BERT Review

Pre-training of Deep Bidirectional Transformers for Language Understanding

Sangho Kim

Contents

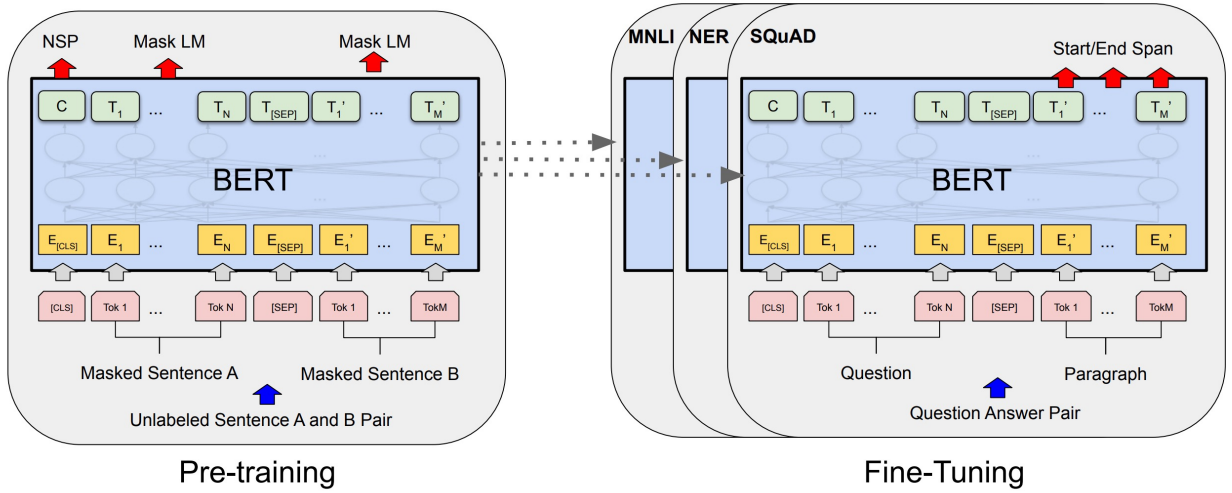
Introduction

- NLP 분야에서 다양한 모델이 제안되었지만 특화된 architecture가 필요한 등 한계점이 존재
- 이를 해결하기 위해 bidirectional representation을 학습하는 방식을 이용한 BERT 소개
- 기존 architecture에 해당하는 GPT를 예로 들자면 unidirectional이라는 문제점이 존재
- 그러나 저자들은 bidirectional이 더 좋은 성능을 낸다고 주장
- GPT는 chatting과 같은 task에 적합하지만 대부분의 NLP task는 훨씬 더 긴 context를 이해하는 능력을 필요로 함
- 그래서 이를 위해서는 bidirectional representation 방식이 더 유리
- 이렇게 BERT를 Pre-training을 하게 되면 하나의 output layer를 추가해 다양한 Task를 수행 가능

Contents

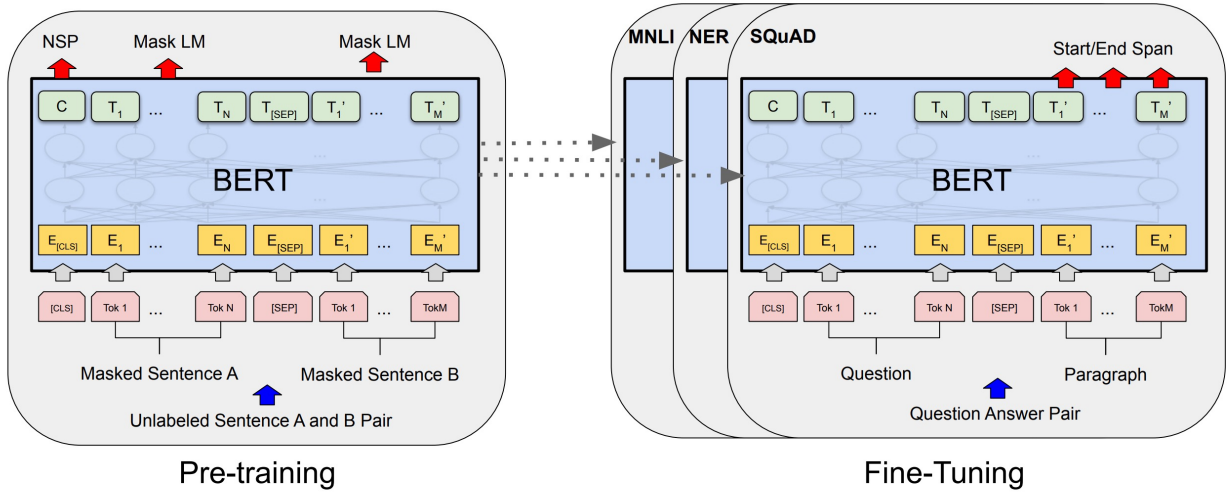
Method

- BERT도 일종의 foundation 모델이라고 볼 수 있다
- 또한 Transformer의 Encoder based architecture로 이루어져 있다
- 아래 그림과 같이 large dataset으로 pre-training을 한 이후 specific task에 맞게 fine-tuning을 수행



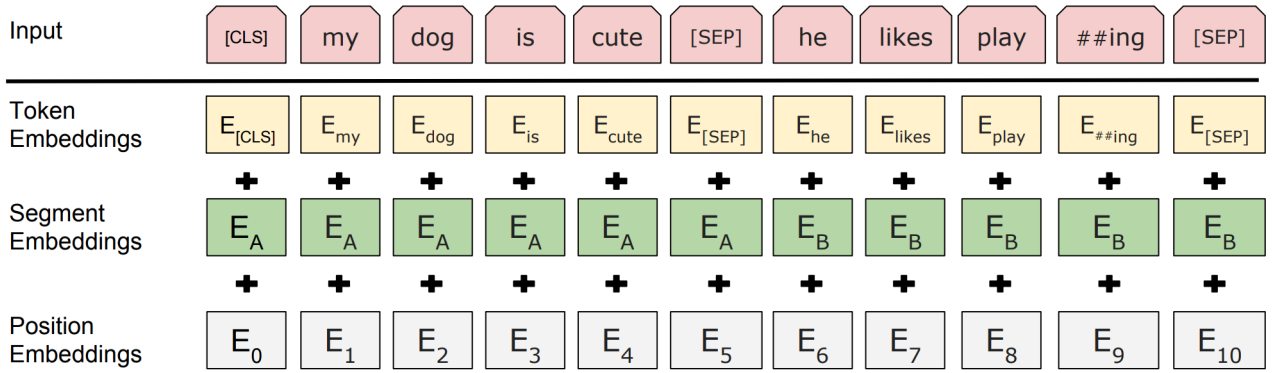
Method

- BERT도 일종의 foundation 모델이라고 볼 수 있다
- 또한 Transformer의 Encoder based architecture로 이루어져 있다
- BERT base(110M)의 경우 L=12, H=768, A=12로 구성, large(340M)의 경우 L=24, H=1024, A=16으로 구성
- 이때 L는 number of layers, H는 hidden dimension, A는 number of heads
- 아래 그림과 같이 large dataset으로 pre-training을 한 이후 specific task에 맞게 fine-tuning을 수행



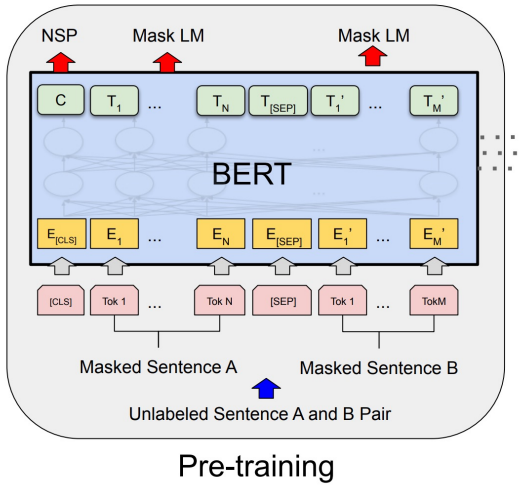
Method

- Input/Output Representations
  - 다양한 downstream task에 fine-tuning이 가능하기 위해선 input representation이 명확해야 함
  - 이를 위해 BERT에서는 3개의 embedding vector를 합쳐 input으로 사용
  - 모든 input sequence의 첫 번째 token은 [CLS] token
  - 각 문장들은 [SEP] token으로 분리 + 각 문장이 A 문장인지 B 문장인지 구분하기 위해 Segment embedding을 사용
  - Token embedding으로는 WordPiece embedding을 사용
  - Position embedidng은 Transformer에서 쓰였던 method와 동일



Method

- 먼저, Pre-training 부분부터 살펴보자
- 2가지 방법을 사용해 모델을 학습하는데
  - 1. Masked LM: input sentence의 일부 토큰을 masking한 뒤 이를 맞추는 작업
  - 2. Next Sentence Prediction: 두 개의 문장을 input으로 받아 두 번째 문장이 첫 번째 문장의 다음 문장인지 맞추는 작업



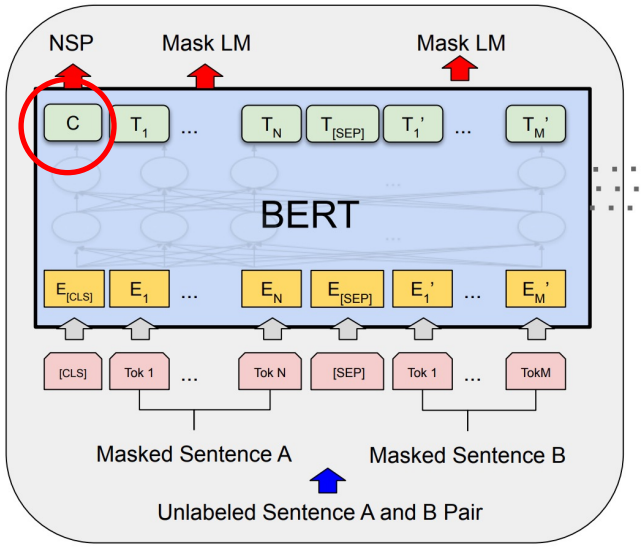
Pre-training

Method

- 1. Masked LM
  - 전체 token 중 15% 정도만 masking 처리를 했을 때 가장 좋은 성능을 보여줌
  - Masking은 pre-training에만 사용되고 fine-tuning 시에는 사용되지 않음
  - 그러나 이럴 경우 pre-training과 fine-tuning 사이의 mismatch가 발생하며 15% masked token을 다음과 같이 처리
    - 80%는 token을 [MASK] token으로 변경
    - 10%는 token을 random word로 변경
    - 10%는 token을 원래 단어 그대로 유지

Method

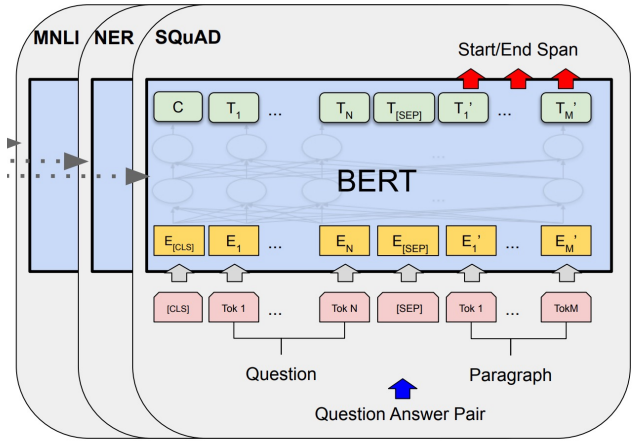
- 2. Next Sentence Prediction
  - NLP 대부분의 task는 두 문장 사이의 관계를 이해하는 것이 중요
  - 이를 위해 BERT는 binarized next sentence prediction을 이용
  - 즉, 두 문장을 제공하고 두 문장이 서로 이어지는 문장인이 분류하는 것이다
  - 이 두 문장이 서로 이어지는 것인지에 대한 여부를 결정하는 토큰이 아래에서 빨간 동그라미에 해당



Pre-training

Method

- 그리고 Fine-tuning 부분을 살펴보면
- 어느 정도의 language understanding이 가능한 상태이고 이후 적용하고자 하는 specific task의 dataset에 맞춰 학습



Fine-Tuning

Method

- 논문에서는 크게 4개의 fine-tuning 기법을 보여주었다
  - 1. Sentence pairs in paraphrasing
  - 2. Hypothesis-Premise pairs in entailment
  - 3. Question-Passage pairs in question answering
  - 4. Degenerate-None pair in text classification or sequence tagigng

Contents

Experiments

- BERT의 실험 결과를 살펴보자
- GPT에 비해 더 좋은 성능을 보여줌으로써 Encoder 구조와 bidirectional method가 효과적인 것을 알 수 있음
- 또한 모든 벤치마크에서 base모델보다 large 모델이 더 성능이 좋으며 사이즈를 키울수록 성능이 좋음을 입증한다

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**

Table 1: GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard). The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

Contents

Ablation studies

- NSP의 적용 여부에 따른 성능을 비교한 실험
- Langauge understanding이 중요한 QNLI task에서 NSP를 적용하지 않을 때 성능이 크게 떨어짐

Tasks	Dev Set MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
$\text{BERT}_{\text{BASE}}$	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

Table 5: Ablation over the pre-training tasks using the $\text{BERT}_{\text{BASE}}$ architecture. "No NSP" is trained without the next sentence prediction task. "LTR & No NSP" is trained as a left-to-right LM without the next sentence prediction, like OpenAI GPT. "+ BiLSTM" adds a randomly initialized BiLSTM on top of the "LTR + No NSP" model during fine-tuning.

Ablation studies

- 모델 사이즈에 따른 성능 비교 실험
- 모델 사이즈가 클수록 성능이 비례적으로 증가하는 것을 알 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

Table 6: Ablation over BERT model size. #L = the number of layers; #H = hidden size; #A = number of attention heads. "LM (ppl)" is the masked LM perplexity of held-out training data.

Ablation studies

- Feature-based architecture로 사용한 것에 대한 실험
- 모든 task에 적용이 가능한 것은 아니라 task-specific model을 추가해서 사용
- Pre-computing을 통해 representation을 생성하고 별도의 모델을 생성하고 이를 input으로 사용하면 비용 절감 가능

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	-

Table 7: CoNLL-2003 Named Entity Recognition results. Hyperparameters were selected using the Dev set. The reported Dev and Test scores are averaged over 5 random restarts using those hyperparameters.

Contents

Conclusion

- BERT는 Encoder network와 bidirectional method를 사용해 효과적으로 성능을 개선
- 이를 통해 다양한 task에 fine-tuning을 가능하게 했고 필요 리소스를 절감
- 다양한 벤치마크 실험을 통해 BERT의 성능을 입증