

# BERT 놓은장치



25일 → 3.4m/s + 강의

26일 → 끝까지 다 + 75의

BERT: transformer 구조를 이용한 language representation 모델

## • Abstract

Bert: Bidirectional Encoder Representation from Transformer

→ original implementation은 단순

unlabeled data (wiki, book data 등)를 모델을 초기 학습시킨 후, NLP task를 위한 labeled data를 transfer learning으로 한다.

→ As a result,   
 pre-train BERT model은 initial output layer를 fine-tuning을 하게 된다.  
 그럼, 다른 NLP task를 위한 task-specific architecture modification으로 SOTA model 만들수 있다.

QnA, language inference (QA와 같은)

= pre-train한 BERT model을 NLP task의 fine-tuning으로 SOTA

→ task-specific 한 퍼센트 단위로 가능 X

## I. Introduction

- NLP tasks  
  └ Sentence-level 3 단계로 하는 task  
    └ token-level 3 단계로 하는 task      → pre-training 단계에서 여러 task를 더 잘 학습시킬 수 있다.

- pre-training 단계 방법

- ① unsupervised feature-based 방법 : deep learning의 feature를 이용하는 것
- ② unsupervised fine-tuning 방법 : input feature를 input 단계에서만 사용
- ③ transfer learning from supervised Data.

feature-based 방법 → AI 논문이나 인터넷, BERT가 무엇인가 알리시나요? (주제)

: task-specific한 architecture를 사용하는 것.

: pre-trained representation을 additional feature 넘는 방법, 즉 2nd network를 통한 사용  
Ex) ELMo

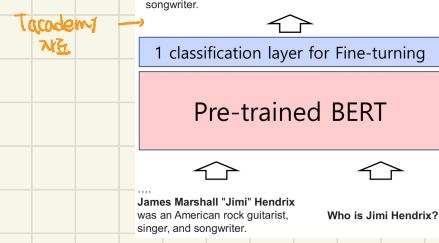
② fine-tuning 단계

: BERT가 네트워크 형태!

: BERT와 SOTA 같은 GPT-2와 같은 방법 사용

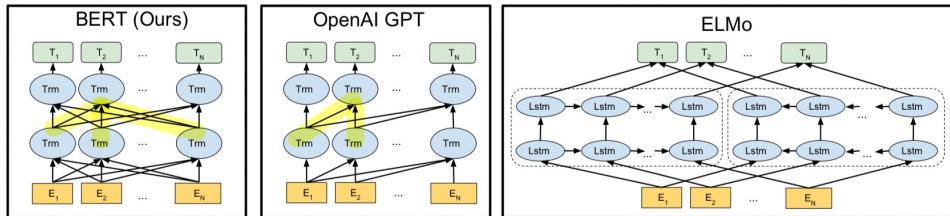
: task-specific한 parameter를 추출하는 fine-tuning

Ex) OpenAI GPT



→ 예전 GPT, ELMo  $\Rightarrow$  pre-training 시에 동일한 objective function으로 학습을 수행.

But, BERT  $\Rightarrow$  pre-trained Language Representation을 학습  $\rightarrow$  매우 효율적



• BERT의 핵심 3가지

- 1) Unidirectional or only bidirectional
- 2) masked language model or next sentence prediction task를 이용한 pretraining
- 3) task dataset or pre-training : task-specific dataset from fine-tuning

BERT의 pretraining의 내용은 2가지

- ① masked language model
- ② next sentence prediction task

\* 가정의 방식은

: 예전 n개의 단어를 가지고 두번 단어를 예측하는 Model을 이루는 것 (n-gram)

→ Unidirectional II

이를 극복하기 위해 ELMo처럼 Bi-LSTM으로 양방향성을 가지게 하지만, 근본적 Shallow한 양방향성.

= 단방향 context 양방향

### ① masked language model (MLM)

: input array에 무작위로特殊 token을 mask 한다

그리고 transformer encoder가 주변 단어 context를 보고 MASK된 단어를 예측하는 문제.

(GPT는 transformer 구조 사용하지만, 앞 단어들만 보고 뒷단어를 예측하는 transformer Decoder 구조 사용)

: BERT는 Input sequence mask된 token은 sequence Transformer encoder로 무작위 token을 예측

$\Rightarrow$  深深 ! deep bidirectional이다. 학습도

→ they're unlabeled corpus의 language model training 후, 이를 basis로 두개의 동일task를 처리하는 network를 별도로 놓는 방식.

GPT  $\rightarrow$  BERT가 일반화하고 미화화하는 model

auto-regressive task

$\rightarrow$  Unidirectional 이면 = 앞 token은 이전 token의 주제

## ② next sentence prediction

- 두 문장을 pre-training NLP 퀴즈에서 넣었을 때 두 문장이 다음으로 올라갈 확률이 아까운 이유는
- pre-training AI  $\Rightarrow$  둘다를 이해하는 두 문장 : 간단하게 추출된 두 문장 = 50 : 50 으로  
불가능, BERT가 이해해 수준!

## 2. Related Work

- ELMo, OpenAI GPT  $\rightarrow$  예측

## 3. BERT

- BERT architecture는 "Attention is all you need"의 시대인 transformer는 NLP 학자들이  
pre-training과 fine-tuning NLP architecture를 통해 대중적인 transfer learning을  
보여주는 계기.

### ※ NLP, transfer learning (cont.)

→ 기존의 만들기전 모델을 사용하여 새로운 언어를 만들면서 차이를 바꿔내고, 이를 더 높이는 방법

→ 예상

: 대상언어 problem을 위한 학습된 모델을 사용해서 문제 해결할 수 O

적용) fine-tuning과 transfer learning의 차이점?

→ 같은 뜻! 

### 3.1 Model Architecture

- BERT는 transformer 중 encoder 부분만 사용

- model의 크기와 따라 가중치 model 사용

**BERT-base** : L=12, H=768, A=12, Total Parameter = 110M

**BERT-large** : L=24, H=1024, A=16, Total Parameter = 340M

↳ 더 큰 데이터 SOTA는 가능!

L : transformer block의 layer 수  $\rightarrow$  영어나 중국어

H : hidden size

A : self-attention head 수 = 챕터의 개수

OpenAI Efficien hyper parameter가 있고

: 95% model의 hyper parameter가 동일하다고, pre-training  
concept이 아닙니다. 성능↑한데도 성능↑한데도 성능↑한데도 성능↑한데도

### 3.2 Input Representation

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

- bertel input : 371개 embedding 간의 합으로 이루어져 있음.
- wordPiece embedding 사용 : BERT english  $\rightarrow$  30000 개의 token 사용.

\* 그럼, wordPiece embedding 어떤가?

: WordPiece tokenizing

He likes playing  $\rightarrow$  He likes #ing,

→ 입력문장을 tokenizing하고, 1 token을 1 token sequence로 만들어 계산에 사용.

→ BPE (Byte Pair Encoding) 알고리즘 이용

: BPE로 기본하여 단어를 의미하는 subword로 절친 tokenizing

- PC sentence의 첫번째 token : [CLS]  $\rightarrow$  transformer 초기화를 가지고 나중 **token sequence의 classification 의도**를 가지게 된다.  
 $\rightarrow$  예전에 전방향 classifier를 넣으면 단일문장 / 연속된 문장의 classification을 동시에 하는 듯이 된다.  
**\* classification task가 아니라 [CLS] token은 무시.**

- Sentence pair는 흔히 **Single sentence** 일정수

1st sentence  $\rightarrow$  2nd sentence2 이루어져 있을 수도 O

(ex) QA task의 경우 [Question, Paragraph] или Paragraph의 대답문 )

그럼!, 두 문장을 각각의 토큰 ① [Seq] token 사용

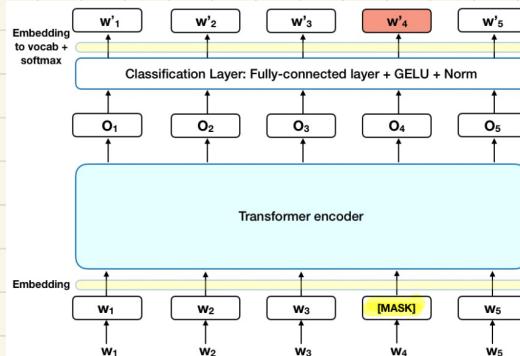
② Segment Embedding 사용

: 이미 문장에는 Sentence A embedding, 두 번째 문장에는 Sentence B embedding을 적용함. ↗ 문장이 1개면 예상 사용.

### 3.3 Pre-training task

: unsupervised prediction task pre-training

#### 3.3.1 Task #1. Masked LM



<https://minio-park7.github.io/nlp/2018/12/12/bert-fbclid=lwAR3S-8lLWEVG6FGUVxoYdwQyAzG0GpOUzVesFBd0ARFg4eFXqCyGLznu7w>

: 입력 단어 중의 일부를 [MASK] token으로 바꿔놓음

15%

: Plain text  $\Rightarrow$  tokenization 하는 방법  $\Rightarrow$  WordPiece

: 문장 전체를 predict X  $\rightarrow$  [MASK] token만 predict하는 pre-training task 수행.

$\rightarrow$  [MASK] token은 pre-training 때만 사용!, fine-tuning 때는  $\text{15\%} \times$

$\rightarrow$  token을 모으면서, 문맥을 파악하는 능력을 강화시킨다

\* 15%의 [MASK] Token은 인데일리, 예상 단어와 함께 훈련!

80%  $\rightarrow$  token  $\Rightarrow$  [MASK] 를 바꿈

15%  $\rightarrow$  .. random word3 바꿈

10%  $\rightarrow$  .. 유사한 단어 그대로 두자  $\rightarrow$  같은 단어에 대한 표본을 bias하게 유해

Pre-training 때 transformer encoder의 학습에서는

어떤 단어를 predict하는지 / 어떤 단어를 random word3 바꿨는지 알 수 X

$\Rightarrow$  transformer encoder는 그동안 token의 distributional contextual representation을 유지하도록 한다.

### 3.3.2 Task #2 : Next Sentence prediction

- 이전하는 이유

• NLP task 다 QA / NLI (Natural Language Inference)는 두 문장 사이의 관계를 이해하는 것이 중요!

• corpus에서 두 문장을 뽑아서 이것이 원래의 corpus에서 연관된 문장인지 맞는 **binarized next sentence prediction task** 수행.

- ↳ 50% : sentence A, B가 같은 next sentence  
↳ 50% : sentence A, B가 corpus에서 random으로 뽑은 (맞지 않는) 두 문장.  
↳ ex).  
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

### 3.4 Pre-training Procedure

→ 기본적인 절차는 LM처럼 수행하는 것과 같음

→ BERT\_English 경우 : Book corpus + Wikipedia

↳ text passage만 적용하여 사용

long contiguous sequences가 학습에 좋음.

- Input pre-processing

• NSP를 위해 sentence를 뽑아서 embedding A, B를 해줌 → 이 두 token이 험자연 길이는  
(50% → 같자, 50% → random) ↳ Segment Embedding ↳ 512개 (OOM 때문)  
• 오크, masking 추가!

- pre-training의 Hyper parameter

- batch size : 256 sequences (256 sequences \* 512 tokens = 128,000 tokens/batch) for 1,000,000 steps  
-> 3.3 billion word corpus의 40 epochs
- Adam optimizer, learning rate : 1e-4,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ , L2 weight decay of 0.01, learning rate warmup over the first 10,000 steps, linear decay of the learning rate
- Dropout prob: 0.1 for all layers
- using gelu activation Hendrycks and Gimpel, 2016
- BERT\_base - 4 TPUs, BERT\_large - 16 TPUs를 사용하여 4일동안 학습

### 3.5 Fine-tuning procedure.

→ sequence-level classification on 토큰 단위 fine-tuning 알고리즘 straightforward.

: [CLS] 토큰의 transformer output을 기준으로 한다.

(input sequence의 토큰 단위 시각화 representation을 기준으로 한다.)

: [CLS] 토큰의 vector → 허위를 가정

CER<sup>H</sup>

: 예상, classifier 를  $\text{softmax}(W)$ 의 output layer로 설정한다.  
 $= W$

$W \in \mathbb{R}^{K \times H}$

: label probabilities  $\in$  standard softmax  $\in$  정답.

P

$$P = \text{softmax}(CW^T)$$

→ W matrix는 BERT의 BCE parameter가 되어 fine-tuning 된다.

→ Span-level / token-level prediction tasks on 토큰 단위, 예상 fine-tuning 단계를 직접 수행함! → ④

#### - fine-tuning의 parameter

: batch size, learning rate, training epochs 등 훈련에涉及到한 hyper-parameter들

? : optimal hyperparameter는 task별로 다르다, 예상은 각각 다르다!

- fine-tuning의 dataset은 선택하는 hyperparameter에 따라 훈련과 validation 단계를 모두 포함

- 디자인된 dataset  $\Rightarrow$  여러 hyperparameter 설정을 exhaustive search하는 경우

↳ 예상치 않음

### 3.6 Comparison of BERT and OpenAI GPT

⑦: 예상되는 차이점

#### OpenAI GPT

Book corpus만을 이용한 pre-training on NLP

[SEP], [CLS] 토큰을 fine-tuning 단위로 추가하여 사용

fine-tuning은 향상화한 learning rate를 사용

#### BERT

Book corpus + Wikipedia

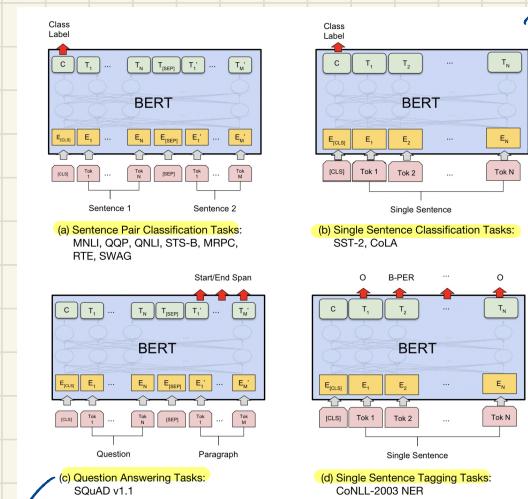
pre-training 단위 사용

(NLP task가 전반적 인보기)

task-specific 단위

## 4. Experiments

: 17개 NLP task을 위한 BERT fine-tuning에 대한 실험 결과.



(c) QA task

: Question에 해당하는 paragraph의 substring을 예상하는 것

=> [SEP] 토큰으로 토큰간 Start/End Span을 예측하는 task

(d) Single Sentence Tagging Task

: NER, 명사수명 등이 single sentence를 각 토큰의 예상 class로 예상하는 classifier

### 4.1. GLUE

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Average
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
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	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
BERT <sub>BASE</sub>	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
BERT <sub>LARGE</sub>	86.7/85.9	72.1	91.1	94.9	60.5	86.5	89.3	70.1	81.9

\* BERT는 어떤 투자 fine-tuning 하는지를 예상할 수 있는지?

## 4.2 SQuAD

: GLUE dataset을 BERT의 fine-tuning 데이터로 쓰는 퀴즈!

↳ task = sequence classification

SQuAD의 task => 질문과 대답문이 같은 substring인지를 찾는 task

: 질문을 A embedding, 대답을 B embedding으로 한다

↓

→ 어떤 문장의哪一部分 substring이 대답문을 찾는 task로 문제를 치운다

⇒ Start vector  $S \in \mathbb{R}^H$  와 end vector  $E \in \mathbb{R}^H$  를 fine-tuning 때 학습하여, 대답의 각 token을 dot product 해서 substring을 찾아낸다

## 4.3 NER

: Token tagging task는 Intent나 slot => CoNLL 2003 NER task를 fine-tuning 해용

→ 각 단위는 Person, Organization, .. 를 tagging 하는 것을

! 각 token에 classifier를 넣으면 예상 class로 예상하는 확률

→ 각 prediction은 global prediction이 영향을 끼친다 X

## 4.4 SWAG

: 이 dataset은 grounded common-sense inference를 추구하기 위한 dataset.

→ 알맞은 주제로, 높은 확률로 어떤 주제는 어떤 주제를 이어갈지 예상하는 task

: 이 dataset을 BERT의 fine-tuning 퀴즈

⇒ 각각의 Input sequence를  $\rightarrow$  sentence A (given sentence) or sentence B (possible continuation) 을  
concat한 것을

⇒ 각 task는 task-specific 퀴즈  $V \in \mathbb{R}^H$  를 학습시켜, Input sentence의 값을 합성한  $C_i \in \mathbb{R}^H$  를 dot product하여 softmax 퀴즈

## 5. Ablation Studies

### 5.1 Effect of Pre-training Tasks

: BERT-base의 동일한 hyper parameter로 훈련 → ablation한 두 가지 다른 model로 성능비교

[No NSP] : MLM을 사용 but NSP(다음 문장 예측)을 없앤 모델

[LTR & No NSP] : MLM 대신 LTR(Left-to-right) 사용, NSP는 없앤 모델 → OpenAI GPT와 동일하게 21%의 training data 사용

Tasks	Dev Set				
	MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
BERT <sub>BASE</sub>	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 + BiLSTM	82.1	84.3	77.5	92.1	77.8
	82.1	84.1	75.7	91.6	84.9

pretraining task로

다른 task를 수행하는 능력이 부족한 경우

No NSP의 경우

: NLP 자체의 task들이 서로 다른 특성을 갖기 때문에 예상치 못함

⇒ NSP task가 예상치 못한 특성을 갖기 때문에 예상치 못함

### 5.2 Effect of Model Size

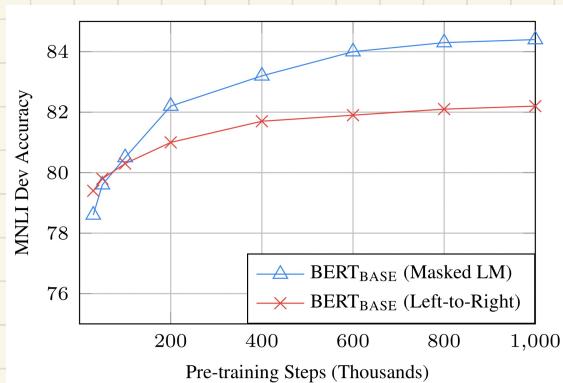
: 모델이 커지면서 성능이 높아짐

#L	Hyperparams			Dev Set Accuracy		
	#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

: BERT의 경우

downstream task를 수행하는 dataset의 규모가 커질수록, pre-training 단계에서 모델의 학습률을 증가시킬 수 있다.

### 5.3 Effect of Number of training steps



Q. fine-tuning阶段 높은 정확률을 얻으려면, pre-training阶段 같은 training step이 필요할까?

A. 예스, 0.5M step에 비해 1M step의 accuracy가 약 1.0% 상승함

Q. MLM과 학습하면 1%의 정확률을 얻을 수 있는 방향전환에 대해서, LTR보다 구현된다거나 하는 것인가

A. 본래는 LTR과 같이 fine-tuning이 out-perform되는 경우가 많았지만, 최근에는

### 5.4 Feature-based Approach with BERT

: Optimizer  $\Rightarrow$  BERT는 pre-training을 진행한 후  $\rightarrow$  down stream task를 학습하는데

기존의 classifier를 부착하는 BERT layer를 다시 학습하는 fine-tuning 방식 사용

하지만 BERT를 Elmo와 같은 feature based approach로 사용하는게 좋다.

- + Transformer Encoder는 NLP task를 represent할 수 X. 그에 NLP task를 수행할 수 있는 Network를 부착하여 쓸 수 O
- + Computational benefit 있을 수 O

Layers	Dev F1
Finetune All	96.4
First Layer (Embeddings)	91.0
Second-to-Last Hidden	95.6
Last Hidden	94.9
Sum Last Four Hidden	95.9
Concat Last Four Hidden	96.1
Sum All 12 Layers	95.5

↑ 0.3% 차이出す  $\Rightarrow$  BERT Feature-based Approach는 좋은 것인가?