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

Abstract & Introduction

- 기존의 pre-training 된 언어 모델의 두가지 접근법 1) feature-based 접근: ELMo와 같이 pre-train된 표현은 추가적인 feature로 활용하여 task 별로 아키텍처를 설계해야 함 2) fine-tuning 접근: GPT와 같이 pre-training 된 모델을 그대로 fine-tuning 해서 downstream 과제에 적용함. 이때 아키텍처 수정은 필요 없음 → 하지만 대부분의 pre-training 된 모델은 단방향 구조를 사용하고 있음 (GPT의 경우도 왼쪽에서 오른쪽으로 학습이 되어 Transformer의 self-attention에서 이전 단어만 참조할 수 있도록 제한됨). 이러한 한계는 문장 수준의 과제에서 비효율적. (특히 질의응답과 같이)
- BERT는 label 이 없는 텍스트에서 양방향으로 심층적인 표현을 pre-train하도록 설계됨 → 모든 layer에서 왼쪽과 오른쪽 문맥을 동시에 고려하도록. Masked Lanuage Model을 pre-train의 목표로 사용함 → **이는 독립적으로 학습된 단방향 모델을 단순히 결합하는 방식(ELMo)과도 차별됨.** → **ELMo 는 양방향을 활용하지만, 독립적으로 학습된 두개의 단방향 (left→right, right→left) 모델로 양방향의 정보를 concatenate하여 정보를 얻는 방식임. 하지만 이는 두개의 단방향 모델을 따로 학습한 후 결합해야 하므로 학습 과정에서 양방향 정보를 직접 공유하지 않음 (각각의 방향을 독립적으로 학습하고 단순히 결합한 것. 따라서 실제 문맥 내에서 동시적인 관계를 학습하지 못함). BERT는 한번에 양방향으로 학습하는 방식을 사용하여 한번에 전체 문맥을 활용하여 표현을 학습할 수 있음 (동시에 양방향 문맥을 고려하므로, 문장 내 단어들간 상호 의존성을 효과적으로 학습할 수 있음)**
- MLM 이란 기존의 단방향 언어 모델고 달리, 입력 토큰의 일부를 랜덤하게 마스킹하고 이 마스킹된 단어를 원래의 단어로 복원하는 작업을 수행함 → 왼쪽과 오른쪽 문맥을 동시에 고려하는 양방향 학습이 가능해짐
- 추가적으로, Next Sentence Prediction task를 추가하여 text-pair 표현을 함께 학습할 수 있도록
- BERT 는 **첫번째 fine-tuning-based 모델** **(fine-tuning 기반 접근법은 pre-trained 모델을 특정 task에 맞게 fine-tuning 하여 성능을 최적화하는 방식을 의미함.**

따라서 추가적인 모델 설계를 할 필요 없이, 사전 훈련된 가중치를 그대로 유지하면 특정 task 에 맞춰 조정이 가능. GPT-1 도 fine-tuning 기반 접근법을 사용한 최초의 모델 중 하나지만, GPT는 단방향 모델임. BERT 는 최초의 양방향 fine-tuning 기반 모델) 로서, 기존의 많은 task-specific 아키텍처보다 좋은 성능을 보임.

Related Work

Unsupervised Feature-based Approaches (Word2Vec, GloVe, ELMo)

- 사전훈련된 단어 임베딩은 처음부터 학습된 임베딩보다 더 나은 성능을 제공함. 이때 왼쪽에서 오른쪽으로의 언어 모델링, 좌우 문맥을 활용한 단어 예측 등 의 방법으로 단어 임베딩 벡터를 사전훈련 해왔음
- 문장 표현을 학습하는 기존 연구는 후보 문장 순위를 매기는 법, 이전 문장을 보고 다음 문장의 단어를 생성하는 법, 노이즈 제거 autoencoder 등의 방식을 사용했음
- ELMo등의 연구에서는 왼쪽-오른쪽 모델과 오른쪽-왼쪽 모델에서 문맥 정보를 추출하여 각 토큰의 문맥적 표현을 두 개의 방향 정보를 concatenate 하는 방식으로 구성함 → task-specific architecture와 결합하여 성능이 더 나아졌음

Unsupervised Fine-tuning Approaches (ULMFit, GPT)

- 초기 연구에서는 비지도 학습된 단어 임베딩을 사용하여 모델을 학습하는 방식을 제안함
- 최근의 연구에선, 문장이나 문서 인코더를 비지도 학습하여, 이를 지도 학습 과제에 미세조정하는 방식을 활용함 → 새로운 task에서 처음부터 모든 파라미터를 학습하지 않아도 된다는 장점. 그리고 이로 인해 pre-trained 된 모델을 fine-tuning 하는 것이 훨씬 효율적이고 강력한 성능을 보인다는 것을 의미함 (ex GPT)
- 이때의 pre-training 에서의 목표는 왼쪽에서 오른쪽으로 학습하는 언어 모델링, autoencoder를 사용함

Transfer Learning from Supervised Data (NLI, Translation)

- 대규모 데이터셋을 활용한 지도 학습이 전이 학습에 강력한 영향을 미친다는 연구 결과들이 있었음
- cv 분야에서도, ImageNet 데이터셋으로 pre-trained 된 모델을 fine-tuning 하는 방식으로 이전 task에서 학습된 특징을 재사용한다는 것이 모델의 성능을 크게 향상시킬 수 있다는 점을 제시

Method

[1] Pre-training: unlabeled data 로 다양한 task 로 학습시킴 → 하나의 아키텍처를 사용함

[2] Fine-tuning: pre-trained params로 초기화 후, downstream task에서 labeled data 사용하여 fine-tuning

- BERT 모델의 아키텍처는 Multi-layer Bidirectional Transformer Encoder로 기존 Transformer 구조와 거의 동일함.

본 논문에서는, 2가지 크기의 BERT 모델 사용

- BERT Base: 12개 layer, 768개 hidden state. 12개 self-attention head → 110M params (GPT와 동일한 크기)
- Bert Large: 24개 layer, 1024개 hidden state. 16개 self-attention head → 340M params

BERT 가 다양한 task를 처리할 수 있도록, input 표현은 단일 문장과 문장 pair을 하나의 토큰 시퀀스로 표현할 수 있도록 설계됨

따라서 문장을 실제 언어에서의 문장이 아닌, 연속된 텍스트로 정의!

WordPiece 임베딩을 사용하여 총 3천개의 토큰으로 이루어진 vocab set을 사용함. → 이때 [CLS] 토큰은 모든 입력 시퀀스의 첫번째 토큰으로 분류를 위함. 이 토큰의 최종 hidden state 는 classification 과제에서 전체 시퀀스를 대표하는 벡터로 사용이 됨. 또한 sentence pair의 경우 두 문장을 하나의 시퀀스로 묶어서 한번에 처리하는데, 이때 [SEP] 토큰을 넣어 두 문장을 구분하거나, 각 토큰이 문장 A, B 중 어디에 속하는지를 나타낸 segment embedding 을 추가하여 묶음

Pre-training BERT

- BERT에선 기존과 마찬가지로 BooksCorpus (800M 단어),Wikipedia (2,500M 단어) 데이터를 사용. 이때 Wiki에선 텍스트 본문만 추출하는 방식 → 문장 수준에서 shiffle 하지 않고, 문서 수준에서 연속적인 긴 시퀀스를 유지하는 corpust 를 사용 (문맥적 연속성 유지하도록)

이때 2가지 unsupervised task로 사전훈련시킴

[Masked Language Model]

- 기존 standard conditional language model은 양방향 학습을 허용한다면, 모델이 각 단어를 볼 수 있게 되어 (leaking 문제) 너무 쉽게 예측

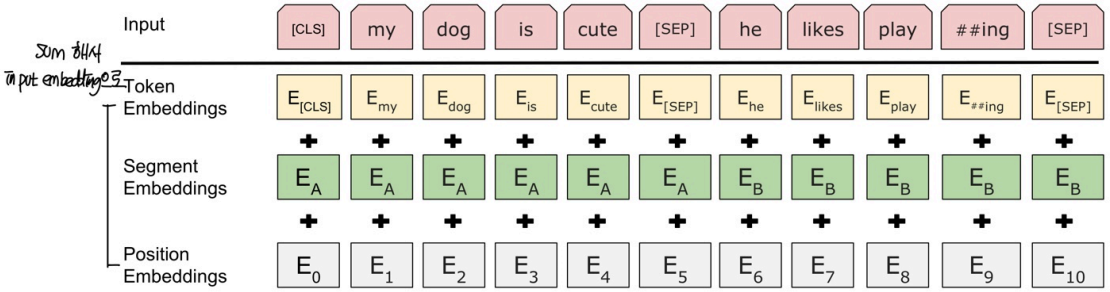
- 따라서 입력 토큰의 일부를 무작위로 Masking 하고 해당 토큰을 예측하도록 하는 MLM 을 사용
- masked token의 최종 hidden vector를 출력 softmax 층을 통해 vocab 내의 원래 단어를 예측하는데 사용됨. 이때 모든 실험에서 각 시퀀스의 WordPiece 중 15%를 무작위로 masking 함. 이때 **denoising autoencoder와 달리, 전체 입력을 재구성 하는 것이 아닌, 마스킹 된 단어만 예측함**

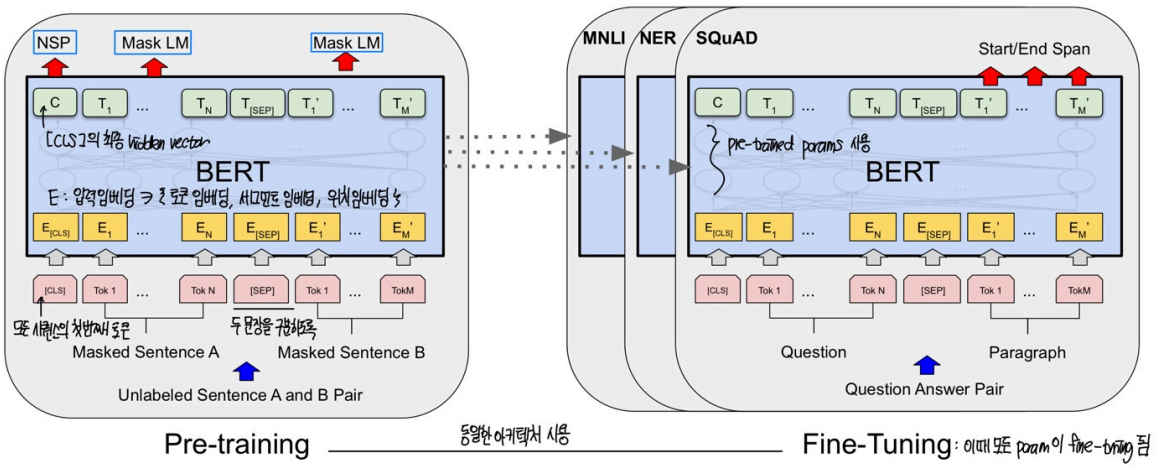
→ 이때 MLM을 사용하면 양방향 학습이 가능하지만, pre-training 엔 [MASK] 토큰이 있지만 fine-tuning 시에는 [MASK] 토큰이 없기 때문에 mismatch 문제가 발생할 수 있음
→ 따라서 마스킹딘 단어를 [MASK] 토큰으로 대체하는 것이 아니라, 15% 로 선정된 토큰 위치 중 80%로 [MASK] 토큰으로 변경 + 10% 로 다른 random 토큰으로 변경 + 10%로 해당 토큰을 변경하지 않고 유지함

[Next Sentence Prediction]

- 질의응답, NLI 등의 task는 두 문장간 관계를 학습하는 것이 중요하므로 기존의 언어 모델 방식으론 문장간 관계를 직접적으로 학습할 수 없음
- 따라서 본 연구에선 모든 단일 언어 corpus 에서 문장 A를 선택할 때, 50% 확률로 A 다음에 오는 문장을 B 로 설정 + 50% 확률로 B를 corpus에서 random 하게 선택한 문장으로 설정 → binary NSP 과제를 추가
- 따라서 [CLS] 토큰의 최종 hidden state인 C 벡터는 NSP 예측에 사용

→ 이전 연구에선, 문장 임베딩만 하위 과제로 transfer가 되었지만, **BERT 에선 모든 파라미터를 end-to-end transfer로 초기화한다는 점에서 차이**





Fine-tuning BERT

- Transformer의 self-attention 매커니즘으로 다양한 downstream 를 모델링
- 기존 연구에서는, text pair를 독립적으로 encoding 하고, bidirectional cross attention 를 적용했음. **하지만 BERT 는 self-attention 하나로 (두 문장을 concatnate 한 상태에서 encoding 하여)**
- 각 task에 대해 BERT 는 end-to-end 방식으로 모든 param을 fine-tuning 함
- 입력 문장 (A, B) 는 paraphrase, 함의, 질의응답 관계의 문장 쌍으로 구성됨. 또한 텍스트 분류, Sequence Tagging 일 때는 입력 문장 (A, B)가 단일 문장과 빈 텍스트 쌍으로 구성이 되어있음
- sentence-level task일 때 (NLI, sentiment 분석 등) [CLS] 토큰의 최종 표현을 출력층으로 전달하여 분류 수행함. token-level task일 때는 (sequence tagging, QA) 개별 토큰 표현을 출력층으로 전달
- pre-training 과 비교했을 때, fine-tuning 은 상대적으로 비용이 굉장히 적게 듦

Experiments

GLUE (General Language Understanding Evaluation) 벤치마크

- 단일 문장 또는 문장 쌍을 하나의 입력 시퀀스로 → [CLS] 의 최종 hidden state C를 전체 시퀀스의 aggregate representation 로 사용
- 이때 fine-tuning 에선 classification layer 가중치에서 새로운 파라미터 가 추가됨 또한 이때 log softmax 로 loss를 계산함
- fine-tuning 에서 → batch size=32, epoch=3, learning rate은 후보값 중 가장 좋은 것을 선택하도록 (이때 BERT Large의 경우 작은 데이터셋에서 학습이 불안정할 수 있기 때문에 여러번 random restart (pre-trained 는 동일하지만, data shuffling 과

마지막 classification layer의 초기화를 다르게) → 가장 좋은 learning rate 선택하도록)

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**

문제가 있는 WNLI 데이터셋 제외. 앙상블은 제외하고 single-model 방식으로 평가

→ 모든 테스크에서 기존 모델보다 우수한 성능.

- SQuAD v1.1 : 100,000개의 크라우드소싱된 질문-답변 쌍으로 구성됨 → Wikipedia 기반으로 문단 내에 정답이 포함된 텍스트 예측하는 문제
- SQaAD v2.0: 제공된 문단에 짧은 답변이 존재하지 않을 가능성을 추가한 문제로 일부 질문은 아예 정답이 없을 수도 있음 → BERT 의 경우, 답변이 없는 질문은 정답 범위의 시작과 끝을 [CLS]로 : 모델이 질문에 대합 답변이 존재하는지 여부를 학습할 수 있음
- SWAG (Situations With Adversarial Generations) : 113000개의 sentence-pair 로 상식적 추론 능력을 평가하는 과제로 주어진 문장에 대해 가장 적절한 다음 문장 4개의 선택지 중에서 고르는 문제임 → BERT 의 경우, (주어진 문장-후속 문장 후보) 인 4개의 입력 시퀀스로 구성
- 새로운 task에서 추가된 param은 하나의 vector로 이 vector와 C의 dot product → softmax 취해서 선택. 3 epoch에서 훈련을 시도

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				
$BERT_{BASE}$ (Single)	80.8	88.5	-	-
$BERT_{LARGE}$ (Single)	84.1	90.9	-	-
$BERT_{LARGE}$ (Ensemble)	85.8	91.8	-	-
$BERT_{LARGE}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$BERT_{LARGE}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

SQuAD 1.1 에서 result

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				
$BERT_{LARGE}$ (Single)	78.7	81.9	80.0	83.1

SQaAD 2.0 에서 result

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

SWAG에서 result

[Ablation Study]

- Effect of Pre-training Tasks : NoNSP (only MLM) , LTR(Left to Right 단방향) + NoNSP 에서 성능 저하가 확인됨
- 특히 SQuAD에선 단방향 모델은 각 단어의 hidden state가 오른쪽 문맥을 보지 못하므로 적절한 정답을 예측하기가 어려움
- ELMo 와 비교해보았을 때도, BERT 보다 비효율적 (학습 비용 증가) + QA 에서 RTL이 질문을 보고 정답을 유추할 수 없다는 한계점이 있음.
- Effect of Model Size: layer 수, hidden unit 수, attetion head 수를 변경해보면서 학습시켜보았음. 이때 모델의 크기가 증가할 수록 모든 데이터셋에서 정확도가 향상된다는 것이 확인됨. 특히, 작은 데이터셋에서도 큰 모델이 성능 향상이 이루어질 수 있다는 것을 증명함. (기존 BiLSTM은 layer를 2→ 4개로 늘려도 성능이 향상되지 않았음.)
- Feature-based Approach + BERT : input을 대소문자를 유지하는 WordPiece 와 출력에서 Conditional Random Field 를 사용하지 않고, 각 토큰의 첫번째 subtoken으로 NER (Named Entity Recognition)를 수행함 → 여기서 성능향상 BERT 가 fine-tuning 뿐만 아니라 feature-based에도 효과적임이 입증됨

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

Ablation over pre-training

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

Ablation over model size

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	-

Feature-based, not fine-tuning

Conclusion

- transfer learning, 이 중 rich unsupervised pre-training 이 매우 필수적인 요소 → 심층 단방향 아키텍처의 이점을 활용 가능. 본 논문에서는 심층 양방향 아키텍처로 확장.