Chapter 2. BERT 이해하기

발표자: 박채원

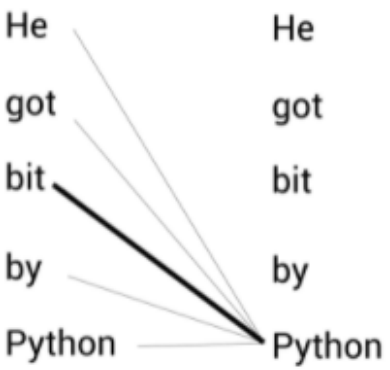
22-07-15

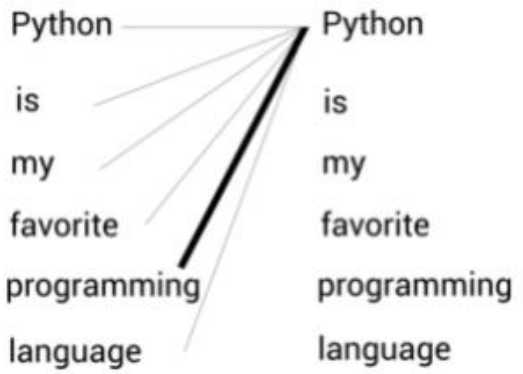
목차

2.1 BERT의 기본 개념

- BERT: Bidirectional Encoder Representation from Transformer
- 질문에 대한 대답, 텍스트 생성, 문장 분류 등과 같은 태스크에서 가장 좋은 성능을 도출

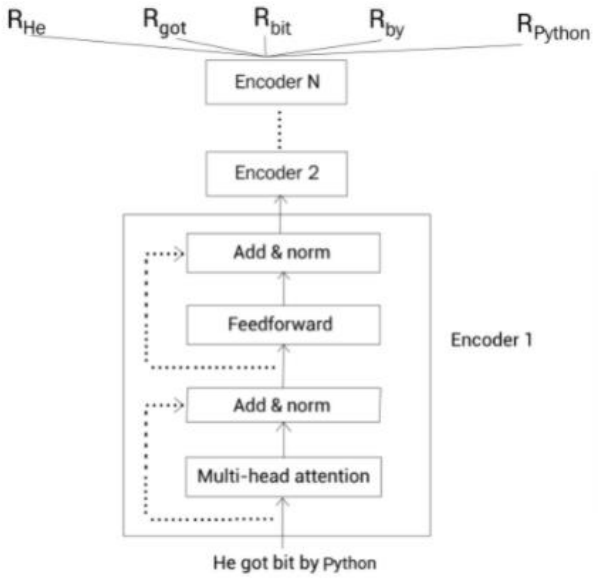
- BERT가 성공한 주된 이유
  - 단어가 나타나는 문맥에 관계없이 각 단어가 고정된 표현을 갖는 word2vec(문맥 독립 모델)과 달리 주변 단어에 의해 동적으로 변하는 단어 표현을 생성하는 BERT (문맥 기반 모델)
  - 동음이의어에 대해 처리해낼 수 있음
  - 특정 단어를 문장의 다른 모든 단어와의 관계를 기반으로 이해하려 시도





2.2 BERT의 동작 방식

- 트랜스포머 모델(인코더-디코더 구조)와 달리 인코더(양방향)만 사용
- 문장을 입력으로 넣어주면, 인코더는 **멀티 헤드 어텐션 메커니즘**을 사용해 문장의 각 단어의 문맥을 이해해 문장에 있는 각 단어의 문맥 표현을 출력으로 반환



- 인코더 레이어 N개 스택
- 각 토큰의 표현 크기는 인코더 레이어의 출력 차원
- 즉, BERT를 사용해 각 단어에 대한 문맥 표현을 얻을 수 있다.

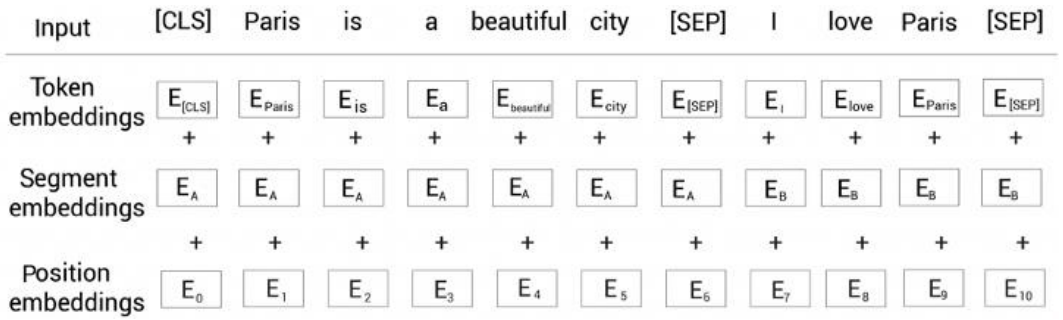
2.3 BERT의 구조

- 2.3.1 BERT-base
  - 12개의 인코더가 스택처럼 쌓인 형태 (L=12)
  - 각 인코더는 12개의 어텐션 헤드 사용 (A=12)
  - 인코더의 피드포워드 네트워크는 768 개의 은닉 유닛으로 구성. 즉 표현의 크기 768 (H=768)

- 2.3.2 BERT-large
  - 24개의 인코더가 스택처럼 쌓인 형태 (L=24)
  - 각 인코더는 16개의 어텐션 헤드 사용 (A=16)
  - 인코더의 피드포워드 네트워크는 1024 개의 은닉 유닛으로 구성. 즉 표현의 크기 1024 (H=1024)

2.4 BERT 사전 학습

- 2.4.1 BERT의 입력 표현
  - 토큰 임베딩
    1. 문장을 토큰화 해 토큰을 추출
    2. 첫번째 문장의 시작 부분에 [CLS] 토큰 추가 & 각 문장의 마지막 부분에 [SEP] 토큰 추가
    3. 토큰 임베딩이라는 임베딩 레이어를 사용해 토큰을 임베딩으로 변환 (토큰 임베딩의 변수들은 사전학습이 진행되면서 학습됨)
  - 세그먼트 임베딩
    - 입력된 두 문장을 구별하는 데 사용
  - 위치 임베딩
    - 트랜스포머가 어떤 반복 메커니즘도 사용하지 않고 모든 단어를 병렬로 처리하기 때문에 단어 순서와 관련된 정보 제공해야 함
    - 말그대로 각 단어의 위치를 나타내는 임베딩 (sin, cos 함수를 사용)



- 입력의 최종 표현
- 모든 임베딩을 합산해 BERT에 입력으로 제공

2.4 BERT 사전 학습

- 워드피스 토크나이저
  - BERT 사용 토크나이저. 하위 단어 토큰화 알고리즘을 기반으로 함

  - 토큰화 할 때 단어가 어휘 사전에 있는지 확인
  - 있으면 그 단어를 토큰으로 사용. 없으면 다시 하위 단어로 분할해 어휘 사전에 있는지 확인. 없으면 다시 분할
  - 개별 문자에 도달할 때까지 어휘 사전을 기반으로 하위 단어를 계속 분할하고 확인
  - 이는 out-of-vocabulary의 단어를 처리하는 데 효과적
  - 토큰 앞의 해시 기호는 하위 단어임을 의미.

Ex) “Let us start pretraining the model” -> [let, us, start, pre, ##train, ##ing, the, model]

2.4 BERT 사전 학습

- 사전 학습
  - 특정 태스크에 대한 방대한 데이터셋으로 모델을 학습시키고, 가중치 저장
  - 새로운 태스크가 주어졌을 때, 사전 학습된 모델의 가중치를 불러와 모델을 초기화하고, 새로운 태스크에 따라 가중치를 조정 -> 파인튜닝

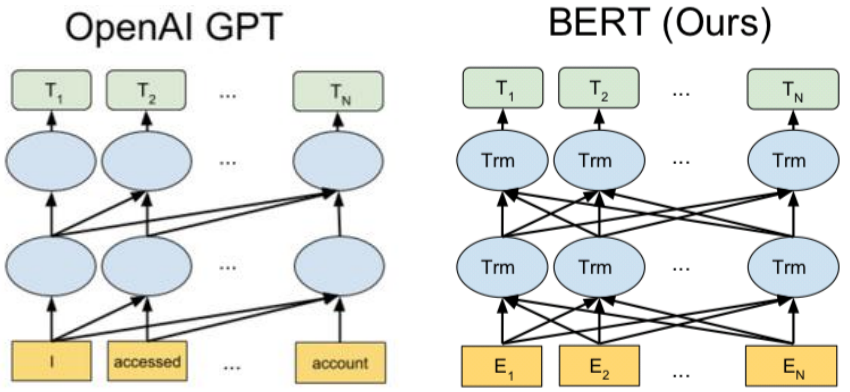
- BERT의 사전학습
  - BERT는 MLM과 NSP라는 두 가지 태스크를 이용해 큰 말뭉치 기반으로 사전학습

2.4 BERT 사전 학습

- 2.4.2 사전 학습 전략
  - 언어 모델링
    - 일반적으로 임의의 문장이 주어지고 단어를 순서대로 보면서 다음 단어를 예측하도록 모델을 학습시키는 것

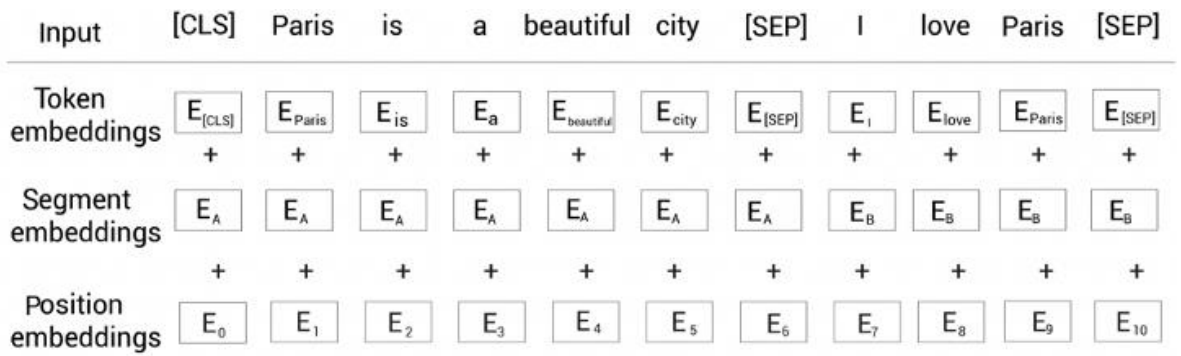
    - 자동 회귀 언어 모델링 (단방향)
      - 예시 문장: “Paris is a beautiful _(city). I love Paris”
      - 전방 예측: Paris is a beautiful _
      - 후방 예측: _. I love Paris

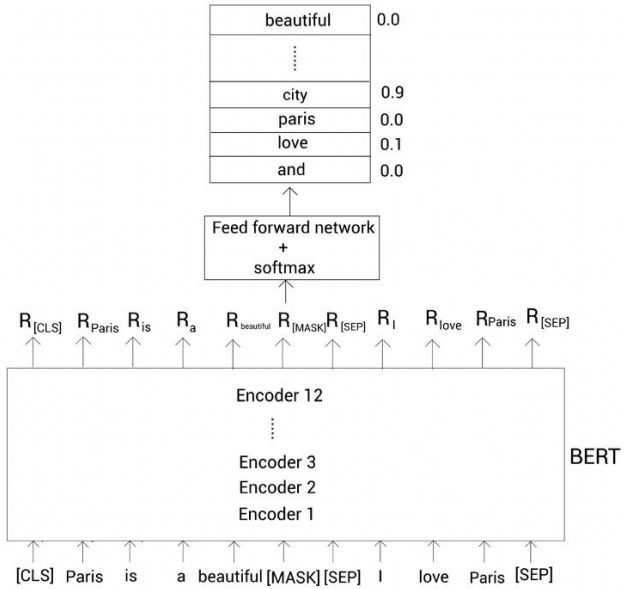
    - 자동 인코딩 언어 모델링 (양방향)
      - 전방 및 후방 예측을 모두 활용
      - 단방향보다 더 정확한 결과를 제공함



2.4 BERT 사전 학습

- 마스크 언어 모델링 (MLM)
  - 자동 인코딩 언어 모델 (양방향)
  - 주어진 입력 문장에서 전체 단어의 15%를 무작위로 마스킹하고 마스크된 단어를 예측하도록 모델 학습
  - -) 사전 학습과 파인 튜닝 사이 불일치가 생김. 사전학습시엔 [MASK] 토큰이 존재하지만, 파인튜닝시엔 [MASK] 토큰 없음
    - 이 문제를 해결하기 위해 80-10-10% 규칙 적용
    - 선택된 15% 토큰에 대해서 80%는 마스킹하고, 10%는 임의 토큰으로 교체하고, 10%는 변경하지 않음(그대로)
    - 소프트맥스로 마스크된 단어일 확률을 뽑아냄





2.4 BERT 사전 학습

- 전체 단어 마스킹 (WWM)
  - 하위 단어가 마스킹되면 해당 하위 단어와 관련된 모든 단어를 마스킹
  - 마스크 비율(15%)을 유지하기 위해 마스킹 하는 동안 마스크 비율이 15%를 초과하면 다른 단어의 마스킹을 무시

Ex) [[CLS], let, us, start, pre, ##train, ##ing, the, model, [SEP]]

-> 마스킹: [[CLS], [MASK], us, start, pre, ##train, [MASK], the, model, [SEP]]

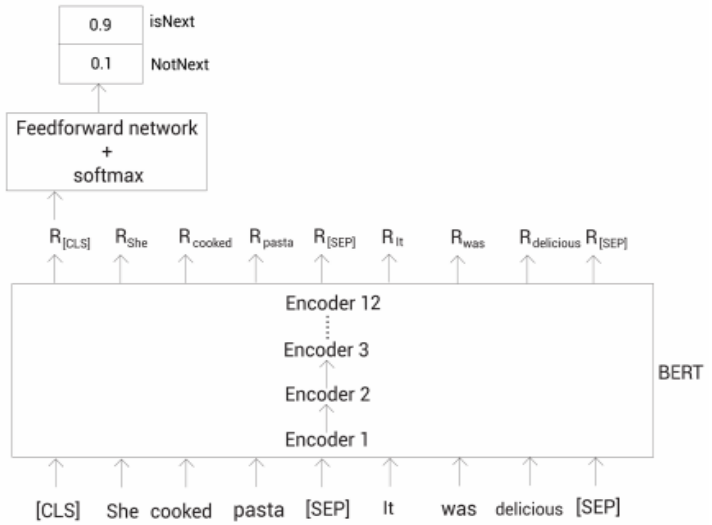
-> 하위 단어 관련 단어 마스킹: [[CLS], [MASK], us, start, [MASK], [MASK], [MASK], the, model, [SEP]]

-> 마스크 비율 조정: [[CLS], let, us, start, [MASK], [MASK], [MASK], the, model, [SEP]]

2.4 BERT 사전 학습

- 다음 문장 예측 (NSP)
  - 두 문장을 입력하고 두 번째 문장이 첫 번째 문장의 다음 문장인지 예측 (isNext or notNext)
  - 목표: 문장 쌍이 isNext 범주에 속하는지 여부를 예측하는 것
  - 본질적으로 이진 분류 태스크 -> 즉 두 문장 사이의 관계를 파악할 수 있다. 이는 질문 응답 및 유사문장탐지와 같은 다운스트림 태스크에서 유용하다.

  - 데이터셋 구축
    - 한 문서에서 연속된 두 문장을 isNext로 표시하고, 각각 다른 문서에서 가져온 두 문장을 notNext로 표시



- 모든 토큰의 집계 표현을 보유하고 있는 [CSL] 토큰 표현을 가져와 소프트맥스 함수를 사용해 피드포워드 네트워크에 공급

2.4 BERT 사전 학습

- 2.4.3 사전 학습 절차
  - 데이터셋 준비
    - 말뭉치에서 두 문장을 샘플링 (50%는 isNext, 50%는 notNext)
    - 두 문장의 토큰 수 합은 512보다 작거나 같아야 함

      [[CLS], we, enjoyed, the, game, [SEP], turn, the radio, on, [SEP]

      -> 80-10-10% 규칙에 따라 마스킹 진행: [[CLS], we, enjoyed, the, [MASK], [SEP], turn, the radio, on, [SEP]

    - 토큰을 입력으로 넣어주고, 마스크된 토큰을 예측하기 위해 모델을 학습시키며, 동시에 연속 문장 여부를 분류하게 된다.
    - MLM과 NSP 작업을 동시에 사용해 BERT를 학습시킨다.

  - 하이퍼 파라미터
    - 100만 스텝, 배치 크기 256, lr=1e-4, $\beta_1 = 0.9$, $\beta_2 = 0.999$, Adam 옵티마이저, 웜업 1만 스텝
    - 학습 초기에는 과감한 변화를 주고, 이후에는 수렴에 가까워지기 때문에 작은 변화를 주어 모델 최적화 -> 학습률 스케줄링
    - 웜업 스텝: 학습률 스케줄링의 일부로, 초기 1만 스텝은 학습률이 0에서 1e-4로 선형적으로 증가. 이후엔 선형적으로 감소
    - 모든 레이어에 0.1 확률의 드롭아웃 적용

    - GELU 활성화 함수: $\mathrm{GELU}(x) = xP(X \leq x) = x\Phi(x) = x \cdot \frac{1}{2}\left[1 + \mathrm{erf}(x/\sqrt{2})\right]$

2.5 하위 단어 토큰화 알고리즘

- OOV 단어 처리에 매우 효과적임
- 단어 수준 토큰화
  - 학습 데이터셋에서 어휘 사전 구축: 텍스트를 공백으로 분할 후 모든 고유 단어를 어휘 사전에 추가
  - 입력 문장 공백으로 분할 후 어휘 사전에 없는 단어는 [UNK] 토큰으로 대체

Ex) vocab = [game, the, I, played, walked, enjoy]

Token = [I, enjoyed, the, game] -> [I, [UNK], the, game]

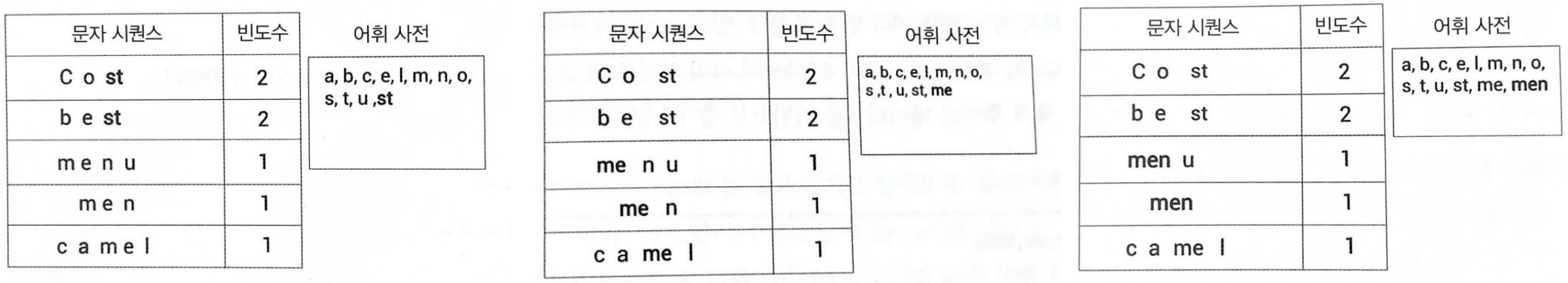
  - 어휘 사전을 더 크게 할 수 있으나, 모델의 메모리 부족 문제와 성능 문제(처리량 부족)를 일으킬 수 있다. 게다가 여전히 알 수 없는 단어는 존재하기 때문에 문제가 발생할 수 있다.
  - 이러한 문제를 해결하기 위해 하위 단어 토큰화 알고리즘을 사용

- 단어를 하위 단어로 분할한다면?

vocab = [game, the, I, play, walk, ed, enjoy]
Token = [I, enjoyed, the, game] -> [I, enjoy, ##ed, the, game]

2.5 하위 단어 토큰화 알고리즘

- 2.5.1 바이트 쌍 인코딩 (BPE)
  - 데이터셋에서 모든 단어를 **빈도수**와 함께 추출
  - 어휘 사전 크기 정의
  - 모든 단어를 문자로 나누고 문자 시퀀스로 만듦
  - 문자 시퀀스에 있는 모든 고유 문자를 어휘 사전에 추가
  - 가장 빈도수가 큰 기호 쌍을 식별해, 이를 어휘 사전에 추가한다. -> 어휘 사전 크기에 도달할 때까지 반복 수행



문자 시퀀스	빈도수
C o st	2
b e st	2
m e n u	1
m e n	1
c a m e l	1

어휘 사전: a, b, c, e, l, m, n, o, s, t, u ,st

문자 시퀀스	빈도수
C o st	2
b e st	2
me n u	1
me n	1
c a me l	1

어휘 사전: a, b, c, e, l, m, n, o, s ,t , u, st, me

문자 시퀀스	빈도수
C o st	2
b e st	2
men u	1
men	1
c a me l	1

어휘 사전: a, b, c, e, l, m, n, o, s, t, u, st, me, men

  - Ex) mean -> [me, an] -> [me, a, n]

2.5 하위 단어 토큰화 알고리즘

- 바이트 수준 바이트 쌍 인코딩 (BBPE)
  - 문자 수준의 시퀀스를 사용하는 대신 바이트 수준의 시퀀스 사용
  - 단어 -> 바이트 시퀀스
  - 다국어 설정에서 유용
  - OOV 단어 처리에 매우 효과적이어서 여러 언어로 어휘 사전을 공유하기 좋다.
  - BBPE maximizes vocabulary sharing across many languages and achieves better translation quality.

Ex) best -> b e s t -> 62 65 73 74

- 2.5.3 워드 피스
  - BPE와 유사하게 작동하지만, 빈도가 아닌 가능도에 따라 기호 쌍을 병합한다.
  - 모든 기호 쌍에 대해 학습된 언어모델의 가능도를 확인
  - 최대 가능도를 가진 것을 병합해 어휘에 추가

가능도: $\frac{p(st)}{p(s)p(t)}$

2.6 마치며

- BERT 개념 이해
  - 문맥에 관계없이 임베딩을 생성하는 word2vec과 달리 단어의 문맥을 이해해 임베딩을 생성할 수 있음
- BERT의 작동 방식
  - 트랜스포머의 인코더 사용
- 사전학습 task
  - MLM
  - NSP
- BERT의 학습 절차
  - 데이터셋, 하이퍼 파라미터 등
  - 세가지 하위 단어 토큰화 알고리즘