2022학년도 1학기 컴퓨터언어학

23강 BERT와 문맥 임베딩

박수지

서울대학교 인문대학 언어학과

2022년 5월 30일 월요일

오늘의 목표

1. Multihead attention의 구조를 간략하게 그릴 수 있다.
2. Position embedding의 필요성을 설명할 수 있다.
3. Byte Pair Encoding을 비롯한 Subword tokenization의 원리를 설명할 수 있다.
4. BERT(Bidirectional Encoder Representations from Transformers)의 구성으로 Masked language modeling과 Next sentence prediction의 작동 방식을 설명할 수 있다.

정정 및 보충

계층 정규화

한 계층의 값들을 표준화(평균이 0, 표준편차가 1이 되도록 만드는 변환)한 후 변환하는 것.

$$\hat{\vec{x}} = \frac{\vec{x} - \mu}{\sigma}$$

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

정규화 vs. 표준화

- 정규화(Normalization): 값들을 특정한 범위(주로 [0,1]) 안에 놓이게 만드는 것.
- 표준화(Standardization): 값들을 평균이 0, 표준편차가 1이 되도록 만드는 것.
  - Standardization: z-score normalization

정정 및 보충

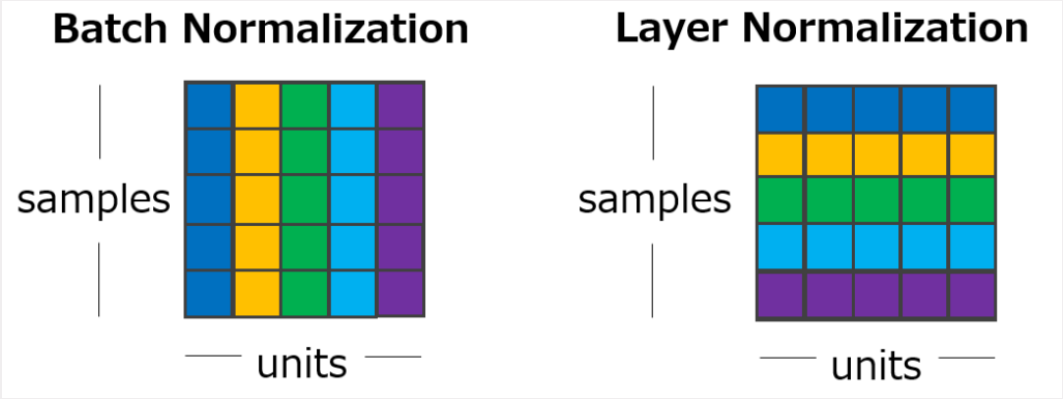


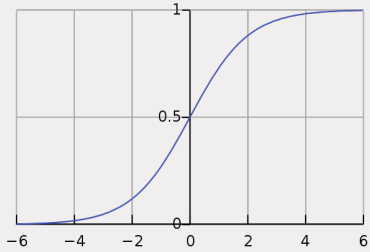
Figure: https://ai-pool.com/a/s/normalization-in-deep-learning

정정 및 보충

Scaled dot product의 목적

내적값의 크기를 줄여서 오버플로 현상을 방지한다.

- 내적값이 커지면 발생하는 추가적인 문제
  - 소프트맥스 함수를 취했을 때 기울기가 0에 매우 가까워지는 기울기 소실(vanishing gradient)이 일어난다.



내적값을 $\sqrt{d_k}$로 나누는 이유: $\vec{q}, \vec{k}$의 분산이 1일 때 내적값의 표준편차가 $\sqrt{d_k}$이기 때문.

복습

셀프-어텐션 계층: 입력 벡터의 세 가지 표상

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

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

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

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

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

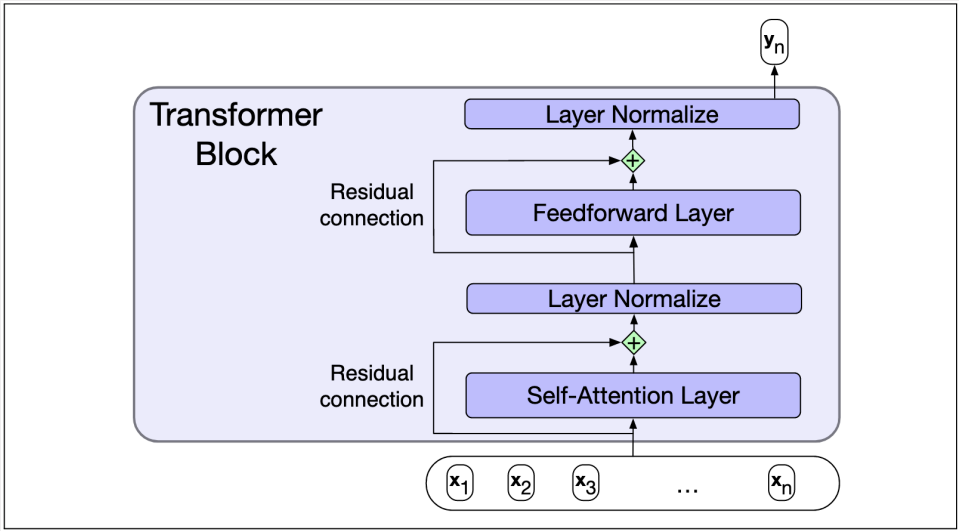


Figure 9.18 A transformer block showing all the layers.

지난 시간에 배운 것

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

셀프 어텐션 계층의 한계

1. 입력 시퀀스의 다양한 속성(통사, 의미, 담화 관계 등)을 골고루 포착하기 어렵다.
2. 입력 시퀀스의 어순을 온전히 반영하지 못한다.

해결

1. Multihead attention
2. Positional embeddings

멀티헤드 어텐션

하나의 어텐션 헤드

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

새로운 제안

한 계층 안에 여러 개의 어텐션 “헤드”를 두자!

목적

한 계층 안에서 입력 시퀀스의 다양한 속성(통사, 의미, 담화 관계) 등을 반영하기 위해

멀티헤드 어텐션

h 개의 헤드로 이루어진 멀티헤드 어텐션

$$i = 1, 2, \cdots, h; \ \mathbf{Q} = \mathbf{X}\mathbf{W}_i^{\mathbf{Q}}; \ \mathbf{K} = \mathbf{X}\mathbf{W}_i^{\mathbf{K}}; \ \mathbf{V} = \mathbf{X}\mathbf{W}_i^{\mathbf{V}};$$

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

$$\text{MultiHeadAttn}\left(\mathbf{X}\right) = \left(\overrightarrow{\text{head}_1} \oplus \overrightarrow{\text{head}_2} \oplus \cdots \oplus \overrightarrow{\text{head}_h}\right)\mathbf{W}^{\mathbf{O}} \in \mathbb{R}^{N \times d}$$

연습

행렬 $\mathbf{W}^{\mathbf{O}}$ 의 형상은?

멀티헤드 어텐션

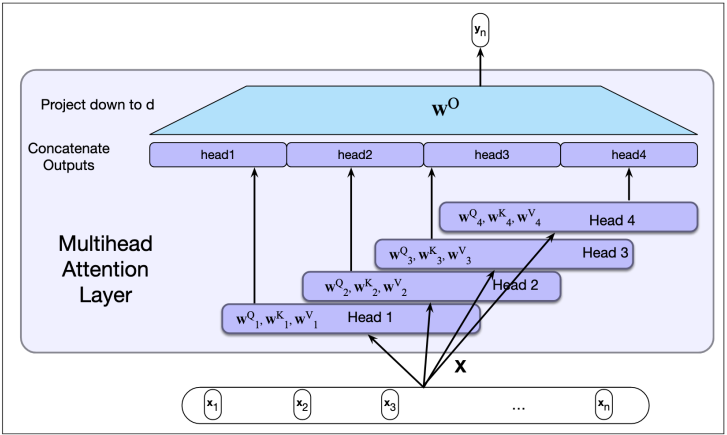


Figure 9.19 Multihead self-attention: Each of the multihead self-attention layers is provided with its own set of key, query and value weight matrices. The outputs from each of the layers are concatenated and then projected down to d, thus producing an output of the same size as the input so layers can be stacked.

어순 반영하기: 위치 임베딩

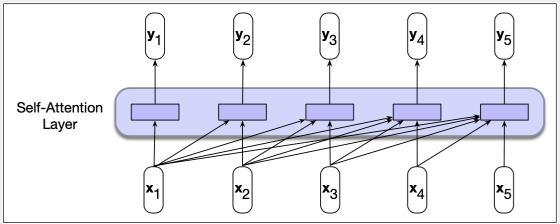


Figure 9.15 Information flow in a causal (or masked) self-attention model. In processing each element of the sequence, the model attends to all the inputs up to, and including, the current one. Unlike RNNs, the computations at each time step are independent of all the other steps and therefore can be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

셀프 어텐션의 문제

x_1 과 x_2 는 서로의 어순을 알지만…
x_3 의 입장에서는 $[x_1, x_2, x_3]$ 이나 $[x_2, x_1, x_3]$ 이나 계산이 똑같다!

목표

$[x_1, x_2, x_3]$ 과 $[x_2, x_1, x_3]$ 이 서로 다른 시퀀스라는 것을 반영하자!

어순 반영하기: 위치 임베딩

목표

$[x_1, x_2, x_3]$과 $[x_2, x_1, x_3]$이 서로 다른 시퀀스라는 것을 반영하자!

해법

단어(토큰) 임베딩에 위치 임베딩을 더해서 트랜스포머의 입력으로 삼는다.

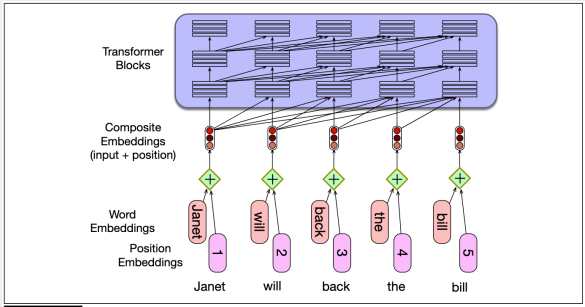


Figure 9.20 A simple way to model position: simply adding an embedding representation of the absolute position to the input word embedding.

효과

'Janet'이 첫 번째 토큰일 때와 두 번째 토큰일 때 서로 다른 임베딩 벡터로 표상된다.

복습

Causal (backword-looking, left-to-right) self-attention

Causal self-attention

$$\vec{y}_i = \sum_{j=1}^{i} \alpha_{ij} \vec{v}_j$$

i번째 입력을 처리할 때 $j \leq i$를 만족하는 j번째 토큰들만 살펴본다.

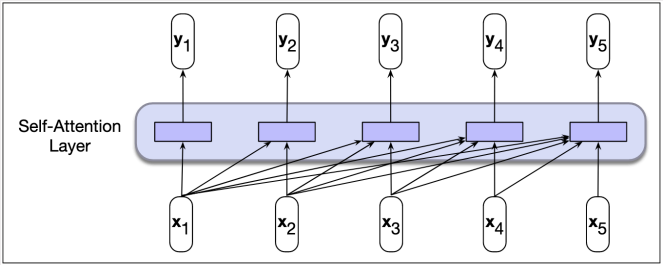


Figure 11.1 A causal, backward looking, transformer model like Chapter 9. Each output is computed independently of the others using only information seen earlier in the context.

새로운 접근

Bidirectional self-attention

Bidirectional self-attn

$$\vec{y}_i = \sum_{j=1}^{n} \alpha_{ij} \vec{v}_j$$

i번째 입력을 처리할 때 $j \leq i$와 무관하게 모든 j번째 토큰들을 다 살펴본다.

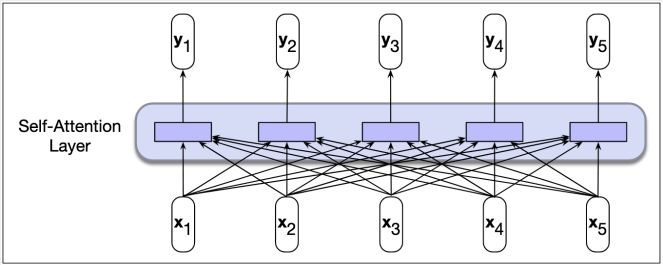


Figure 11.2 Information flow in a bidirectional self-attention model. In processing each element of the sequence, the model attends to all inputs, both before and after the current one.

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

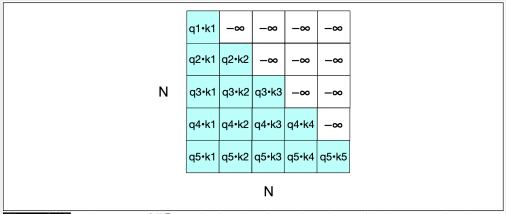


Figure 9.17 The $N \times N$ $\mathbf{QT}^\intercal$ matrix showing the $q_i \cdot k_j$ values, with the upper-triangle portion of the comparisons matrix zeroed out (set to $-\infty$, which the softmax will turn to zero).

Figure: Causal self-attention

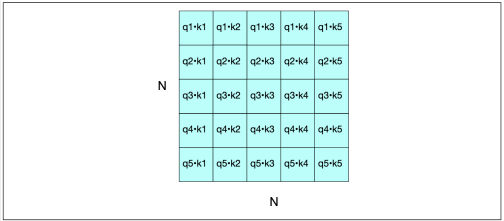


Figure 11.3 The $N \times N$ $\mathbf{QK}^\intercal$ matrix showing the complete set of $q_i \cdot k_j$ comparisons.

Figure: Bidirectional self-attention

BERT: Bidirectional Encoder Representations from Transformers

NAACL 2019

BERT

- Bidirectional
  - 양방향 셀프-어텐션 계층을 사용하여
- Encoder Representations
  - 토큰을 벡터로 표상하는 모형
  - cf. decoder: 벡터로부터 토큰을 생성하는 모형
- from Transformers

기말고사에 나옴

https://twitter.com/bertsesame

BERT: Bidirectional Encoder Representations from Transformers

NAACL 2019

BERT의 설정

- (subword) vocab size $|V| = 30000$
- hidden size $d = 768$
- 12 transformer blocks
- 12 multihead attention layers per block

잠깐

"subword"가 무엇인가?

Subword tokenization

단어를 더 작은 단위로 분리하는 것

- cf. 형태소 분석

효과

- 형태소가 무엇인지 기계에게 알려주지 않아도 형태소 분석과 비슷한 결과를 낸다.
- vocabulary size를 효율적으로 줄일 수 있다.

종류

- BPE(Byte Pair Encoding)
  - WordPiece ← BERT에서 사용하는 토크나이저.
  - Unigram
  - SentencePiece

예시: Byte Pair Encoding

전제 텍스트가 단어 단위로 이미 분리되어 있다.

데이터 (hug) (hug) (pug) (pun) (bun) (hugs)

목표 vocab size $|V| = 10$

1. 일단 모든 단어를 문자 단위로 분리한다.

   tokenization (h u g) (h u g) (p u g) (p u n) (b u n) (h u g s)
   vocabulary {h, u, g, p, n, b, s} $\Rightarrow$ $|V| = 7$
   pairs h-u, u-g, h-u, u-g, p-u, u-g, p-u, u-n, b-u, u-n, h-u, u-g, g-s

2. 위의 토큰쌍 중 가장 자주 나온 u-g를 하나의 토큰 ug로 통합한다.

   tokenization (h ug) (h ug) (p ug) (p u n) (b u n) (h ug s)
   vocabulary {h, u, g, p, n, b, s, ug} $\Rightarrow$ $|V| = 8$
   pairs h-ug, h-ug, p-ug, p-u, u-n, b-u, u-n, h-ug, ug-s

예시: Byte Pair Encoding (계속)

3. 위의 토큰쌍 중 가장 자주 나온 h-ug를 하나의 토큰 hug로 통합한다.

tokenization (hug) (hug) (p ug) (p u n) (b u n) (hug s)
vocabulary {h, u, g, p, n, b, s, ug, hug} $\Rightarrow$ $|V| = 9$
pairs p-ug, p-u, u-n, b-u, u-n, hug-s

4. 위의 토큰쌍 중 가장 자주 나온 u-n을 하나의 토큰 un으로 통합한다.

tokenization (hug) (hug) (p ug) (p un) (b un) (hug s)
vocabulary {h, u, g, p, n, b, s, ug, hug, un} $\Rightarrow$ $|V| = 10$

5. 목표 vocab size를 달성했으므로 종료한다.

BERT 언어 모형의 훈련 목표

1. Masked language model
   - 지금까지 배운 언어 모형: 다음 단어 예측하기
     - `Please turn your homework __.`
   - 마스크 언어 모형: 빈 칸 채우기
     - `Please turn ____ homework in.`
2. Next sentence prediction

예시: KLUE RoBERTa large

`https://huggingface.co/klue/roberta-large`

Masked language modeling

문장에서 가려진 부분에 들어갈 토큰이 무엇인지 예측하는 문제.

특수 토큰

- `[MASK]`: 토큰을 가릴 때 사용하는 토큰.

훈련 데이터

- 전체 토큰의 15%를 추출하여 훈련에 사용한다.
  이 15% 중에서...
  - 80%를 `[MASK]`로 교체한다.
  - 10%를 임의의 토큰으로 교체한다.
  - 나머지 10%는 그대로 둔다.

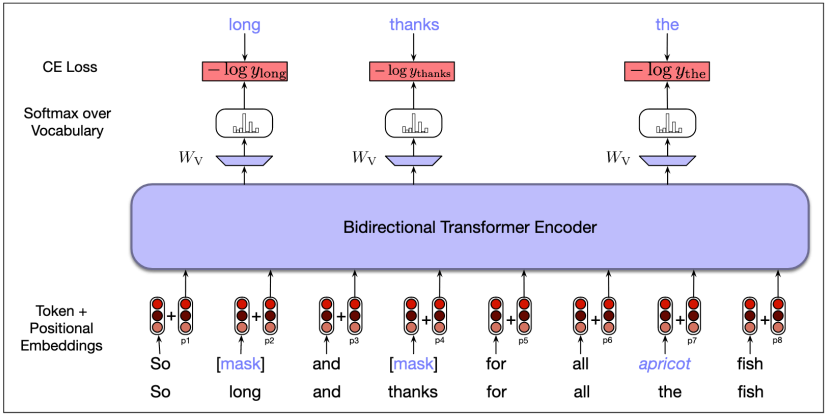


Figure 11.5 Masked language model training. In this example, three of the input tokens are selected, two of which are masked and the third is replaced with an unrelated word. The probabilities assigned by the model to these three items are used as the training loss. (In this and subsequent figures we display the input as words rather than subword tokens; the reader should keep in mind that BERT and similar models actually use subword tokens instead.)

Next sentence prediction

시퀀스의 두 번째 문장이 첫 번째 문장 다음에 나오는 것인지 아닌지 예측하는 문제

특수 토큰

- [CLS](classfication): 시퀀스를 분류할 때 사용하는 토큰.
- [SEP](separator): 문장의 경계를 지을 때 사용하는 토큰.

입력 벡터의 구성요소

- Token embeddings: 토큰 고유의 의미를 표상하는 벡터.
- Segment embeddings: 토큰이 앞뒤 어느 문장에 속하는지를 표상하는 벡터.
- Positional embeddings: 토큰의 시퀀스 내 위치가 몇 번째인지를 표상하는 벡터.

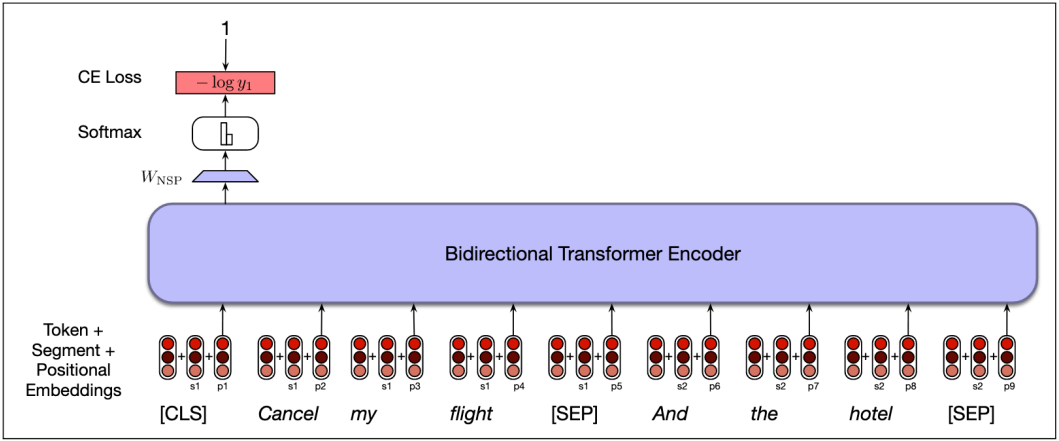


Figure 11.7 An example of the NSP loss calculation.

문맥 임베딩

BERT 훈련 결과

BERT 모형의 출력 벡터가 문맥을 반영한 임베딩이 된다.
입력 시퀀스가 $[x_1, \cdots, x_i, \cdots, x_n]$일 때 출력 벡터 $\vec{y}_i$를 x_i의 문맥 $[x_1, \cdots, x_n]$을 고려한 임베딩 벡터로 해석할 수 있다.

비교: static embedding

단어 w마다 고정된 임베딩 벡터가 존재하고 문맥에 따라 바뀌지 않는다.

남은 문제

Transfer learning: BERT를 감성분석, 품사 태깅 등 NLP 과제에 어떻게 적용할 것인가?

- 보강에서 만나요…

다음 시간에 할 일

HuggingFace Transformers 실습