

Transformers & Pretraining

Problems of Vocab)

기존 방법론에서 전체 흐름

embedding layer: Pretrained Embedding 사용

model parameter: 초기화

=> 학습데이터 부족, out-of-vocab 문제, 다른 문장에도 동일 embedding 사용

Embedding process



1. 문자열 입력

2. 학습된 토크나이저 통해 토큰화

3. 각 토큰별 학습된 embedding vector 변환

* 학습 데이터에 존재하지 않는 단어 모두 UNK로 토큰화 (다른 의미에도 같은 token으로)

∴ 변형, 오탈자, 신조어에 대응 불가

Subword Token

- 단어 이하 단위에서 언어 구조 추론

- Byte-Pair-Encoding (BPE)

- 문법/의미/활용 포착

Bidirectional
양쪽 방향 형용사

Byte-Pair-Encoding

1. Word를 문자 단위로 분해

Connect

2. 전체 Corpus에서 가장 자주 등장하는 연속된 두 문자열 병합

C#A O## n##n##e##d##l##f##

3. 병합된 토큰을 기반으로 분해 결과 조합

n## n## \Rightarrow nn##

4. 종료 조건 (병합 횟수, 토큰 개수 등)을 만족할 때까지 1~3 반복 C#A O## nn##e##d##l##f##

do ooots \rightarrow do, oo, oo, ts

4ever \rightarrow 4, ever

\therefore 변형, 오탈자, 신조이 대응 가능, OOV 문제에서 자유로움

Fixed Word Embedding

- 각 단어에 사전에 학습된 embedding vector 부여

- Context 고려 못한 Embedding 방식

- 다른 뜻에도 동일 vector mapping Buffalo n. 짜개는 들은 V. 들이 한다

- embedding layer로 Context 반영 불가

\Rightarrow 각 task마다 대량 data 필요

\Rightarrow Pretrain (사전학습) 등장!

Pretrain (사전학습)

- Input의 일부를 훈련하고 훈련된 input을 복원하면서 언어의 의미적 / 문법적 구조 학습

Downstream Task

- Pretrain된 Weight를 initial weight로 삼아서 학습 진행

\Rightarrow Fine-tuning

- 다양한 task 적용 가능

- 문장 일부 변형, 해당 부분 단어 예측 (단편적 사실, 문법 정보, 상호참조, 감성, 추론, 단순 계산)

Encoder & Decoder 역할 다름

Pretrain에 사용되는 objective function

- Language Modeling

$$L = \sum \log P(w_t | w_0, \dots, w_{t-1})$$

이전 token 바탕으로 현재 token 예측 \Rightarrow 빌드 label 없이도 됨 Big Data 확보 가능

- SGD with Pretrain/Finetune

$$\hat{\theta} = \min_{\theta} L_{\text{pretrain}}(\theta)$$

$$\min_{\theta} L_{\text{finetune}}(\theta) \quad (\text{initial weight} = \hat{\theta})$$

Random initializing θ 에 비해 $\hat{\theta}$ 가 global optimum에 가까울 수 있음

작은 data로도 잘 학습, 쉽게 수렴

Pretrained model

1. Decoders - 일반적인 Language Model, 생성모델

2. Encoders - 양방향 토큰 정보 활용

3. Encoder-Decoders - encoder가 양방향 정보 종합, decoder가 생성에 집중

Pretrain task: 사전학습시 수행하는 pretrain을 위한 task

Downstream task: 실제 수행할 구체적인 task

Head: pretrain된 모델에 불어 downstream task에 맞춘 layer

finetuning 이전의 학습이 전혀 안 된 상태

Pretrained model: pretrain task를 수행한 layer

finetuning 이전에 어느정도 학습된 상태

GPT-1

- transformer decoder만 사용 (단방향)
- 일반적인 LM을 통해 pretrain
- 대량 데이터 확보 가능 (\because label 필요X)
- transformer보다 parameter 수 증가 (T: 6.5M, G: 110M) *pretrain* 효과 발생

총 4가지 방법론 finetuning

① Classification: 긍정·부정, 문법 오류 여부

② Entailment: 주어진 문장 관계분류

③ Similarity: 두 문장 간 유사도 파악

④ Multiple Choice: 주어진 문제에 대한 보기 중 정답 고르기

문장 종료 토큰 = <Extract>, 문장 마지막에 생성

Teacher Forcing 이용해 학습

<Extract> 위치의 출력값을 head에 입력

↳ 문장 전체 정보 담기도록 유도

Pretrain layer 사용함수로 성능 향상
Pretrain 진행한 model 함수로

BERT

- BERT의 pretrain task는 MLM & NSP 두 가지
- transformer encoder만 사용 (양방향)

MLM (Masked Language Modeling)

$$L_{MLM} = - \sum_i c_i m \log P(w_i | w_0, \dots, w_n; \theta)$$

- 양방향 token 정보 활용, LM 목록 확장 사용 불가

\Rightarrow 입력 token의 15%를 번영하고 이를 예측

80% [MASK], 10% 랜덤 다른 token, 10% 원본 토큰

\therefore Inference 상황과 동일한 환경, 모델이 언어 구조를 학습

NSP (Next Sentence Prediction)

- 많은 downstream task에서 문장 간 관계 학습
- 이를 반영한 pretrain task 퀴즈
- 두 문장 입력 50% 학습 데이터에서 연속된 문장 50% 서로 다른 문장에서 사용된 문장
 - ↳ 연속된 문장 여부 판단
- 문장 간 관계 학습, <CLS> 토큰이 입력 문장 정보 담음

* RoBerta에서 NSP가 학습에 도움 X 주장

BERT의 input

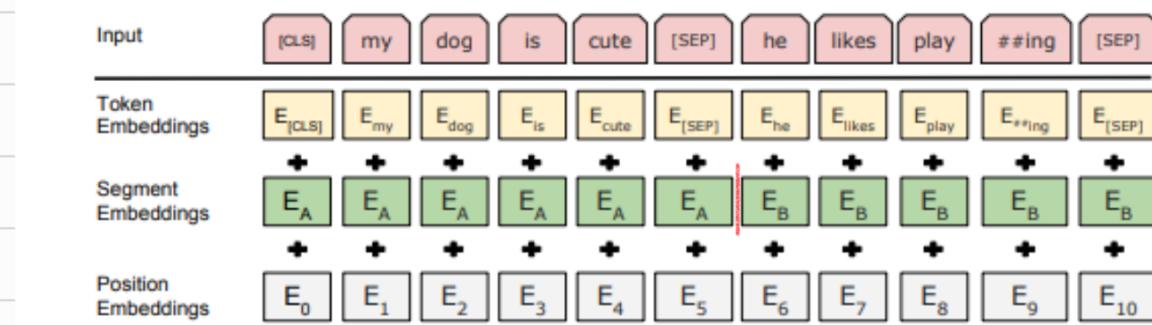


Figure 2: BERT input representation. The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

Token, Segment, Position 3가지 embedding vector를 더해 input embedding 생성

Token: Wordpiece 기반 토큰 단위 embedding (BPE와 유사, 무도기반)

Segment: 두 문장을 구분하기 위한 embedding

Position: transformer의 positional encoding

T5

(답)

- transformer 구조를 가지므로 dataset과 model을 카우고 pretrain 과정 정교화
- 매우 다양하고 downstream task에 따른 성능 평가 진행
- pretrain task 3: span corruption 처리
 - ↳
 - encoder 암호로 mask 사용
 - decoder 암호로 mask ID와 해당 문장 생성
 - encoder는 MLM을, decoder는 LM을 수행하는 pretrain task
 - Open Domain