BERT

: Pre-training of Deep Bidirectional Transformers for Language Understanding

발표자 한나경

Table of contents

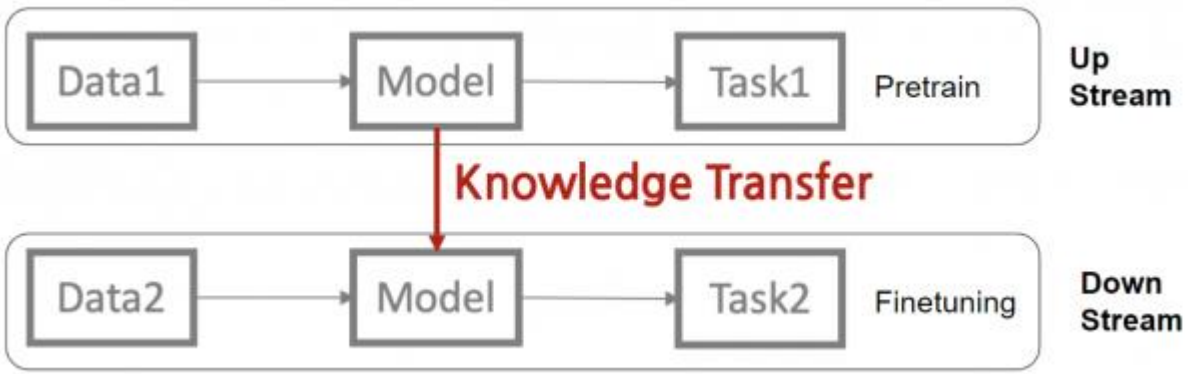
Transfer Learning (전이 학습)

개념 : 특정 데이터를 학습한 모델을 다른 데이터에 재사용

원리 : 인간이 사전에 학습한 배경지식을 활용하여 다른 Task를 수행하는 것과 같다.

BookCorpus, Wikipedia 데이터셋과 같은 대규모 말뭉치를 학습한 모델은 0부터 학습한 모델들보다 더 빠르게 수렴하고, 좋은 성능을 기록

-> 대규모 말뭉치를 사전 학습함으로써 다양한 언어학적 지식/문맥을 익혔기 때문



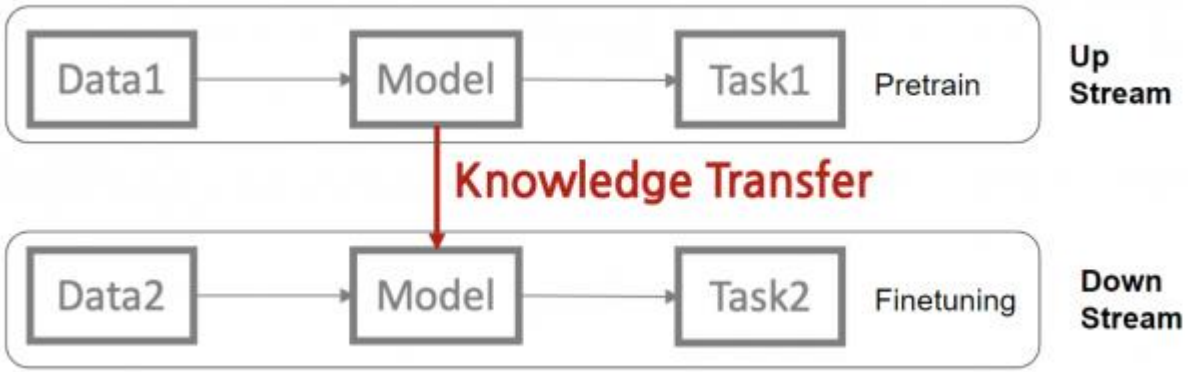
용어 정리

Language Model : 문장 혹은 단어가 등장할 확률을 반환하는 함수

Upstream Task / Pre-train : 다음 단어 맞히기, 빈칸 채우기 등 대규모 말뭉치로 언어 모델 학습

Downstream Task / Fine-tuning : 문서 분류, 개체 명 인식 등 구체적 과제 학습

Embedding / Representation : Pre-train 완료된 언어 모델의 일부 출력 값 혹은 그 자체



BERT = **B**idirectional **E**ncoder **R**epresentations from **T**ransformers

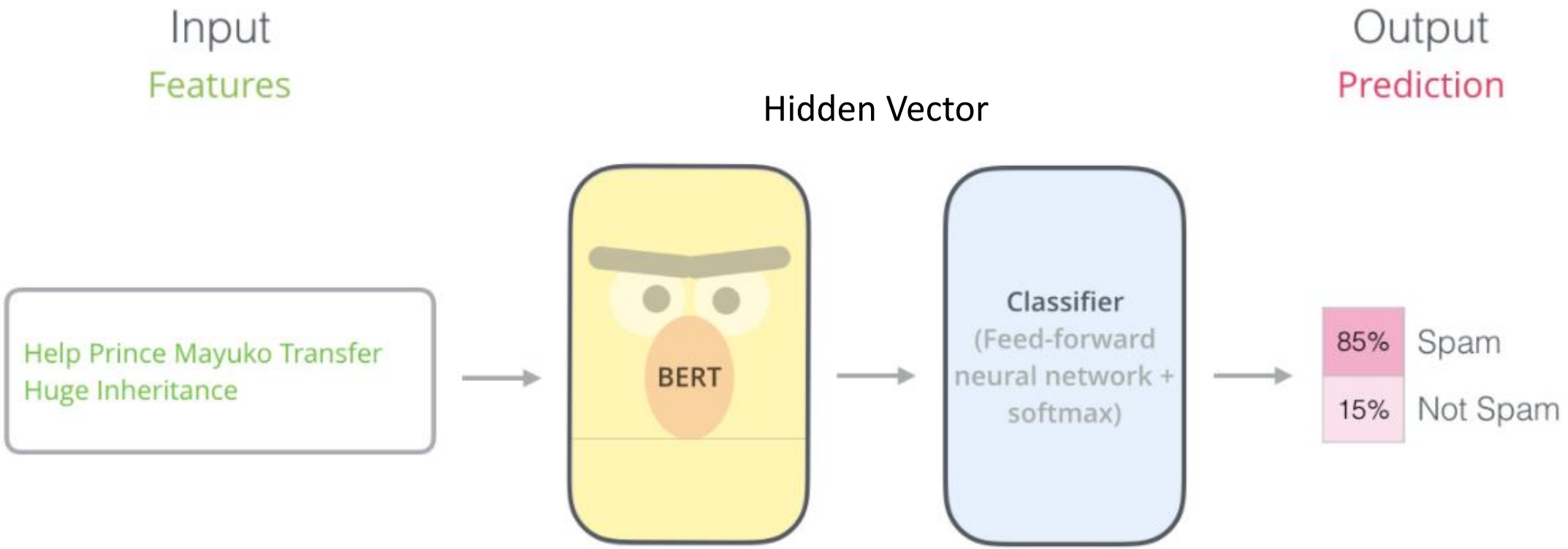
개념 : 대용량 Unlabeled data로 모델을 Pre-train한 뒤, 특정 Task를 가진 Labeled data로 Fine-tuning (Transfer learning)을 하는 모델

GPT는 Unidirectional (단방향성), ELMo는 Shallow Bidirectional (얕은 양방향성)
-> BERT는 모든 Layer에서 좌측/우측에 있는 모든 내용을 학습

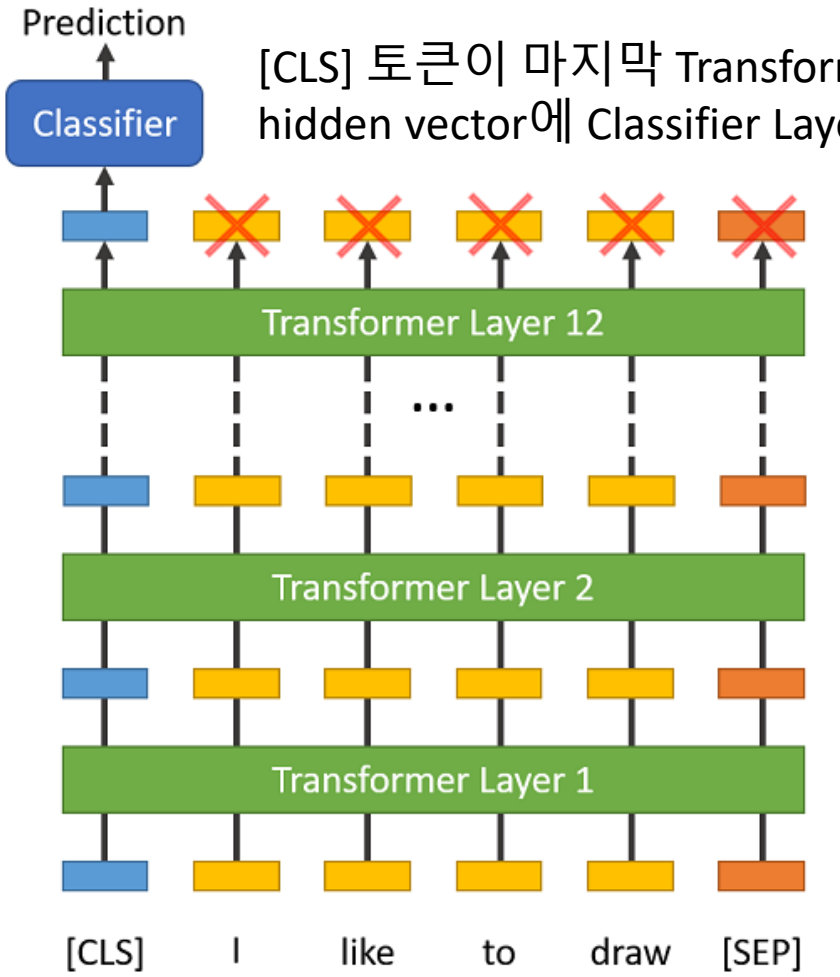
deep bidirectional : deep을 강조하여 기존의 모델과의 차별성 강조

하나의 Output layer만 추가하면 Question Answering, Language Inference와 같은 NLP의 다양한 주요 Task에서 SOTA를 달성할 수 있음
-> Pre-trained BERT 모델의 확장성이 넓고, 성능 역시 기존의 모델들을 뛰어넘는다는 것을 의미

BERT = **B**idirectional **E**ncoder **R**epresentations from **T**ransformers



BERT = **B**idirectional **E**ncoder **R**epresentations from **T**ransformers



[CLS] 토큰이 마지막 Transformer Layer를 통과하고 나온 hidden vector에 Classifier Layer를 부착

BERT를 활용하는 두 가지 방법

1. **Feature-based Approach :** 사전 학습된 모델 Feature를 특정 Task를 수행하는 곳에 Freeze한 채로 붙여서 사용하는 방식 / ELMo

2. **Fine-tuning Approach :** 대량의 데이터로 학습된 모델을 특정 Task에 맞는 Dataset으로 Fine-Tuning 시키는 것, Feature-based와 달리 Fine-Tuning 과정에서 Pre-trained Feature들 또한 갱신 / GPT

Language Modeling에서 Pre-training은 NLP Task 성능을 높이는데 크게 기여하였음

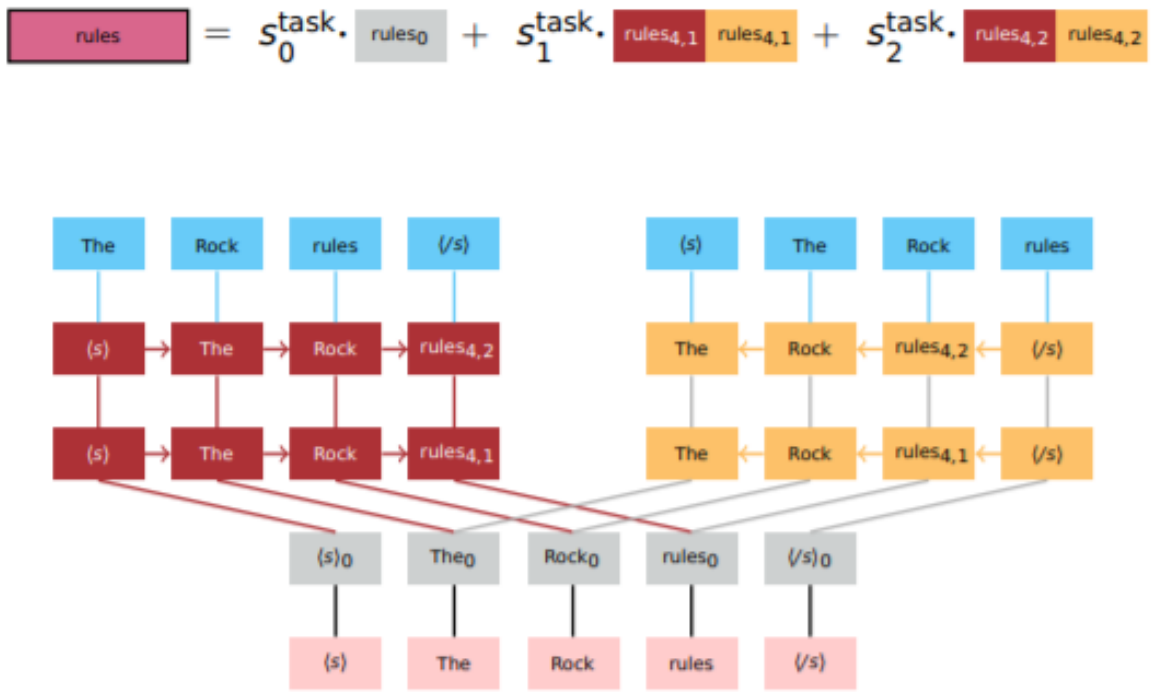
-> NLP Task는 크게 2가지로 나눌 수 있음

1. **문장 단위 Task :** 문장 간의 관계를 분석, 예측 / Natural Language Inference, Paraphrasing
2. **토큰 단위 Task :** 토큰 단위의 정밀한 Output을 반환 / Named Entity Recognition(NER), Question Answering

ELMo

: Left to Right 방식과 Right to Left 방식에서 추출된 Embedding을 Concat하여 사용하는

Shallow Bidirection



GPT

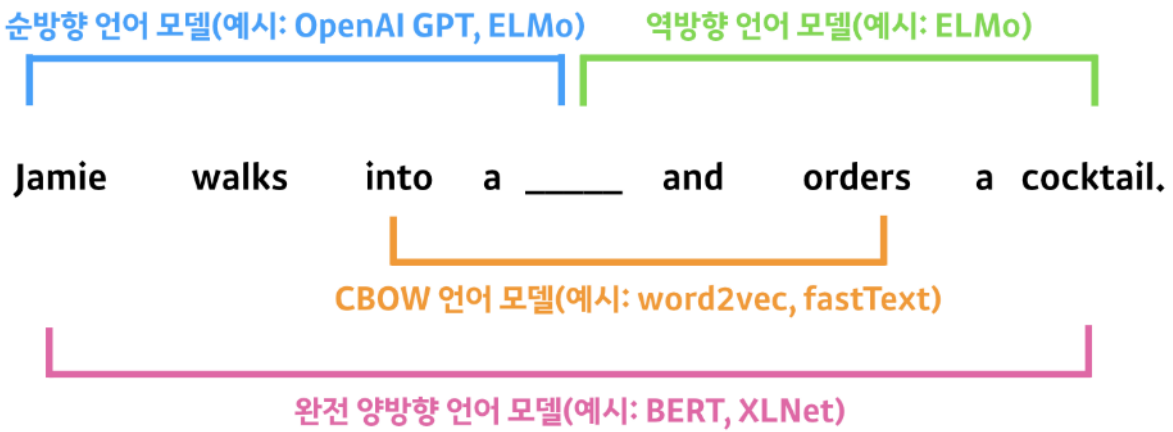
: Left to Right의 단방향성(Unidirection) 방식으로 학습 진행

단방향성 접근은 Pre-trained Representation의 성능을 크게 제한

-> 각각의 토큰이 자신의 위치 이전에 있는 토큰만을 접근할 수 있음을 의미

-> 문장 단위 Task에서는 최적의 성능 X, 토큰에 민감한 Task에서는 매우 높은 성능 향상O

이 논문에서는 양방향성을 고려한 Fine-tuning Approach 모델인 BERT를 소개한다.



BERT 프레임워크는 2가지 스텝으로 구성

1. Pre-training : Unlabeled 데이터를 통해 기본적인 문맥 이해를 위한 학습 진행
2. Fine-tuning : Pre-training 파라미터로 가중치가 초기화되어 있으며, 이를 목표하는 down-stream Task에 맞게 미세 조정하는 과정

BERT의 장점

: Pre-training 아키텍처와 실제 down-stream Task에 적용되는 아키텍처에 큰 차이가 없음

-> 사전 학습이 완료된 모델에 간단한 레이어만 추가하면 쉽게 적용 가능

B E R T - M o d e l A r c h i t e c t u r e

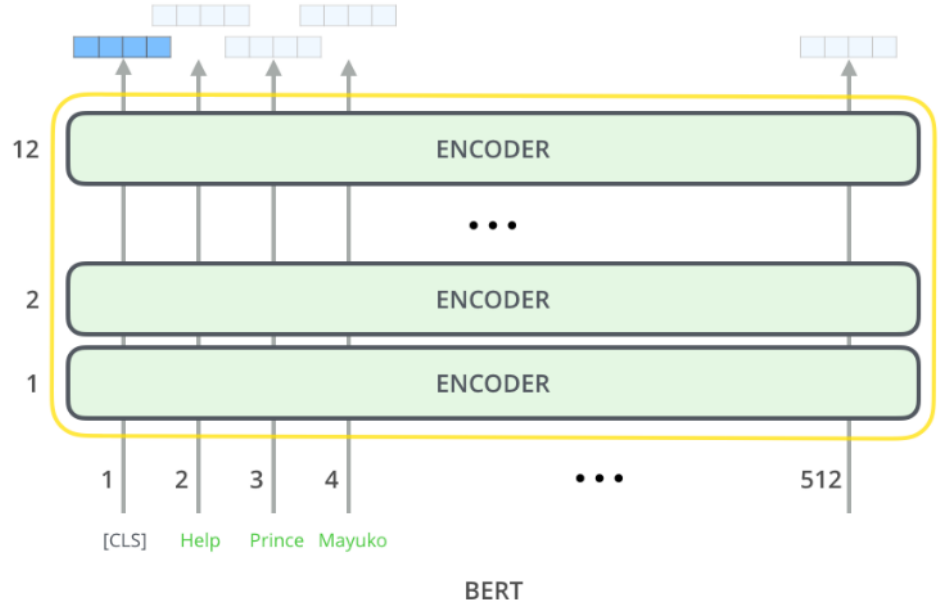
BERT의 아키텍처

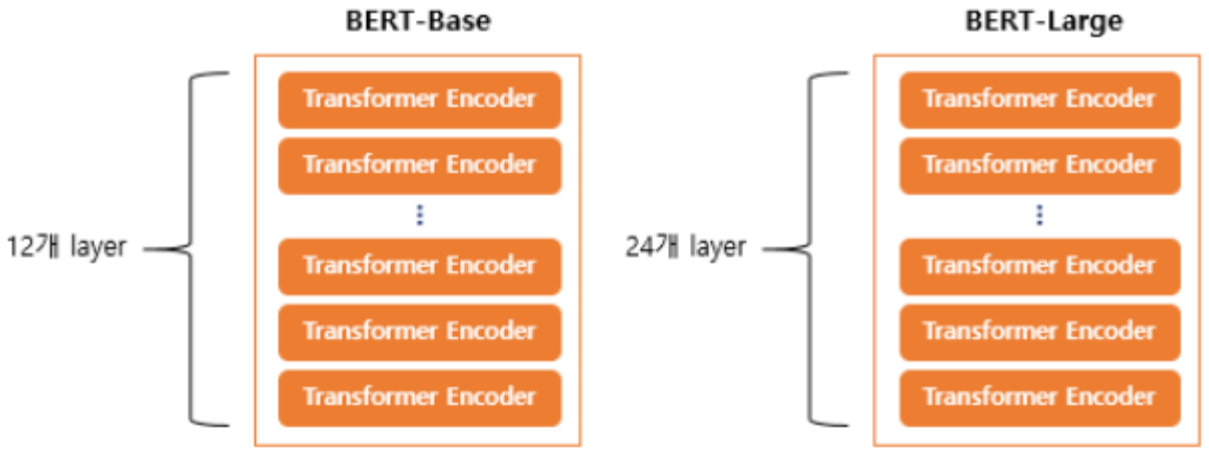
: Multi-layer bidirectional Transformer로 여러 개의 트렌스포머 레이어를 겹겹이 쌓아올림

L : Layers , H = the hidden size , A = self-attention heads

BERT (BASE) : L = 12 , H = 768 , A = 12 , Total parameters = 110M

BERT (LARGE) : L = 24 , H = 1024 , A = 16 , Total parameters = 340M





BERT-Input/Output Representations

BERT를 여러 Down-stream Task에 적용할 수 있도록 하기 위해, 단일 문장과 문장 쌍 모두를 하나의 인풋으로 받을 수 있도록 함

-> ‘Sentence’는 실제 언어학적인 문장의 의미보다는 연속적인 text들의 배열을 의미

-> ‘Input Sequence’는 입력으로 들어오는 토큰들의 집합으로, 1문장일 수도 2문장일 수도 있음

WordPiece Embedding

BERT는 Tokenizing을 할 때 BertTokenizer 방식을 사용

-> 토큰이 단어 집합에 존재한다면 분리하지 않음

-> 토큰이 단어 집합에 존재하지 않는 OOV (Out of Vocabulary) 어휘에 대해 서브 워드 단위로 분절하는 방식 사용

-> 분절된 서브워드들 중 단어 내부에 위치하는 것은 앞에 '##'이 붙음

```
from transformers import BertTokenizer

tokenizer = BertTokenizer.from_pretrained('bert-base-case')

tokenizer.tokenize("This isn't too surprising.")
# ['This', 'isn', "'", 't', 'too', 'surprising', '.']

tokenizer.tokenize("Encode me!")
# ['En', '##code', 'me', '!']

tokenizer.tokenize("Snuffleupagus?")
# ['S', '##nu', '##fle', '##up', '##agu', '##s', '?']

tokenizer.vocab_size
# 28996
```

[CLS]

- 분류 테스크 수행 시 문장의 앞에 [CLS]를 부착해야 함
- 쌍으로 구성된 문장들도 하나의 Sequence로 표현
- 모든 레이어를 거쳐 나온 [CLS] 토큰은 곧 문장 전체의 representative한 벡터로 간주

[SEP]

- 모든 문장의 끝마다 [SEP]를 부착해야 함
- 두 개의 문장쌍이 주어지는 Task를 해결하기 위한 Artifact이지만, 한 개의 문장을 input으로 할 때도 [SEP] 토큰을 이용해야 함

BERT - Input / Output Representations

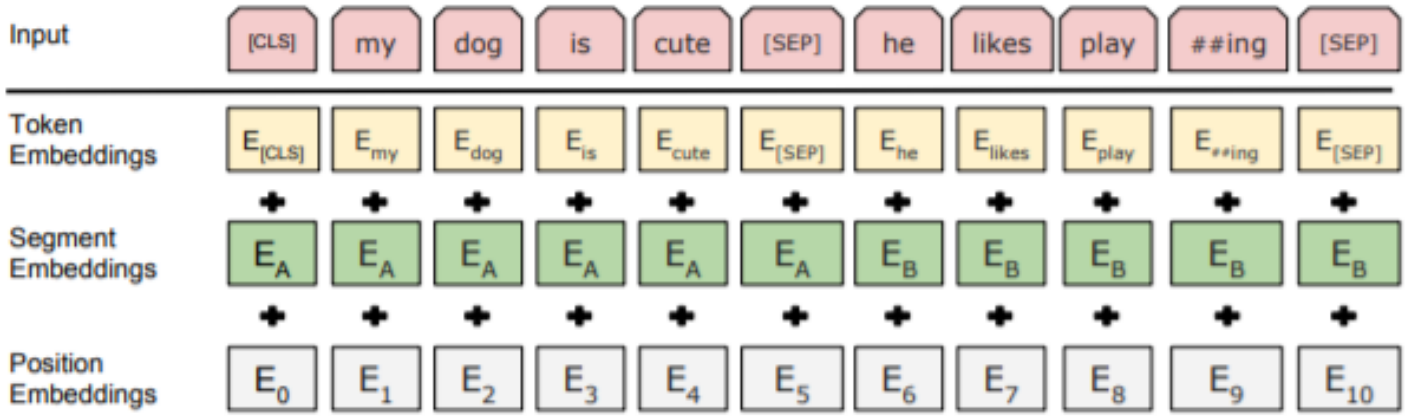
BERT에서는 총 3가지의 Embedding Vector를 합쳐서 input으로 사용

-> Input = Token Embeddings + Segment Embeddings + Position Embeddings

Token Embedding : Word embedding

Segment Embedding : 두 개의 문장을 구분하기 위한 embedding

Position Embedding : 위치 정보 표현을 위한 embedding



Pre-training BERT

2개의 비지도 학습을 통해 Pre-training 됨

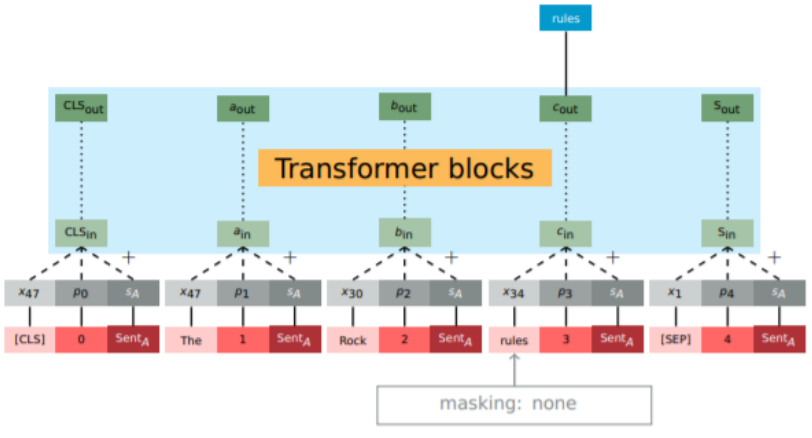
1. Masked LM
2. Next Sentence Prediction (NSP)

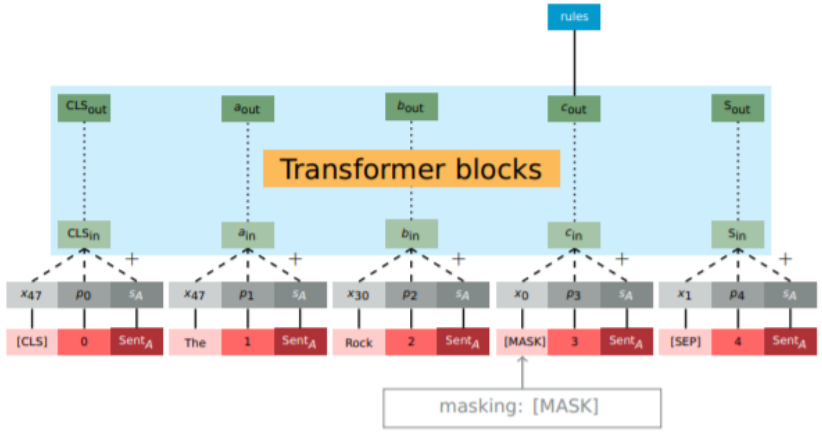
*비지도 학습 : 입력 데이터만을 제공하여 데이터의 패턴을 스스로 식별한다.

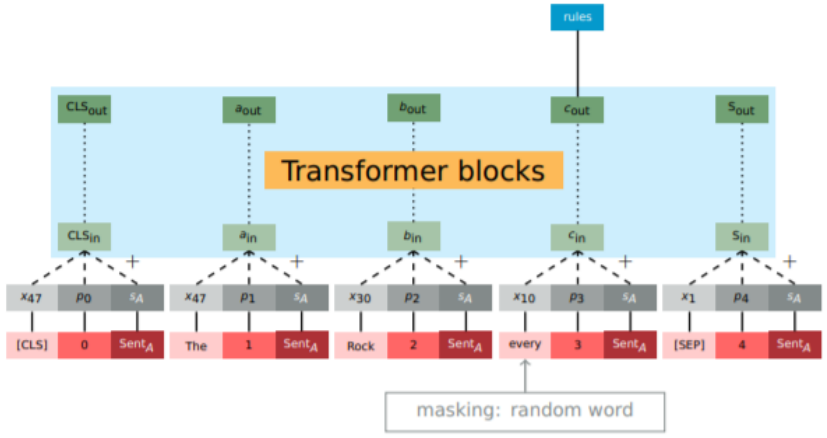
-> 정답이 없는 데이터를 사용하므로 지도 학습과 반대이다.

Pre-training BERT – Masked LM

- 이전 타임스텝들의 토큰이 주어졌을 때, 다음에 등장할 토큰이 무엇인지 맞히는 언어 모델링 Task과 기본적인 아이디어는 동일
- 하지만 기존 Task와 동일하다면 BERT의 장점인 Bidirectionality를 살릴 수 없음
- MLM을 이용 -> 전체 토큰 중 랜덤으로 15%를 마스크를 씌워, 해당 단어가 무엇인지를 맞히는 방식으로 언어 모델링 실시
- 마스킹은 80% [MASK], 10% [RANDOM], 10% 마스크를 씌우지 않음







Pre-training BERT – Next Sentence Prediction (NSP)

- 문장 간의 관계성을 파악하는 것이 중요한 NLP Down-stream Task가 많이 존재
- 따라서 문장 간의 논리적인 관계를 Pre-training 단계에서 학습시키는 것이 유용

Positive: Actual sentence sequences

- [CLS] the man went to [MASK] store [SEP]
- he bought a gallon [MASK] milk [SEP]
- Label: IsNext

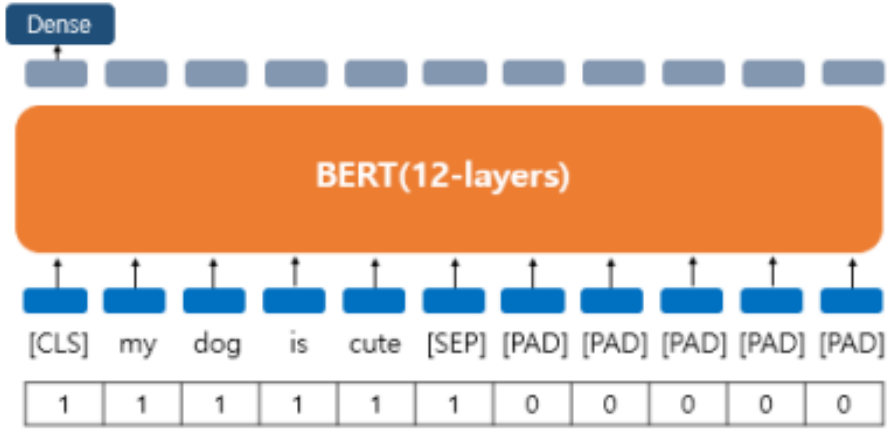
<- IsNext / NotNext의 비율은 50:50

Negative: Randomly chosen second sentence

- [CLS] the man went to [MASK] store [SEP]
- penguin [MASK] are flight ##less birds [SEP]
- Label: NotNext

Pre-training BERT – Pre-training Data

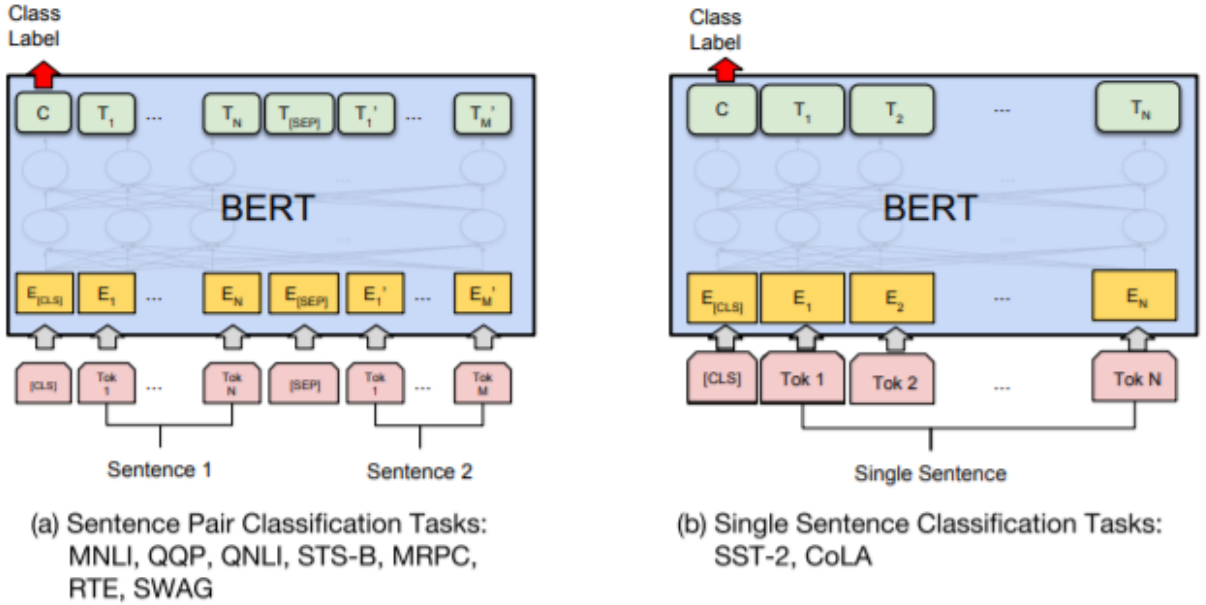
- 사용한 데이터 : BookCorpus, English Wikipedia
- Pre-training procedure는 일반적인 LM과 동일하되, IsNext/NotNext 문장이 50:50의 비율로 섞인 문장 쌍의 임베딩을 input으로 제시하고, 총 토큰의 수가 최대 512개를 넘지 않도록 한다.
- 이후 Masked LM을 위한 Masking을 실시



문장의 최대 길이인 512보다 작을 경우에는 padding을 할 수 있는 mask가 필요

-> 들어온 위치는 1, 안 들어온 위치는 0으로 만든다

Fine-tuning BERT



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

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

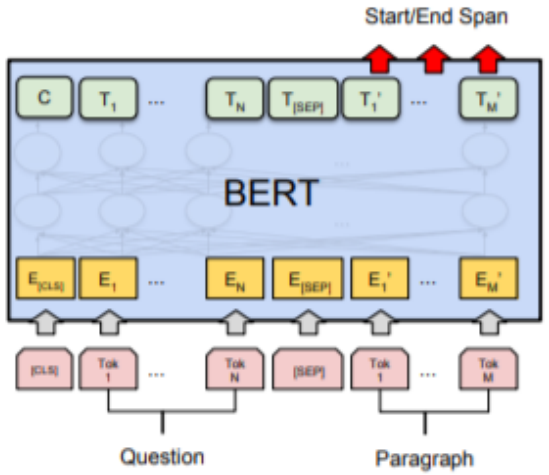
1. Sentence pairs in paraphrasing

: 2개의 문장을 입력 받고, 두 문장 간의 관계를 구하는 문제

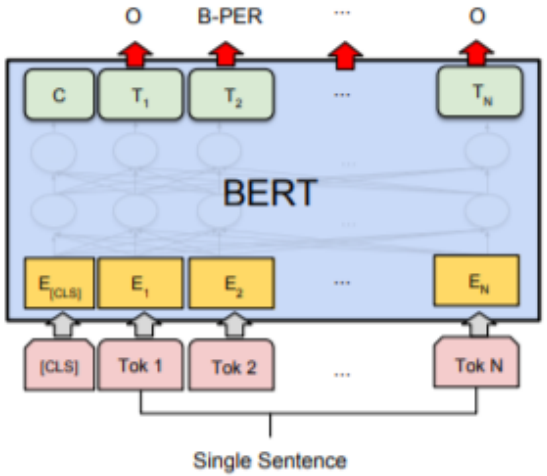
2. Hypothesis-Premise pairs in entailment

: 한 문장을 입력하고, 문장의 종류를 분류하는 문제

Fine-tuning BERT



(c) Question Answering Tasks:
SQuAD v1.1



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

3. Question-Passage pairs in question answering

: 질문(Q)과 문단(A)을 넣으면, 문단 내에 원하는 정답 위치의 시작과 끝을 구하는 문제

4. **Degenerate-None pair** in text classification or sequence tagging

: 입력 문장 토큰들의 개체 명이나 품사 등을 구하는 문제

Experiments

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	79.9	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
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

- 모든 Task에서 SOTA를 갱신
- BERT BASE보다 BERT LARGE가 더 성능이 잘 나옴

Part 5

Conclusions

- 본 논문에서 가장 주요한 것은 Bidirectional Pre-training의 효과를 거의 변화 없이 down-stream task에 적용할 수 있도록 아키텍처를 제시했다는 점이다.

감사합니다