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

BERT: 언어 이해를 위한 심층 양방향 트랜스포머의 사전 훈련

기존 Transformer의 Encoder만을 활용하며, Masked LM을 활용한 빈 칸 맞추기, Next sentence prediction 등 사전 훈련을 통해 정확성을 높이 올렸다. 한 이 논문에서는 Fine-tuning 기반 위주로 작성되었으나 후술하는 Feature-based에서도 좋은 결과를 보여주며, 최소한의 파라미터 수정만으로 대부분의 NLP 과제에서 좋은 성적을 내는 모습을 보여주었다.

초록

BERT는 모든 층에서 왼쪽 및 오른쪽 문맥을 동시에 고려하여 심층 양방향 표현을 사전 훈련하는 새로운 언어 표현 모델입니다. 사전 훈련된 BERT 모델은 작업별 구조 수정을 크게 할 필요없이 직문 답변과 언어 추론과 같은 넓은 분야의 작업에서 state-of-the-art 모델을 만들기 위해 하나의 출력 레이어만 추가해도 파인튜닝될 수 있다. BGLUE, MultiNLI, SQuAD v1.1 및 SQuAD v2.0 테스트에서 성능 향상을 기록했습니다.

1. 도입

사전 훈련된 언어 표현을 다운스트림 작업에 적용하는 두 가지 기존 전략이 있습니다.
*Downstream task : 사전 학습한 가중치를 활용해, 학습하고자 하는 본 문제를 Downstream task(하위 문제)라고 한다.

- Pre-training의 종류
Fuature-based: ELMo와 같은 접근 방식은 사전 훈련된 표현들을 추가 기능으로 포함하는 작업별 구조를 사용한다.
Fine-tuning: GPT와 같은 미세 조정 접근 방식은 최소한의 작업별 파라미터를 도입하고, 모든 사전 훈련된 파라미터를 미세 조정하여 다운스트림 작업을 훈련합니다.

- Pre-training 모델의 문제점
1. 단방향 언어 모델을 사용하여 일반적인 언어 표현을 학습합니다.
2. 현재 기술은 미세 조정 접근 방식에서 사전 훈련된 표현의 힘을 제한합니다.
3. GPT에서는 왼쪽에서 오른쪽으로 파라미터를 사용하여 각각의 토큰들은 이전 토큰에만 영향을 받습니다.
4. 양방향 문맥을 통합하는 것이 중요한 질문 응답과 같은 토큰 수준 작업에 미세 조정 접근 방식을 적용할 때 매우 해로울 수 있습니다.

- 해결 방법
따라서 이 논문은 양방향 학습이 가능한 BERT모델을 제안함으로써, fine-tuning 기반의 방법을 향상시킨다.
1. BERT는 "masked language 모델"(MLM) 사전 훈련 목표를 사용하여 앞서 언급한 단방향성 제약을 완화합니다.
2. 입력에서 일부 토큰을 무작위로 마스킹하고, 문맥만을 기반으로 마스킹된 단어의 원래 어휘 ID를 예측하는 것입니다.
3. MLM 목표는 표현이 왼쪽과 오른쪽 문맥을 융합할 수 있게 하여 심층 양방향 Transformer를 사전 훈련할 수 있게 합니다.

2. 관련 연구

2.1 비지도 피처 기반 접근 방식

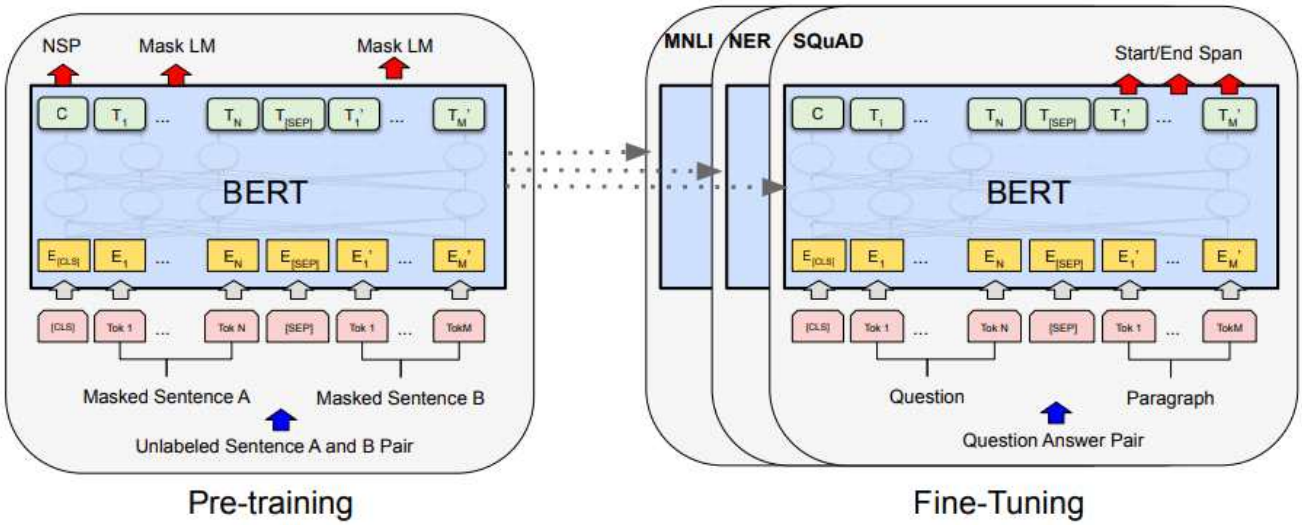
단어 임베딩: 비신경 방법과 신경 방법을 통해 학습되며, 사전 훈련된 단어 임베딩은 현대 NLP 시스템에서 중요한 역할을 함.
문장 및 문단 임베딩: 다음 문장의 순위를 매기거나, 이전 문장 표현을 통해 다음 문장을 생성하는 등의 방법을 사용하여 학습.
ELMo: 왼쪽-오른쪽 및 오른쪽-왼쪽 언어 모델에서 문맥에 민감한 피처를 추출하여 NLP 벤치마크에서 최첨단 성능을 달성.

2.2 비지도 학습 미세 조정 접근 방식

초기 작업: 레이블이 없는 텍스트에서 단어 임베딩 파라미터만 사전 훈련.
최근 작업: 레이블이 없는 텍스트로부터 문장 또는 문서 인코더를 사전 훈련하고, 다운스트림 작업을 위해 미세 조정.
장점: 처음부터 학습해야 하는 파라미터가 적어 효율적.

2.3 지도 데이터로부터의 전이 학습

지도 작업 전이 학습: 대규모 데이터셋을 사용한 지도 작업으로부터의 전이 학습이 효과적임을 보여줌.
컴퓨터 비전: ImageNet으로 사전 훈련된 모델을 미세 조정하여 전이 학습의 중요성 입증.



위 그림은 BERT의 전체 사전 훈련 및 미세 조정 절차를 보여줍니다. 출력 층을 제외하면 사전 훈련과 미세 조정 모두 동일한 아키텍처가 사용됩니다. 동일한 사전 훈련된 모델 매개변수가 다양한 다운스트림 작업을 초기화하는 데 사용됩니다. [CLS]는 모든 입력 예제 앞에 추가되는 특수 기호이며, [SEP]는 질문과 답변을 구분하는 등 특수 구분자 토큰입니다.

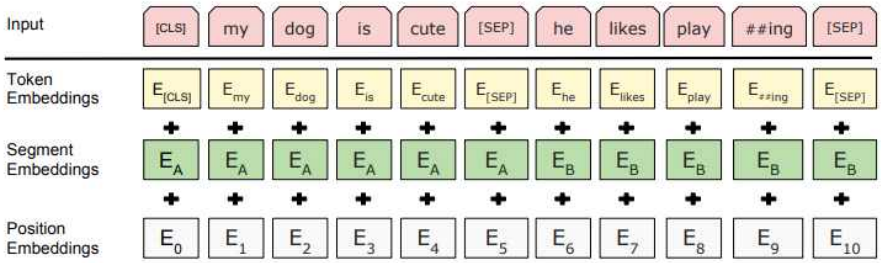
사전 훈련 (Pre-training)
Masked Sentence A와 Masked Sentence B 쌍을 사용합니다.
BERT는 마스크된 문장 A와 문장 B의 마스크된 단어들을 예측합니다.
NSP(Next Sentence Prediction)와 Masked LM(Masked Language Model) 목표를 사용합니다.

미세 조정 (Fine-Tuning)
다양한 다운스트림 작업 (MNLI, NER, SQuAD 등)을 위한 모델 초기화를 위해 사전 훈련된 매개변수를 사용합니다.
질문 응답 작업의 경우, 질문과 문단 쌍을 입력으로 받으며, [CLS]와 [SEP] 토큰이 사용됩니다.
시작 및 종료 Span을 예측하여 답변을 찾습니다.

3 BERT

모델 아키텍처
BERT의 모델 구조는 다중 층 양방향 트랜스포머 인코더입니다.
이 작업에서 레이어의 수를 L, 은닉의 크기를 H, 셀프 어텐션 헤드의 수는 A로 나타냅니다.
BERTBASE(L=12, H=768, A=12, 총 매개변수=110M)와 BERTLARGE(L=24, H=1024, A=16, 총 매개변수=340M).

입력/출력 표현
BERT가 다양한 다운스트림 작업을 처리할 수 있도록 하기 위해, 단일 문장과 문장 쌍(예: 질문, 답변)을 명확하게 나타낼 수 있습니다.
모든 시퀀스의 첫 번째 토큰은 항상 특수 분류 토큰([CLS])입니다. 이 토큰에 해당하는 마지막 은닉 상태는 분류 작업을 위한 집합 시퀀스 표현으로 사용됩니다.
우리는 두 가지 방법으로 문장을 구분합니다. 첫째, 특수 토큰([SEP])으로 문장을 구분합니다. 둘째, 각 토큰에 대해 학습된 임베딩을 추가하여 해당 토큰이 문장 A에 속하는지 문장 B에 속하는지를 나타냅니다.
주어진 토큰에 대해, 그 입력 표현은 해당 토큰, 세그먼트 및 위치 임베딩을 합산하여 구성됩니다. 이 구성의 시각화는 그림 2에서 볼 수 있습니다.



BERT 모델의 입력 표현을 시각화한 것입니다. 입력 임베딩은 토큰 임베딩, 세그먼트 임베딩, 위치 임베딩의 합으로 구성됩니다.
Input 시퀀스: [CLS]는 분류 토큰, [SEP]는 문장을 구분하는 특수 토큰입니다.
Token Embeddings: 각 단어와 특수 토큰에 대해 고유한 임베딩이 있습니다.
Segment Embeddings: 문장 A와 문장 B를 구분하기 위해 사용됩니다.
Position Embeddings: 입력 시퀀스 내에서 각 토큰의 위치를 나타내는 임베딩입니다.

3.1 BERT 사전 훈련

Task #1: MLM
직관적으로, 깊은 양방향 모델이 왼쪽-오른쪽 모델이나 왼쪽-오른쪽 및 오른쪽-왼쪽 모델의 얕은 연결보다 더 강력하다고 믿을 수 있습니다. 깊은 양방향 표현을 학습하기 위해, 단순히 입력 토큰의 일정 비율을 무작위로 마스킹하고, 그런 다음 그 마스크된 토큰을 예측합니다. 이 경우, 마스크된 토큰에 해당하는 최종 은닉 벡터는 출력 소프트맥스에 입력됩니다. 모든 실험에서, 각 시퀀스의 모든 WordPiece 토큰 중 15%를 무작위로 마스킹합니다. 전체 입력을 재구성하는 대신 마스크된 단어만 예측합니다. 학습 데이터 생성기는 예측을 위해 무작위로 토큰 위치의 15%를 선택합니다. i번째 토큰이 선택되면, 80%의 확률로 [MASK] 토큰으로 대체하고, 10%의 확률로 무작위 토큰으로 대체하고, 10%의 확률로 변경하지 않습니다.

Task #2: NSP
질문 응답(QA) 및 자연어 추론(NLI)과 같은 많은 중요한 다운스트림 작업은 두 문장 간의 관계를 이해하는 데 기반.
문장 관계를 이해하는 모델을 학습하기 위해, 단일 언어 말뭉치로부터 이진화된 다음 문장 예측 작업을 사전 훈련합니다.
이전 작업에서는 문장 임베딩만 다운스트림 작업으로 전이되는 반면, BERT는 엔드 과제 모델 파라미터를 초기화하기 위해 모든 매개 변수를 전이한다.

3.2 BERT 미세 조정

트랜스포머의 셀프 어텐션 메커니즘은 BERT가 단일 텍스트 또는 텍스트 쌍을 포함하는 많은 다운스트림 작업을 적절한 입력과 출력을 바꾸어 모델링할 수 있게 합니다. 셀프 어텐션으로 연결된 텍스트 쌍을 인코딩하는 것은 두 문장 사이의 양방향 크로스 어텐션을 효과적으로 포함합니다.

입력에서는, 사전 훈련에서의 문장 A와 문장 B가 패러프레이싱(다른 말로 바꾸어 표현)의 문장 쌍, 수반되는 가설-전제 쌍, 문제의 질문-통과 쌍, 텍스트 분류 또는 시퀀스 태깅의 변질된 텍스트-∅ 쌍과 유사합니다.
출력에서는, 토큰 표현이 시퀀스 태깅 또는 질문 응답과 같은 토큰 수준 작업을 위한 출력 레이어에 입력되고, [CLS] 표현은 수반 또는 감정 분석과 같은 분류를 위한 출력 레이어에 입력됩니다.

Pre-training과 비교했을 때, fine-tuning은 상대적으로 적은 비용으로 수행할 수 있다.

4 실험

4.1 GLUE(General Language Understanding Evaluation)

우리는 배치 크기 32를 사용하여 모든 GLUE 작업에 대해 3 에포크 동안 데이터를 미세 조정합니다. 각 작업에 대해 Dev 세트에서 최상의 미세 조정 학습률을 선택했습니다. BERTLARGE의 경우 소규모 데이터셋에서 미세 조정이 불안정할 수 있어 여러 번의 무작위 재시작을 수행하고 Dev 세트에서 최상의 모델을 선택했습니다. 무작위 재시작을 할 때는 동일한 사전 훈련 체크포인트를 사용하지만 다른 미세 조정 데이터 셔플링과 분류층 초기화를 수행합니다.

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**

이전 최신 기술(GPT) 대비 각각 4.5%와 7.0%의 평균 정확도 향상을 달성했습니다. BERTBASE와 OpenAI GPT는 주의 마스킹을 제외하면 모델 아키텍처 측면에서 거의 동일합니다.

* BERTBASE와 BERTLARGE는 모든 작업에서 이전 시스템보다 뛰어난 성능을 보여줌.
* BERTLARGE는 특히 소규모 데이터셋에서 BERTBASE보다 뛰어난 성능을 보임.

- 세부 결과
System (시스템): 평가된 시스템의 이름
MNLI-(m/mm): MNLI 작업의 결과 (매치드/미스매치드)
QQP: Quora Question Pairs 작업의 결과
QNLI: Question Natural Language Inference 작업의 결과
SST-2: Stanford Sentiment Treebank 작업의 결과
CoLA: Corpus of Linguistic Acceptability 작업의 결과
STS-B: Semantic Textual Similarity Benchmark 작업의 결과
MRPC: Microsoft Research Paraphrase Corpus 작업의 결과
RTE: Recognizing Textual Entailment 작업의 결과
Average: 각 작업의 평균 점수

4.2 SQuAD v1.1

스탠포드 질문 응답 데이터셋은 100,000개의 크라우드소싱된 질문/답변 쌍으로 구성되어 있습니다. 구절 내에서 답변 텍스트 범위를 예측하는 것입니다.

입력 시퀀스: 질문과 구절을 하나의 패킹된 시퀀스로 나타냄.
미세 조정: 시작 벡터와 종료 벡터 도입, 로그 가능성의 합을 학습 목표로 함.
결과: BERT 모델이 상위 리더보드 시스템을 능가하며, TriviaQA 미세 조정 데이터를 사용하지 않더라도 높은 성능을 유지.

System	Dev		Test	
	EM	F1	EM	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**

Table 2: SQuAD 1.1 results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.

#1 Ensemble - nlnet: nlnet 앙상블 시스템
#2 Ensemble - QANet: QANet 앙상블 시스템

BiDAF+ELMo (Single): 단일 모델
R.M. Reader (Ensemble): 앙상블 모델

BERT_BASE (Single): 단일 BERT_BASE 모델
BERT_LARGE (Single): 단일 BERT_LARGE 모델
BERT_LARGE (Ensemble): BERT_LARGE 앙상블 모델
BERT_LARGE (Sgl.+TriviaQA): 단일 BERT_LARGE 모델에 TriviaQA 미세 조정 추가
BERT_LARGE (Ens.+TriviaQA): BERT_LARGE 앙상블 모델에 TriviaQA 미세 조정 추가

BERT_LARGE (Ens.+TriviaQA) 모델이 테스트에서 최고 성능을 보여줌.
BERT_LARGE (Sgl.+TriviaQA) 단일 모델도 상위 앙상블 시스템을 능가.
* TriviaQA 미세 조정 없이도 BERT 모델이 기존 시스템보다 높은 성능을 보임.

4.3 SQuAD v2.0

SQuAD 2.0 과제는 짧은 답변이 없는 경우를 포함하도록 확장되었습니다. 우리는 질문이 답변이 없는 경우 [CLS] 토큰에서 시작하고 끝나는 답변 범위를 갖는 것으로 처리했습니다. 답변 범위의 시작과 끝 위치에 대한 확률 공간을 [CLS] 토큰의 위치를 포함하도록 확장했습니다. 예측을 위해, 답변이 없는 범위의 점수를 가장 좋은 null이 아닌 범위의 점수와 비교하여 예측했습니다. 이 모델에 TriviaQA 데이터를 사용하지 않았으며, 2 에포크 동안 학습했습니다. 이전 최상의 시스템에 비해 +5.1 F1 향상을 관찰했습니다.

System	Dev		Test	
	EM	F1	EM	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

Table 3: SQuAD 2.0 results. We exclude entries that use BERT as one of their components.

#1 Single – MIR-MRC (F-Net): 단일 모델 MIR-MRC (F-Net)
#2 Single – nlnet: 단일 모델 nlnet

unet (Ensemble): 앙상블 모델 unet
SLQA+ (Single): 단일 모델 SLQA+

BERT_LARGE (Single): 단일 BERT_LARGE 모델

BERT_LARGE (Single) 모델이 테스트에서 최고 성능을 보여줌.
BERT_LARGE 모델이 기존 출판된 모델과 상위 리더보드 시스템보다 높은 성능을 보임.

4.4 SWAG

SWAG 데이터셋은 문장 쌍 완성 예제를 통해 상식 추론을 평가하는 과제입니다.

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

Table 4: SWAG Dev and Test accuracies. †Human performance is measured with 100 samples, as reported in the SWAG paper.

SWAG 개발(Dev) 및 테스트(Test) 데이터셋에 대한 정확도를 보여줍니다.
BERT_LARGE는 테스트 데이터에서 86.3%의 정확도를 기록하였습니다.

5 Ablation Studies(소거 실험)

모델이나 알고리즘의 기능(feature)들을 하나씩 제거해보면서 이 기능이 전체 성능에 얼마나 영향을 미치는 지 확인해보는 것이다.

	Dev Set				
Tasks	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.

BERTBASE: 표준 BERTBASE 모델.
No NSP: 다음 문장 예측 없이 훈련된 모델.
LTR & No NSP: 다음 문장 예측 없이 왼쪽에서 오른쪽으로의 언어 모델로 훈련된 모델.
+ BiLSTM: "LTR + No NSP" 모델 상단에 임의로 초기화된 BiLSTM을 추가한 모델.

5.1 사전 학습 작업의 효과

- BERTBASE 모델이 전반적으로 가장 높은 성능을 보임.
- 다음 문장 예측 작업(NSP)이 없을 때 성능이 다소 감소함.
- LTR & No NSP 모델은 성능이 더 크게 감소함.
- BiLSTM을 추가해도 성능이 크게 개선되지 않음.

5.2 모델 크기의 효과

모델 크기의 영향: 더 큰 BERT 모델은 더 높은 정확도를 가져옵니다.
사전 학습의 중요성: 충분히 사전 학습된 경우, 작은 데이터셋에서도 큰 모델의 이점을 얻을 수 있습니다.
GLUE 작업: GLUE 작업의 모든 데이터셋에서 더 큰 모델이 일관된 성능 향상을 보여줍니다.
기존 연구와의 비교: BERT 모델은 기존 문헌의 모델보다 훨씬 큰 규모로 성능을 개선합니다.

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.

표 6: BERT 모델 크기 분석

#L: 레이어 수 (number of layers)
#H: 히든 사이즈 (hidden size)
#A: 어텐션 헤드 수 (number of attention heads)

LM (ppl): 홀드아웃(hold-out) 훈련 데이터의 masked LM perplexity
MNLI-m: MNLI-matched 정확도
MRPC: MRPC 정확도
SST-2: SST-2 정확도

* BERT 모델 크기가 커질수록(레이어 수, 히든 사이즈, 어텐션 헤드 수가 증가할수록) 성능이 향상됨을 보여줍니다.
이 결과는 더 큰 모델이 더 복잡한 표현을 학습할 수 있음을 나타내며, 모델 크기를 늘리는 것이 자연어 처리 작업에서 유리할 수 있다고 판단됩니다.

5.3 BERT의 Feature-based 접근 방식

Feature-based 접근 방식: 사전 학습된 모델에서 고정된 기능을 추출하여 사용하는 방식.
미세 조정 접근 방식과 feature-based 접근 방식 비교 결과:
BERTLARGE는 최첨단 방법들과 경쟁력 있는 성능을 보여주며, feature-based 접근 방식에서도 좋은 성능을 보임.

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.

Fine-tuning approach: BERT 모델을 직접 미세 조정하여 NER 작업에서 높은 성능을 보입니다.
Feature-based approach: 미세 조정을 하지 않고 BERT 모델에서 고정된 특징을 추출하여 사용했습니다. 가장 높은 Dev F1 점수는 Concat Last Four Hidden 방법으로 96.1을 기록했습니다.
* Fine-tuning approach와 Feature-based approach 모두 강력한 성능을 보여줍니다.

6. 결론

최근 언어 모델을 통한 전이 학습이 언어 이해 시스템에서 중요한 역할을 한다는 것이 입증되었습니다. 이러한 접근법은 자원이 적은 작업에서도 심층 단방향 아키텍처의 혜택을 제공합니다. 이 논문에서는 이러한 결과를 심층 양방향 아키텍처로 확장하여 동일한 사전 학습된 모델이 다양한 자연어 처리(NLP) 작업을 성공적으로 수행할 수 있음을 보여줍니다.

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

- BERT에 대한 추가 구현 세부 사항은 부록 A에 제시됩니다.
- 실험에 대한 추가 세부 사항은 부록 B에 제시됩니다.
- 추가 절제 연구는 부록 C에 제시됩니다.

A BERT에 대한 추가 세부 사항

A.1 사전 학습 작업의 설명

BERT의 추가 세부 사항에서는 마스킹된 LM과 다음 문장 예측 작업의 절차와 예시를 설명합니다. 마스킹된 LM은 토큰의 15%를 마스킹하거나 무작위로 변경하여 모델이 분포적 맥락 표현을 유지하게 합니다. 다음 문장 예측 작업은 두 문장이 연속되는지 여부를 예측하도록 모델을 훈련합니다. 이 절차들은 BERT가 더 나은 언어 이해 능력을 갖추도록 돕습니다.

A.2 사전 학습 절차

사전 학습 절차에서는 두 개의 텍스트 스팬을 샘플링하여 A와 B 임베딩을 부여하고, 50%의 경우 실제 다음 문장을, 50%의 경우 무작위 문장을 사용합니다. 배치 크기는 256 시퀀스, 학습 단계는 1,000,000이며, Adam 옵티마이저와 gelu 활성화 함수를 사용합니다. BERTBASE와 BERTLARGE의 훈련은 각각 4일이 소요되었으며, 긴 시퀀스는 학습 속도를 위해 마지막 10% 단계에서만 사용됩니다.

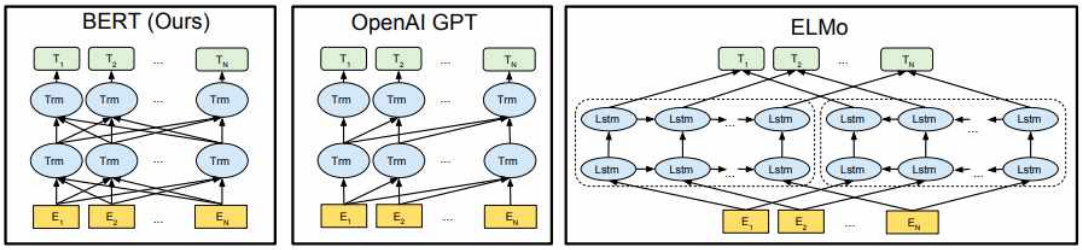


Figure 3: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTMs to generate features for downstream tasks. Among the three, only BERT representations are jointly conditioned on both left and right context in all layers. In addition to the architecture differences, BERT and OpenAI GPT are fine-tuning approaches, while ELMo is a feature-based approach.

그림 3: 사전 학습 모델 아키텍처의 차이점

BERT: 양방향 Transformer 사용.
OpenAI GPT: 좌에서 우로 읽는 Transformer 사용.
ELMo: 좌우로 읽는 LSTM을 독립적으로 학습하여 특징 생성.

BERT는 모든 레이어에서 좌우 문맥을 모두 고려하여 학습된 표현을 제공하며, BERT와 OpenAI GPT는 미세 조정 접근 방식을, ELMo는 특징 기반 접근 방식을 사용합니다.

A.3 미세 조정 절차

미세 조정 절차에서는 사전 학습과 동일한 하이퍼파라미터를 사용하지만, 배치 크기, 학습률, 학습 에포크 수는 작업에 따라 다르게 설정합니다. 일반적으로 배치 크기는 16 또는 32, 학습률은 5e-5, 3e-5, 2e-5 중에서 선택하며, 에포크 수는 2, 3, 4 중에서 선택합니다. 큰 데이터 세트는 하이퍼파라미터 선택에 덜 민감하므로, 철저한 검색을 통해 최적의 모델을 선택합니다.

A.4 BERT, ELMo, OpenAI GPT 비교

BERT, ELMo, OpenAI GPT의 차이점을 비교합니다. BERT와 OpenAI GPT는 미세 조정 접근 방식을 사용하는 반면, ELMo는 피처 기반 접근 방식을 사용합니다. BERT와 GPT의 학습 방법 간의 주요 차이점은 학습 데이터, 학습 단계, 배치 크기 및 미세 조정 학습률입니다. 실험 결과, BERT의 주요 개선 사항은 두 가지 사전 학습 작업과 양방향성에서 비롯된다는 것을 입증합니다.

A.5 다양한 작업에 대한 미세 조정 예시

그림 4에서는 BERT를 다양한 작업에 미세 조정하는 예시를 보여줍니다. 작업별 모델은 BERT에 하나의 추가 출력 레이어를 통합하여 형성되며, 최소한의 매개변수만 새로 학습합니다. (a)와 (b)는 시퀀스 수준의 작업이고, (c)와 (d)는 토큰 수준의 작업입니다. 그림에서 E는 입력 임베딩을, Ti는 토큰 i의 문맥적 표현을 나타내며, [CLS]는 분류 출력, [SEP]는 비연속적인 토큰 시퀀스를 구분하는 특수 기호입니다.

B. 상세 실험 설정

B.1 GLUE 벤치마크 실험에 대한 상세 설명

MNLI: 문장 쌍의 함축, 모순, 중립을 예측.
QQP: 두 Quora 질문이 의미적으로 동등한지 판별.
QNLI: 질문과 문장 쌍이 정답을 포함하는지 여부를 판별.
SST-2: 영화 리뷰 문장의 감정을 이진 분류.
CoLA: 영어 문장의 언어적 허용 가능성을 예측.
STS-B: 문장 쌍의 의미적 유사성을 점수로 매김.
MRPC: 온라인 뉴스 문장 쌍의 의미적 동등성 판별.
RTE: 적은 데이터로 문장 함축 여부를 이진 분류.
WNLI: 자연어 추론 데이터셋으로, 공정한 비교를 위해 제외.

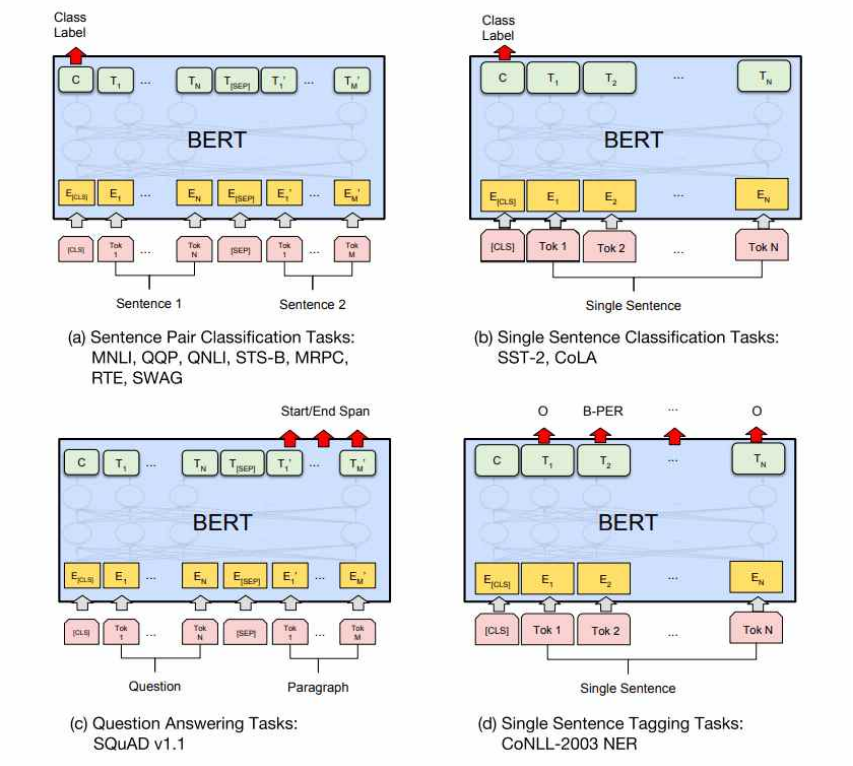


그림 4: 다양한 작업에서 BERT의 파인 튜닝에 대한 설명
이 그림은 하나의 추가 출력 레이어를 사용하여 BERT를 다양한 작업에 맞게 파인 튜닝하는 방법을 설명합니다.

(a) 문장 쌍 분류 작업
작업: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG
설명: 모델은 두 개의 문장(문장 1과 문장 2)을 입력으로 받습니다. 각 문장은 토큰 임베딩, 세그먼트 임베딩(E_A는 문장 1, E_B는 문장 2) 및 위치 임베딩으로 인코딩됩니다. [CLS] 및 [SEP] 특별 토큰은 시퀀스의 시작과 문장 사이의 구분을 나타냅니다. [CLS] 토큰에 해당하는 최종 은닉 상태(C)는 분류를 위해 사용됩니다.

(b) 단일 문장 분류 작업
작업: SST-2, CoLA
설명: 모델은 단일 문장을 처리합니다. 입력 시퀀스에는 문장의 시작을 나타내는 특별 [CLS] 토큰과 문장 토큰들이 포함됩니다. [CLS] 토큰에 해당하는 최종 은닉 상태(C)는 분류를 위해 사용됩니다.

(c) 질문 응답 작업
작업: SQuAD v1.1
설명: 입력은 질문과 단락으로 구성됩니다. 각각은 문장 쌍 분류 작업과 유사하게 인코딩되며, 세그먼트 임베딩이 질문과 단락을 구분합니다. 답변 범위의 시작 및 끝 위치는 토큰 표현(T)에서 예측됩니다.

(d) 단일 문장 태깅 작업
작업: CoNLL-2003 NER
설명: 모델은 토큰 수준 태깅을 위해 단일 문장을 처리합니다. 입력에는 [CLS] 특별 토큰과 문장 토큰들이 포함됩니다. [CLS] 토큰에 해당하는 최종 은닉 상태는 분류를 위해 사용됩니다.

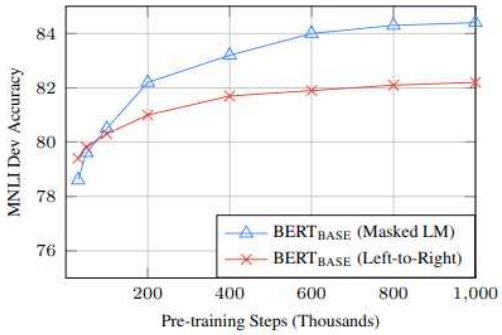
C 추가 실험 연구

C.1 학습 단계 수의 효과

추가 실험 연구에서, BERT가 높은 정확도를 달성하기 위해 많은 양의 사전 훈련이 필요하며, MLM 사전 훈련은 LTR 사전 훈련보다 약간 느리게 수렴하지만 더 나은 성능을 보인다는 것을 확인했습니다.

C.2 다양한 마스킹 절차에 대한 절제 실험

다양한 마스킹 절차에 대한 절제 실험에서, 파인 튜닝은 다양한 마스킹 전략에 대해 강인한 반면, 특징 기반 접근 방식에서는 MASK 전략만 사용하는 것이 문제가 있었음을 확인했습니다. RND 전략만 사용하는 것도 성능이 떨어지는 것으로 나타났습니다.



BERT 모델이 많은 사전 훈련 단계가 필요함을 보여주며, 마스킹된 언어 모델(MLM)이 왼쪽에서 오른쪽(LTR) 모델보다 더 높은 정확도를 제공함을 시사합니다. MLM 모델은 초기부터 LTR 모델보다 우수한 성능을 보이지만, 수렴 속도는 약간 더 느립니다.

Masking Rates			Dev Set Results		
MASK	SAME	RND	MNLI	NER	
			Fine-tune	Fine-tune	Feature-based
80%	10%	10%	84.2	95.4	94.9
100%	0%	0%	84.3	94.9	94.0
80%	0%	20%	84.1	95.2	94.6
80%	20%	0%	84.4	95.2	94.7
0%	20%	80%	83.7	94.8	94.6
0%	0%	100%	83.6	94.9	94.6

Table 8: Ablation over different masking strategies.

이 테이블은 다양한 마스킹 전략에 따른 절제 실험 결과를 보여줍니다.

MASK: 마스킹된 단어를 [MASK] 토큰으로 대체하는 비율.
SAME: 마스킹된 단어를 그대로 두는 비율.
RND: 마스킹된 단어를 무작위 단어로 대체하는 비율.

MNLI Fine-tune: MNLI 데이터셋에서의 파인 튜닝된 모델의 성능.
NER Fine-tune: NER 데이터셋에서의 파인 튜닝된 모델의 성능.
NER Feature-based: NER 데이터셋에서 피처 기반 접근 방식을 사용한 모델의 성능.

MNLI Fine-tune:
마스킹 전략이 80% MASK, 10% SAME, 10% RND일 때 가장 높은 정확도(84.4%)를 달성합니다.
100% MASK만 사용한 경우와 SAME과 RND만 사용한 경우 성능이 떨어집니다.

NER Fine-tune:
대부분의 마스킹 전략에서 높은 정확도(95.2-95.4%)를 보이며, 100% MASK 전략에서도 성능이 좋습니다(94.9%).
RND만 사용한 경우에도 성능이 높습니다(94.9%).

NER Feature-based:
80% MASK, 20% SAME 전략에서 가장 높은 정확도(94.7%)를 달성합니다.
100% MASK 전략에서 성능이 떨어집니다(94.0%).

이 절제 실험 결과는 다양한 마스킹 전략이 모델의 성능에 미치는 영향을 보여줍니다. 전반적으로 파인 튜닝된 모델은 마스킹 전략에 덜 민감하지만, 피처 기반 접근 방식에서는 적절한 마스킹 전략이 성능에 큰 영향을 미칠 수 있습니다. 특히, 마스킹된 단어를 [MASK] 토큰으로 대체하는 비율을 80%로 설정하고, 나머지 20%를 SAME 또는 RND로 설정하는 것이 좋은 성능을 보이는 경향이 있습니다.