2022학년도 1학기 컴퓨터언어학

22강 셀프 어텐션 계층과 트랜스포머 소개

박수지

서울대학교 인문대학 언어학과

2022년 5월 25일 수요일

오늘의 목표

1. Attention 점수 계산을 일반화하여 query, key, value에 관한 식으로 표현할 수 있다.
2. 셀프 어텐션 계층(self-attention layer)에서 입력 시퀀스를 처리하는 과정을 설명할 수 있다.
3. 잔차 연결(residual connection)이 무엇인지 설명할 수 있다.
4. 레이어 정규화(layer normalization)의 목적과 방법을 설명할 수 있다.


he hit me with a pie
il α11 α21 α31 α41 α51 α61
a α12 α22 α32 α42 α52 α62
m’ α13 α23 α33 α43 α53 α63
entarté α14 α24 α34 α44 α54 α64

복습

부호화기-복호화기 모형에서의 attention

도착어 문장의 현재 단계에서 출발어 문장의 각 토큰을 얼마나 주목할 것인가?

복호화기의 직전 단계 은닉 상태 $\vec{h}^{d}_{i-1} \in \mathbb{R}^{d_h}$

부호화기의 모든 단계 은닉 상태 $\vec{h}^{e}_{j} \in \mathbb{R}^{d_h}\ (j = 1, 2, \cdots, n)$

Attention 점수 $a_{ij} = \text{score}\left(\vec{h}^{d}_{i-1}, \vec{h}^{e}_{j}\right) = \vec{h}^{d}_{i-1} \cdot \vec{h}^{e}_{j} \in \mathbb{R}$

Attention 가중치 $\alpha_{ij} = \text{softmax}\left(a_{ij}\right) \in]0, 1[\quad (j = 1, 2, \cdots, n)$

문맥 벡터 $\vec{c}_i = \sum_{j=1}^{n} \alpha_{ij} \vec{h}^{e}_{j} \in \mathbb{R}^{d_h}$

Causal (or backward looking) self-attention layer

$(\vec{x}_1, \cdots, \vec{x}_n) \mapsto (\vec{y}_1, \cdots, \vec{y}_n)$

1. 입력 시퀀스의 각 토큰을 처리할 때 현재 토큰 및 이전의 모든 토큰에 접근한다. (이후의 토큰은 고려하지 않는다.)
   $\Rightarrow$ 자기회귀적인(autoregressive) 방법으로 언어 모형을 만들 수 있다.
2. 각 토큰에서 수행하는 계산은 다른 토큰의 계산과 독립적으로 이루어진다. (토큰을 순차적으로 처리하지 않는다.)
   $\Rightarrow$ 순전파와 훈련을 병렬적으로 진행할 수 있다.

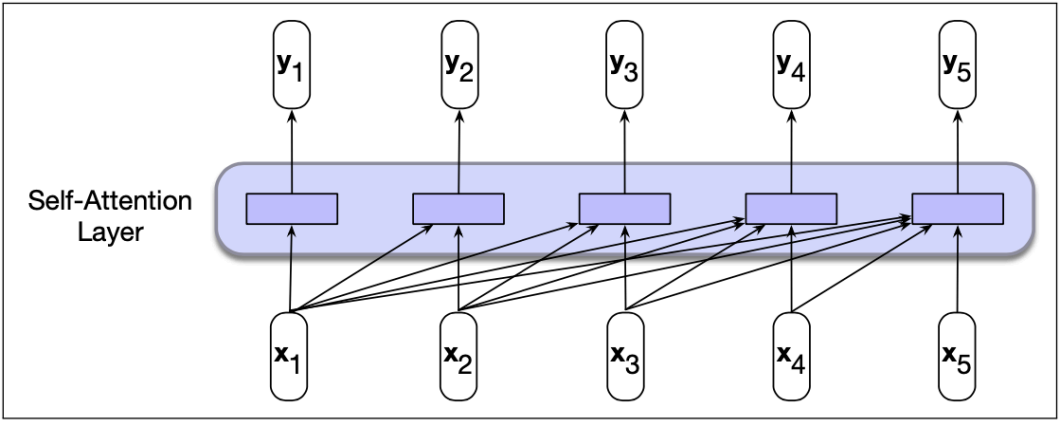

Figure 9.15 Information flow in a causal (or masked) self-attention model. In processing each element of the sequence, the model attends to all the inputs up to, and including, the current one. Unlike RNNs, the computations at each time step are independent of all the other steps and therefore can be performed in parallel.

“Causal” self-attention

입력 $\vec{x}_i \in \mathbb{R}^d$

선행 요소 $\vec{x}_j \in \mathbb{R}^d \quad \forall j \leq i$

Attention scores $a_{ij} = \text{score}\left(\vec{x}_i, \vec{x}_j\right) = \vec{x}_i \cdot \vec{x}_j \in \mathbb{R}$

Attention weights $\alpha_{ij} = \text{softmax}\left(\text{score}\left(\vec{x}_i, \vec{x}_j\right)\right) \in]0,1[\quad \forall j \leq i$

출력 $\vec{y}_i = \sum_{j=1}^{i} \alpha_{ij} \vec{x}_j \in \mathbb{R}^d$

예시

입력 $\vec{x}_3 \in \mathbb{R}^d$

선행 요소 $\vec{x}_1, \vec{x}_2, \vec{x}_3 \in \mathbb{R}^d$

Attention scores $a_{3j} = \text{score}\left(\vec{x}_3, \vec{x}_j\right) = \vec{x}_3 \cdot \vec{x}_j \in \mathbb{R} \quad (j = 1, 2, 3)$

Attention weights $\alpha_{3j} = \text{softmax}\left(a_{3j}\right) = \dfrac{\exp\left(a_{3j}\right)}{\exp\left(a_{31}\right) + \exp\left(a_{32}\right) + \exp\left(a_{33}\right)}$

출력 $\vec{y}_3 = \alpha_{31}\vec{x}_1 + \alpha_{32}\vec{x}_2 + \alpha_{33}\vec{x}_3 \in \mathbb{R}^d$

셀프-어텐션 계층에서 일어나는 일

입력 $\vec{x}_i$에 대응하는 출력 $\vec{y}_i$를 얻는 방법

$$\vec{y}_i = \sum_{j=1}^{i} \alpha_{ij} \vec{x}_j = \sum_{j=1}^{i} \text{softmax}\left(\text{score}\left(\vec{x}_i, \vec{x}_j\right)\right) \vec{x}_j = \sum_{j=1}^{i} \text{softmax}\left(\vec{x}_i \cdot \vec{x}_j\right) \vec{x}_j$$

관찰

우변에서 입력 토큰들의 임베딩 벡터 $\vec{x}_i$가 세 가지 다른 역할로 사용된다.

$$\text{output}_i = \sum_{j=1}^{i} \text{softmax}\left(\text{query}_i \cdot \text{key}_j\right) \text{value}_j$$

Attention score 수정 (1)

query, key, value 구별

Transformer의 접근 방법: 입력 벡터의 세 가지 표상

$$\vec{y}_i = \sum_{j=1}^{i} \text{softmax}\left(\vec{q}_i \cdot \vec{k}_j\right) \vec{v}_j$$

세 가지 역할을 각기 다른 벡터로 나타내기 위해 가중치 매개변수 행렬 $\mathbf{W^Q}$, $\mathbf{W^K}$, $\mathbf{W^V}$를 도입한다.

query $\vec{q}_i = \vec{x}_i^{\intercal}\mathbf{W^Q} \in \mathbb{R}^{d_k}$ ⋯"current focus of attention"

key $\vec{k}_i = \vec{x}_i^{\intercal}\mathbf{W^K} \in \mathbb{R}^{d_k}$ ⋯"preceding input(s)"

value $\vec{v}_i = \vec{x}_i^{\intercal}\mathbf{W^V} \in \mathbb{R}^{d_v}$

예시

입력 $\vec{x}_3 \in \mathbb{R}^d$ (열벡터)

Query $\vec{q}_3 = \vec{x_3}^\intercal \mathbf{W^Q} \in \mathbb{R}^{d_k}$ (행벡터)

선행 요소 $\vec{x}_1, \vec{x}_2, \vec{x}_3 \in \mathbb{R}^d$ (열벡터)

Keys $\vec{k}_1 = \vec{x_1}^\intercal \mathbf{W^K}, \vec{k}_2 = \vec{x_2}^\intercal \mathbf{W^K}, \vec{k}_3 = \vec{x_3}^\intercal \mathbf{W^K} \in \mathbb{R}^{d_k}$ (행벡터)

Attention scores $a_{3j} = \text{score}\left(\vec{x}_3, \vec{x}_j\right) = \vec{q}_3 \cdot \vec{k}_j \in \mathbb{R} \quad (j = 1, 2, 3)$

Attention weights $\alpha_{3j} = \text{softmax}\left(a_{3j}\right) = \dfrac{\exp\left(a_{3j}\right)}{\exp\left(a_{31}\right) + \exp\left(a_{32}\right) + \exp\left(a_{33}\right)}$

Values $\vec{v}_1 = \vec{x_1}^\intercal \mathbf{W^V}, \vec{v}_2 = \vec{x_2}^\intercal \mathbf{W^V}, \vec{v}_3 = \vec{x_3}^\intercal \mathbf{W^V} \in \mathbb{R}^{d_v}$ (행벡터)

출력 $\vec{y}_3 = \alpha_{31}\vec{v}_1 + \alpha_{32}\vec{v}_2 + \alpha_{33}\vec{v}_3 \in \mathbb{R}^{d_v}$

참고 원 논문(Vaswani et al., 2017에서는 $d_k = d_v = 1024$로 설정했음.

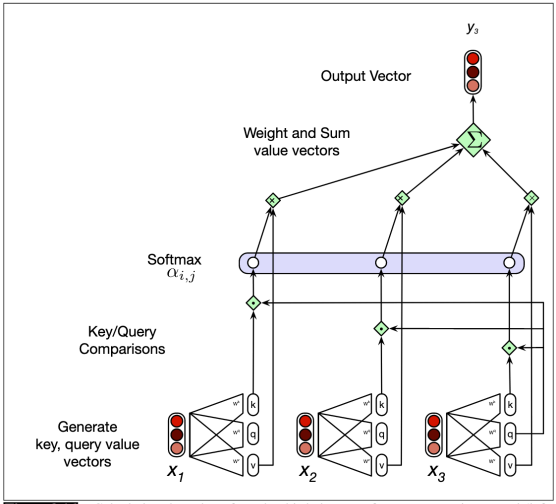

Figure 9.16 Calculating the value of $\mathbf{y}_3$, the third element of a sequence using causal (left-to-right) self-attention.

관찰

i $= 3$일 때(3번째 토큰을 살펴볼 때) 사용하는 벡터

Query $\vec{q}_3$

Keys $\vec{k}_1, \vec{k}_2, \vec{k}_3$

Values $\vec{v}_1, \vec{v}_2, \vec{v}_3$

주의

내적을 계산하기 위해서는 $\vec{q}_i$와 $\vec{k}_j$의 차원이 같아야 한다. ($\vec{v}_j$의 차원은 달라도 됨)

내적 계산의 문제

$$\vec{q}_i \cdot \vec{k}_j = q_{i1}k_{j1} + q_{i2}k_{j2} + \cdots + q_{id_k}k_{jd_k} \in \mathbb{R}$$

- 1024차원 벡터의 내적은 1024개의 곱을 더하는 것이다.
  $\Rightarrow$ 내적값이 커지면 지수함수(exp)를 취했을 때 overflow가 일어날 수 있다.

Overflow

```
>>> exp(709)
8.218407461554972e+307
>>> exp(710)
<stdin>:1: RuntimeWarning: overflow encountered in exp
inf
```

Attention score 수정 (2)

scaled dot-product

해결: scaled dot-product attention

내적값을 차원 d_k의 제곱근으로 나누어 준다.

$$\text{score}\left(\vec{x}_i, \vec{x}_j\right) = \frac{\vec{q}_i \cdot \vec{k}_j}{\sqrt{d_k}}$$

남은 단계

입력 시퀀스 전체 $(\vec{x}_1, \cdots, \vec{x}_N)$에 한꺼번에 적용하기

데이터

- N개의 토큰 시퀀스를…
  …N개의 d차원 임베딩 벡터(행벡터)를 쌓은 행렬 $\mathbf{X}$로 표기하자.

$$\mathbf{X} = \left[\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_N\right]^T = \begin{bmatrix} \vec{x}_1^T \\ \vec{x}_2^T \\ \vdots \\ \vec{x}_N^T \end{bmatrix} \in \mathbb{R}^{N \times d}$$

가중치 매개변수

- Queries, keys, values

$$\mathbf{W^Q} \in \mathbb{R}^{d \times d_k} \Rightarrow \mathbf{Q} = \mathbf{X}\mathbf{W^Q} \in \mathbb{R}^{N \times d_k}$$
$$\mathbf{W^K} \in \mathbb{R}^{d \times d_k} \Rightarrow \mathbf{K} = \mathbf{X}\mathbf{W^K} \in \mathbb{R}^{N \times d_k}$$
$$\mathbf{W^V} \in \mathbb{R}^{d \times d_v} \Rightarrow \mathbf{V} = \mathbf{X}\mathbf{W^V} \in \mathbb{R}^{N \times d_v}$$

사실

시퀀스 길이가 N이면 score $(\vec{x}_i, \vec{x}_j)$의 모든 가능한 조합은 $(N \times N)$ 행렬로 표현 가능하다.

$$\mathbf{Q} = \begin{bmatrix} \vec{q}_1 \\ \vec{q}_2 \\ \vdots \\ \vec{q}_N \end{bmatrix}, \mathbf{K} = \begin{bmatrix} \vec{k}_1 \\ \vec{k}_2 \\ \vdots \\ \vec{k}_N \end{bmatrix} \in \mathbb{R}^{N \times d_k}$$이면 $\mathbf{K}^\intercal = \begin{bmatrix} \vec{k}_1^\intercal \, \vec{k}_2^\intercal \cdots \vec{k}_N^\intercal \end{bmatrix} \in \mathbb{R}^{d_k \times N}$이므로

$$\mathbf{QK}^\intercal = \begin{bmatrix} \vec{q}_1\vec{k}_1^\intercal & \vec{q}_1\vec{k}_2^\intercal & \cdots & \vec{q}_1\vec{k}_N^\intercal \\ \vec{q}_2\vec{k}_1^\intercal & \vec{q}_2\vec{k}_2^\intercal & \cdots & \vec{q}_2\vec{k}_N^\intercal \\ \vdots & \vdots & \ddots & \vdots \\ \vec{q}_N\vec{k}_1^\intercal & \vec{q}_N\vec{k}_2^\intercal & \cdots & \vec{q}_N\vec{k}_N^\intercal \end{bmatrix} = \begin{bmatrix} \vec{q}_1 \cdot \vec{k}_1 & \vec{q}_1 \cdot \vec{k}_2 & \cdots & \vec{q}_1 \cdot \vec{k}_N \\ \vec{q}_2 \cdot \vec{k}_1 & \vec{q}_2 \cdot \vec{k}_2 & \cdots & \vec{q}_2 \cdot \vec{k}_N \\ \vdots & \vdots & \ddots & \vdots \\ \vec{q}_N \cdot \vec{k}_1 & \vec{q}_N \cdot \vec{k}_2 & \cdots & \vec{q}_N \cdot \vec{k}_N \end{bmatrix} \in \mathbb{R}^{N \times N}$$

Attention weights $[\alpha_{ij}] = \left[\text{softmax}_j\left(\frac{\vec{q}_i \cdot \vec{k}_j}{\sqrt{d_k}}\right)\right] = \text{softmax}\left(\frac{\mathbf{QK}^\intercal}{\sqrt{d_k}}\right) \in \mathbb{R}^{N\times N}$

최종

$$\text{SelfAttention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{QK}^\intercal}{\sqrt{d_k}}\right)\mathbf{V} \in \mathbb{R}^{N\times d_v}$$

문제

현재 보고 있는 토큰 이후의 토큰은 고려하지 않는다는 조건을 반영해야 한다.

해결

$i < j$인 경우 $\vec{q}_i \cdot \vec{k}_j = -\infty$로 설정한다.

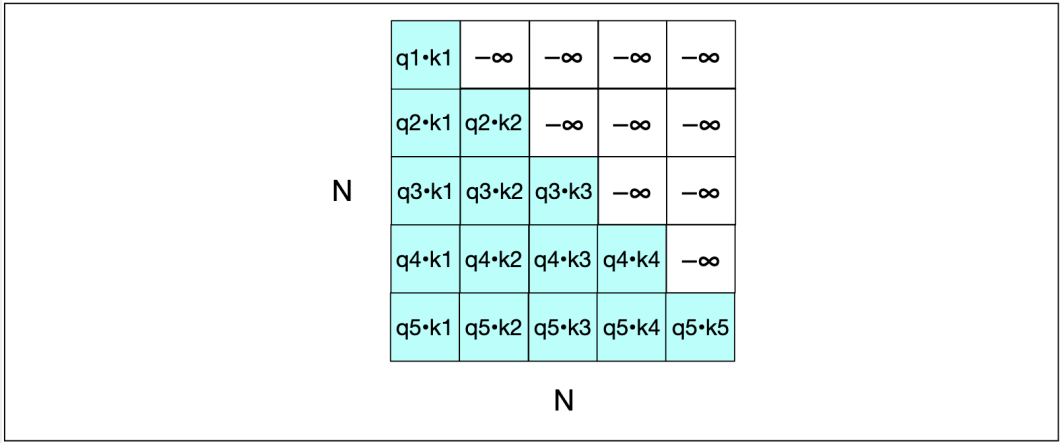

Figure 9.17 The $N \times N$ $\mathbf{QT}^\intercal$ matrix showing the $q_i \cdot k_j$ values, with the upper-triangle portion of the comparisons matrix zeroed out (set to $-\infty$, which the softmax will turn to zero).

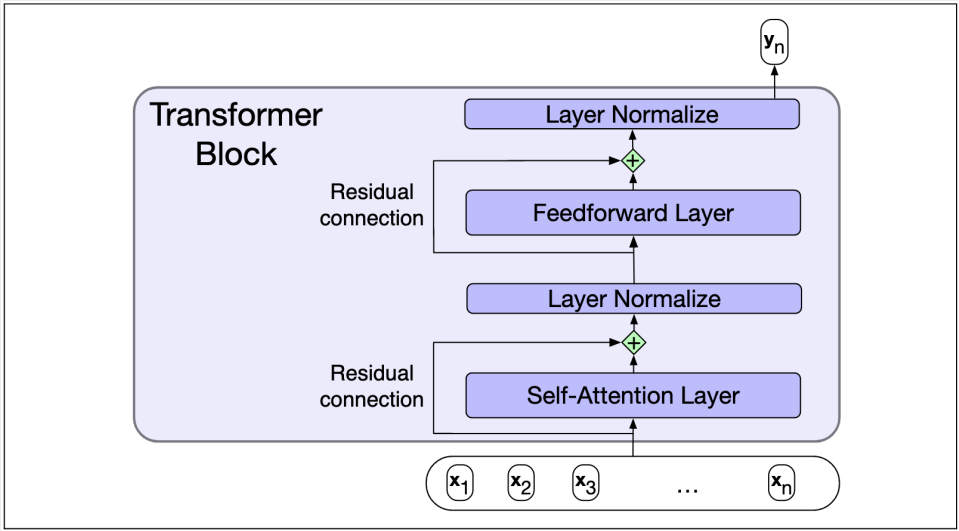

Figure 9.18 A transformer block showing all the layers.

$$\vec{z} = \text{LayerNorm}\left(\vec{x} + \text{SelfAttention}\left(\vec{x}\right)\right)$$
$$\vec{y} = \text{LayerNorm}\left(\vec{z} + \text{FFNN}\left(\vec{z}\right)\right)$$

트랜스포머 블록의 구성요소

- 셀프-어텐션 계층 …방금 배웠음
- 순방향 계층 …전에 배웠음
- 잔차 연결(Residual connection) …아직 모름
- 계층 정규화(Layer normalization) …아직 모름

$$\vec{z} = \text{LayerNorm}\left(\vec{x} + \text{SelfAttention}\left(\vec{x}\right)\right)$$
$$\vec{y} = \text{LayerNorm}\left(\vec{z} + \text{FFNN}\left(\vec{z}\right)\right)$$

잔차 연결

정보를 낮은 계층에서 중간 계층을 건너뛰고 높은 계층으로 직접 보내는 연결

$$\vec{z} = \text{LayerNorm}\left(\vec{x} + \text{SelfAttention}\left(\vec{x}\right)\right)$$
$$\vec{y} = \text{LayerNorm}\left(\vec{z} + \text{FFNN}\left(\vec{z}\right)\right)$$

계층 정규화

한 계층에 속하는 값들을 정규화(평균이 0, 표준편차가 1이 되도록 만드는 변환)하는 것.

$$\hat{\vec{x}} = \frac{\vec{x} - \mu}{\sigma}$$
$$\text{LayerNorm}\left(\vec{x}\right) = \gamma\hat{\vec{x}} + \beta$$

- μ $\vec{x}$의 평균
- σ $\vec{x}$의 표준편차
- γ, β 매개변수

오늘 배운 것

- 셀프 어텐션 계층: query, key, value, scaled dot product
- 트랜스포머 블록: 잔차 연결, 레이어 정규화

남은 것

- Multihead attention, Positional embeddings (SLP3 9.7 나머지)
- Contexual embeddings: BERT and GPT-2,3 (SLP3 11장)