BERT(Bidirectional Encoder Representations from Transformer)

Introduction

Pre-trained Language Model은 자연어 처리 task의 성능을 향상시킬 수 있다.

PLM(pre-trained language model)을 적용하는 방법은 현재 크게 2가지로 구분된다

1) Feature-based 방식 (대표적으로 ELMO)

특정한 task에 알맞는 구조를 보유하고, pre-trained representation을 특성으로 추가해서 사용하는 구조

2) Fine-tuning 방식 (대표적으로 GPT)

특정 task에 특화된 파라미터를 최소화 하고, 사전 학습된 파라미터를 fine-tuning 한다.

두 방식 모두 pre-training 하는 과정에서는 동일한 objective function(목적함수)를 사용하고, 동일하게 unidirectional language model (단방향 언어모델) 을 사용한다.

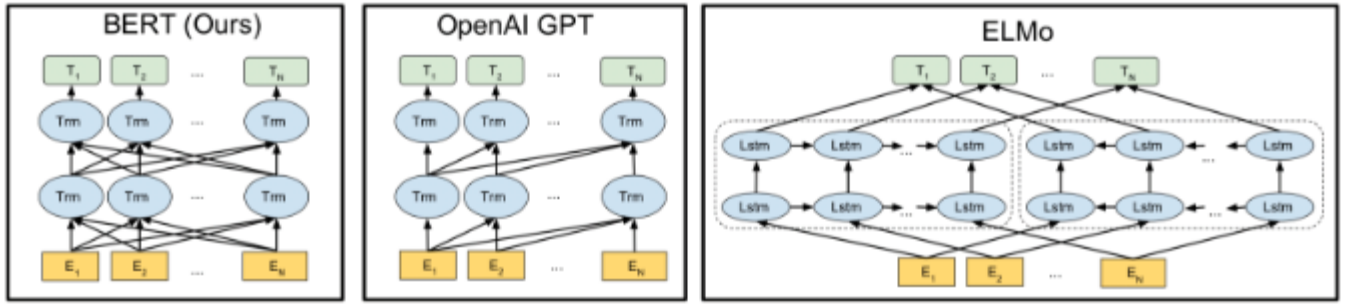
- GPT는 Transformer를 사용한 구조이기는 하지만, Decoder의 사용하기 때문에 이전 토큰만을 고려하여 언어 모델을 구성한다.
- ELMo는 biLSTM을 사용하지만, forward, backward와 같은 일련의 방향이 존재하는 언어모델 입니다. 또한 얕은 concatenate를 통해 언어 모델을 구성=> 따라서 깊은 양방향 특징을 학습하기에는 최선의 방법X

이러한 이유로 단방향의 언어모델은 pre-trained representation의 성능을 저해하는 요소가 될 수 있다. 특히 fine-tuning approach의 경우, 사전 학습 과정에서 단방향의 언어모델이 다양한 언어 구조의 선택 가능성을 제한할 수 있다.

Introduction

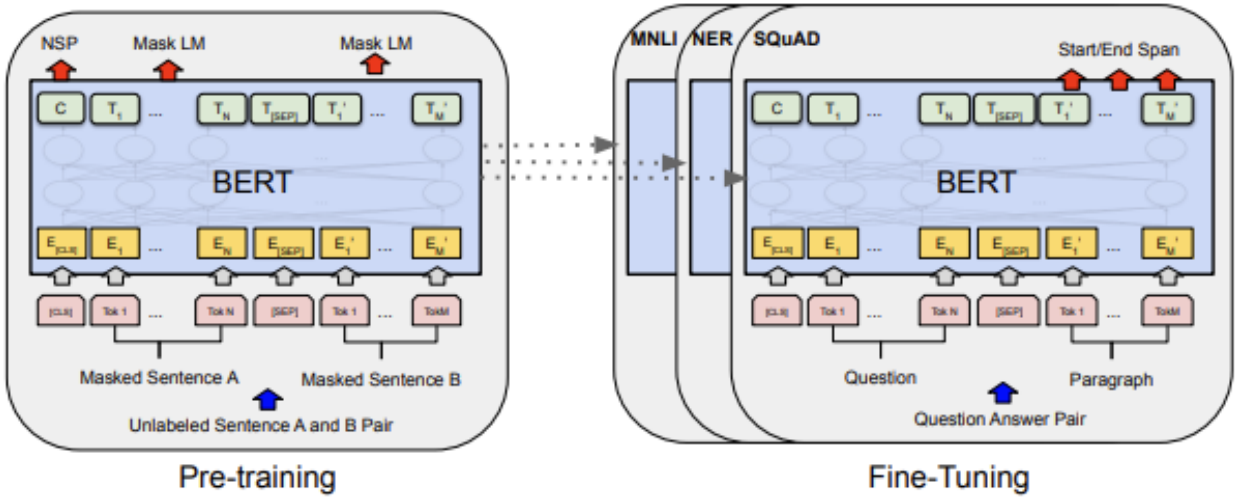
그래서 이 연구에서는 fine-tuning 방식을 개선하는 BERT (Bidirectional Encoder Representations from Transformers) 를 제안한다.

- BERT는 masked language model (MLM) 을 사용해서 성능을 향상시킨다.
- MLM은 임의로 몇 개의 토큰을 mask하고, mask 된 토큰을 예측하는 것을 목표로 한다. 이러한 방식은 기존의 left-to-right 구조와는 다르게 왼쪽 문맥과 오른쪽 문맥을 모두 학습할 수 있도록 한다.



- 추가적으로, next-sentence-prediction(NSP) task를 함께 수행하게 된다.

Bert



Bert는 크게 pre-training 단계와 fine-tuning 단계, 두가지 단계로 구분한다.

- Pre-training 단계에서는 레이블링 하지 않는 데이터를 기반으로 학습을 진행하고, Fine-tuning 과정에서 모델은 pre-train된 파라미터로 초기화된다.
- 이후 모델을 레이블링된 데이터로 fine-tuning 한다.

=> 실제 task에서 사용하는 모델은 초기에 동일한 파라미터로 시작하지만, 최종적으로는 서로 다른 fine-tuned 된 모델을 보유하게 된다. BERT 는 pre-trained 된 모델과 fine-tuned 된 모델 사이의 구조적 차이가 거의 없게 된다.

Bert

BERT는 multi-layer bidirectional Transformer encoder를 사용한다.

모델의 크기에 따라 base 모델과 large 모델을 제공한다.

- BERT_base : L=12, H=768, A=12, Total Parameters = 110M
- BERT_large : L=24, H=1024, A=16, Total Parameters = 340M

*L : transformer block의 layer 수

*H : hidden size

*A : self-attention heads 수

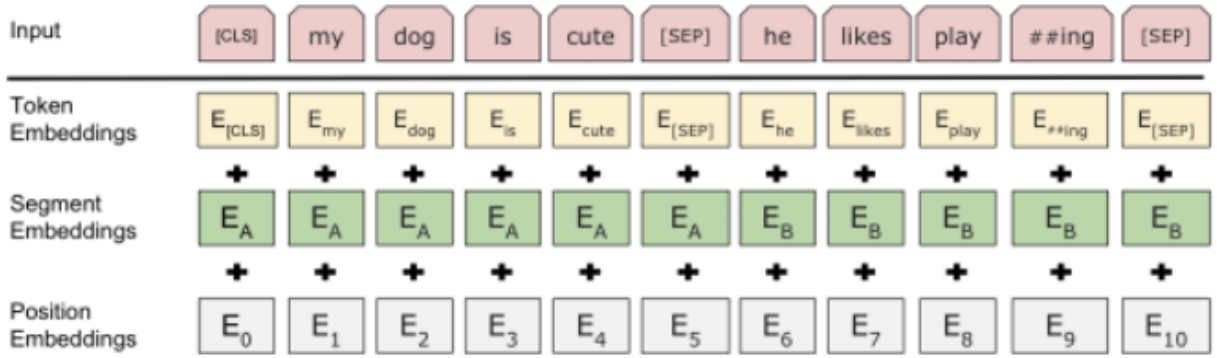
BERT_base의 경우 GPT 와의 비교를 목적으로 동일한 크기로 제작함.

둘의 차이점)

BERT: 현재 토큰의 좌/우를 모두 참조할 수 있는 bidirectional self-attention을 수행함

GPT: 현재 토큰의 왼쪽에 있는 문맥만 참조가 가능함

Bert: Input/Output Representations



BERT의 INPUT은 3가지 embedding 값의 합으로 이루어져있다.

Token Embedding: wordpiece embedding을 사용(gpt는 bpe사용)

- Sequence 의 첫 토큰은 [CLS] 라는 토큰을 사용함.
  여기에 간단한 classifier를 붙이면 단일 문장, 또는 연속된 문장의 classification을 쉽게 할 수 있게됨.
- 쌍으로 구성된 문장들도 하나의 sequence로 표현됨.
- 두 문장을 하나의 sequence 로 표현하기 위해, 먼저 두 문장을 [SEP]라는 토큰으로 구분하고, 각 문장의 단어들을 임베딩으로 표현함.

Segment Embedding: BERT는 NSP를 위해 두개의 문장을 함께 사용한다.

- 단어가 첫번째 문장에 속하는지 두번째 문장에 속하는지 알려준다.
- [0 0 0 0 0 0 1 1 1 1] 과 같이 표현하며 해당 Vector를 Token Embedding 차원수와 같게 맞추어 임베딩 해준다 (학습 가능함)

Position Embedding: 트랜스포머와 같은 방식의 위치 임베딩

wordpiece
전체 글자 중 각 단어가 따로 등장한 것을 분모로 잡고 분자로는 같이 등장한 빈도수로 설정해 우도로 단어 병합
Bpe
단어의 pair의 빈도수 순서대로 병합

BERT

Pre-training BERT

전통적인 left-to-right/right-to-left LM을 사용해서 pre-train하는 ELMo, GPT와는 다르게, BERT는 2개의 unsupervised task를 이용해서 학습한다.

Task #1 : Masked LM

Bidirectional 하게 처리하면(양방향) 간접적으로 예측하려는 단어를 참조하게 되고, multi-layered 구조에서 간접 참조한 단어를 예측할 수 있다. 그래서 bert는 일정 비율의 token을 mask하게 된다.(전체의 15%, token은 pre-trainin에만 사용되고, fine-tunin 시에는 사용되지 않음)

Mask 토큰은 다음과 같이 몇가지 rule을 적용한다.

- 80%의 경우 : token을 [MASK]로 바꾼다. eg., my dog is hairy -> my dog is [MASK]
- 10%의 경우 : token을 random word로 바꾼다. eg., my dog is hariy -> my dog is apple
- 10%의 경우 : token을 원래의 단어로 그대로 놔둔다.

최종적으로 cross-entropy loss 를 사용해서 기존의 토큰을 예측하도록 학습된다.

BERT

Task #2 : Next Sentence Prediction (NSP)

- Question-answering(QA), Natural Language Interference(NLI) 등의 task는 두 문장 사이의 관계를 이해해야 하는 task이다.
- LM을 통해서 학습하기 쉽지 않기 때문에, NSP라는 task에 대해서도 함께 학습을 진행한다.

학습 방법)

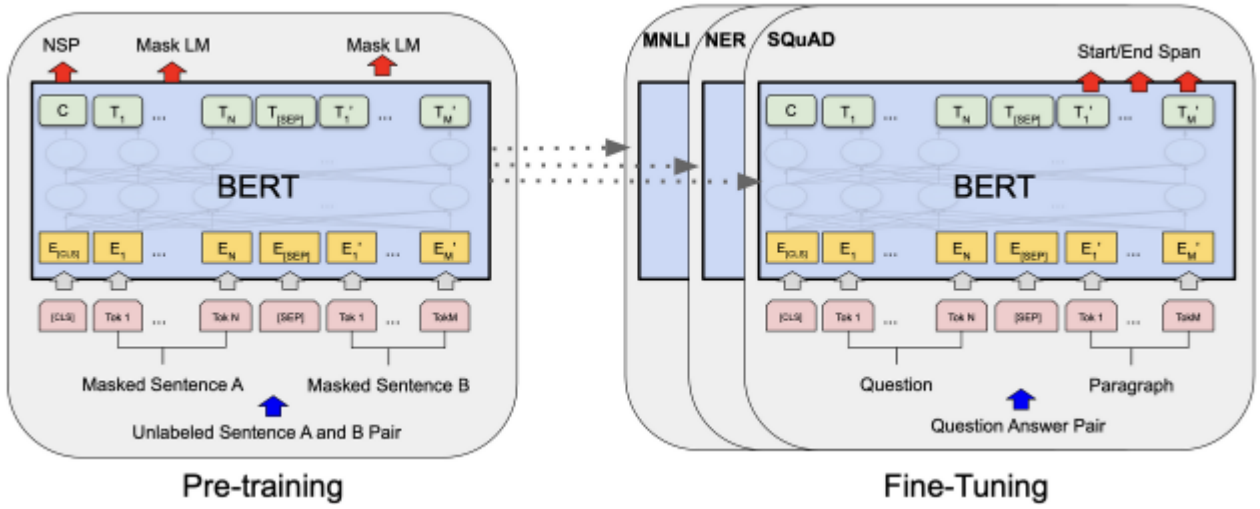
1. 두 문장 A, B를 입력
2. 이때, 50%의 경우 두 번째 문장인 B는 실제로 A의 다음 문장으로 구성되고, 나머지 50%의 경우 B는 전혀 관계가 없는 임의의 문장으로 제공
3. B가 다음 문장인 경우 IsNext, 임의의 문장인 경우 NotNext 라고 레이블링 하고 BInary classification을 학습한다.

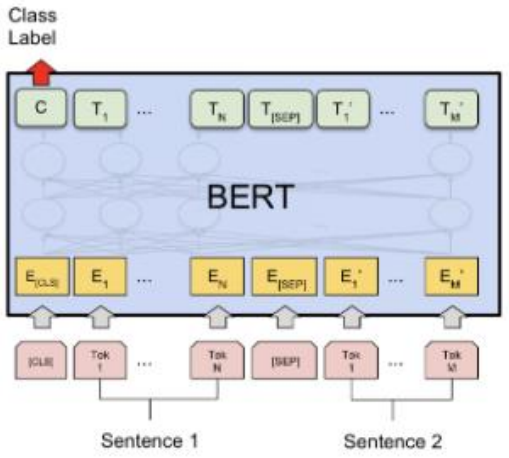
Pre-training Data

Pre-training corpus로는 다음과 같은 데이터를 사용했다. Wikipedia의 경우 text passage 만 사용했고, 목록이나 표 등은 모두 제외했다. 긴 문맥을 학습하기 위해서 Billion Word Benchmark 와 같이 섞인 문장으로 구성된 데이터를 사용하지 않았다.

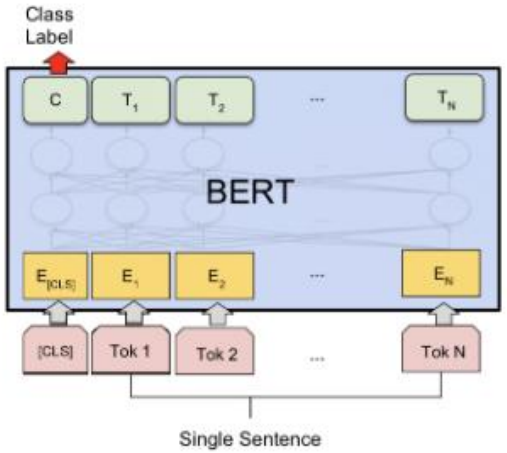
- BooksCorpus (800M words)
- English Wikipedia (2,500M words)

BERT





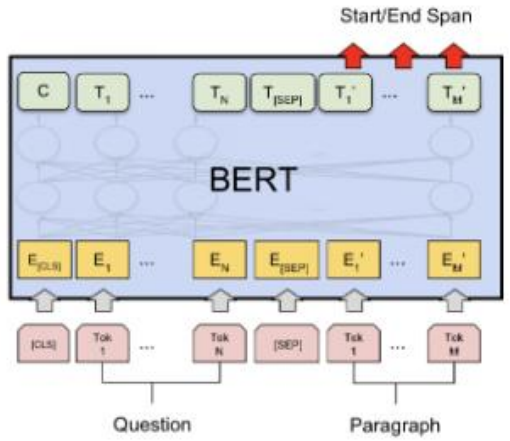
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



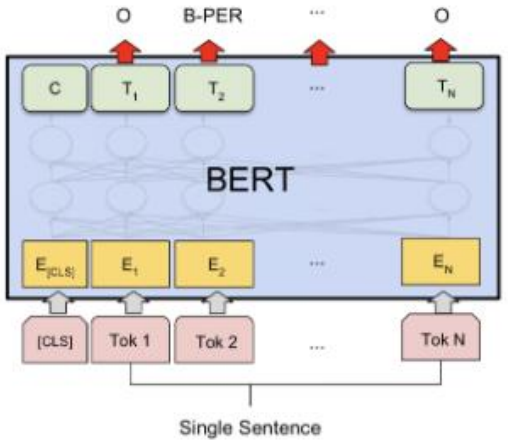
(b) Single Sentence Classification Tasks: SST-2, CoLA

Fine-tuning

- task에 알맞는 입력과 출력을 모델에 입력으로 제공해서 파라미터들을 해당 task에 맞게 end-to-end로 업데이트한다. 이 때, [CLS] 토큰은 classification을 수행하기 위해 사용된다
- Sequence tagging(d)과 QA(c) 같은 토큰 단위의 down-stream task는 토큰의 representation을 output layer로 넘겨주는 형태로 학습이 이루어진다.
- 감성분석(b), entailment(a)와 같은 classification task는 NSP 방법 그대로 [CLS] 토큰의 representation을 output layer로 넘겨주어 학습하면 된다.
- (d)의 경우는 Named Entity Recognition(NER)이나 형태소 분석과 같이 single sentence에서 각 토큰이 어떤 class를 갖는지 모두 classifier 적용하여 정답을 찾아낸다.



(c) Question Answering Tasks: SQuAD v1.1



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Fine-tuning 시의 dataset의 크기가 클수록 hyperparameter에 영향을 덜 받고 잘 training 된다고 본 연구에서는 말한다.

Experiments

BERT fine-tuning을 이용한 11개의 NLP task를 수행하였다.

대표적인 GLUE benchmark dataset 실험

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$BERT_{BASE}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$BERT_{LARGE}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

- 모든 task에 대해 SOTA를 달성
  (문장들은 문법적으로 옳은지, 그른지, 리뷰의 감성분석, 2개의 문장이 의미적으로 같은지 다른지 등등의 task)
- 데이터의 크기가 작아도 fine-tuning 때는 좋은 성능을 낼 수 있다.

Ablation Studies

Tasks	Dev Set MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
$BERT_{BASE}$	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

No NSP

masked LM(MLM) 으로만 학습되고 NSP는 사용하지 않는 경우 성능이 떨어짐을 알 수 있다.

LTR & No NSP

left-context-only model 을 사용하고 NSP도 사용하지 않는 경우 모든 task에 대해서 MLM에 비해 성능이 떨어진다.