BERT: Pre-training of Deep Bidirectional Transformers forLanguage Understanding

tag	NLP
☑	☐
Paper Link	https://arxiv.org/abs/1810.04805?source=post_page
published	NAACL 2019

▼ References

BERT 논문정리 · MinhoPark

Follow me :

https://mino-park7.github.io/nlp/2018/12/12/bert-논문정리/?fbclid=IwAR3S-8iLWEVG6FGUVxoYdwQyA-zG0GpOUzVEsFBd0ARFg4eFXqCyGLznu7w#4-experiments

BERT: Bidirectional Transformers for Language Understanding

An Ed edition

https://reniew.github.io/47/

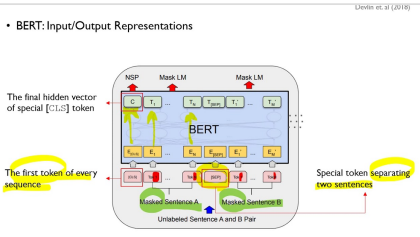
08-5: BERT

고려대학교 산업경영공학과 일반대학원

Unstructured Data Analysis

08-5: GPT

https://www.youtube.com/watch?v=IwtexRHoWG0



▼ 목차

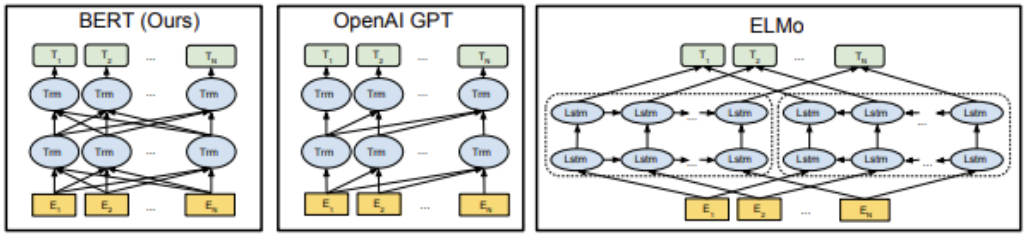
Abstract

BERT(**B**idirectional **E**ncoder **R**epresentations from **T**ransformers)

- 모든 레이어에 **양방향 문맥**에서 공동으로 조절함으로써 라벨이 없는 텍스트로부터 깊은 양방향 표현들을 사전 훈련하도록 디자인됨
- 작업별 모델 구조를 수정할 필요 없이도 **단 하나의 출력 레이어를 추가**하여 파인튜닝 가능
- QA, NLI를 포함한 11개의 자연어 처리 작업에서 새로운 SOTA 결과를 달성

Introduction & Related Work

- 사전 훈련된 language representations을 downstream task에 적용하기 위해서 2가지 전략 존재
  - **Feature-based Approach ex) ELMO**
    - task-specific architectures
    - pre-trained representations (embedding layer)를 추가적인 feature로 사용
  - **Fine-tuning Approach ex) GPT, Transformer**
    - 일반적인 task를 위해 학습한 모델을 이용하고, 이후에 downstream task에서 특정 task에 맞게 파라미터를 미세하게 조정



📌 **ELMO**

- Left-to-right, Right-to-left 구조
- 정방향 LSTM과 역방향 LSTM을 각각 훈련시키는 방식으로 양방향 언어 모델

  → **shallow** concatenation of independently trained LMs

GPT

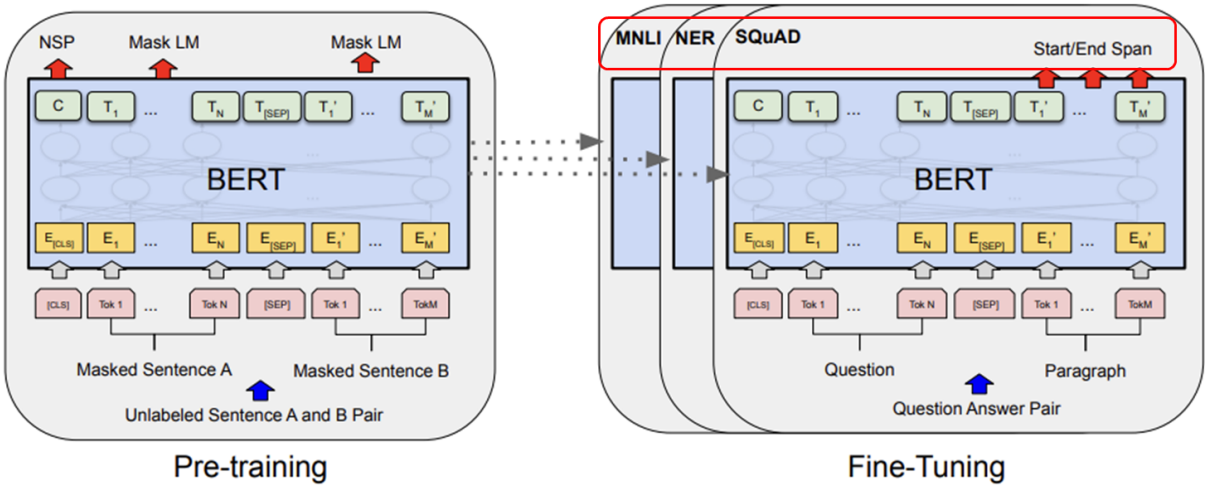
- Left-to-right 구조
- 모든 토큰이 트랜스포머의 셀프 어텐션 레이어에서 이전 토큰에만 참조 가능

- 특히 이러한 **단방향 언어 모델**은 **fine-turning에서 한계**가 존재
  - 양방향에서의 문맥 정보가 모두 중요한 **token-level task**(ex. Named Entity Recognition, Question Answering)**에서 높은 성능을 보이지 못 하는 것**

⇒ 따라서, **Masked Language Model(MLM)**을 사용하여 **양방향의 문맥을 학습**할 수 있는 BERT 모델을 고안

BERT

pre-training → fine-tuning



- pre-training : unlabeled data에 대해 학습한 후, fine-tuning 진행
- fine-tuning : 사전훈련된 파라미터로 초기화되고, downstream task에 맞는 labeled data에 맞게 하이퍼파라미터를 조정
- 서로 **다른 task들에 대해서도 같은 모델 구조**를 갖는다

📌 [CLS] : 모든 시퀀스의 시작을 알려주는 첫 번째 토큰

[SEP] : 2개의 문장을 구분해주는 시퀀스

T_N : N번째 토큰의 final hidden vector

C : CLS 토큰의 final hidden vector

1) Model Architecture

- BERT는 **multi-layer bidirectional Transformer encoder**로 구성
- BERT 모델 크기에 따라 $BERT_{BASE}$ 와 $BERT_{LARGE}$ 로 나눌 수 있음
  - $BERT_{BASE}$: L=12, H=768, A=12 (110M parameters)

    (GPT-1과 성능 비교를 위해 동일한 크기로 설계)
  - $BERT_{LARGE}$: L=24, H=1024, A=16 (340M parameters)

📌 L : encoder block 개수
D : hidden vector size
A : self-attention head 개수

2) Input/Output Representations

- 여러 downstream tasks에 유연하게 대응할 수 있도록 하나의 문장 또는 2개의 문장 (ex. 질문 - 답변) 도 입력 가능
- CLS 토큰의 final hidden vector는 **해당 sequence의 통합적인 representation으로 분류 작업에 사용됨**
- [SEP] 토큰을 이용해 2개의 문장을 구분

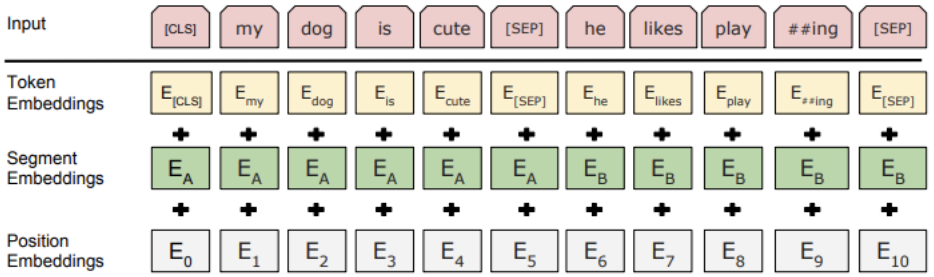


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

INPUT EMBEDDING = **Token Embedding + Segment Embedding + Position Embedding**

- **Token Embedding** : 모델이 이해할 수 있는 고차원 벡터로 변환하는 역할
  - WordPiece Embedding with a 30,000 token vocabulary
- **Segment Embedding :** E_A, E_B
  - 모든 토큰에 문장 A인지 B인지 표시하는 학습된 임베딩을 추가하여 두 개의 문장을 구분
  - 하나의 문장만 들어간다면, E_A 만을 사
- **Position Embedding :** $E_0, E_1, ..., E_{10}$
  - 학습을 통해 위치 정보 표현
  - 512 토큰 길이에 맞게 학습된 임베딩 값을 사용

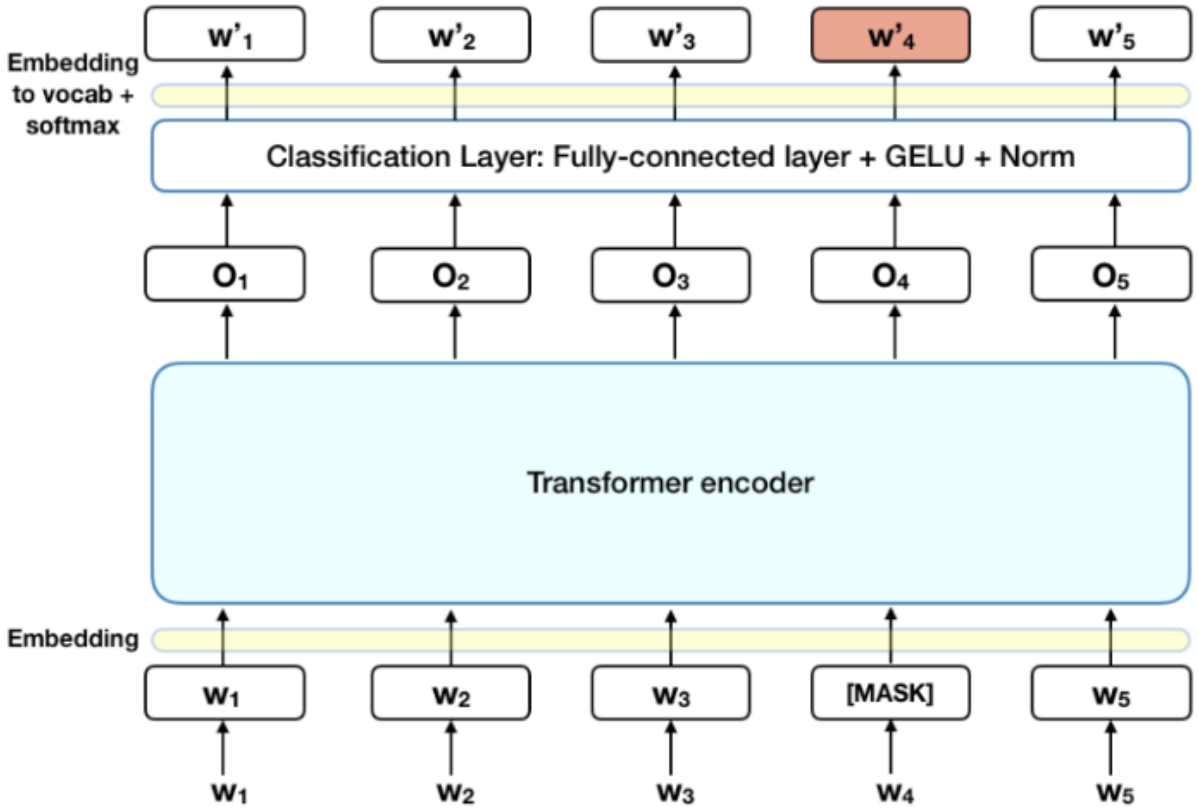
📌 논문에서 정의하는 문장과 토큰

- 문장 (sentence) : 하나의 연속적인 단어의 나열 형태로, 완벽한 문장 형태가 아니어도 됨
- 토큰 (token) : BERT의 입력 token sequence, 1개 또는 2개의 문장을 일컫음

3) Pre-training BERT

BERT의 pre-train을 위해 2가지 Unsupervised tasks 이용

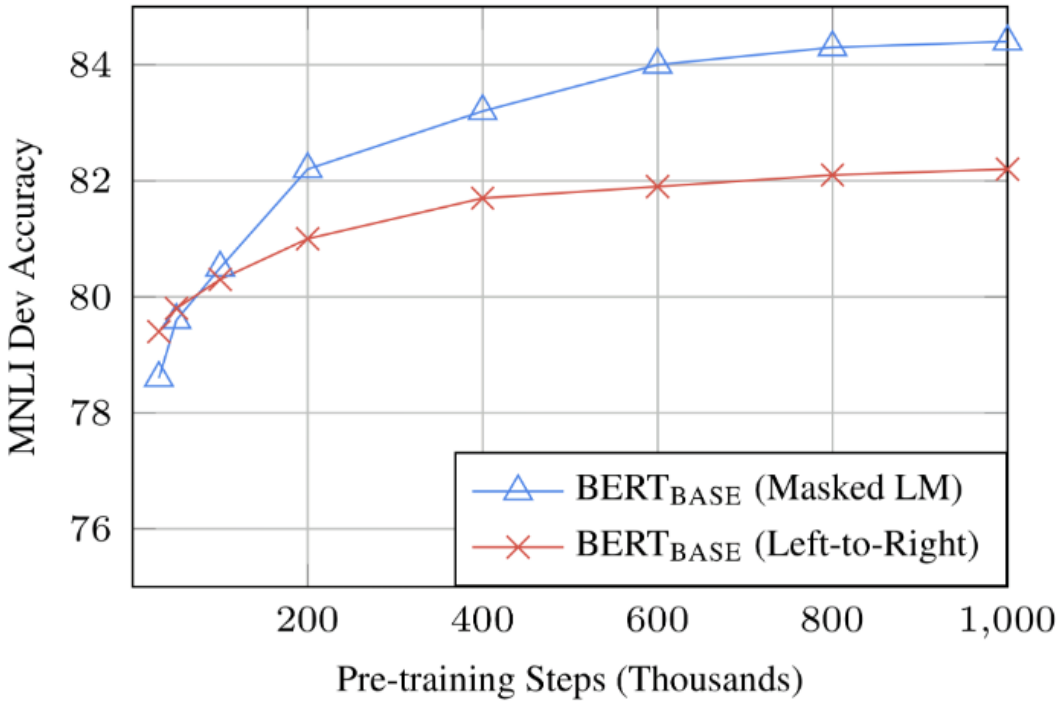
3.1) Task #1: Masked LM



BERT MLM

- 양방향의 문맥을 제대로 학습하기 위해 전체 단어 중 **15%의 단어를 [MASK] token으로 치환**하여 [MASK] token만 예측하면 됨 (↔ denoising auto-encoders)
- [MASK] token 은 **Pre-training시에만 사용**되고, Fine-tuning에는 사용되지 않음
- **문제점**
  - Masking을 사용하는 Pre-training과 그렇지 않은 Fine-tuning phase 사이에 불일치가 발생
- **해결책**

  `my dog is hairy` 전체 토큰 중 15%에 해당하는 토큰 중에서
  - 80% : token을 [MASK]로 바꾼다. `my dog is [MASK]`
  - 10% : token을 random token으로 바꾼다. `my dog is apple`
  - 10% : token을 원래 단어로 놔둔다. `my dog is hairy`
- MLM으로 학습하면 15%의 단어만 맞추는 것으로 학습을 진행하기 때문에 수렴 속도가 LTR 보다 훨씬 느리지만, **out-perform 성능은 더 빠르게 나타남**



Ablation Studies - Effect of Number of Training Steps

3.2) Task #2: Next Sentence Prediction (NSP)

- QA나 NLI와 같이 **두 문장 사이의 관계를 이해**하는 것이 중요하나, 문장 단위의 언어 모델은 이러한 관계를 파악하는 것이 어려움
- 그래서 BERT에서는 두 개의 문장을 준 후에 이 문장이 이어지는 다음 문장인지 아닌지를 맞추는 binary classification task 수행
  - 50% : sentence A, B가 실제 next sentence
  - 50% : sentence A, B가 corpus에서 random으로 뽑힌(관계가 없는) 두 문장

$$\text{Input} = \texttt{[CLS] the man went to [MASK] store [SEP]}$$
$$\texttt{he bought a gallon [MASK] milk [SEP]}$$
$$\text{Label} = \texttt{IsNext}$$

$$\text{Input} = \texttt{[CLS] the man [MASK] to the store [SEP]}$$
$$\texttt{penguin [MASK] are flight ##less birds [SEP]}$$
$$\text{Label} = \texttt{NotNext}$$

4) Pre-training Procedure

- Data : 위키피디아, book corpus 데이터
- NSP를 위해 sentence를 뽑아서 Sentence A, B embedding 입력 (토큰의 길이 ≤ 512)
- masking 작업 후 mask token 예측

Fine-tuning BERT

- Task별 input과 output layer를 추가하고, end-to-end 방식으로 파라미터를 학습

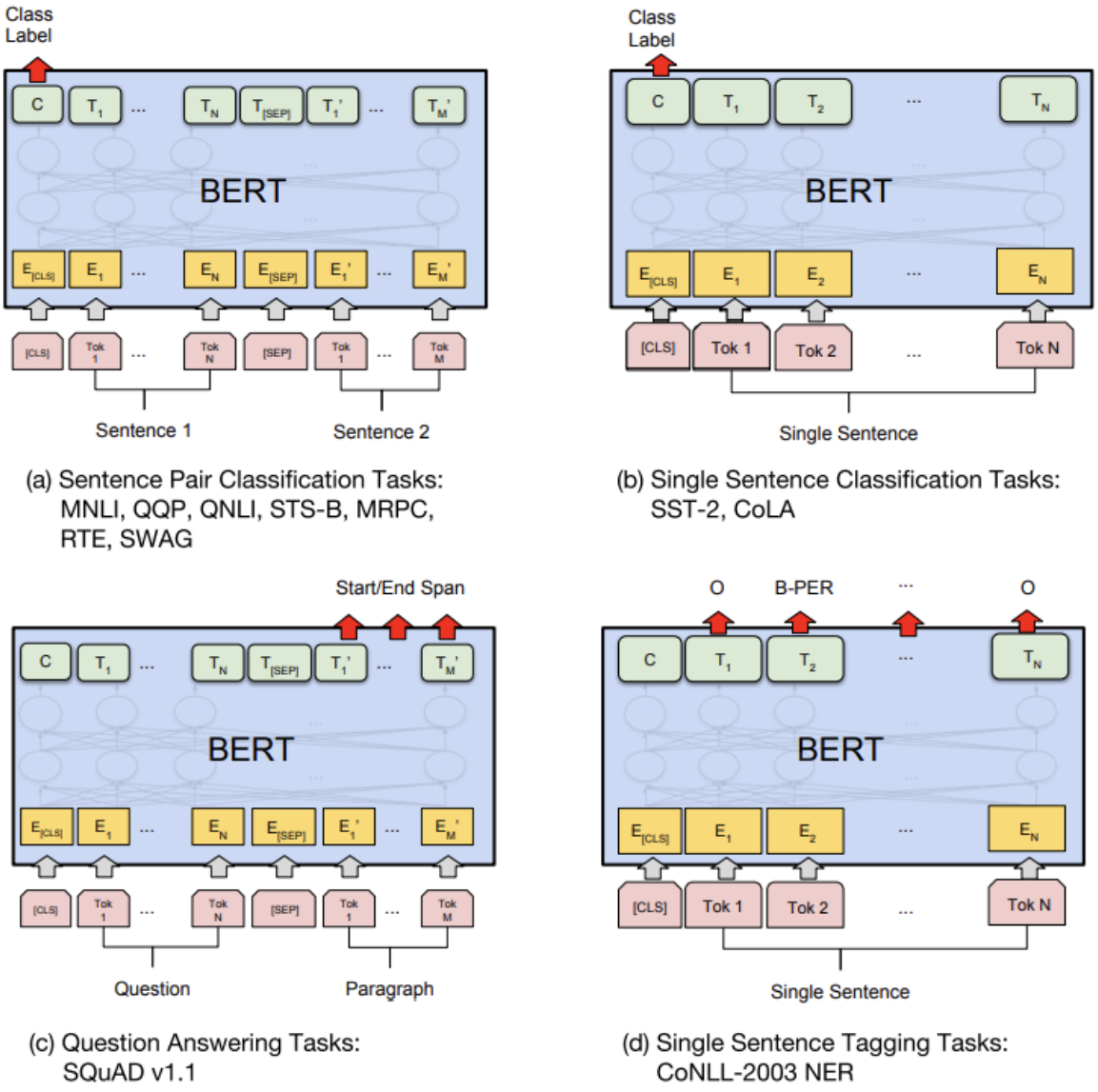


Figure 4: Illustrations of Fine-tuning BERT on Different Tasks.

Task별 Fine-tuning 방식

- **token-level task : (c), (d)**

- (c) : BERT output의 마지막 토큰들이 답변 문장의 시작 index와 끝 index를 출력하게 끔 학습
- (d) : single sentence에서 각 토큰이 어떤 class를 갖는지 **모두 classifier** 적용

- **sequence-level task : (a), (b)**
  - class K * [CLS] token의 벡터 차원 H 의 classification layer 생성

    $C \in R^H, W \in R^{K \times H}$
  - softmax를 통과하여 label probabilities 도출

    $P = softmax(CW^T)$

Experiments

- 총 11개의 NLP task에 대해 모두 SOTA 달성

** 자세한 실험 세팅 및 결과는 논문 참고

Ablation Studies

1) Effect of Pre-training Tasks

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

pre-training task ablation result

- Task 실험
  - No NSP : NSP task 제거
  - LTR & No NSP : MLM 대신 **Left-to-Right (LTR)** 사용 + NSP task 제거
- pre-training task를 **하나라도 제거하면 성능이 굉장히 떨어지는 것**을 확인할 수 있음
  - No NSP : NLI 계열의 task에서 성능이 상대적으로 많이 하락

    → **문장 간의 논리적인 구조 파악에 중요한 역할**을 하고 있음을 방증할 수 있음

2) Effect of Model Size

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

- 모델의 크기가 클수록 정확도 향상

3) Feature-based Approach with BERT

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	-

- 마지막 레이어에 Bi-LSTM을 부착시켜, 해당 레이어만 학습 시키는 방법론을 사용
- BERT는 **Feature-based Approach**에서도 효과적

📌 **Feature-based Approach의 장점**

- Transformer encoder는 모든 NLP task를 represent하지는 못하므로, 특정 NLP task를 수행할 수 있는 네트워를 부착할 수 있음
- 계산 효율성

Conclusion

- **deep bidirectional architecture 를 통해 동일한 사전 훈련 모델로 전분야의 NLP task에서 SOTA를 달성**

+개선점

- BERT는 GPT-1과 같이 매우 많은 파라미터를 가지고 있으며, computational cost가 높으며, MLM 방식으로 인해 수렴 속도가 LTR 방식보다 낮은 것을 확인할 수 있음
- BERT는 사전 훈련 시 대규모 데이터셋을 필요로 하기 때문에 개인이 쉽게 학습하기 어려움