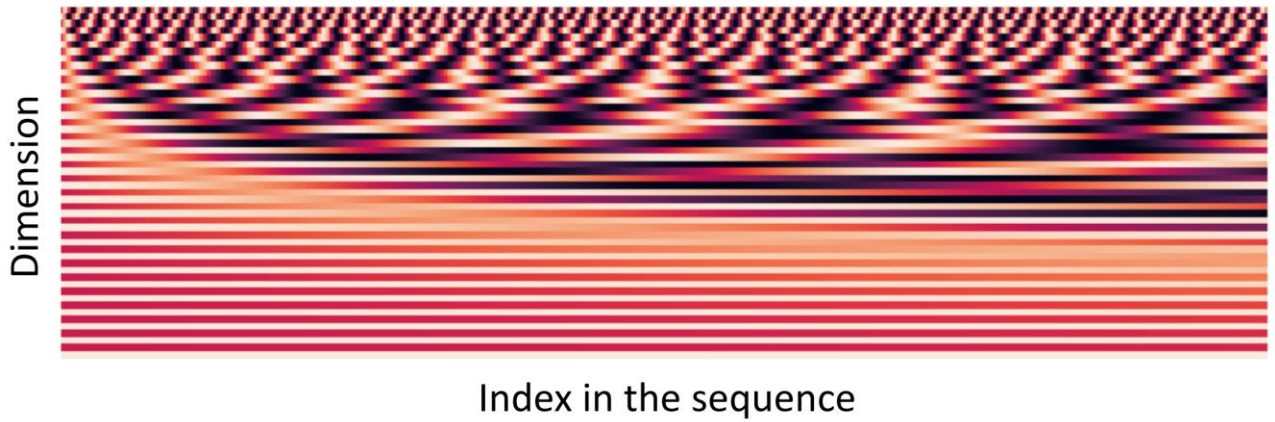
01 From recurrence (RNN) to attention-based NLP models

3) Barriers and solutions for Self-Attention as a building block

문제 1: sequence order

Sinusoidal position representations: concatenate sinusoidal functions of varying periods:

$$p_i = \begin{pmatrix} \sin(i/10000^{2*1/d}) \\ \cos(i/10000^{2*1/d}) \\ \vdots \\ \sin(i/10000^{2*\frac{d}{2}/d}) \\ \cos(i/10000^{2*\frac{d}{2}/d}) \end{pmatrix}$$



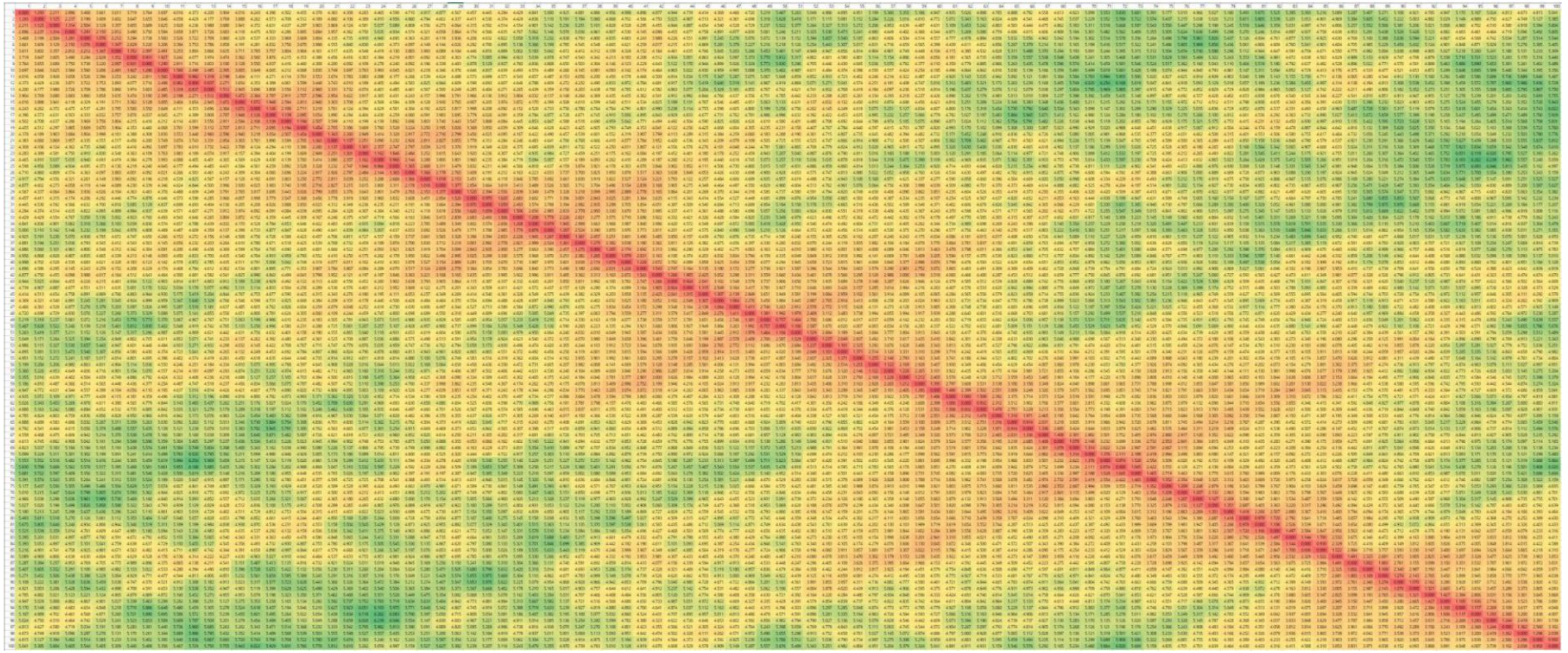
Pros: 절대적 위치 아닌 상대적인 비교가 중요하므로 함수주기 보다 sequence가 길어도 정보를 잃지 않는다.

Cons: 학습이 불가능 하다.

01 From recurrence (RNN) to attention-based NLP models

3) Barriers and solutions for Self-Attention as a building block

- The further the two positions, the larger the distance



01 From recurrence (RNN) to attention-based NLP models

3) Barriers and solutions for Self-Attention as a building block

• 학습가능한 position representations

matrix $p \in \mathbb{R}^{d \times T}$ 학습가능한 parameter p_i matrix의 열이 되도록 한다.

- Relative linear position attention [Shaw et al., 2018] : 상대적 위치

Model	Position Information	EN-DE BLEU	EN-FR BLEU
Transformer (base)	Absolute Position Representations	26.5	38.2
Transformer (base)	Relative Position Representations	**26.8**	**38.7**
Transformer (big)	Absolute Position Representations	27.9	41.2
Transformer (big)	Relative Position Representations	**29.2**	**41.5**

- Dependency syntax-based position [Wang et al., 2019] : 구조

Model Architecture	**Zh⇒En**					**En⇒De**
	MT03	MT04	MT05	MT06	Avg	WMT14
Hao et al. (2019c)	-	-	-	-	-	28.98
Transformer-Big	45.30	46.49	45.21	44.87	45.47	28.58
+ *Structural* PE	45.62	47.12⇑	45.84	45.64⇑	46.06	28.88
+ *Relative* Sequential PE	45.45	47.01	45.65	45.87⇑	46.00	28.90
+ *Structural* PE	45.85⇑	47.37⇑	46.20⇑	46.18⇑	46.40	29.19⇑

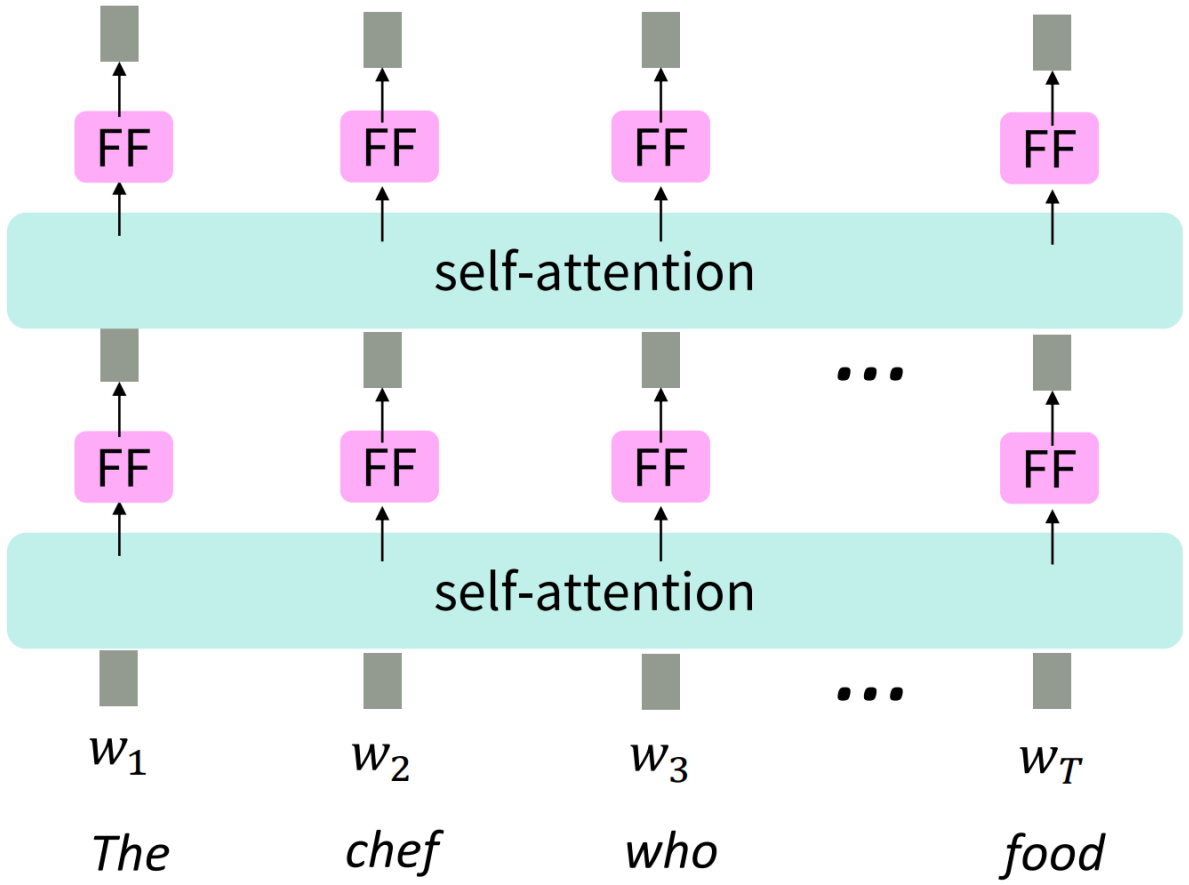
Pros: 현재 데이터에 맞게 학습이 가능하다.

Cons: 함수길이 보다 sequence가 길면 정보를 잃는다.

01 From recurrence (RNN) to attention-based NLP models

3) Barriers and solutions for Self-Attention as a building block

문제 2: nonlinearities in self-attention



$$m_i = MLP(\text{output}_i)$$
$$= W_2 * \text{ReLU}(W_1 \times \text{output}_i + b_1) + b_2$$

01 From recurrence (RNN) to attention-based NLP models

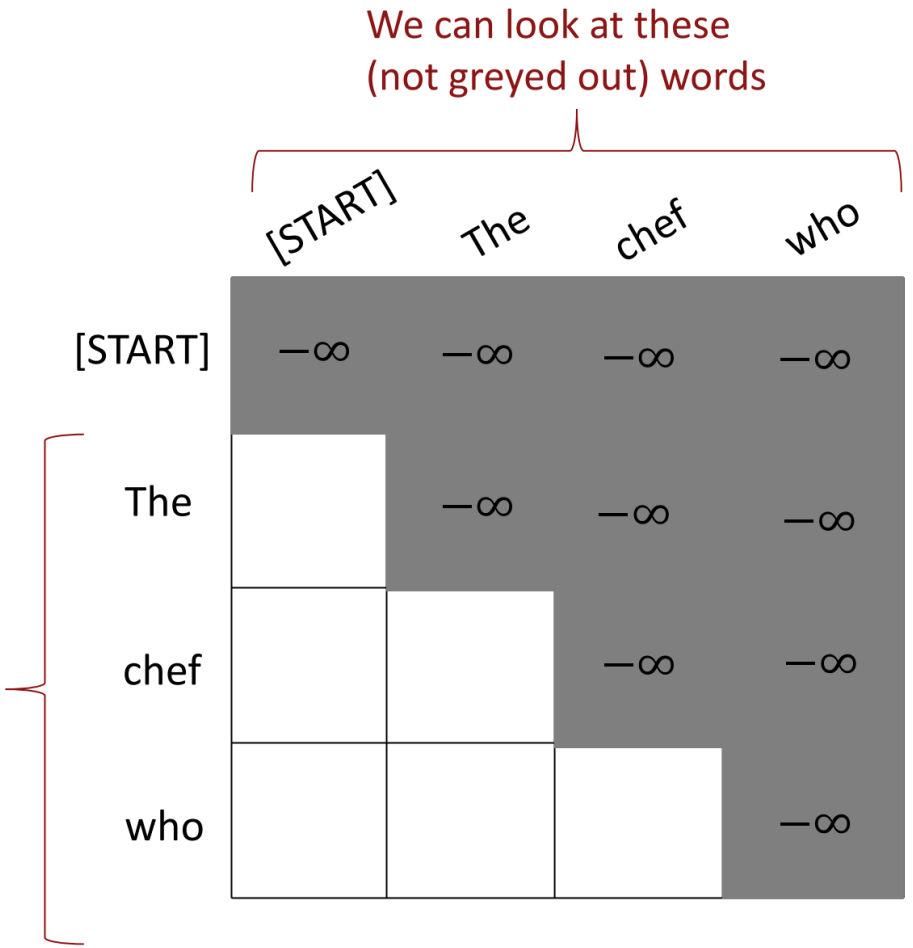
3) Barriers and solutions for Self-Attention as a building block

문제 3: decoder에서 미래 sequence 정보를 보지 못해야 한다

key index
: 나머지 word의 label

query index
: 지금 탐색중인 대상

$$e_{ij} = \begin{cases} q_i^\top k_j, j < i \\ -\infty, j \geq i \end{cases}$$



02 Introducing the Transformer model

Sub-content

1. Transformer model structure

2. Query, Key, Value for Transformer

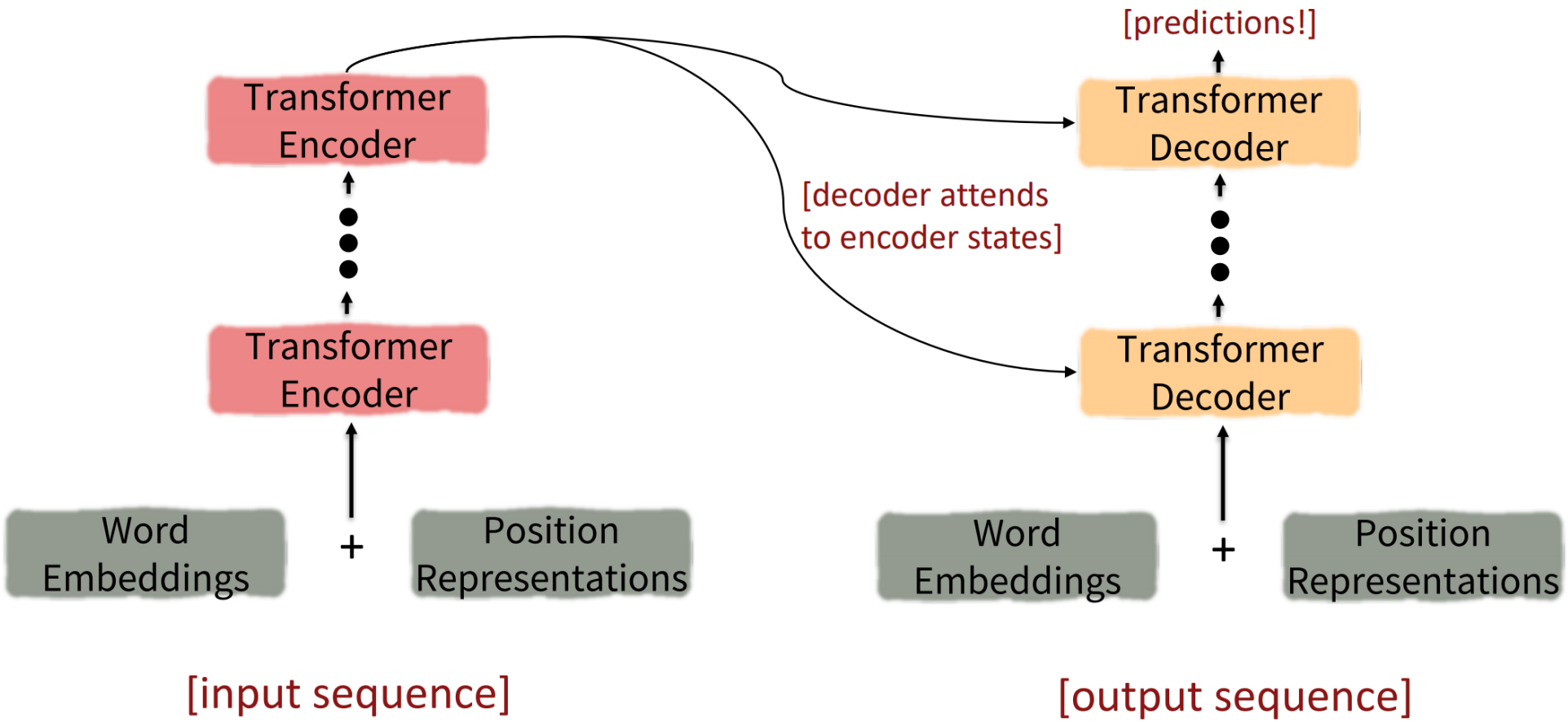
3. Multi-head attention

4. Tricks to help with training!
   - Residual connections
   - Layer normalization
   - Scaling the dot product

02 Introducing the Transformer model

Transformer model structure

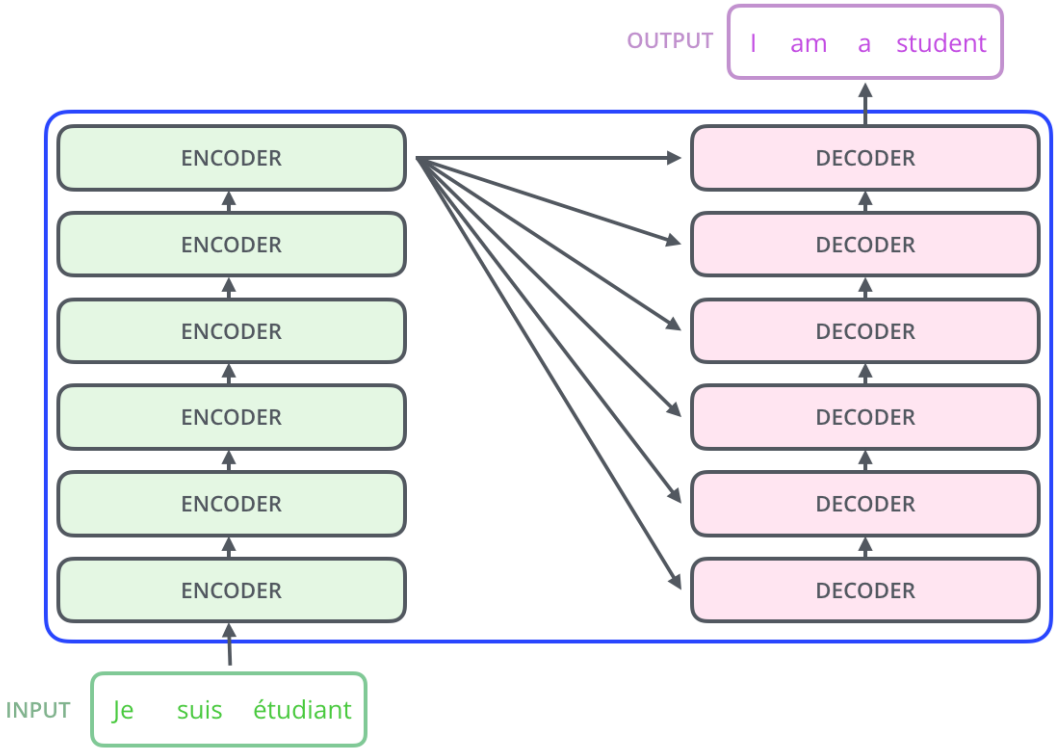
Transformer Encoder and Decoder

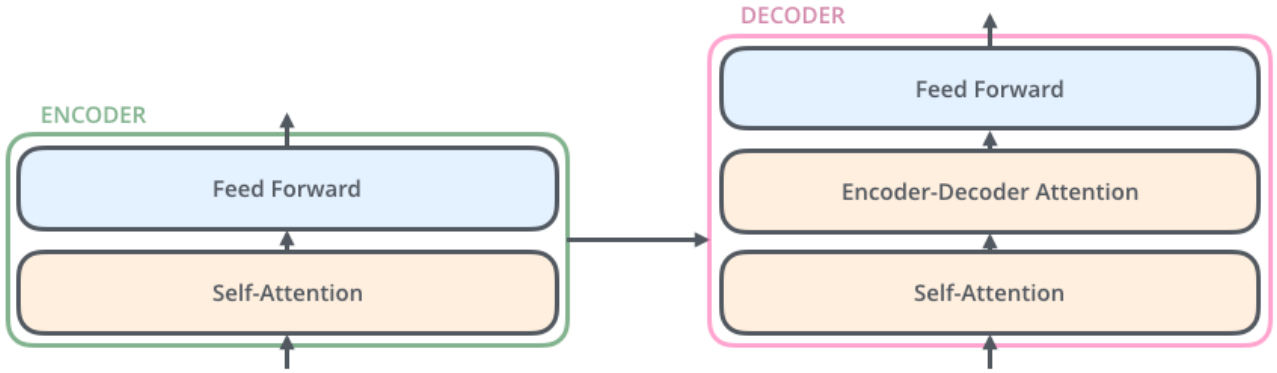


02 Introducing the Transformer model

Transformer model structure

Alamar (Transformer)





각 6개씩의 Encoder와 Decoder

Decoder에는 Cross-Attention 추가로 존재

02 Introducing the Transformer model

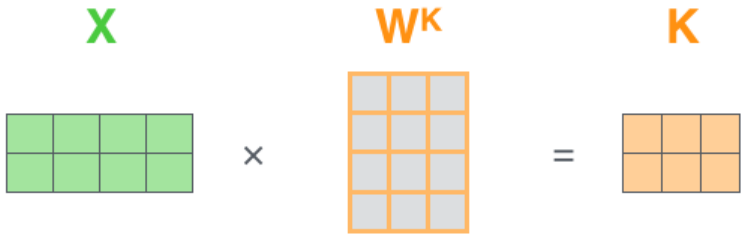
Query, Key, Value for Transformer

Let $x_1, \ldots, x_T$ be input vectors to the Transformer encoder; $x_i \in \mathbb{R}^d$

$k_i = Kx_i$, where $K \in \mathbb{R}^{d \times d}$ is the key matrix.

$q_i = Qx_i$, where $Q \in \mathbb{R}^{d \times d}$ is the query matrix.

$v_i = Vx_i$, where $V \in \mathbb{R}^{d \times d}$ is the value matrix.



Query, Key, Value가 조금씩 다르며 각각의 역할에 맞게 사용된다.

행렬 Q, K, V가 우리가 찾아야하는 미지수!

02 Introducing the Transformer model

Query, Key, Value for Transformer

- Input vector가 d dimension을 가질 때, 그를 결합한 행렬을 X라 하자.

$$X = [x_1; \ldots; x_T] \in \mathbb{R}^{T \times d}$$

- Query, Key, Value를 계산하기 위해 행렬 X와 행렬 K, Q, V를 dot product한다.

$$XK \in \mathbb{R}^{T \times d}, XQ \in \mathbb{R}^{T \times d}, XV \in \mathbb{R}^{T \times d}$$

- Attention score 계산을 위해 다음의 행렬 연산을 수행한다. $XQ(XK)^\top$

$$XQ \quad K^\top X^\top = XQK^\top X^\top \in \mathbb{R}^{T \times T}$$

02 Introducing the Transformer model

Query, Key, Value for Transformer

- Input vector가 d dimension을 가질 때, 그를 결합한 행렬을 X라 하자.

$$X = [x_1; \ldots; x_T] \in \mathbb{R}^{T \times d}$$

- Query, Key, Value를 계산하기 위해 행렬 X와 행렬 K, Q, V를 dot product한다.

$$XK \in \mathbb{R}^{T \times d}, XQ \in \mathbb{R}^{T \times d}, XV \in \mathbb{R}^{T \times d}$$

- Attention score 계산을 위해 다음의 행렬 연산을 수행한다. $XQ(XK)^\top$

$$XQ \quad K^\top X^\top = XQK^\top X^\top \in \mathbb{R}^{T \times T}$$

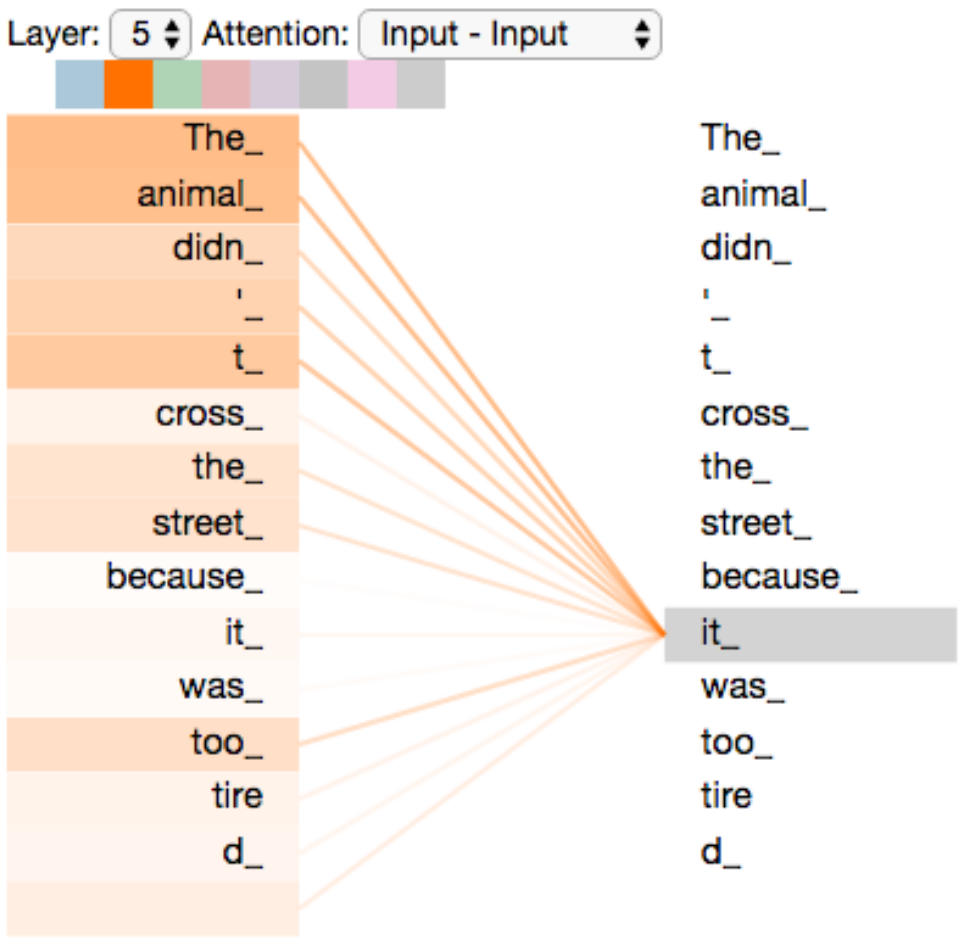
- Attention score 결과를 softmax를 통해 가중치를 얻고 그를 가중합하여 outpu을 얻는다.

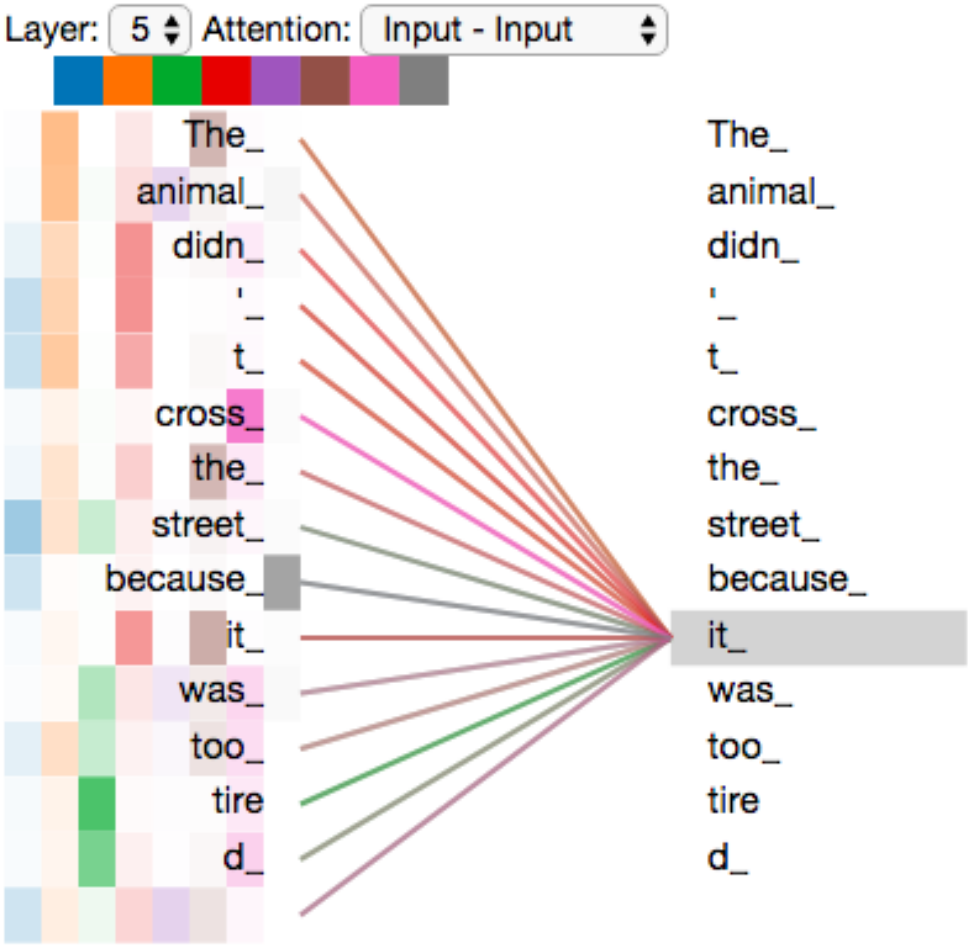
$$\text{softmax}\left(XQK^\top X^\top\right) XV = \quad \text{output} \in \mathbb{R}^{T \times d}$$

02 Introducing the Transformer model

Multi-headed attention

- 한번에 여러 부분에 집중하기 위해 여러 개의 attention head구성





02 Introducing the Transformer model

Multi-headed attention

1) This is our input sentence*

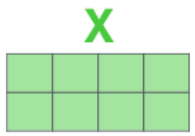
2) We embed each word*

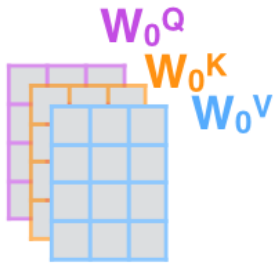
3) Split into 8 heads. We multiply X or R with weight matrices

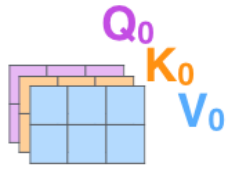
4) Calculate attention using the resulting Q/K/V matrices

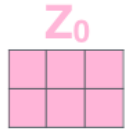
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

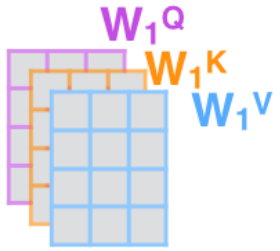
Thinking Machines

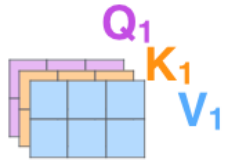


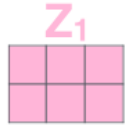


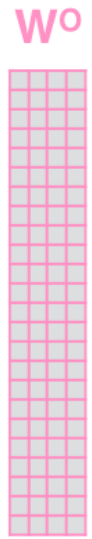


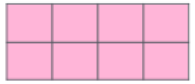




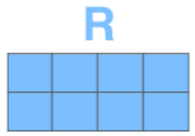


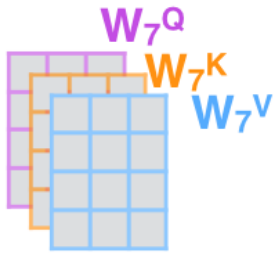


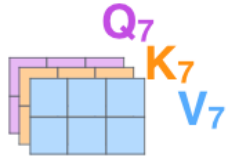


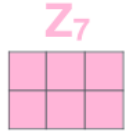


* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one









02 Introducing the Transformer model

Multi-headed attention

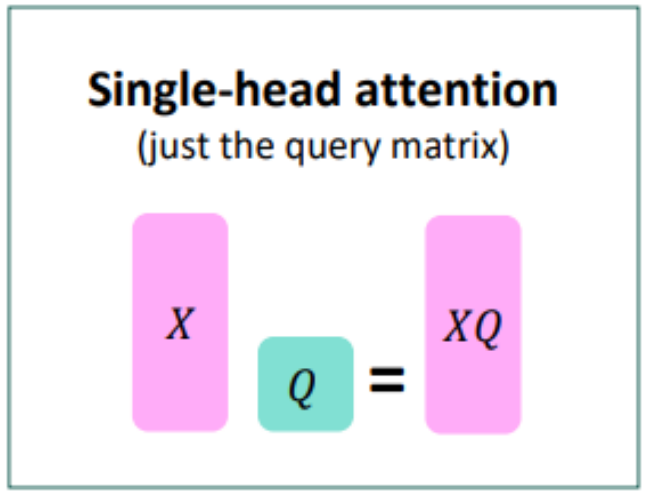
- 각각의 head는 다른 부분에 집중하고, 다른 value vector를 산출한다.

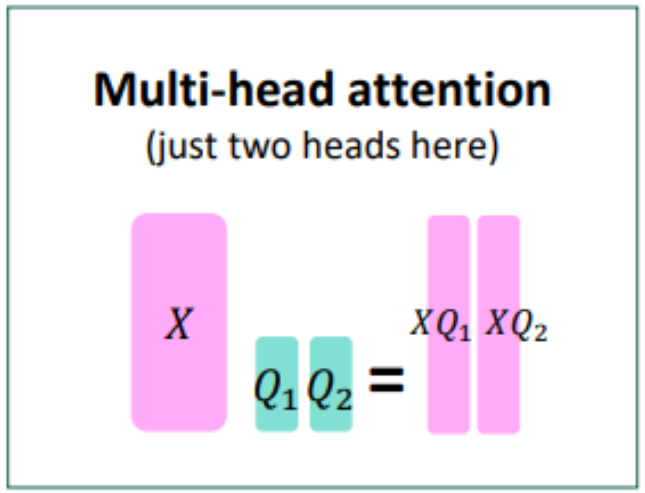
$Q_\ell, K_\ell, V_\ell \in \mathbb{R}^{d \times \frac{d}{h}}$ h 는 head의 개수, $\ell = 1,2,\cdots, h$

$$\text{output}_\ell = \text{softmax}\left(XQ_\ell K_\ell^\top X^\top\right) * XV_\ell, \text{ where } \text{output}_\ell \in \mathbb{R}^{d/h}$$

$$\text{output} = Y[\text{output}_1; \ldots; \text{output}_h], \text{ where } Y \in \mathbb{R}^{d \times d}$$

- Multi-head attention을 수행하여도 결국 input과 같은 크기의 output을 산출한다.
- Multi-head attention을 수행하여도 총 계산 량은 동일하다.





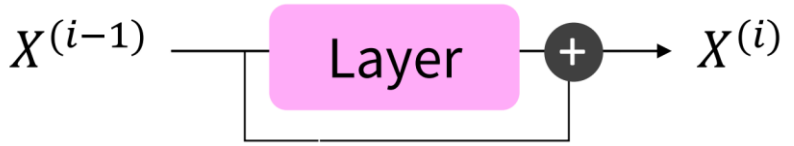
02 Introducing the Transformer model

Tricks to help with training! - Residual connection

- Residual connection: 자기 자신을 더해준다.

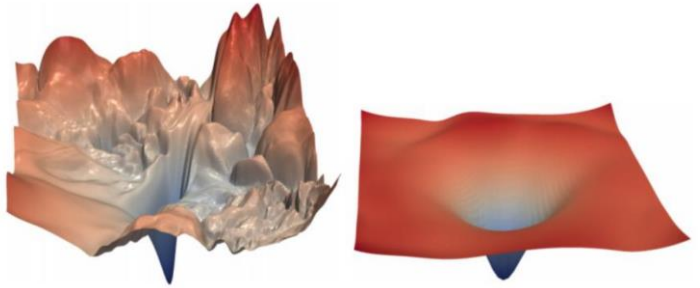
$$X^{(i)} = \text{Layer}(X^{(i-1)})$$

$$X^{(i)} = X^{(i-1)} + \text{Layer}(X^{(i-1)})$$



- 이전 시전보다 얼마나 변화했는지를 학습한다고도 해석해볼 수 있음

- New f(x) = old f(x)+x라고 표현할 때, 미분결과 f'(x)+1로 gradient가 아주 작아도 1만큼을 이어주는 효과가 있다.



[no residuals] [residuals]

[Loss landscape visualization, Li et al., 2018, on a ResNet]

02 Introducing the Transformer model

Tricks to help with training! - Layer normalization

- Layer 내에서 하나의 input sample x에 대해서 모든 feature에 대한 평균과 분산을 구해 normalization

Let $x \in \mathbb{R}^d$ be an individual (word) vector in the model.

Let $\mu = \sum_{j=1}^{d} x_j$; this is the mean; $\mu \in \mathbb{R}$.

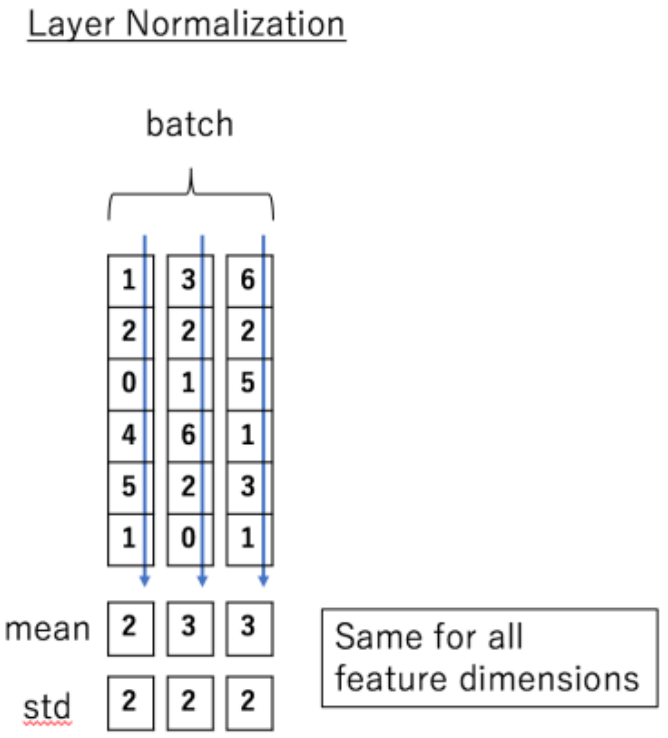
Let $\sigma = \sqrt{\frac{1}{d} \sum_{j=1}^{d} (x_j - \mu)^2}$; this is the standard deviation; $\sigma \in \mathbb{R}$.

Let $\gamma \in \mathbb{R}^d$ and $\beta \in \mathbb{R}^d$ be learned "gain" and "bias" parameters. (Can omit!)

$$\text{output} = \frac{x - \mu}{\sqrt{\sigma + \epsilon}} * \gamma + \beta$$

Normalize by scalar mean and variance

Modulate by learned elementwise gain and bias



https://mlexplained.com/2018/01/13/weight-normalization-and-layer-normalization-explained-normalization-in-deep-learning-part-2/

02 Introducing the Transformer model

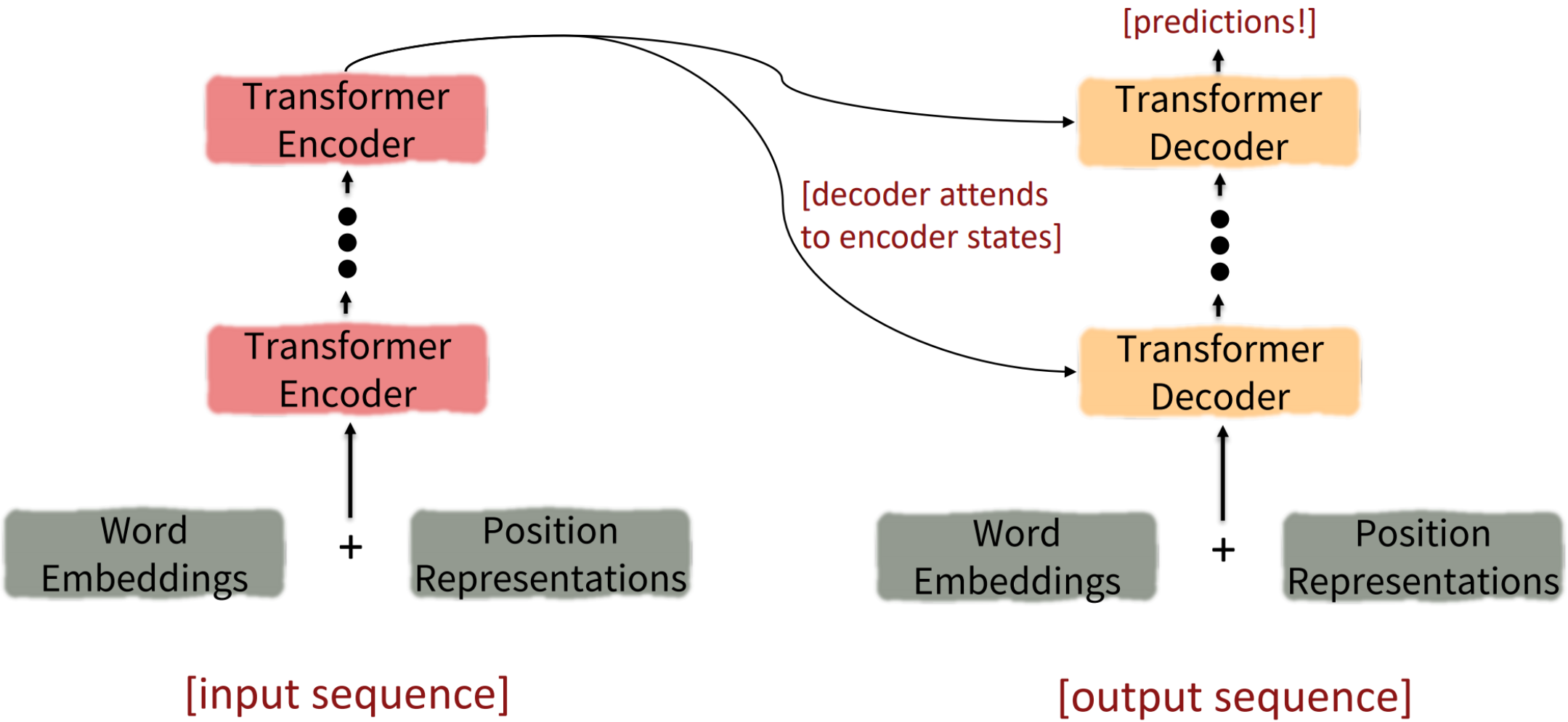
Tricks to help with training! - Scaling the dot product

- Attention score 좀 더 다양한 vector 들에게 분배

$$\text{output}_\ell = \text{softmax}\left(XQ_\ell K_\ell^\top X^\top\right) * XV_\ell \longrightarrow \text{output}_\ell = \text{softmax}\left(\frac{XQ_\ell K_\ell^\top X^\top}{\sqrt{d/h}}\right) * XV_\ell$$

02 Introducing the Transformer model

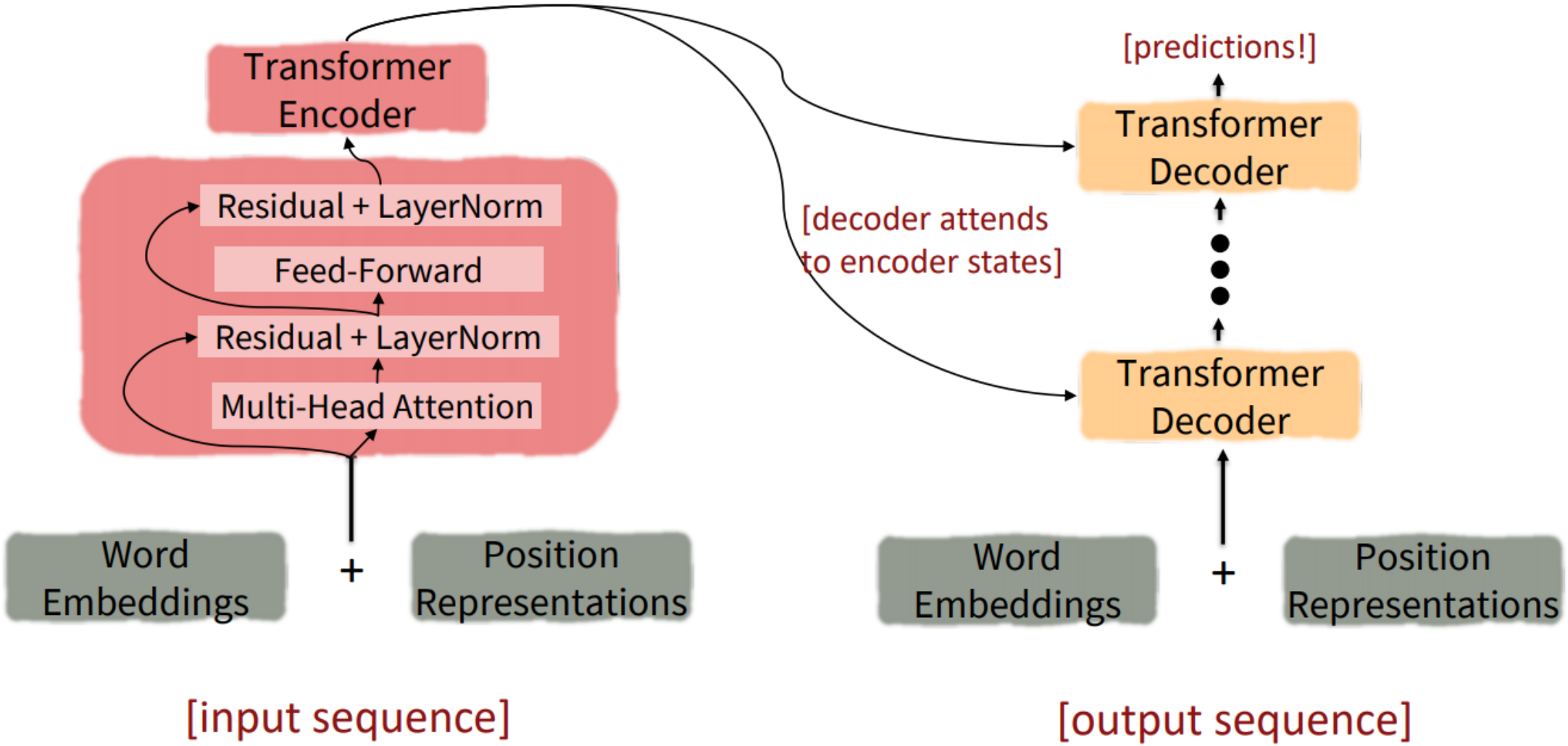
Encoder & Decoder Detail



02 Introducing the Transformer model

Encoder & Decoder Detail

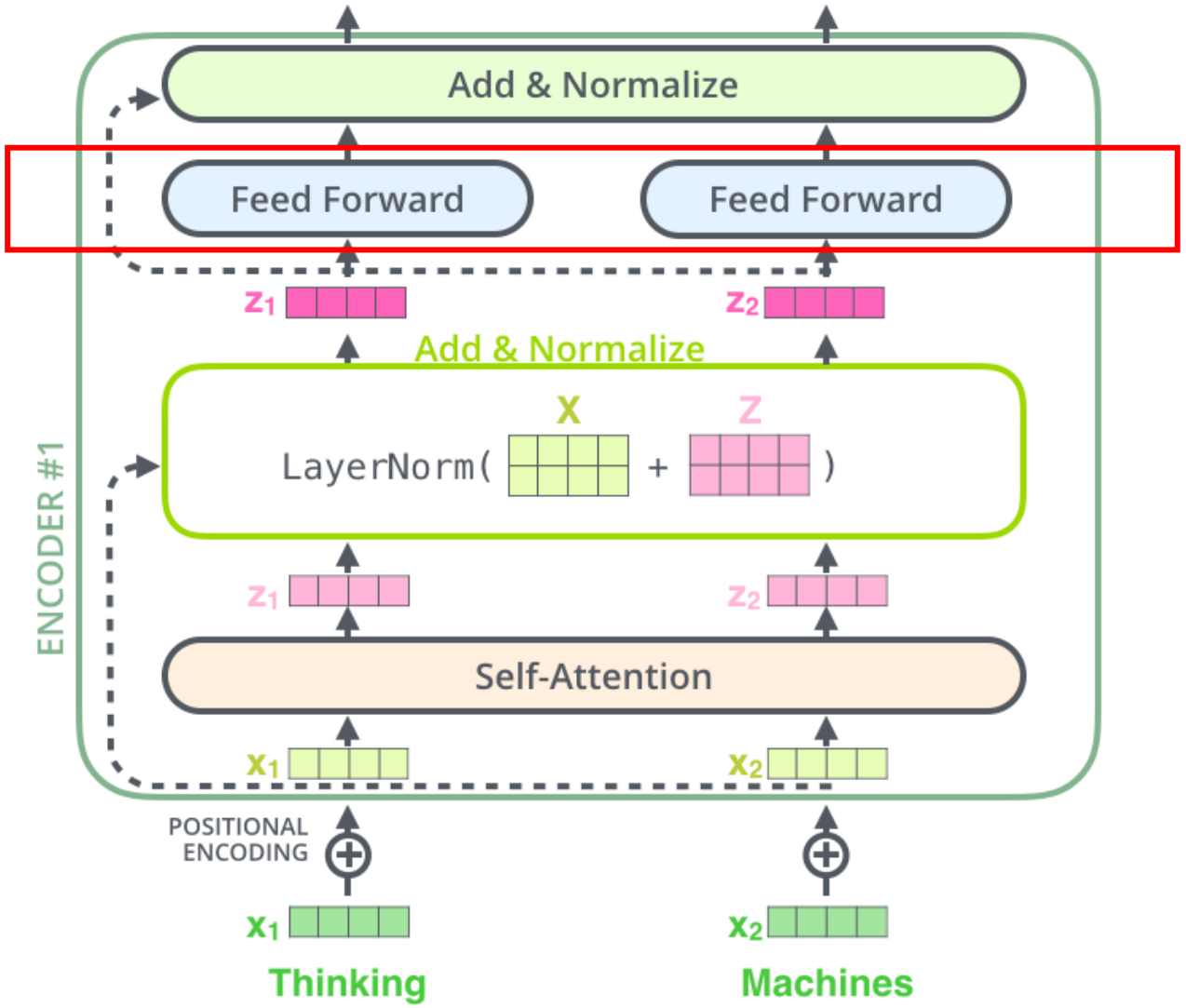
Looking back at the whole model, zooming in on an Encoder block:



02 Introducing the Transformer model

Encoder & Decoder Detail

Alamar (Transformer)



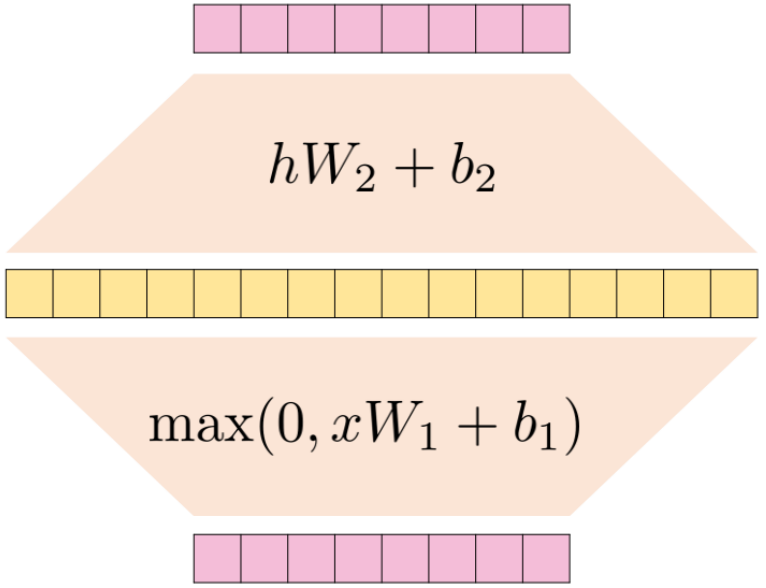
02 Introducing the Transformer model

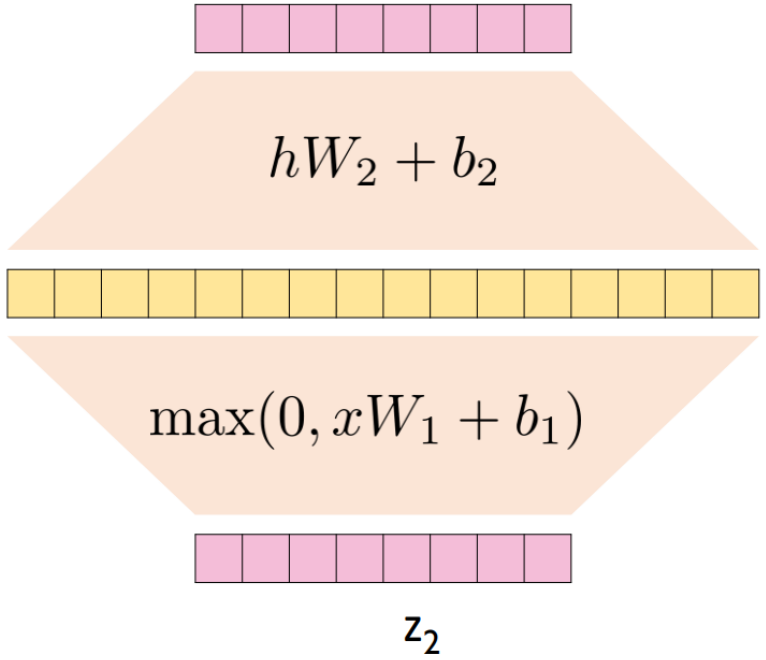
Encoder & Decoder Detail

Vaswani et. al (2017), Kim (2019)

- Position-wise Feed-Forward Networks
  - Fully connected feed-forward network
  - sequence 마다 독립적으로 적용된다.
  - 하나의 layer안에서는 동일한 weight parameter를 공유한다.
  - 다른 layer에서는 다른 weight를 사용한다.

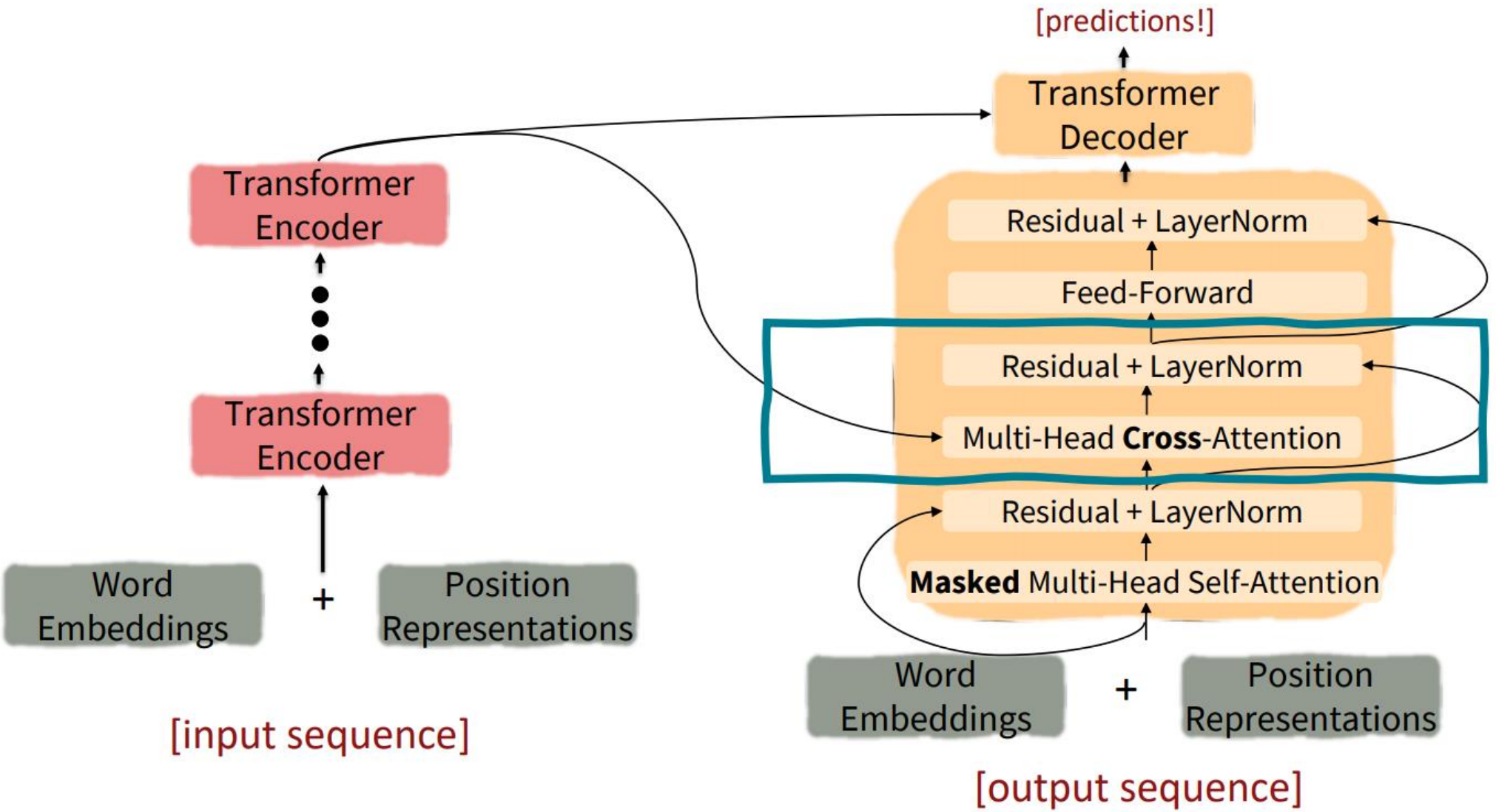
$$m_i = MLP(\text{output}_i)$$
$$= W_2 * \text{ReLU}(W_1 \times \text{output}_i + b_1) + b_2$$





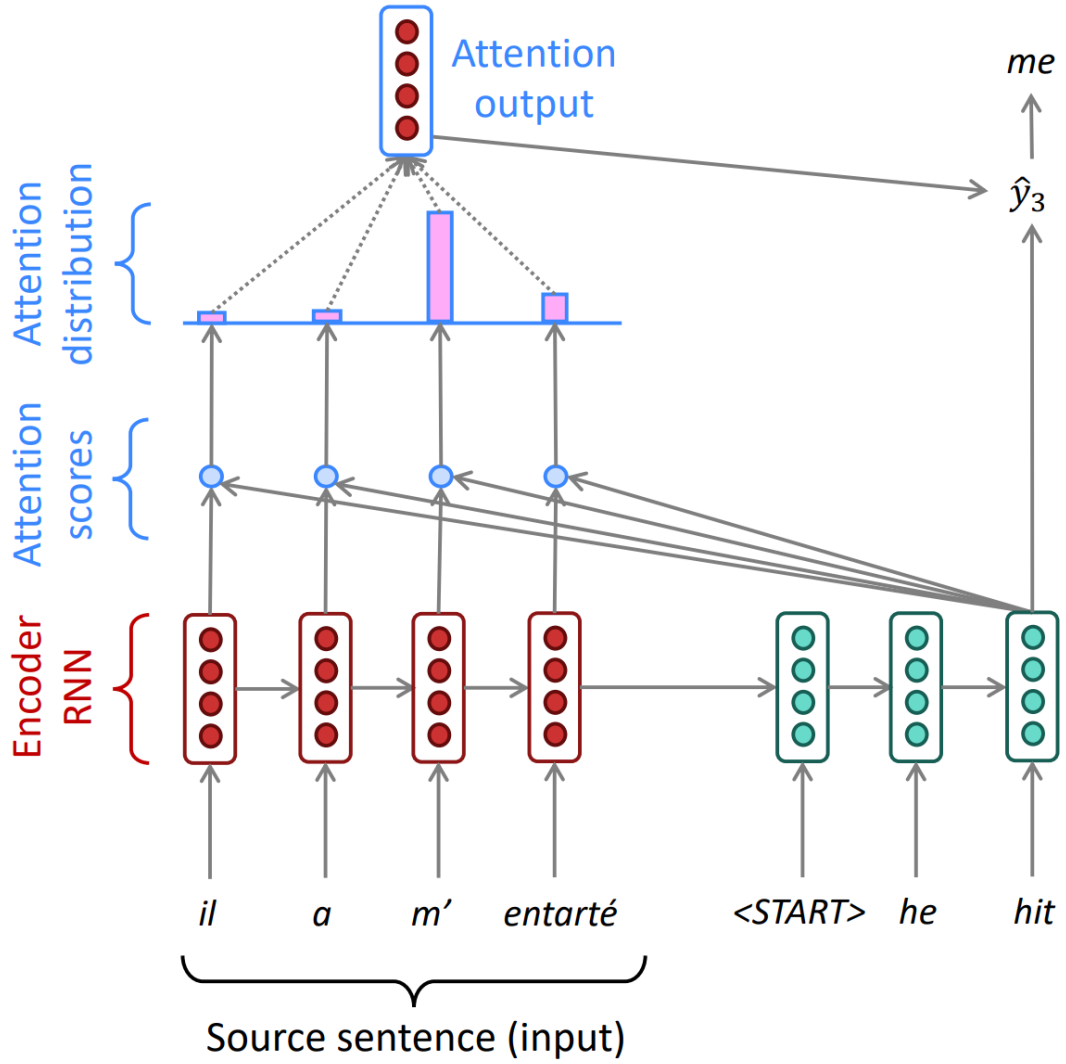
02 Introducing the Transformer model

Encoder & Decoder Detail



02 Introducing the Transformer model

Encoder & Decoder Detail



02 Introducing the Transformer model

Encoder & Decoder Detail

- h: encoder의 output vector
  z: decoder의 input vector

$$\text{Let } H = [h_1; \ldots; h_T] \in \mathbb{R}^{T \times d}$$
$$\text{Let } Z = [z_1; \ldots; z_T] \in \mathbb{R}^{T \times d}$$

- Decoder에서 현재 처리하는 단어의 Query를 가져오고, Encoder의 Key로 탐색한 결과를 Encoder의 Value로 가중 합한다.
- Attention score 계산을 위해 다음의 행렬 연산을 수행한다. $ZQ(HK)^\top$

$$ZQ \quad K^\top H^\top = ZQK^\top H^\top \in \mathbb{R}^{T \times T}$$

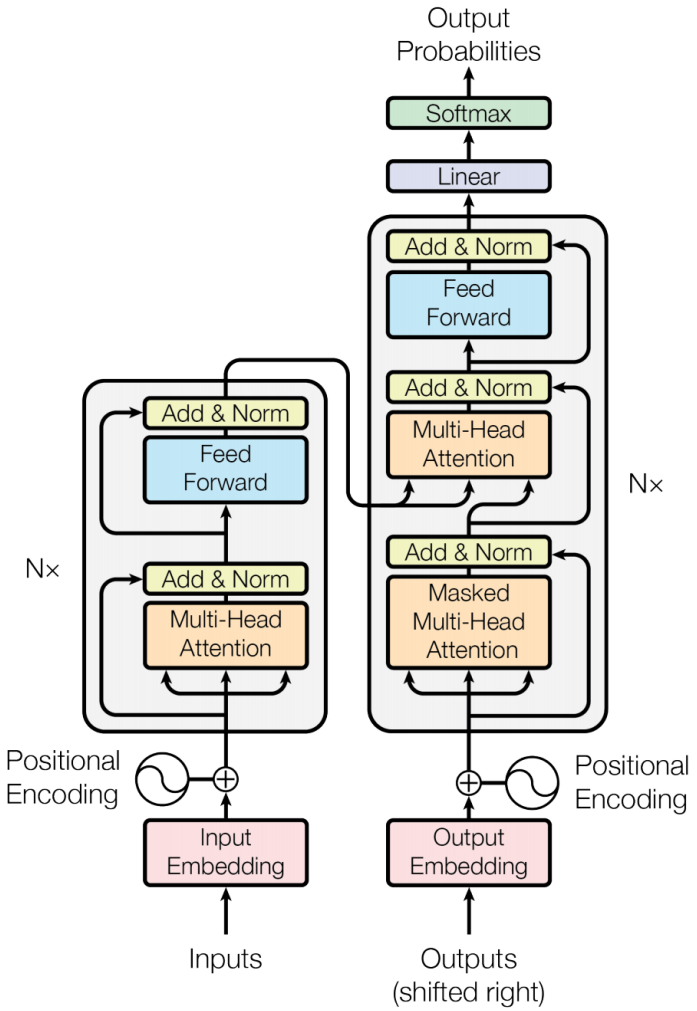
- Attention score 결과를 softmax를 통해 가중치를 얻고 그를 가중합하여 output을 얻는다.

$$\text{softmax}\left(ZQK^\top H^\top\right) HV = \text{output} \in \mathbb{R}^{T \times d}$$

02 Introducing the Transformer model

Transformer 총정리

강형원 선배님 Decision Transformer 발표자료 참고



- **Sequence-to-sequence 모델**
- **Self-Attention**
  - 단어 처리시 문장 내의 다른 단어들로 부터 힌트를 받아 현재 단어를 인코딩

- **Positional Encoding**
  - 단어들을 set으로 취급하여 한번에 행렬로 input하기 때문에 위치정보가 없음
  - 추가적인 position representation을 더해 positional encoding 수행

- **Encoder-Decoder 구조**
  - 각 Encoder와 Decoder를 동일한 개수 사용
  - Multi-head self-attention
  - Position-wise feed-forward network
  - Residual connection
  - Layer Normalization

03 Great Result with Transformers

우수한 성능

• Machine Translation

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**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$

[Test sets: WMT 2014 English-German and English-French]

[Vaswani et al., 2017]

03 Great Result with Transformers

우수한 성능

• Document generation

Model	Test perplexity	ROUGE-L
seq2seq-attention, $L = 500$	5.04952	12.7
Transformer-ED, $L = 500$	2.46645	34.2
Transformer-D, $L = 4000$	2.22216	33.6
Transformer-DMCA, no MoE-layer, $L = 11000$	2.05159	36.2
Transformer-DMCA, MoE-128, $L = 11000$	1.92871	37.9
Transformer-DMCA, MoE-256, $L = 7500$	1.90325	38.8

[Liu et al., 2018]; WikiSum dataset

04 Drawbacks and variants of Transformers

Drawbacks

- Quadratic compute in self-attention
  - sequence length가 증가함에 따라 연산 량이 2차식으로 증가 $O(T^2 d)$
  - d가 1000정도로 고정 되어있다고 할 때
    - 짧은 문장의 경우 $T \leq 30$; $T^2 \leq \mathbf{900}$로 연산이 가능하나
    - 긴 글의 경우 $\boldsymbol{T} \geq \mathbf{10,000}$라면 병렬처리가 가능하여도 문제가 된다.

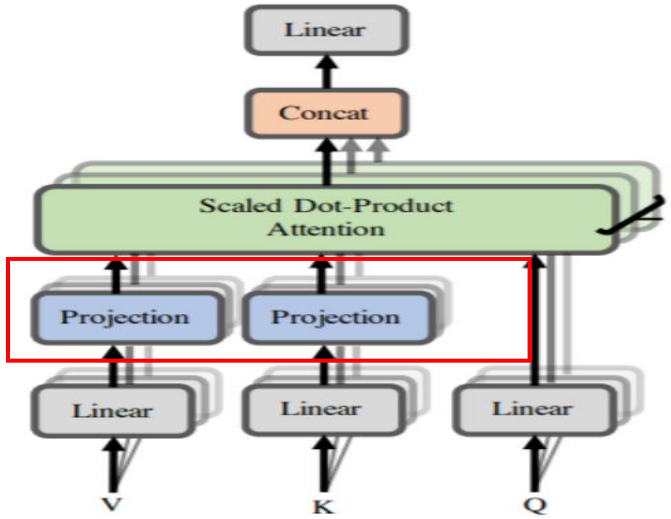
- Position representations
  - 논문에서 제시한 절대적인 position representations가 과연 최선일까?
    - Relative linear position attention [Shaw et al., 2018]
    - Dependency syntax-based position [Wang et al., 2019]

04 Drawbacks and variants of Transformers

Drawbacks

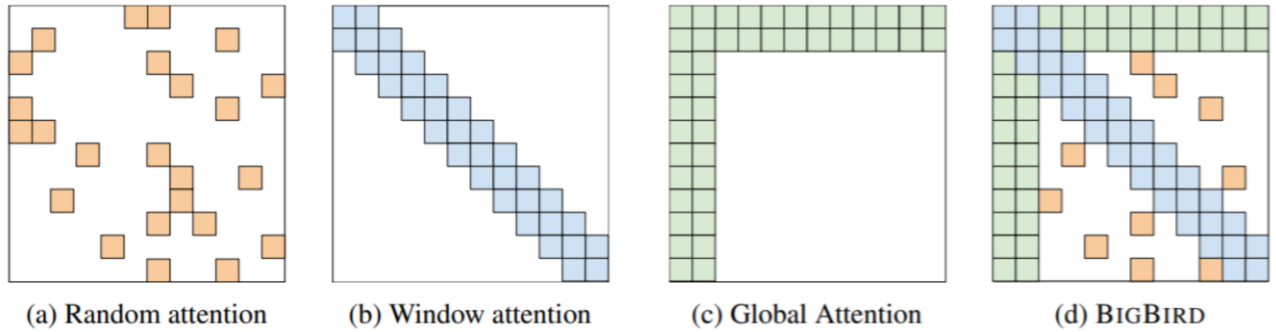
- Improving on Quadratic compute in self-attention Cost

- Linformer [Wang et al., 2020]



Key idea: projection을 통해 value와 key의 sequence length dimension을 낮춘다.

- BigBird [Zaheer et al., 2021]



Key idea: 모든 pair 사이의 attention을 계산하지 않고 window, Global, Random을 적절히 조합한 만큼만 계산한다.

감사합니다