

# Transformer

## Sequence to Sequence Model

Transformer 모델은 입력 시퀀스 (Context)를 기반으로 출력 시퀀스를 생성하는 모델입니다.

구성 요소

- Word Embedding**
  - 단어를 임베딩 ID로 변환 (512차원)
  - PyTorch: `nn.Embedding`
- Positional Encoding**
  - Transformer는 RNN과 달리 순서를 모르므로 위치 정보를 추가
  - 구현 방법
    - 정현 코사인 (sin)
    - 정현 사인 (cos)

## Encoder

Encoder는 입력 시퀀스를 block 단위로 6번 처리

각 block은 다음을 포함

- Multi-Head Self Attention**
  - 단어 간의 관계를 학습 (자기 자신에 대한 attention)
- Feed Forward Network**
  - MLP (Multi-Layer Perceptron)
  - 비선형 활성화 함수

추가 구성 요소

- Dropout
- Layer Normalization**

## Decoder

Decoder는 다음을 포함

- Encoder의 출력을 attention
- Auto-regressive** (자기 회귀)

Decoder 구조

Decoder는 3개의 block + LayerNorm으로 구성

- Masked Multi-Head Self-Attention (자기 자신에 대한 attention)
- Encoder-Decoder Attention (Encoder의 출력을 attention)

- Masked Multi-Head Self-Attention**
  - Decoder의 이전 출력을 attention
  - Decoder의 현재 출력을 attention
- Encoder-Decoder Attention**
  - Decoder가 Encoder의 출력을 attention



## Decoder

- 0000000000000000 <B0S> 000000
- 000000 000000000000“0”000000
- 0000000000“0 0”
- ...00000

□ □

## Teacher Forcing

## Residual Connection

□□□□

`"00" "0" "0000" "0000000000x0000F(x)000000000000x + F(x)`

□□□□□□

- 如何求取  $y = F(x) + x$  的梯度
- 如何求取  $\partial L / \partial x$  的梯度

$$\frac{\frac{\partial L}{\partial y}}{\frac{\partial L}{\partial x}} = \frac{\frac{\partial L}{\partial y}}{\frac{\partial F(x)}{\partial x} + 1}$$

- $\frac{\partial F(x)}{\partial x}$  1

## Layer Normalization

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

- **BatchNorm** 对mini-batch中所有feature的dimension进行归一化
- NLP 需要对BatchNorm 对mini-batch中所有batch 进行归一化
- **Layer Normalization**对每个feature的dimension进行归一化 **Batch Normalization**对每个feature的dimension进行归一化

### Explanation

self-attention

5/5



 image-20250427121123206

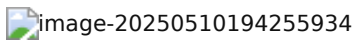
- \$a^1,a^2,a^3,a^4\$
- 

$$\begin{aligned} Q &= \begin{bmatrix} q^1 & q^2 & q^3 & q^4 \end{bmatrix} \ \&= \begin{bmatrix} a^1 & a^2 & a^3 & a^4 \end{bmatrix} W^q \ [1em] \ K = \begin{bmatrix} k^1 & k^2 & k^3 & k^4 \end{bmatrix} \ \&= \begin{bmatrix} a^1 & a^2 & a^3 & a^4 \end{bmatrix} W^k \ [1em] \ V = \begin{bmatrix} v^1 & v^2 & v^3 & v^4 \end{bmatrix} \ \&= \begin{bmatrix} a^1 & a^2 & a^3 & a^4 \end{bmatrix} W^v \end{aligned}$$

- $$2. \quad \mathbf{A} = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} \\ \alpha_{4,1} & \alpha_{4,2} & \alpha_{4,3} & \alpha_{4,4} \end{bmatrix} = \mathbf{Q} \cdot \mathbf{K}^T =$$

3.  $\sqrt{d_k}$  **softmax**  $d_k$   $key/query$

4.

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$
[illegible]

- $\text{aaaaaa} \$a^i, a^j, a^m, a^n \$ \text{aaaa} \$W^q W^k W^v \$ \text{aaaaaaaaaaaaaaaaaaaa} \$Q^K V \$$
- $\text{aaa} \$W^q W^k W^v \$ \text{aaaaaa}$

**•** □□□□□□□□□□

- ☐ ☐ ☐ ☐ ☐ ☐

$$K_1 = \begin{bmatrix} k^{1,1} & k^{1,j} & k^{1,m} & k^{1,n} \\ k^{i,2} & k^{j,2} & k^{m,2} & k^{n,2} \end{bmatrix} \quad K_2 = \begin{bmatrix} k^{1,1} & k^{1,j} & k^{1,m} & k^{1,n} \\ k^{i,2} & k^{j,2} & k^{m,2} & k^{n,2} \end{bmatrix}$$

$$V_1 = \begin{bmatrix} v^{1,1} & v^{2,1} & v^{m,1} & v^{n,1} \\ v^{1,2} & v^{2,2} & v^{m,2} & v^{n,2} \end{bmatrix} \quad V_2 = \begin{bmatrix} v^{1,1} & v^{2,1} & v^{m,1} & v^{n,1} \\ v^{1,2} & v^{2,2} & v^{m,2} & v^{n,2} \end{bmatrix}$$

- 计算点积

$$\text{head}_1 = \text{softmax}\left(\frac{Q_1 K_1^T}{\sqrt{d_k}}\right) V_1 \quad \text{head}_2 = \text{softmax}\left(\frac{Q_2 K_2^T}{\sqrt{d_k}}\right) V_2$$

- 拼接

$$\text{multihead} = \text{head}_1 \parallel \text{head}_2$$

- 输出

$$\text{output} = \text{multihead} \cdot W^O$$

## Self-attention

计算点积

计算点积

### Sequence Labeling

计算点积

- 计算点积

image-20250418130432286

计算点积

计算点积

计算点积

- 计算点积
- 计算点积
- 计算点积

计算点积

- 计算点积

- 计算点积

- $Q = X \times W^Q$
- $K = X \times W^K$
- $V = X \times W^V$

计算点积

- 计算点积

- dot product

- $Q \cdot K^T$
- $K \cdot V^T$

- $QK^T$ 의 행렬 곱셈 결과에 softmax를 적용

- $QK^T$ 의 행렬 곱셈 결과에 softmax를 적용하여 attention score를 계산
- $\text{softmax}(QK^T)$ 의 결과에  $V$ 를 곱하여 attention score를 계산
- $\text{softmax}(QK^T)$ 의 결과에  $V$ 를 곱하여 attention score를 계산
  - $QK^T$ 의 행렬 곱셈 결과에 softmax를 적용
  - $QK^T$ 의 행렬 곱셈 결과에 softmax를 적용
  - $V$ 를 곱하여 attention score를 계산

## Multi-head Self-attention

Multi-head Self-attention은 다음과 같이 계산된다.

- $a^i$ 를 계산
  - $b^{i,1}$ 
    - $q^{i,1}k^{i,1}k^{j,1}$ 의 곱셈 결과에  $b^{i,1}$ 를 곱
  - $b^{i,2}$ 
    - $q^{i,2}k^{i,2}k^{j,2}$ 의 곱셈 결과에  $b^{i,2}$ 를 곱
  - $b^i$ 
    - $b^{i,1}$ 과  $b^{i,2}$ 의 곱셈 결과에  $b^i$ 를 곱하여 attention score를 계산
- $a^i$ 를 계산



image-20250420112828589

## Masked Multi-Head Self-Attention

Masked Multi-Head Self-Attention은 다음과 같이 계산된다.

Masked Multi-Head Self-Attention은 다음과 같이 계산된다. "A B C D" token을 사용하여 계산한다.

Masked Multi-Head Self-Attention은 다음과 같이 계산된다.  $\begin{bmatrix} 1 & \text{masked} & \text{masked} & \text{masked} \\ 1 & 1 & \text{masked} & \text{masked} \\ \text{masked} & 1 & 1 & \text{masked} \\ \text{masked} & 1 & 1 & 1 \end{bmatrix}$ 를 사용하여 계산한다.



image-20250702191639250

## Word Embedding

Word Embedding은 One-Hot Encoding을 사용하여 계산된다.

- One-Hot Encoding을 사용하여 계산
- One-Hot Encoding을 사용하여 계산

## Positional Encoding

- Self-attention을 사용하여 계산
- Self-attention을 사용하여 계산

Positional Encoding은 다음과 같이 계산된다.

1. Positional vector를 사용하여 계산
2. Positional vector를 사용하여 계산

$$\text{Final\_embedding} = \text{Token\_embedding} + \text{Positional\_encoding}$$

000000

□□□□□□□□□□□□□□

□□□□□□□□□□

111

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

1.  $\$pos\$d_{\{model\}}$
  2.  $\$2i\$d_{\{model\}}\$2i\backslash e_{d_{\{model\}}}$
  3.  $\$d_{\{model\}}\$d_{\{model\}}\$d_{\{model\}}$
- I am a kid
    - I pos 0
      - am pos 1
    - am pos = 1
      - $i = 0$ 
        - $\sin(1/10000^{0/d_{\{model\}}})$
        - $\cos(1/10000^{0/d_{\{model\}}})$
      - $i = 1$ 
        - $\sin(1/10000^{2/d_{\{model\}}})$
        - $\cos(1/10000^{2/d_{\{model\}}})$

□ □

- [illegible]

□ □ □ □ □ □ □ □

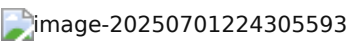
$$\underbrace{\begin{bmatrix} \text{PE}_{(\text{pos} + \Delta, 2i)} & \text{PE}_{(\text{pos} + \Delta, 2i+1)} \end{bmatrix}}_{\{ \text{pos} + \Delta \}}$$

$$\underbrace{\begin{bmatrix} \cos(\Delta \theta_i) & \sin(\Delta \theta_i) \\ -\sin(\Delta \theta_i) & \cos(\Delta \theta_i) \end{bmatrix}}_{\text{}} \underbrace{\begin{bmatrix} PE_{\text{pos}, 2i} \\ PE_{\text{pos}, 2i+1} \end{bmatrix}}_{\text{}} \begin{bmatrix} \text{pos} \\ \text{pos} \end{bmatrix}$$

$$\Delta \theta_i = \frac{1}{10000^{\frac{2i}{d_{\text{model}}}}}$$

$$\underbrace{\begin{bmatrix} \cos(\Delta \theta_i) & \sin(\Delta \theta_i) \\ -\sin(\Delta \theta_i) & \cos(\Delta \theta_i) \end{bmatrix}}_{\text{rot}(\Delta \theta_i)}$$

$$\Delta \theta_i \text{pos}$$



$$\begin{aligned} & \begin{bmatrix} \text{PE}_{(\text{pos} + \Delta, 2i)} \\ \text{PE}_{(\text{pos} + \Delta, 2i+1)} \end{bmatrix} = \begin{bmatrix} \sin((\text{pos} + \Delta) \cdot \theta_i) \cos((\text{pos} + \Delta) \cdot \theta_i) \\ \sin((\text{pos} + \Delta) \cdot \theta_i) \sin((\text{pos} + \Delta) \cdot \theta_i) \end{bmatrix} \\ & \begin{bmatrix} \sin(\text{pos} \cdot \theta_i) \cos(\Delta \cdot \theta_i) + \cos(\text{pos} \cdot \theta_i) \sin(\Delta \cdot \theta_i) \\ \cos(\text{pos} \cdot \theta_i) \cos(\Delta \cdot \theta_i) - \sin(\text{pos} \cdot \theta_i) \sin(\Delta \cdot \theta_i) \end{bmatrix} \\ & \begin{bmatrix} \cos(\Delta \theta_i) & \sin(\Delta \theta_i) \\ -\sin(\Delta \theta_i) & \cos(\Delta \theta_i) \end{bmatrix} \begin{bmatrix} \sin(\text{pos} \cdot \theta_i) \cos(\text{pos} \cdot \theta_i) \\ \sin(\text{pos} \cdot \theta_i) \sin(\text{pos} \cdot \theta_i) \end{bmatrix} \\ & \begin{bmatrix} \cos(\Delta \theta_i) & \sin(\Delta \theta_i) \\ -\sin(\Delta \theta_i) & \cos(\Delta \theta_i) \end{bmatrix} \begin{bmatrix} \text{PE}_{(\text{pos}, 2i)} \\ \text{PE}_{(\text{pos}, 2i+1)} \end{bmatrix} \end{aligned}$$

# Transformer

$$\begin{aligned} w_i &= 10000^{\frac{2i}{d_{\text{model}}}} \theta_i = \text{pos} / 10000^{\frac{2i}{d_{\text{model}}}} = \text{pos} / w_i \\ & \end{aligned}$$

- $w_i \theta_i \text{pos}$
- $w_i \theta_i \text{pos} \Delta$   
 $\text{pos} = 1000 \frac{1000}{10000} = 0.1$
- $w_i \sin(\text{pos} / 10000) \text{pos} 20000 \pi$

- $QK^T$
- $QK^T$

$$\begin{aligned} q_i &= W_q(x_i + p_i) \quad k_j = W_k(x_j + p_j) \\ q_i^T k_j &= (x_i + p_i)^T W_q^T W_k (x_j + p_j) = \underbrace{x_i^T W_q^T W_k x_j}_{\text{}} + \underbrace{x_i^T W_q^T W_k p_j + p_i^T W_q^T W_k x_j}_{\text{}} + \underbrace{p_i^T W_q^T W_k p_j}_{\text{}} \end{aligned}$$

$$\begin{aligned} q_i^T k_j &= x_i^T W_q^T W_k x_j + \underbrace{p_i^T W_q^T W_k p_j}_{\text{}} \quad \beta_{i-j} = x_i^T W_q^T W_k x_j \\ & + \underbrace{\beta_{i-j}}_{\text{}} \quad \beta_{i-j} \text{head}_h \beta_{i-j}^h \end{aligned}$$

# T5

# ALibi



QK\$



$m_h = 2^{\frac{8 \times h}{n_{head}}}$   
 $m = \begin{bmatrix} 2^{\frac{8 \times 1}{n_{head}}} & 2^{\frac{8 \times 2}{n_{head}}} & \cdots & 2^{\frac{8 \times n_{head}}{n_{head}}} \end{bmatrix}$   
 $D_{ij} = -(i - j)$   
 $\text{softmax}(q_i K^T + m \cdot [- (i - 1), \dots, -2, -1, 0])$

RoPE

QK

Attention\_score =  $q_m \cdot k_n^T$   
 $q^{\text{rot}} = q_m \cdot R_m$   
 $k^{\text{rot}} = k_n \cdot R_n$   
 $\begin{aligned} q^{\text{rot}} \cdot k^{\text{rot}} &= q_m R_m R_n^T k_n^T = q_m R_{m-n} k_n^T \end{aligned}$   
RoPE

1.

2.

$m, n$   $m - n$

- $d$ RoPE $d/2$  $d$

$R_{\theta, m} = \begin{bmatrix} \cos \theta_1 & -\sin \theta_1 & 0 & \cdots & 0 \\ \sin \theta_1 & \cos \theta_1 & 0 & \cdots & 0 \\ 0 & 0 & \cos \theta_2 & -\sin \theta_2 & 0 \\ 0 & 0 & \sin \theta_2 & \cos \theta_2 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \end{bmatrix}$

- 

$\begin{bmatrix} x_i \\ x_{i+1} \end{bmatrix}$

$\begin{bmatrix} \cos \theta_i & -\sin \theta_i \\ \sin \theta_i & \cos \theta_i \end{bmatrix}$   
 $\begin{bmatrix} x_i \\ x_{i+1} \end{bmatrix}$

$(m - n)$

$\theta_i = \frac{1}{10000^{\frac{2i}{d_{model}}}}$   
 $R_{m-n} = \begin{bmatrix} \cos((m - n)\theta_i) & \sin((m - n)\theta_i) \\ -\sin((m - n)\theta_i) & \cos((m - n)\theta_i) \end{bmatrix}$   
 $R_n^T R_m = R_{m-n}$   
 $R_{-m} = R^T_m$

$$R_{nq}^T R_{mk} = q^T R_{\{m-n\}k}$$

$$q_n^{\text{rot}} = R_{nq}^T k_m^{\text{rot}} = R_{mk}$$

$$Q^T K$$

**KV Cache**

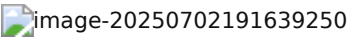
- $Q$
- $K$
- $Q^T K^T V$
- $V$

token  $K^T V$

- $Q$
- $K$
- $V$
- $V$

KV Cache “”  $K^T V$

**KV Cache**



**KV Cache**



**Pre-NormPost-Norm**

**Pre-Norm** **Post-Norm** Transformer LN Layer Normalization

- 
- Transformer\*\* (warm-up)\*\*
-