

BERT: Pre-training of Bidirectional Transformers for Language Understanding

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre- training of deep bidirectional transformers for language under- standing. arXiv preprint arXiv:1810.04805, 2018.

소개

- 많은 NLP task에서 language model pre-training은 유의미한 성능 향상을 보여줌
- Pre-train된 language representation을 적용하는 방식에는 크게 두 가지가 존재
 - 1. Feature-based
 - 2. Fine-tuning

소개

1. Feature-based

- 특정 task에 맞춰진 구조의 모델에 pre-trained representation을 추가적인 feature로 사용
- 추가적으로 사용되는 pre-trained representation은 학습하지 않음
- Ex: ELMo²

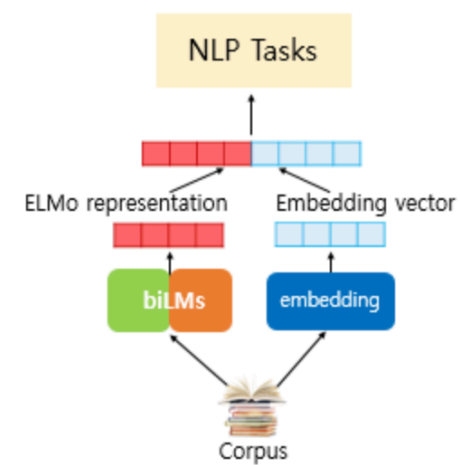
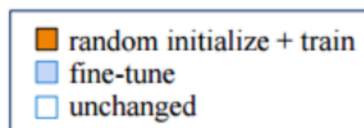
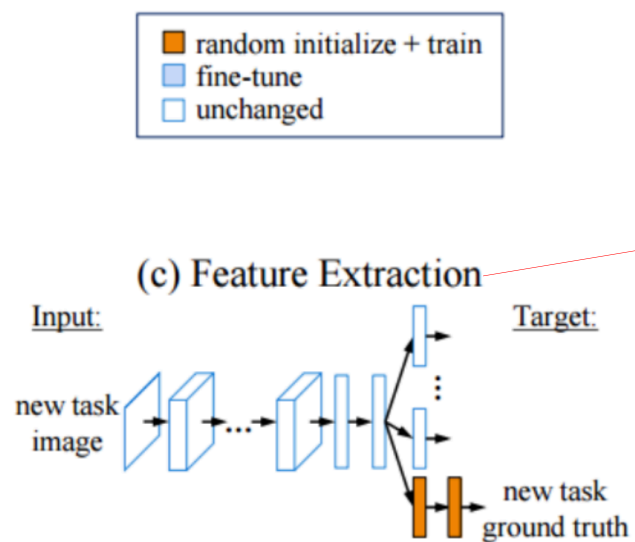
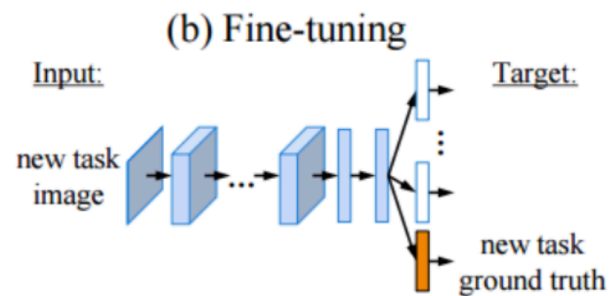
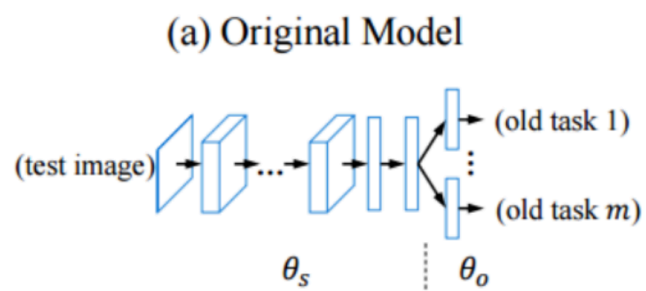
2. Fine-tuning

- Task에 따라 달라지는 파라미터들을 최소화하여 모델을 구성
- Task마다 pre-trained representation을 포함한 모든 파라미터들을 재학습
- Ex: OpenAI GPT³

2. Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In NAACL.

3. Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, OpenAI.

소개

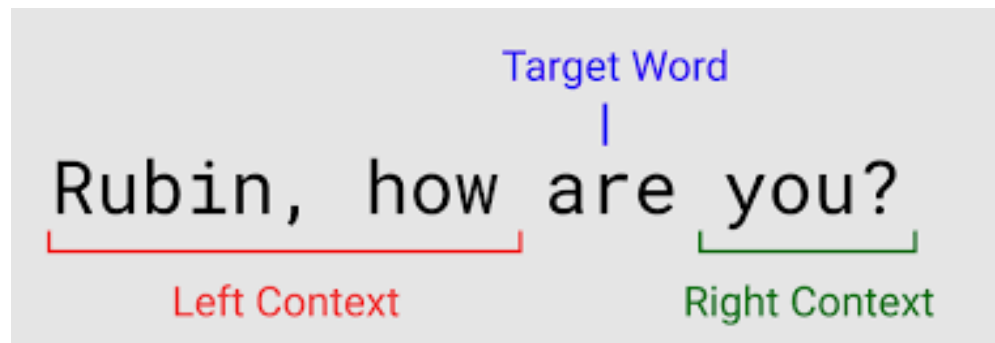


소개

- 기존에는 앞의 두 가지 방식 모두 unidirectional language model을 사용하여 general language representation을 학습

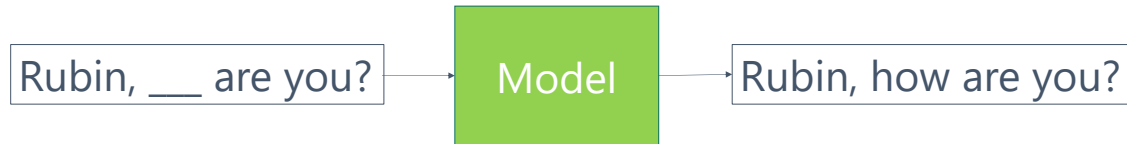


- 이는 양방향에서의 문맥이 중요한 언어 task에서 치명적인 문제



소개

- BERT는 "masked language model(MLM)"을 사용함으로 이 문제를 해결



- 추가적으로 "next sentence prediction" task를 동시에 학습함으로 성능을 향상시킴



BERT

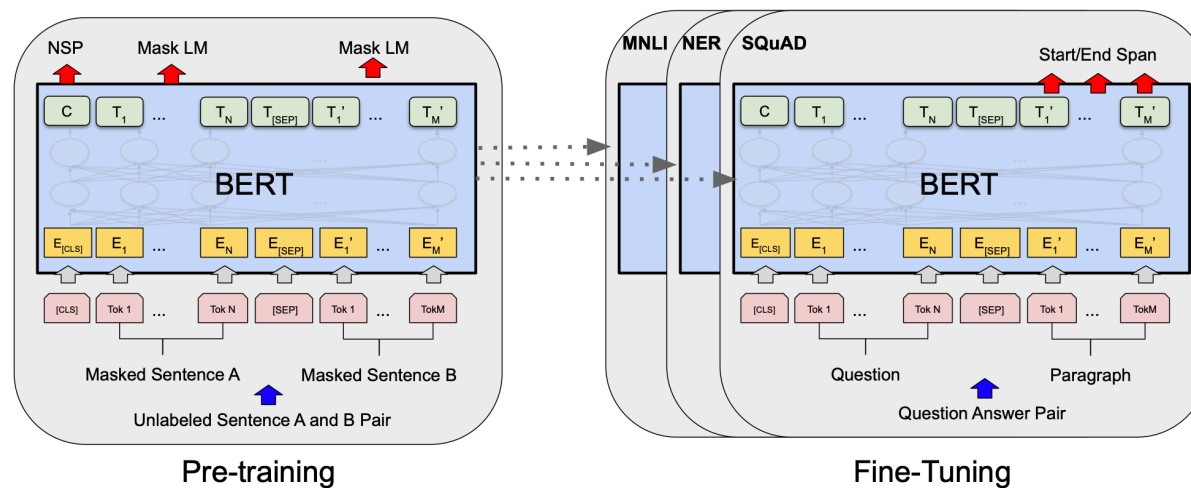
• 두 단계로 나눠서 진행

1. Pre-training

- Label이 없는 데이터를 사용해 앞서 언급된 pre-training task를 학습

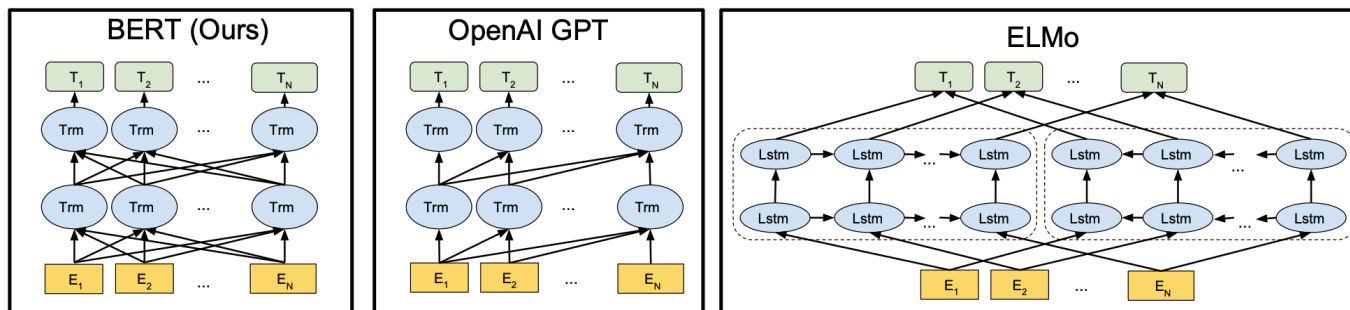
2. Fine-tuning

- 상기 단계에서 pre-train된 파라미터들에 실제로 사용될 여러 task에 맞춰진 최소한의 파라미터들만을 더하여 재학습



BERT 모델 구조

- Transformer⁴의 encoder 부분을 차용
 - Bidirectional한 구조
- 2 가지 크기의 모델을 실험
 - BERT_{BASE}
 - 비교를 위해 OpenAI GPT(Transformer의 decoder 부분 사용)와 같은 크기로 설계
 - Total Parameters = 110M
 - BERT_{LARGE}
 - Total Parameters = 340M



4. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems, pages 6000–6010.

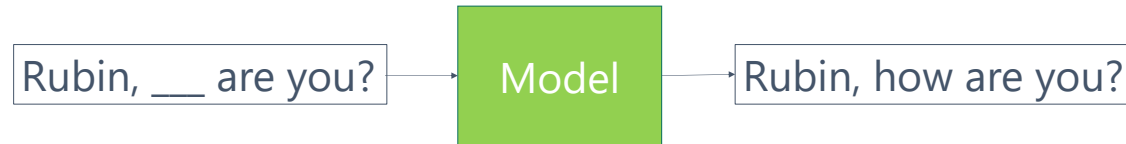
BERT 입력

- 한 token sequence로 하나의 문장 혹은 한 쌍의 문장을 입력
 - 한 쌍의 문장을 입력하는 경우 각 문장 사이를 특수한 token [SEP]으로 분리
 - 각 토큰이 문장 A 혹은 문장 B에 속하는지를 학습한 embedding 추가
- 30,000 token 어휘를 가진 WordPiece embedding 사용
- 모든 sequence의 첫 token은 언제나 특수한 토큰 [CLS]로 고정
 - 이 token으로부터의 출력을 sequence 단위 representation으로 사용
- WordPiece embedding, 문장 segment embedding, positional embedding(기존 transformer에서 사용)을 더하여 최종 입력 완성

Input	[CLS]	my	dog	is	cute	[SEP]	he	likes	play	##ing	[SEP]
Token Embeddings	$E_{[CLS]}$	E_{my}	E_{dog}	E_{is}	E_{cute}	$E_{[SEP]}$	E_{he}	E_{likes}	E_{play}	$E_{\# \# ing}$	$E_{[SEP]}$
	+	+	+	+	+	+	+	+	+	+	+
Segment Embeddings	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B	E_B
	+	+	+	+	+	+	+	+	+	+	+
Position Embeddings	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}

BERT pre-training

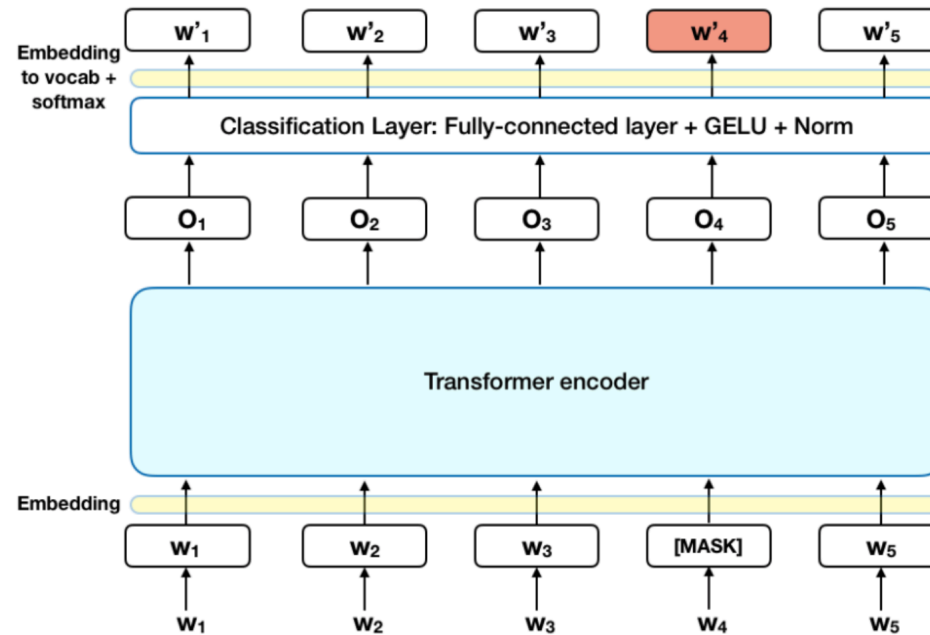
1. Masked Language Model(Masked LM)



- Left-to-right LM 혹은 right-to-left LM의 경우 양방향을 보게 된다면 정답을 보는 것과 같은 효과
- 이를 완화하기 위해 항상 다음 token을 예측하는 것이 아닌 무작위로 중간 token을 지우고 해당 token을 예측하는 masked LM 사용

BERT pre-training

1. Masked Language Model(Masked LM)



- 각 sequence에서 무작위로 15% 정도의 token을 채택
 - 채택된 token의 결과에 대해서만 loss 계산

BERT pre-training

1. Masked Language Model(Masked LM)

- 채택된 15%의 토큰에 대해 다음의 작업을 진행
 - 80%는 masking

80%: my dog is hairy → my dog is [MASK]

- 10%는 무작위로 다른 token 할당

10%: my dog is hairy → my dog is apple

- 10%는 본래 token 재사용

10%: my dog is hairy → my dog is hairy

- 실제 task에서는 masking된 어휘가 등장하지 않기 때문에 이 차이를 완화하기 위한 조치

BERT pre-training

2. Next Sentence Prediction(NSP)



- 두 문장 간의 관계를 기반으로 한 task(ex: QA, NLI) 성능 향상을 위해 해당 관계를 학습하고자 함
- 이를 완화하기 위해 항상 다음 token을 예측하는 것이 아닌 무작위로 중간 token을 지우고 해당 token을 예측하는 masked LM 사용

BERT pre-training

2. Next Sentence Prediction(NSP)

Input = [CLS] the man went to [MASK] store [SEP]
 he bought a gallon [MASK] milk [SEP]

Label = IsNext

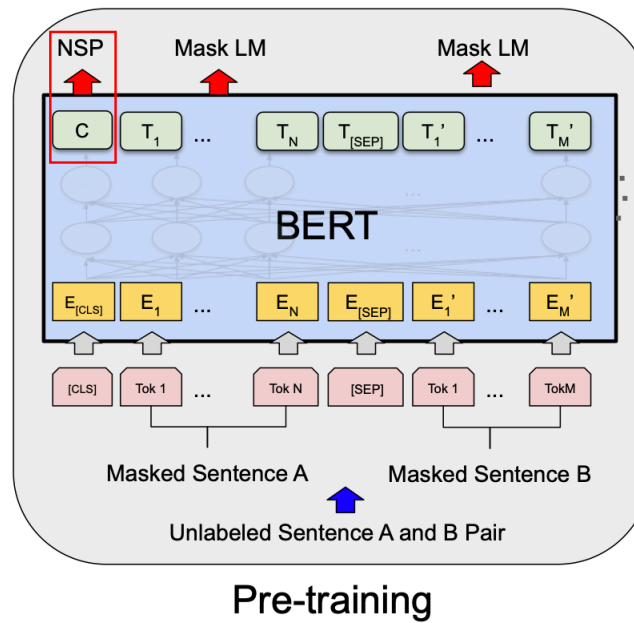
Input = [CLS] the man [MASK] to the store [SEP]
 penguin [MASK] are flight ##less birds [SEP]

Label = NotNext

- 50%의 경우 실제로 연속된 문장으로 sequence 구성
- 나머지의 경우 두 문장을 무작위로 채택해 sequence 구성

BERT pre-training

2. Next Sentence Prediction(NSP)



- CLS token의 출력으로 나온 C를 사용하여 예측

BERT pre-training

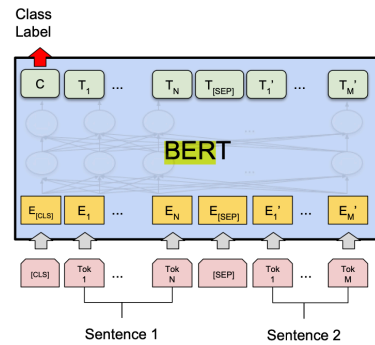
- Data
 - BooksCorpus(800M 단어)
 - English Wikipedia(2,500M 단어)
 - 연속된 sequence를 추출하기 위해 문장들이 무작위로 배치되지 않은 데이터를 사용하는 것이 중요

BERT pre-training

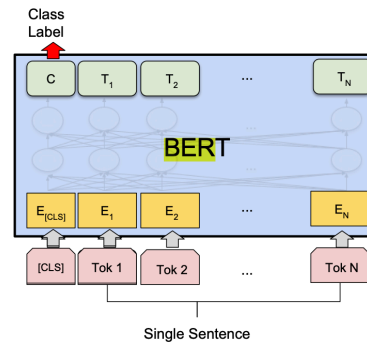
- 학습 절차

- 두 문장에 속한 token 수의 합이 512를 넘지 않도록 sequence를 구성
- Batch size: 256 sequences
- Training steps: 1,000,000 steps
- Optimizer: Adam
 - Learning rate = $1e-4$
 - $\beta_1 = 0.9$
 - $\beta_2 = 0.999$
 - L2 weight decay = 0.01
 - Learning rate warmup over the first 10,000 steps, and linear decay of the learning rate
- Dropout probability: 0.1
- gelu activation
- Loss: mean masked LM likelihood + mean NSP likelihood
- Pre-training 시간을 줄이기 위해 90%의 training step에서는 sequence 길이를 128로 조정(나머지 10% step에서 sequence 길이 512 사용)

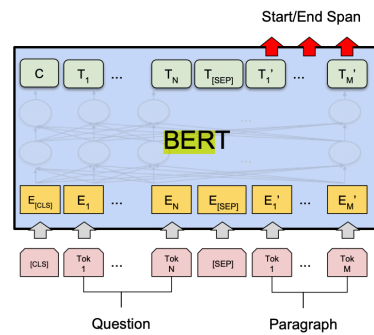
BERT fine-tuning



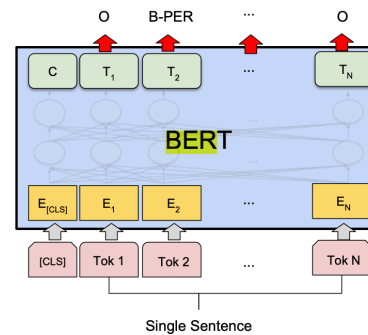
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



(b) Single Sentence Classification Tasks:
SST-2, CoLA



(c) Question Answering Tasks:
SQuAD v1.1

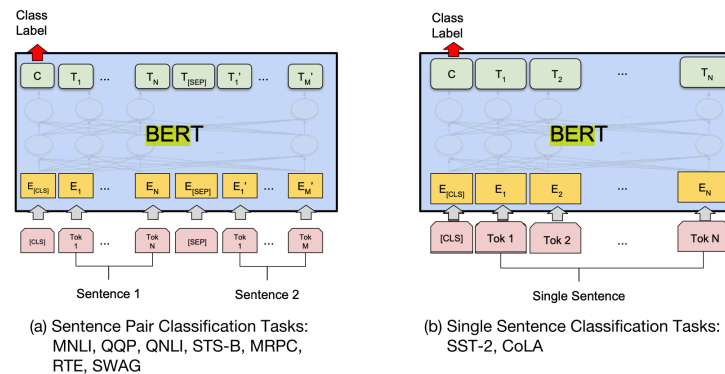


(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

- 각 task에 맞는 입출력을 가지도록 수정한 뒤 재학습

Fine-tuning 실험

- General Language Understanding Evaluation(GLUE)



- 단일 sequence 혹은 한 쌍의 sequence를 입력으로 하는 sequence-level task들로 구성
- ex: MNLI(한 쌍의 문장이 주어졌을 때 문장 간의 관계를 예측), SST-2(단일 문장으로 구성된 영화 리뷰로부터 감성 분석), ...
- [CLS] token으로부터 pre-trained된 BERT를 사용하여 산출된 $C \in \mathbb{R}^H$ 벡터와 새로 추가된 classification layer weights $W \in \mathbb{R}^{K \times H}$ ($K = \#$ of labels)를 사용하여 출력 산출
- $\text{softmax}(CW^T)$
- 위의 출력과 label을 비교하여 전체 모델 재학습

Fine-tuning 실험

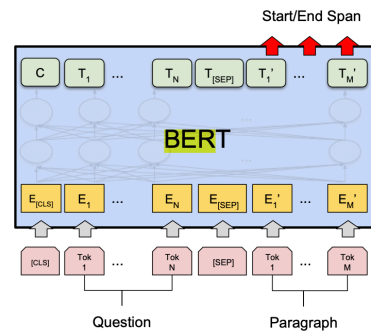
- General Language Understanding Evaluation(GLUE)

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
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

- BERT_{LARGE} 모델은 작은 데이터셋에서 안정적이지 않아 random initialization을 여러번 시도하여 가장 높은 성능의 모델을 채택

Fine-tuning 실험

- Stanford Question Answering Dataset(SQuAD v1.1)

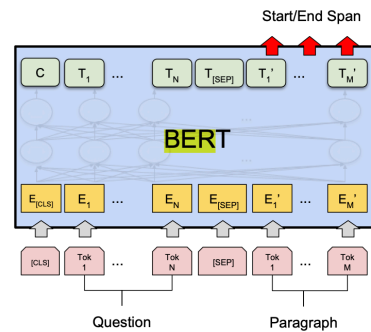


(c) Question Answering Tasks:
SQuAD v1.1

- 100K 개의 문답 쌍으로 구성
- 질문 그리고 답이 포함된 지문이 주어졌을 때 지문에서 답을 찾는 task

Fine-tuning 실험

- Stanford Question Answering Dataset(SQuAD v1.1)



(c) Question Answering Tasks:
SQuAD v1.1

- 새로운 벡터 2개 추가
 - $S \in \mathbb{R}^H$ (정답의 시작 부분을 찾기 위한 벡터)
 - $E \in \mathbb{R}^H$ (정답의 끝 부분을 찾기 위한 벡터)
- i 번째 단어가 정답의 시작일 확률: $P_i = \frac{e^{ST_i}}{\sum_j e^{ST_j}}$
- i 보다 크거나 같은 k 번째 단어가 정답의 끝일 확률도 같은 방식으로 구한 뒤 $ST_i + ET_k$ 가 제일 높은 span을 출력하도록 학습
- 학습 objective 함수는 옳은 시작과 끝 위치의 log-likelihood의 합

Fine-tuning 실험

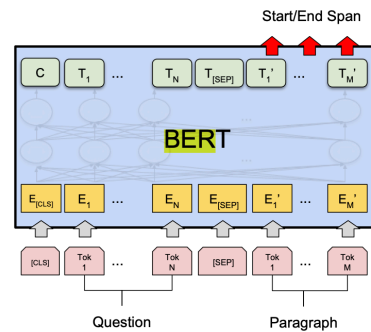
- Stanford Question Answering Dataset(SQuAD v1.1)

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				
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 fine-tuning하기 전 TriviaQA 데이터를 사용하여 fine-tuning 진행

Fine-tuning 실험

- SQuAD v2.0



(c) Question Answering Tasks:
SQuAD v1.1

- SQuAD 1.1 task의 (질문, 지문) 쌍 중 지문 부분에 답이 없는 경우도 추가한 task
- 답이 없는 경우는 [CLS] token에서 답이 시작하고 끝나도록 학습
- 답이 없는 경우의 점수 $S_{null} = SC + EC$
- 지문 안에서 답일 확률이 제일 높은 점수 $S_{i,j} = \max_{j \geq i} ST_i + ET_j$
- $S_{i,j} > S_{null} + \tau$ (τ = hyperparameter) 이면 $S_{i,j}$ 를 정답으로 출력 그 외에는 정답이 없는 것으로 출력

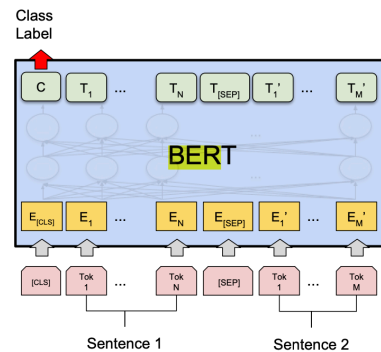
Fine-tuning 실험

- SQuAD v2.0

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				
BERT _{LARGE} (Single)	78.7	81.9	80.0	83.1

Fine-tuning 실험

- Situations with Adversarial Generations(SWAG)



(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG

- 113K 개의 문장 쌍으로 구성
- 문장이 주어졌을 때 4 가지 선택지 중 이어질 문장으로 가장 올바른 것을 고르는 task
- 문장과 선택지 4개를 하나씩 각각 붙인(ex: (문장, 선택지 1), (문장, 선택지 2), ...) 4 개의 sequence를 입력
- Sequence-level task이므로 [CLS] token의 embedding C를 사용하여 normalization, classification 진행

Fine-tuning 실험

- Situations with Adversarial Generations(SWAG)

System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
OpenAI GPT	-	78.0
BER T _{BASE}	81.6	-
BER T _{LARGE}	86.6	86.3
Human (expert) [†]	-	85.0
Human (5 annotations) [†]	-	88.0

모델 변형 결과

• Pre-training task의 효과

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

- No NSP -> QNLI, MNLI, SQuAD 성능 크게 하락
- No bidirectionality -> 전체적으로 성능 하락했지만 MRPC, SQuAD에서 특히 크게 하락
- 오른쪽 context가 아예 없는 것에 대한 보상으로 BiLSTM 추가
 - SQuAD 성능은 좋아졌지만 여전히 Masked LM 사용한 것에 못 미치고, 나머지 task에서는 오히려 성능이 하락
- ELMo처럼 LTR, RTL 모델을 concat하는 것은 밑의 이유로 실험하지 않음
 - 학습 시간 두 배로 소요
 - QA 등의 task에서 RTL은 정답을 보는 격이므로 합당하지 않음
 - 한 층에서 bidirectional한 것보다 성능이 안 좋을 것이 자명하기 때문

모델 변형 결과

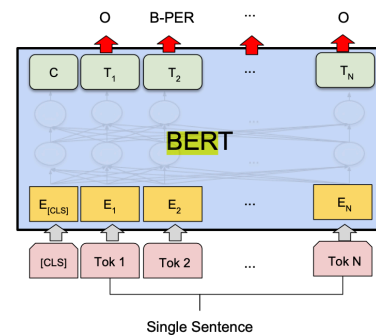
• 모델 크기의 효과

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

- Fine-tuning 시 5 번 무작위로 초기화하여 학습한 결과의 평균
- 모델 크기가 클수록 성능이 향상됨
 - 비교적 사이즈가 작은 MRPC 역시 해당
- 모델이 충분히 pre-train됐다면 작은 scale의 task에서도 모델 크기를 키우는 것이 성능을 향상시킨다는 것을 보여줌

모델 변형 결과

- Feature-based 접근의 효과



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

- Feature-based 접근에는 다음의 장점이 존재
 - Transformer encoder 구조가 모든 task를 represent할 수 없기 때문에 task에 적합한 모델 구조를 사용할 수 있다는 장점
 - 학습 비용 절감
- CoNLL-2003 Named Entity Recognition(NER) task로 두 접근 성능 비교
- Tagging task에서 흔히 사용되는 CRF(conditional random field) 사용하지 않음
- 대신 첫 번째 sub-token의 representation을 token-level classification의 입력으로 사용

모델 변형 결과

• Feature-based 접근의 효과

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		
BERT _{LARGE}	96.6	92.8
BERT _{BASE}	96.4	92.4
Feature-based approach (BERT _{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	-

- BERT의 한 개 혹은 그 이상의 층에서의 결과를 조합해 feature로 사용
- 2 층, 768 차원의 BiLSTM 분류기의 입력으로 사용
- 가장 좋은 성능의 feature-based 모델은 fine-tuning 모델과 성능 차이가 크지 않음
- BERT는 두 가지 접근 모두에서 좋은 성능을 보여줌

결론

- Bidirectionality가 중요함을 입증