

BERT 놓은장치



25일 → 3.4m/s + 강의

26일 → 끝까지 다 + 75의

BERT: transformer 구조를 이용한 language representation 모델

• Abstract

Bert: Bidirectional Encoder Representation from Transformer

→ original implementation은 단순

unlabeled data (wiki, book data 등)를 모델을 초기 학습시킨 후, NLP task를 위한 labeled data를 transfer learning으로 한다.

→ As a result,
 pre-train BERT model은 initial output layer를 fine-tuning을 하게 된다.
 그럼, 다른 NLP task를 위한 task-specific architecture modification으로 SOTA model 만들수 있다.

QnA: language inference (인문학)

= pre-train한 BERT model을 NLP task의 fine-tuning으로 SOTA

→ task-specific 한 퍼셉트론 네트워크 X

I. Introduction

- NLP tasks
 └ Sentence-level 3: 문장 단위 task
 └ token-level 3: 토큰 단위 task ↗ pre-training 단계에서 다른 task를 더 잘 학습시킬 수 있다.

- pre-training 단계 주제
 └ deep learning의 feature를 이용하는 것

- ① unsupervised feature-based 퍼셉트론
- ② unsupervised fine-tuning 퍼셉트론
- ③ transfer learning from supervised Data.

feature-based 방법 → AI 논문이나 인터넷, BERT가 무엇인가 알리시나요? (주제)

: task-specific한 architecture를 사용하는 것.

: pre-trained representation을 additional feature로 넣는 방법, 즉 2개의 network를 병행 사용
Ex) ELMo

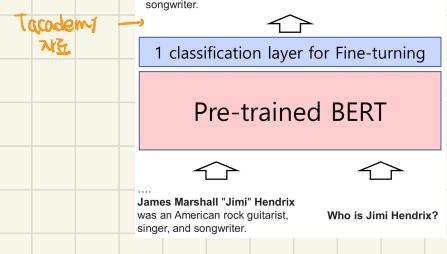
② fine-tuning 단계

: BERT가 네트워크이다!

: BERT와 SOTA 같은 GPT-2와 같은 방법 사용

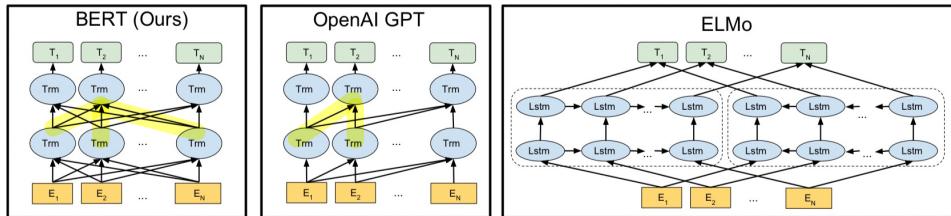
: task-specific한 parameter를 추출하는 fine-tuning

Ex) OpenAI GPT



→ 예전 GPT, ELMo \Rightarrow pre-training 시에 동일한 objective function으로 학습을 수행.

But, BERT \Rightarrow pre-trained Language Representation을 학습 \rightarrow 매우 효율적



• BERT의 핵심 3가지

- 1) Unidirectional or only bidirectional
- 2) masked language model or next sentence prediction task를 이용한 pretraining
- 3) task dataset or pre-training : task-specific dataset from fine-tuning

BERT의 pretraining의 내용은 2가지

- ① masked language model
- ② next sentence prediction task

* 가정의 방식은

: 예전 n개의 단어를 가지고 두번 단어를 예측하는 Model을 이루는 것 (n-gram)

→ Unidirectional II

이를 극복하기 위해 ELMo처럼 Bi-LSTM으로 양방향성을 가지게 하지만, 근본적 Shallow한 양방향성.

= 단방향 context 양방향

① masked language model (MLM)

: input array에 무작위로特殊 token을 mask 한다

그리고 transformer encoder가 주변 단어 context를 보고 MASK된 단어를 예측하는 문제.

(GPT는 transformer 구조 사용하지만, 앞 단어들만 보고 뒷단어를 예측하는 transformer Decoder 구조 사용)

: BERT는 Input sequence mask된 token은 sequence Transformer encoder로 무작위 token을 예측

\Rightarrow 深深 ! deep bidirectional이다. 학습도

→ they're unlabeled corpus의 language model training 후, 이를 basis로 두개의 동일task를 처리하는 network를 별도로 놓는 방식.

GPT \rightarrow BERT가 일반화하고 미화화하는 model

auto-regressive task

\rightarrow Unidirectional 이면 = 앞 token은 이전 token의 주제

② next sentence prediction

- 두 문장을 pre-training NLP 퀴즈에서 넣었을 때 두 문장이 다음으로 올라갈 확률이 아까운 이유는
- pre-training AI \Rightarrow 둘다를 이해하는 두 문장 : 간단하게 추출된 두 문장 = 50 : 50 으로
불가능, BERT가 이해해 수준!

2. Related Work

- ELMo, OpenAI GPT \rightarrow 예측

3. BERT

- bert architecture는 "Attention is all you need"의 전신 transformer는 NLP 학자들
pre-training 과 fine-tuning NLP architecture를 통해 대중적인 transfer learning 을
보여주는 계약.

※ NLP, transfer learning (cont.)

→ 기존의 만들기전 모델을 사용하여 새롭게 문제를 만들고, 학습을 반복해하고, 성과를 더 높이는 방법

→ 예상

: 대체로 예상은 기존 모델을 사용해서 문제 해결할 수 O

실험) fine-tuning과 transfer learning의 차이점?

→ 같은 뜻! 

3.1 Model Architecture

- BERT는 transformer 중 encoder 부분만 사용

- model의 크기와 디자인 model 제작

BERT-base : L=12, H=768, A=12, Total Parameter = 110M

BERT-large : L=24, H=1024, A=16, Total Parameter = 340M

↳ 더 어렵고 SOTA는 아님!

L : transformer block의 layer 수 \rightarrow 더 많은 층수

H : hidden size

A : self-attention head 수 = 층수의 배수

OpenAI Efficien hyper parameter가 풍부

: AI : model의 hyper parameter가 풍부하다, pre-training
concept이 아닙니다. 성능↑ 학습률↑ 보다 빠르게 학습할 수 있다.

3.2 Input Representation

Input	[CLS]	my	dog	is	cute	[SEP]	he	likes	play	#ing	[SEP]
Token Embeddings	$E_{[\text{CLS}]}$	E_{my}	E_{dog}	E_{is}	E_{cute}	$E_{[\text{SEP}]}$	E_{he}	E_{likes}	E_{play}	$E_{\#ing}$	$E_{[\text{SEP}]}$
Segment Embeddings	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B	E_B
Position Embeddings	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}

- bertel input : 371개 embedding 간의 합으로 이루어져 있음.
- wordPiece embedding 사용 : BERT english \rightarrow 30000 개의 token 사용.
* 어떤 wordPiece embedding인가?

: Wordpiece tokenizing

He likes playing \rightarrow He likes *ing,

→ 입력문장을 tokenizing하고, 1 token을 1 token sequence로 만들어 계산에 사용.

→ BPE (Byte Pair Encoding) 알고리즘 이용

: BPE로 기본하여 단어를 의미하는 subword로 절친 tokenizing

- One sentence의 첫번째 token : [CLS] \rightarrow transformer 초기화를 가지고 나중 **token sequence의 classification 의도**를 가지게 된다.
→ 예전에 전방향 classifier를 볼이면 단일문장 / 연속된 문장의 classification을 동시에 하는 걸까 같다.
* classification task가 아니라 [CLS] token은 무시.

- Sentence pair는 두개의 Single sentence로 이루어짐

1st sentence \rightarrow 2nd sentence로 이루어져 있을 수도 있다

(ex) QA task의 경우 [Question, Paragraph] или Paragraph의 대답문)

그럼!, 두 문장을 각각의 토큰 ① [Seq] token 사용

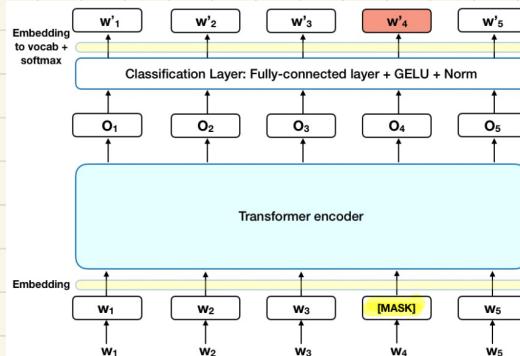
② Segment Embedding 사용

: 이미 문장에는 Sentence A embedding, 두 번째 문장에는 Sentence B embedding을 적용함. ↗ 문장이 1개면 예상 사용.

3.3 Pre-training task

: unsupervised prediction task pre-training

3.3.1 Task #1. Masked LM



<https://minio-park7.github.io/nlp/2018/12/12/bert-fbclid=lwAR3S-8lLWEVG6FGUVxoYdwQyAzG0GpOUzVesFBd0ARFg4eFXqCyGLznu7w>

: 입력 단어 중의 일부를 [MASK] token으로 바꿔놓음

15%

: Plain text \Rightarrow tokenization 하는 방법 \Rightarrow WordPiece

: 문장 전체를 predict X \rightarrow [MASK] token만 predict하는 pre-training task 수행.

\rightarrow [MASK] token은 pre-training 때만 사용!, fine-tuning 때는 $\text{15\%} \times$

\rightarrow token을 모으면서, 문맥을 파악하는 능력을 강화하기 위해

* 15%의 [MASK] Token은 인데일리, 예상 단어와 함께 훈련!

80% \rightarrow token \Rightarrow [MASK] 를 바꿈

15% \rightarrow .. random word3 바꿈

10% \rightarrow .. 유사한 단어 그대로 두자 \rightarrow 같은 단어에 대한 표본을 bias하게 유해

Pre-training 때 transformer encoder의 학습에서는

어떤 단어를 predict하는지 / 어떤 단어를 random word3 바꿨는지 알 수 X

\Rightarrow transformer encoder는 그동안 token의 distributional contextual representation을 유지하도록 한다.

3.3.2 Task #2 : Next Sentence prediction

- 이전하는 이유

• NLP task 다 QA / NLI (Natural Language Inference)는 두 문장 사이의 관계를 이해하는 것이 중요!

• corpus에서 두 문장을 뽑아서 이것이 원래의 corpus에서 연관된 문장인지 맞는 **binarized next sentence prediction task** 수행.

- 50% : sentence A, B가 같은 next sentence
50% : sentence A, B가 corpus에서 random으로 뽑은 (맞지 않는) 두 문장.
ex).

Input = [CLS] the man went to [MASK] store [SEP] he bought a gallon

[MASK] milk [SEP] Label = IsNext

Input = [CLS] the man [MASK] to the store [SEP] penguin [MASK] are
flight ##less birds [SEP] Label = NotNext

3.4 Pre-training Procedure

→ 기본적인 절차는 LM처럼 수행하는 것과 같음

→ BERT_English 경우 : Book corpus + Wikipedia

↳ text passage만 적용하여 사용

long contiguous sequences가 학습에 좋음.

- Input pre-processing

• NSP를 위해 sentence를 뽑아서 embedding A, B를 해줌 → 이 두 token이 힙처럼 길이는
(50% → 같은, 50% → random) ↳ Segment Embedding ≤ 512개 (OOM 때문)
• 오크, masking 추가!

- pre-training의 Hyper parameter

- batch size : 256 sequences (256 sequences * 512 tokens = 128,000 tokens/batch) for 1,000,000 steps
-> 3.3 billion word corpus의 40 epochs
- Adam optimizer, learning rate : 1e-4, $\beta_1 = 0.9$, $\beta_2 = 0.999$, L2 weight decay of 0.01, learning rate warmup over the first 10,000 steps, linear decay of the learning rate
- Dropout prob: 0.1 for all layers
- using gelu activation Hendrycks and Gimpel, 2016
- BERT_base - 4 TPUs, BERT_large - 16 TPUs를 사용하여 4일동안 학습

3.5 Fine-tuning procedure.