Bert 논문 리뷰 (투빅스 8주차 NLP 과제)

논문 : BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding

1. Abstract

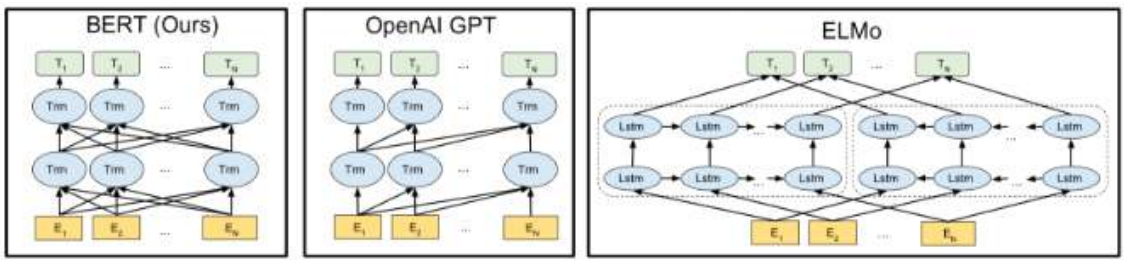
해당 논문에서는 Bidirectional Encoder Representations from Transformers 줄여서 BERT라는 모델에 대한 소개를 했다. 일반적으로 BERT는 unlabeled 데이터에 대해서 사전학습을 진행한 후 특정 task를 가지고 있는 labeled 데이터로 fine-tuning하는 모델이다. 그리고 하나의 output layer만으로 pre-trained BERT 모델에 추가하면 기존의 모델들에 비해 더 단순하면서도 좋은 성능을 보여준다.

2. Introduction

언어 모델들은 문장 전체의 task 또는 token 단위의 task를 분석하여 예측하는 것을 목표로 한다. 이때 미리 훈련된 언어표현들을 down stream task에 적용하는 기존 전략에는 2가지가 존재한다.

첫째로는 ELMo와 같은 feature based approach이며 두 번째로는 GPT와 같은 fine-tuning approach이다. 두 방법 모두 pre-training 시 동일한 목적함수로 학습을 수행하는데 특히 fine-tuning approach에서 pre-trained representation을 제한한다고 말한다.

아래의 그림이 기존의 2가지 approach의 예시인 GPT와 ELMo, 그리고 논문에서 강조한 BERT의 구조이다.



ELMo, GPT는 일반적인 language model을 사용하고 있고 이는 모두 단방향에 의존하는 모델이다. 물론 ELMo에서는 Bi-LSTM으로 양방향성을 가지려고 노력하지만, 순방향과 역방향의 출력을 concat하여 사용하기 때문에 각각의 모델 자체는 단방향이라고 말할 수 있다.

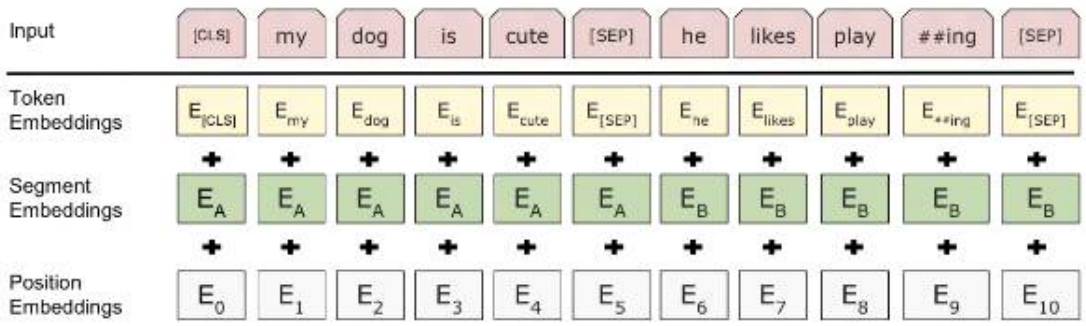
즉, 단방향 language model의 경우에 그 성능을 떨어뜨린다고 말하고 있다. 따라서 deep bidirectional context가 중요하며 이를 만족하는 BERT를 소개하였다.

BERT의 2가지 특징 중 하나는 Masked Language Model(MLM)이다. BERT에서는 input 전체와 masked된 token을 한번에 transformer encoder에 넣고 원래 token 값을 예측하기 때문에 deep bidirectional하다고 할 수 있다. 그리고 또다른 특징은 next sentence prediction이다. 이는 두 문장을 pre-training시에 같이 넣어줘서 두 문장이 이어지는 문장인지를 예측해주는 것이다.

3. BERT

BERT는 Pre-training part와 Fine-tuning part로 나눌 수 있는데 Pre-training part에서는 unlabeled data를 이용하여 초기 파라미터를 설정하고 이를 통해 학습된 모델은 Fine-tuning part에서 labeled dat를 이용해 fine-tuning된다.

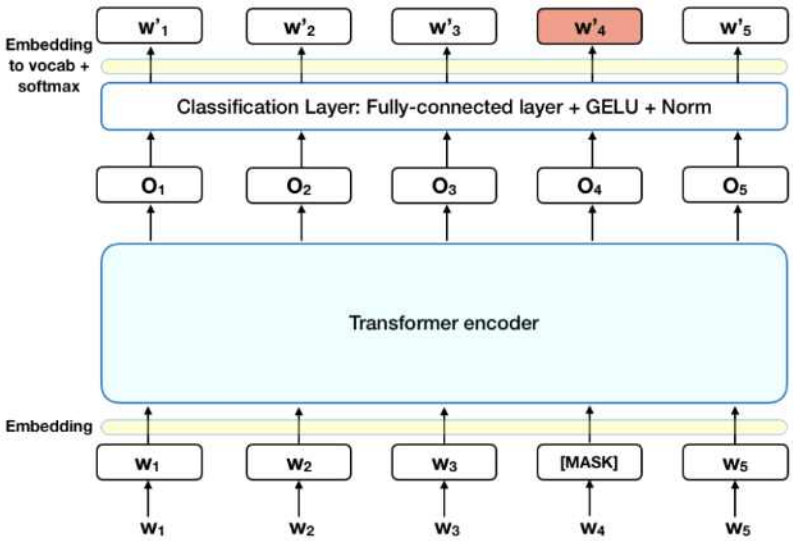
model의 arcitecture은 multi-layer bidirectional transformer encoder이다. 즉 양방향 transformer encoder를 여러 층 쌓는 구조이다. BERT의 특징인 MLM과 NSP를 수행하기 위해 self-attention 구조를 가진 transformer encoder구조를 이용했음을 확인할 수 있다.



먼저 BERT의 INPUT prepresentation을 살펴보면, 위와 같이 3가지의 embedding vector를 합쳐 하나의 input으로 사용함을 확인할 수 있다. WordPiece embedding을 사용하는 token embedding, transformer에서 사용한 방식과 동일한 position embedding을 가지고 있다. 또한 input sequence는 문장의 한 쌍으로 구성되어 있고 이는 [SEP] 토큰으로 분리되어 있다. 이 토큰과 더불어 segment embedding을 사용해 문장을 구분해주는 역할도 존재한다.

이후 Pre-training Tasks에 대한 BERT의 2가지 특징에 대해 설명하도록 하였다. introduction에서 설명했던 2가지 unsupervised prediction task인 MLM과 NSP이다.

1. MLM : input tokens의 일정 비율을 masking한 후 이를 예측하는 과정.



2. NSP : 두 문장 사이의 관계를 이해하기 위한 과정. 문장 A와 B를 선택할 때, 50%는 실제 A의 다음 문장인 B(isnext), 나머지 50%는 랜덤문장 B(notnext)를 고르게 함.(아래의 예시 문장)

```
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
```

Fine-tuning 과정은 앞서 Pre-training보다는 간단한데 입력의 개수에 따라서 앞서 설명한 것처럼 알맞게 하나의 sequence로 생성해 input으로 설한다. 그리고 task에 따라서 알맞게 하이퍼 파라미터를 조정하여 fine-tuning을 진행한다. fine-tuning 시에 dataset이 크기가 클수록 하이퍼 파라미터의 영향을 덜받고 잘 training 됨을 관측할 수 있으며 pre-training에 비해 더 빠르게 학습되며 비용이 덜 든다.

4. Experiments

BERT의 fine-tuning을 이용한 11개의 NLP task 결과를 보여주고 있다.

1) GLUE results

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	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$BERT_{BASE}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$BERT_{LARGE}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

- 모든 task에 대해 SOTA를 달성
- 데이터의 크기가 작아도 fine-tuning 때는 좋은 성능을 냄.

- BERT_large가 BERT_base에 비해 훨씬 좋은 성능을 냄.

2) SQuAD v1.1

- 기존의 glue는 task가 sequence classification이지만 이는 질문과 지문이 주어지고 그중 substring인 정답을 맞추는 task.
- A embedding을 질문, B embedding을 지문으로 처리하여 지문에서 정답이 되는 substring의 처음과 끝을 찾는 task 문제로 변형하여 해결.

3) SQuAD v2.0

- 이는 기존의 v1.1의 문제 정의를 확장하여 보다 현실적인 상황을 반영하여 짧은 대답이 없을 가능성을 허용해 문제를 해결하도록 확장시킴.

4) SWAG(Situations With Adversarial Generations)

- 앞의 문장이 주어지고 보기로 주어지는 4가지 문장중에서 잘 이어지는 문장을 찾는 task.

5. 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

BERT_base와 동일한 파라미터로 실험을 진행하였지만 2가지 다른 모델(No NSP, LTR & No NSP)에 대해서 진행한 결과를 보여주고 있다.

* No NSP: MLM은 사용하지만, 다음 문장 예측 (NSP)를 없앤 모델
* LTR & No NSP : MLM 대신 Left-to-Right (LTR) 을 사용하고, NSP도 없앤 모델(GPT모델과 동일, 더 많은 training data 사용)

- 위의 그림에서 볼 수 있듯이, pre-training task를 하나라도 제거하면 성능이 굉장히 떨어짐.
- No NSP의 경우에는 NLI계열의 task에서 성능이 많이 하락하게 되는데, 이는 NSP task가 문장간의 논리적인 구조 파악에 중요한 역할을 하고 있음을 확인할 수 있음.
- MLM대신 LTR을 쓰게 되면 성능 하락은 더욱더 심해짐. BiLSTM을 더 붙여도, MLM을 쓸 때보다 성능이 하락하는 것으로 보아, MLM task가 더 Deep

Bidirectional한 것을 확인할 수 있음.

Hyperparams				Dev Set Accuracy		
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

위 표를 보면 결국 모델의 크기가 커질수록 fine-tuning의 정확도를 더 높여주는 결과를 보여주고 있다. 즉, 모델의 크기가 모델의 성능에 직접적인 영향을 준다는 것도 알 수 있다.

Layers	Dev F1
Finetune All	96.4
First Layer (Embeddings)	91.0
Second-to-Last Hidden	95.6
Last Hidden	94.9
Sum Last Four Hidden	95.9
Concat Last Four Hidden	96.1
Sum All 12 Layers	95.5

위와 같이 BERT를 feature based approach에도 사용가능하다. feature-based approach는 모든 NLP task를 represent하지는 못하므로 특정 NLP task를 수행할 수 있는 Network를 부착하여 쓸 수 있고 computational benefit을 얻을 수 있다.

그 결과 위에서 확인할 수 있듯이, Concat Last Four Hidden값을 사용하면, Finetune All과 단지 0.3 score 차이밖에 나타나지 않음을 알 수 있다. 이를 통해, BERT는 Feature-based Approach에서도 효과적이라고 할 수 있다.