23.07.13 _ 5주차

딥러닝 논문 요약 및 구현 스터디

발표자

윤상현 (버디)

BERT

Pre-training of Deep Bidirectional Transformers for Language Understanding

들어가기 앞서,

BERT의 등장 배경

기존 SOTA 모델이었던, ELMo와 GPT의 단점을 보완하기 위해 만들어짐

왼쪽에서 오른쪽으로만 읽으며 학습했던 기존의 Language Model(이하 LM)은
문장의 문맥 이하에 한계가 존재했음 -> **Idea!**

Idea : 양방향으로 학습하여 문맥을 더 잘 이해하도록 만들자!

좋은 논문으로 평가되는 이유 2가지

1. **기존 모델들의 단점을 보완 / Fine-Tuning이 쉬움**
2. **자연어 처리 Task에서 매우 좋은 성능**

Abstract

Abstract

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement).

요약

- BERT는 양방향으로 문맥을 고려하는 새로운 언어 표현 모델이다.
- BERT는 Transfomer의 incoder 구조와 MLM, NSP 기법을 사용한다.
- Fine-tuning이 ELMo와 GPT보다 쉽다.
- 여러 벤치마크에서 최고의 성능을 달성했다.

I. Introduction

Byte Pair Encoding

- 단어를 임베딩하는데 큰 문제점은 모든 단어를 token화시키고 학습시킬 수 없다는 것
- 사람 조차도 모르는 단어가 나오면 찾아보게 되는데 AI는 데이터가 없는 부분에 대한 질문은 매우 취약할 수 밖에 없음
- 이러한 상황에서 보통 모델은 OOV(Out-Of-Vocabulary)을 발생
- 이를 위해 나타난 서브워드 분리(Subword segmenation) 작업은 하나의 단어는 더 작은 단위의 의미있는 여러 서브워드들 (Ex) birthplace = birth + place)의 조합으로 구성된 경우가 많음
- 하나의 단어를 여러 서브워드로 분리해서 단어를 인코딩 및 임베딩하겠다는 의도를 가진 전처리 작업
- 이때, 대표적인 것이 BPE

WordPiece Tokenizer

- BPE의 변형 알고리즘
- 가장 먼저 BPE처럼 모든 단어를 나누고 그 후에 가능도에 따라 단어들을 병합
- WordPiece Tokenizer는 모든 단어의 맨 앞에 _를 붙이고, 단어는 서브 워드(subword)로 통계에 기반하여 띄어쓰기로 분리
  - 언더바는 문장 복원을 위한 장치
  - Jet → _J et
- WordPiece Tokenizer이 수행된 결과로부터 다시 수행 전의 결과로 돌리는 방법
  - 현재 있는 모든 띄어쓰기를 전부 제거하고, 언더바를 띄어쓰기로 바꾸기
- 언더바는 기존의 문장에 있었던 띄어쓰기를 보여주는 장치

I. Introduction

- 자연어 처리 분야에서는 대부분의 최신 모델들이 사전 학습된 언어 표현을 사용한다.

- 사전 학습된 언어 표현 모델들은 정적인 모델과 동적인 모델로 나뉘며,
  각각 특징 기반 접근법과 미세 조정 기반 접근법을 사용한다.

- 기존의 사전 학습된 언어 표현 모델들은 단방향이거나 얕은 양방향으로 제한되어 있다.

- 이 논문에서는 BERT라는 새로운 언어 표현 모델을 제안한다.

- BERT는 깊은 양방향 표현을 사전 학습하기 위해 트랜스포머의 인코더와
  Masked Language Model, Next Sentence Prediction이라는 두 가지 방법을 사용한다.

- BERT는 다양한 자연어 처리 작업에 적용할 수 있으며, 높은 성능을 보여준다.

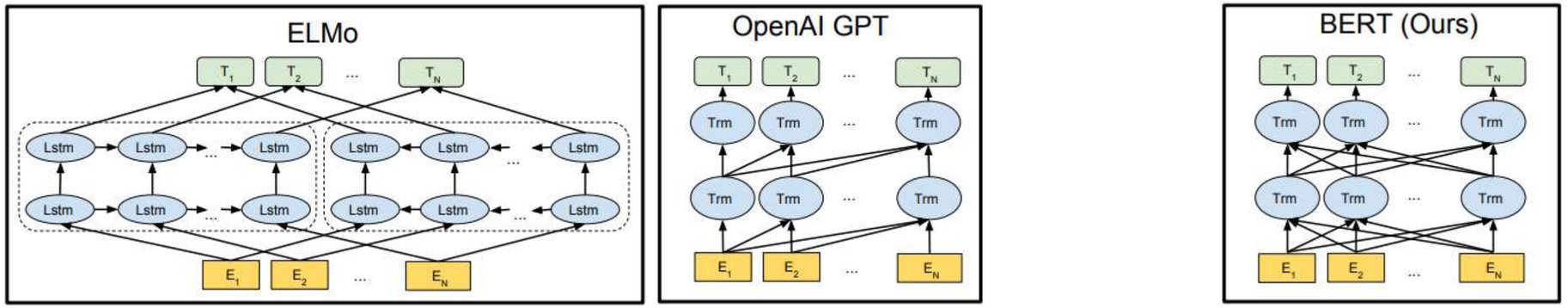
II. Related Work

1. Unsupervised Feature-based 접근 방법

- Unsupervised Feature-based 접근 방법은 사전 학습된 언어 모델에서
  얻은 특징을 추가적인 작업별 구조에 입력으로 사용하는 방법

- ELMo는 LSTM 기반의 언어 모델을 사용하며,
  각 층에서 **왼쪽 문맥과 오른쪽 문맥을 따로따로 고려**하는 얕은 양방향 모델

- ELMo는 다양한 자연어 처리 작업에서 성능 향상을 보여주었으며,
  사전 학습된 언어 모델의 내부 표현이 유용하다는 것을 보여줌



II. Related Work

2. Unsupervised Fine-tuning 접근 방법

- 사전 학습된 언어 모델의 파라미터를 고정하지 않고,
  모든 파라미터를 작업별 데이터셋으로 미세 조정하는 방법

- GPT는 사전 학습된 언어 모델에 하나의 추가적인
  선형 출력층만을 붙여서 미세 조정하는 방식으로 작동

- GPT는 트랜스포머 기반의 언어 모델을 사용하며,
  왼쪽 문맥만을 고려하는 단방향 모델

- GPT는 ELMo보다 더 깊고 넓은 신경망 구조를 사용하며,
  특징 기반 접근법보다 **미세 조정 기반 접근법이 더 효과적**이라는 것을 보여줌

II. Related Work

3. 전이 학습 from Supervised Data

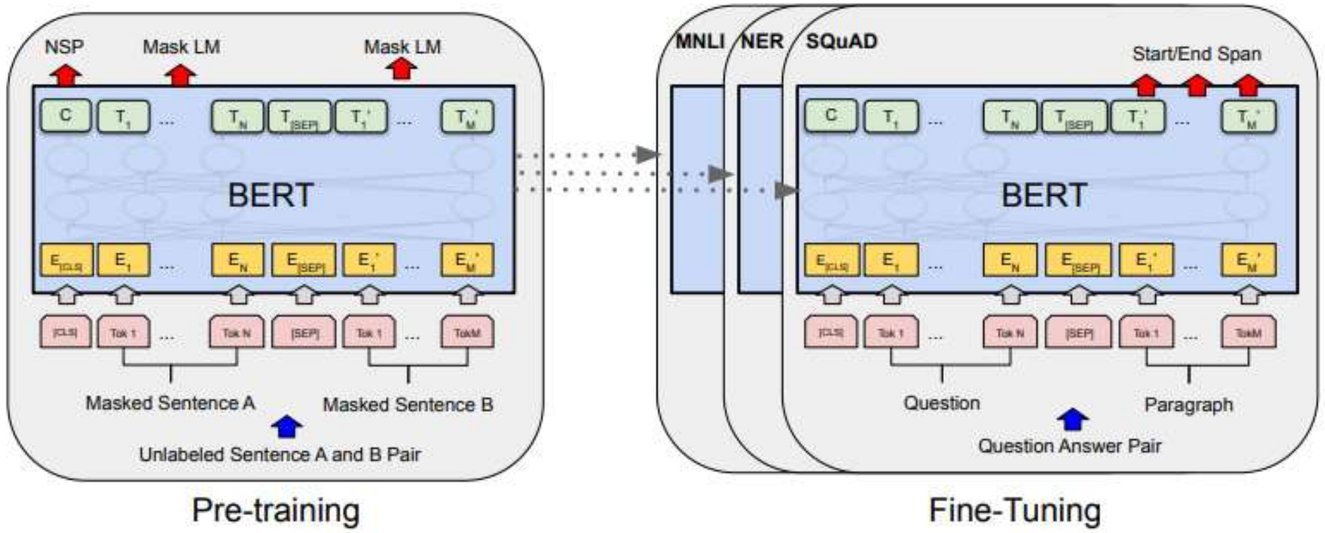
- 전이 학습 from Supervised Data은 사전 학습된 언어 모델을 레이블이 있는 데이터로 추가적으로 학습하는 방법

- ULMFiT는 사전 학습된 언어 모델을 작업별 데이터로 세 단계로 미세 조정하는 방식으로 작동

- ULMFiT는 AWD-LSTM 기반의 언어 모델을 사용하며, 각 단계에서 다른 학습률과 드롭아웃 비율을 적용

- ULMFiT는 작업별 데이터가 적은 경우에도 높은 성능을 보여주었으며, **사전 학습된 언어 모델에 추가적인 지도학습이 유리**하다는 것을 보여줌

III. BERT - Model Architecture

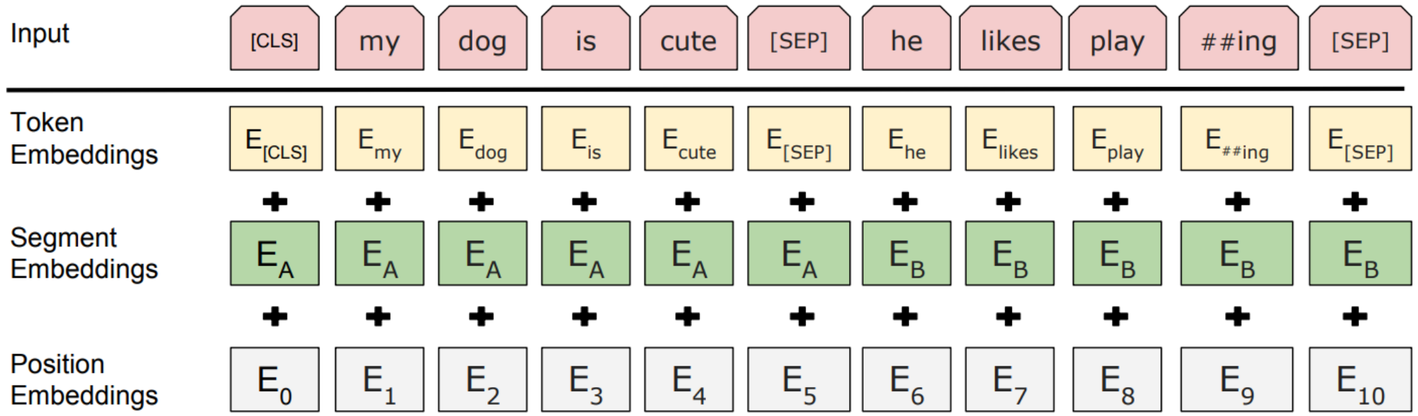


L : 레이어의 수
H : 히든 사이즈
A : self-attention heads

- transformer 중에서도 encoder 부분만을 사용
- BERT_base : L=12, H=768, A=12, Total Parameters = 110M
- BERT_large : L=24, H=1024, A=16, Total Parameters = 340M

- BERT_base 모델의 경우, OpenAI GPT모델과 hyper-parameter가 동일
- Pre-training concept을 바꾸어 주는 것만으로도 훨씬 높은 성능

III. BERT - Input/Output Representations



L : 레이어의 수
H : 히든 사이즈
A : self-attention heads

- transformer 중에서도 encoder 부분만을 사용
- BERT_base : L=12, H=768, A=12, Total Parameters = 110M
- BERT_large : L=24, H=1024, A=16, Total Parameters = 340M

- BERT_base 모델의 경우, OpenAI GPT모델과 hyper-parameter가 동일
- Pre-training concept을 바꾸어 주는 것만으로도 훨씬 높은 성능

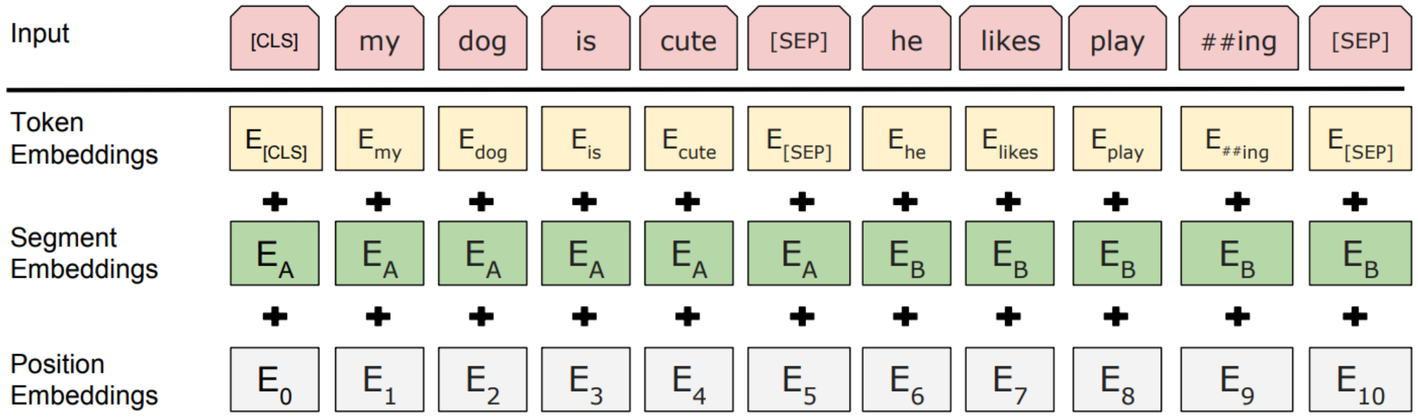
BERT에서 사용하는 **tokenizer는 WordPiece 임베딩 방식**

'문장(Senetence)' 는 실제 언어의 문장 대신에 유사한 텍스트의 임의 범위가 될 수 있음
'시퀀스(Sequence)' 는 BERT에 대한 입력 토큰 시퀀스를 말하며 두 개의 문장을 함께 패킹(합친)한 것일 수 있음

Rule
1. 문장의 쌍은 하나의 시퀀스로 함께 묶임
    1. 토큰 ([SEP], [SEP])을 통해 기존의 문장들을 분리
    2. 모든 토큰에 이것이 문장 A인지 B인지 표시하는 학습된 임베딩을 추가

2. 모든 문장의 첫 번째 토큰은 항상 [CLS]
3. embedding을 E로 표시
4. 각 단어의 tokenizer embedding, Segment Embedding(문장 쌍이라면 어디 속하는지), 그리고 Position Embedding

III. BERT - Input/Output Representations



L : 레이어의 수
H : 히든 사이즈
A : self-attention heads

- transformer 중에서도 encoder 부분만을 사용
- BERT_base : L=12, H=768, A=12, Total Parameters = 110M
- BERT_large : L=24, H=1024, A=16, Total Parameters = 340M

- BERT_base 모델의 경우, OpenAI GPT모델과 hyper-parameter가 동일
- Pre-training concept을 바꾸어 주는 것만으로도 훨씬 높은 성능

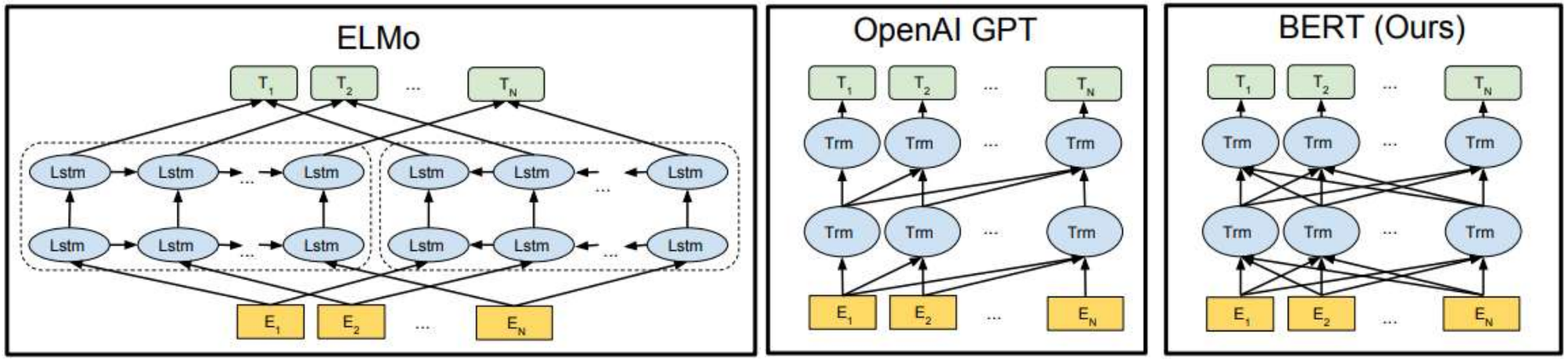
4. 각 단어의 tokenizer embedding, Segment Embedding(문장 쌍이라면 어디 속하는지), 그리고 Position Embedding

1. Token(WordPiece) Embedding : 실질적인 입력이 되는 워드 임베딩. 임베딩 벡터의 종류는 단어 집합의 크기로 30,522개.

2. Position Embedding : 위치 정보를 학습하기 위한 임베딩. 임베딩 벡터의 종류는 문장의 최대 길이인 512개.

3. Segment Embedding : 두 개의 문장을 구분하기 위한 임베딩. 임베딩 벡터의 종류는 문장의 최대 개수인 2개.

III. BERT - Pre-training



핵심 idea : 두 가지 **방향**과 두 가지 **비지도 학습**

Masked Language Modeling, MLM	**Next Sentence Prediction**, NSP
임의로 선택된 단어를 마스크(가리기)하고, 그 단어를 예측하는 과정 Ex) I made a bank deposit "bank"라는 단어를 마스크하고, "[MASK]"라는 토큰으로 대체한 후, "[MASK]"가 "bank"라는 단어임을 예측하는 것 → 모델은 문장의 양쪽에 있는 단어들을 모두 참고하여 문맥을 파악할 수 있음	두 개의 문장이 주어졌을 때, 두 번째 문장이 첫 번째 문장의 다음에 오는 문장인지 아닌지를 예측하는 과정 Ex) I made a bank deposit. He went to the park. 두 번째 문장이 첫 번째 문장과 관련이 없으므로 NSP는 "아니오"라고 예측 → 모델은 두 문장 사이의 관계를 파악

III. BERT - Pre-training

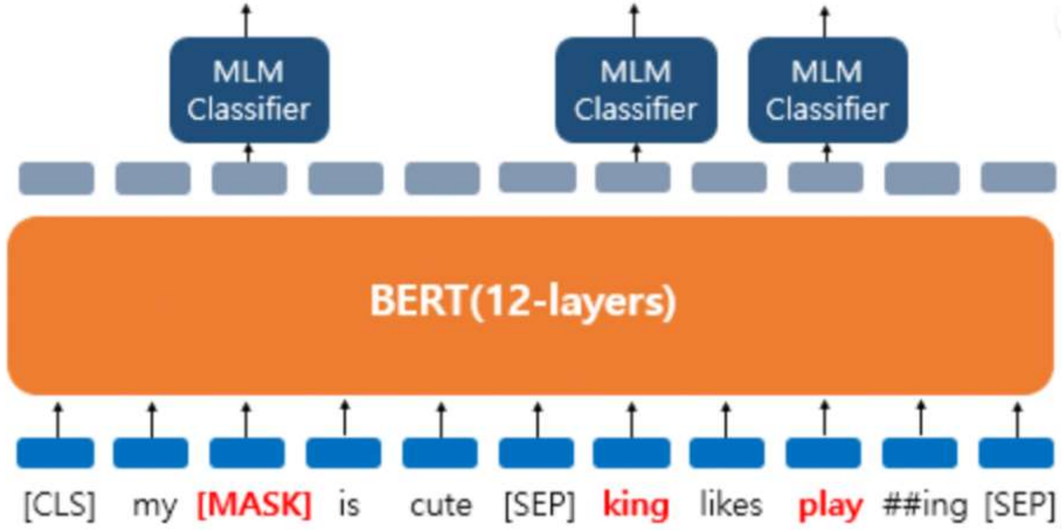
Masked LM (MLM)

사용이유

언어 모델이 아무런 제약조건 없이 Bidirectional하게 학습을 하게 되면 간접적으로 예측하려는 단어를 참조하게 되고,
multi-layered 구조에서 해당 단어를 예측할 수 있게 된다. 즉, 학습이 제대로 되지 않음

해결 방안

다음 단어가 무엇이 오는지 예측하는 학습이 아니라, 문장 내에서 무작위로 입력 토큰의 일정 비율을 마스킹하고, 그 다음 마스킹된 토큰들을 예측
이 과정을 MLM 이라고 함.



마스크 토큰에 해당하는 마지막 hidden vector는
표준 LM에서와 같이 어휘를 통해 출력 소프트맥스로 주어지고 단어를 예측

BERT는 문장을 복구하는 것이 목표가 아닌 해당 단어의 예측이 목표
BERT는 WordPiece 토큰의 15%를 무작위로 각 시퀀스에서 마스킹

If, i 번째 토큰이 선택
(1) 80%는 [MASK] 토큰으로 교체하거나
(2) 10%는 다른 토큰(단어)으로 교체하거나,
(3) 10%는 변경되지 않은 i 번째 토큰을 사용한다.

fine-tuning 중에 [MASK] 토큰이 나타나지 않기 때문에,
사전 훈련과 fine-tuning 사이에 불일치를 만들어내는 단점

III. BERT - Pre-training

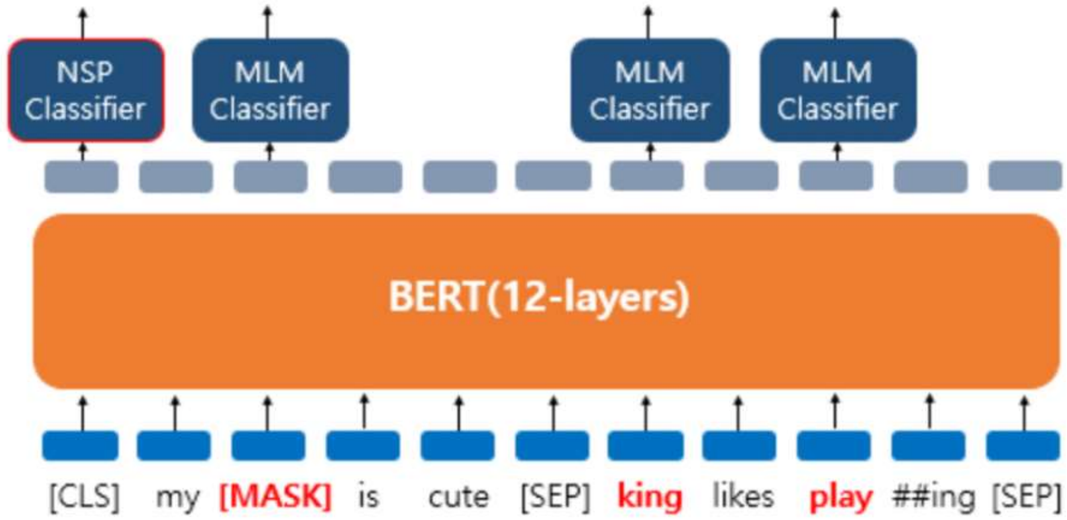
Next Sentence Prediction (NSP)

사용이유

Question Answering Task를 위해 두개의 문장을 제공, 이어지는지 분류되는지 훈련함
이때, 문장들은 50% 확률로 이어지는 문장들

구성 요소

[SEP] – 각 문장이 다른 문장임을 보여주기 위해서 문장과 문장 사이에
[CLS] – 가장 앞에 있는 토큰



트랜스포머 인코더를 거친 후, [CLS] 토큰의 표현을 출력층에 넣고, 이진 분류를 수행
출력층은 시그모이드 함수와 크로스 엔트로피 손실 함수를 사용
만약 B가 A의 다음에 오는 문장이라면 1, 아니라면 0을 출력

NSP를 통해 BERT는 질의 응답, 자연어 추론, 텍스트 요약 등과 같은 문단
수준의 작업에 유용한 지식을 습득

논문의 Ablation Study에서는 해당 NSP 훈련을 하지 않을 경우 모델의
성능이 많이 감소한다고 명시

III. BERT - Fine-tuning BERT

How?

BERT 모델 자체가 Transformer를 통해 Attention Encoding 진행
그 모델을 다시 사용해서 Task에 맞게 문제를 해결하도록 fine-tuning하기 때문에 큰 조작을 가하지 않는다.

문제에 따라 Fine-tuning 하는 방식이 달라지게 되고 논문에서는 네 개의 방법을 제시

Paraphrasing

문장 A와 문장 B가 동의어인지 판단하는 작업

Hypothesis-premise pairs in entailment

문장 A가 문장 B의 추론인지 판단하는 작업

Question answering

문장 A에 대한 답변이 문장 B에 있는지 판단하고,
답변의 위치를 찾는 작업

tagging, text pair classification

문장 A의 텍스트 분류나 시퀀스 태깅을 하는 작업

1. 사전 훈련된 BERT 모델을 로드

2. 작업에 맞게 입력과 출력을 조정.
입력으로는 문장 A와 문장 B를 [SEP] 토큰으로 연결,
[CLS] 토큰을 앞에 붙임.
출력으로는 토큰 단위의 작업에서는 각 토큰의 표현을 출력층에 넣고,
분류 작업에서는 [CLS] 토큰의 표현을 출력층에 넣음

3. 작업에 해당하는 레이블이 있는 데이터를 준비
데이터는 입력과 출력의 쌍으로 구성.

4. 모델의 파라미터를 업데이트하기 위해 데이터를 이용하여 학습
학습 시에는 손실 함수와 최적화 알고리즘을 사용

5. 학습된 모델을 평가하고 성능을 측정
평가 시에는 정확도, F1 점수 등의 지표를 사용

III. BERT - Fine-tuning BERT

Paraphrasing

How?

두 문장이 동의어인지 아닌지를 나타내는 이진 레이블을 추가

Ex) "The man jumped over the fence"와 "The fence was jumped over by the man"은 동의어이므로 1을 레이블로 붙인다.
반면, "The man jumped over the fence"와 "The man climbed over the wall"은 동의어가 아니므로 0을 레이블로 붙인다.

Hypothesis-premise pairs in entailment

How?

두 문장이 추론 관계인지 모순 관계인지 중립 관계인지를 나타내는 다중 레이블을 추가

Ex) "He is a teacher"와 "He works at a school"은 추론 관계이므로 entailment를 레이블로 붙인다.
반면, "He is a teacher"와 "He works at a hospital"은 모순 관계이므로 contradiction을 레이블로 붙인다.
그리고, "He is a teacher"와 "He likes coffee"은 중립 관계이므로 neutral을 레이블로 붙인다.

III. BERT - Fine-tuning BERT

Question answering

How?

답변의 시작과 끝 위치를 나타내는 정수 쌍을 추가

Ex) "Who is the president of the United States?"와 "Joe Biden is the president of the United States"가 주어졌을 때,
답변은 "Joe Biden"이므로 시작 위치는 0, 끝 위치는 1을 레이블로 붙인다.
만약 답변이 없다면 시작과 끝 위치를 모두 0으로 붙인다.

Tagging, text pair classification

How?

텍스트 분류 작업에서는 문장의 카테고리를 나타내는 다중 레이블을 추가

Ex) "I love this movie"는 감정 분석 작업에서 긍정적인 문장이므로 positive를 레이블로 붙인다.
시퀀스 태깅 작업에서는 각 토큰의 태그를 나타내는 다중 레이블을 추가.
예를 들어, "I love this movie"는 품사 태깅 작업에서 PRON VERB DET NOUN라는 태그를 레이블로 붙인다.

IV. Experiments

GLUE

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
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

SQuAD 1.1

System	Dev EM	Dev F1	Test EM	Test F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
Published				
BiDAF+ELMo (Single)	-	85.6	-	85.8
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

SQuAD 2.0

System	Dev EM	Dev F1	Test EM	Test F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	86.3	89.0	86.9	89.5
#1 Single - MIR-MRC (F-Net)	-	-	74.8	78.0
#2 Single - nlnet	-	-	74.2	77.1
Published				
unet (Ensemble)	-	-	71.4	74.9
SLQA+ (Single)	-		71.4	74.4
Ours				
$\text{BERT}_{\text{LARGE}}$ (Single)	78.7	81.9	80.0	83.1

SWAG

System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
OpenAI GPT	-	78.0
$\text{BERT}_{\text{BASE}}$	81.6	-
$\text{BERT}_{\text{LARGE}}$	**86.6**	**86.3**
Human (expert)†	-	85.0
Human (5 annotations)†	-	88.0

V. Ablation Studies

Ablation = 학습이 사용되는데 AI 시스템의 일부를 제거한 것

제안한 요소가 모델에 어떠한 영향을 미치는지 확인하고 싶을 때,
이 요소를 포함한 모델과 포함하지 않은 모델을 비교하는 것
시스템의 인과관계(causality)를 간단히 알아볼 수 있음

System	Dev F1	Test F1
ELMo (Peters et al., 2018a)	95.7	92.2
CVT (Clark et al., 2018)	-	92.6
CSE (Akbik et al., 2018)	-	**93.1**
Fine-tuning approach		
$\text{BERT}_{\text{LARGE}}$	96.6	92.8
$\text{BERT}_{\text{BASE}}$	96.4	92.4
Feature-based approach ($\text{BERT}_{\text{BASE}}$)		
Embeddings	91.0	-
Second-to-Last Hidden	95.6	-
Last Hidden	94.9	-
Weighted Sum Last Four Hidden	95.9	-
Concat Last Four Hidden	96.1	-
Weighted Sum All 12 Layers	95.5	-

1)Embedding만 사용
2)두번째 부터 마지막 Hidden을 사용
3)마지막 Hidden 만을 사용
4)마지막 4개의 Hidden을 가중합
5)마지막 4개의 Hidden을 concat
6)모든 12개의 층의 값을 가중합