BERT :

Pre-training of Deep Bidirectional Transformers for Language Understanding

2024.02.19

이은주

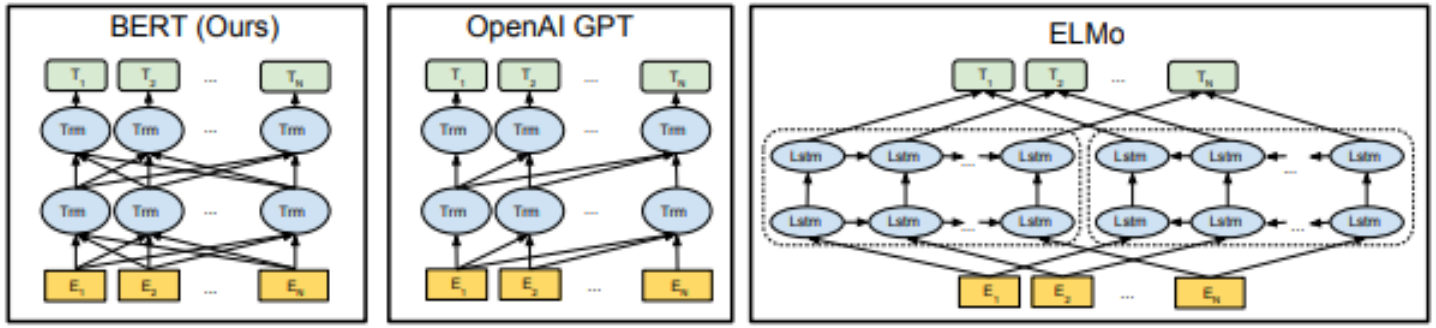
Intro

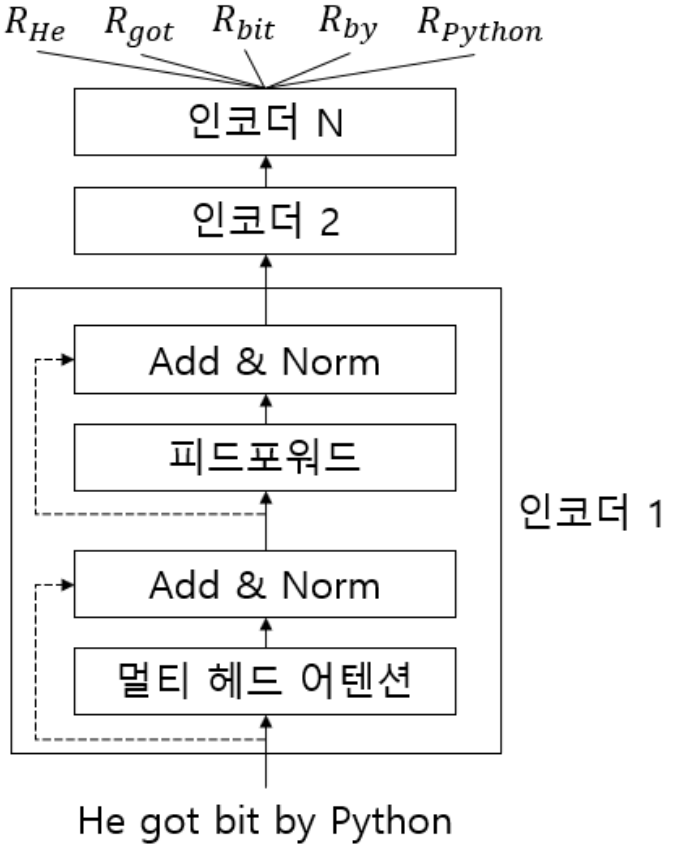
Google에서 발표한 자연어처리(NLP)논문

GPT-1 : Unsupervised Fine-tuning 방식.
left-to-right구조로 token이 이전 것에만 참여 가능.

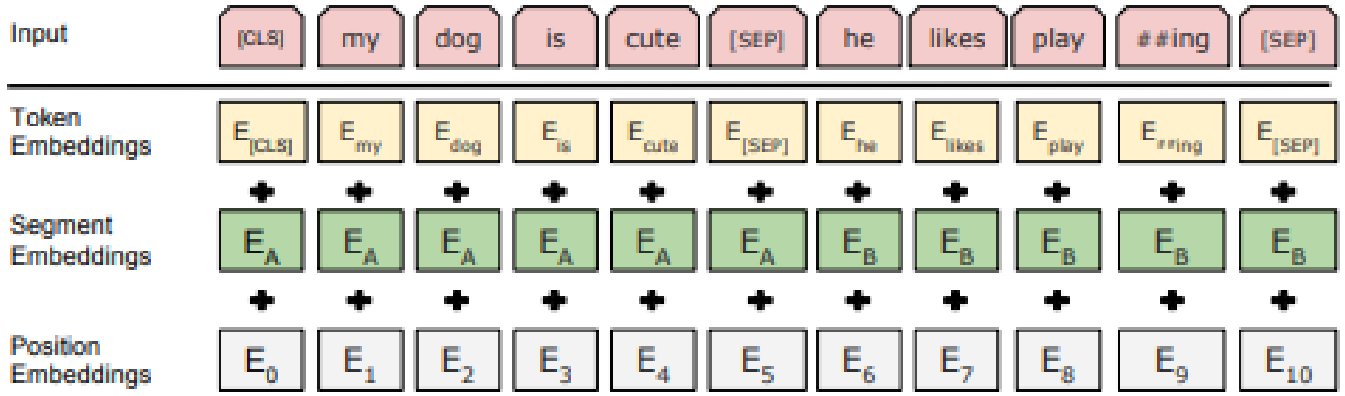
ELMO : Feature-based 방식. left-to-right, right-to-left구조

BERT : Transformers로 부터의 양방향 Encoder 표현





BERT Input Representation



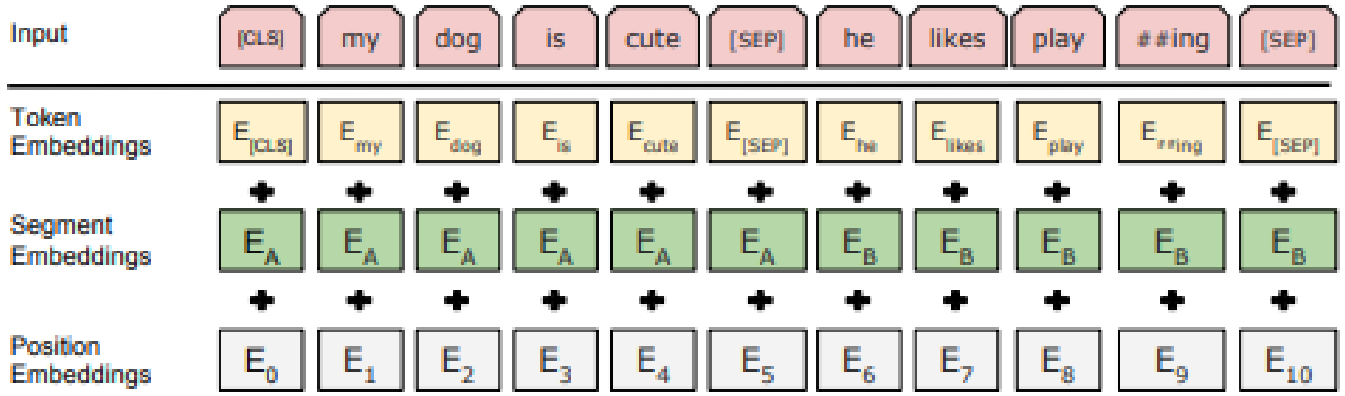
1) Token Embeddings : word piece 임베딩 방식

1.
```
A : Paris is a beautiful city
B : I love paris
tokens = [Paris, is , a, beautiful, city, I, love, Paris]
```

2.
```
token = [[CLS], Paris, is , a, beautiful, city, I, love, Paris]
```

3.
```
token = [[CLS], Paris, is , a, beautiful, city, [SEP], I, love, Paris, [SEP]]
```

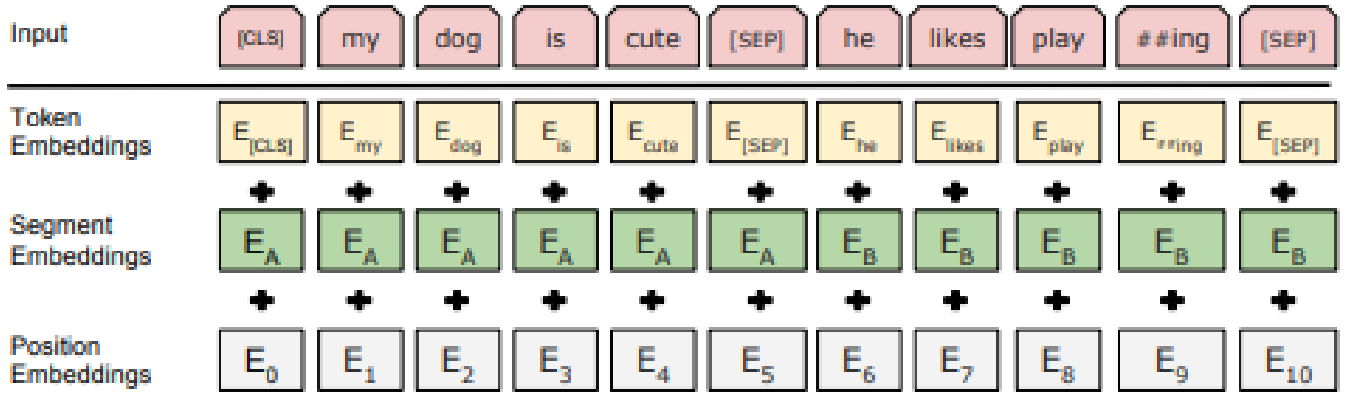
BERT Input Representation



2) Segment Embedding : 주어진 두 문장 구분할 때 사용

입력	[CLS]	Paris	is	a	beautiful	city	[SEP]	I	love	Paris	[SEP]
세그먼트 임베딩	E_A	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B

BERT Input Representation



3) Position Embedding : 토큰의 순서를 인코딩

BERT는 Transformer의 Encoder를 사용. 이때 Self-Attention을 사용하므로 입력 위치에 대해 고려하지 못하므로 Position Embedding사용

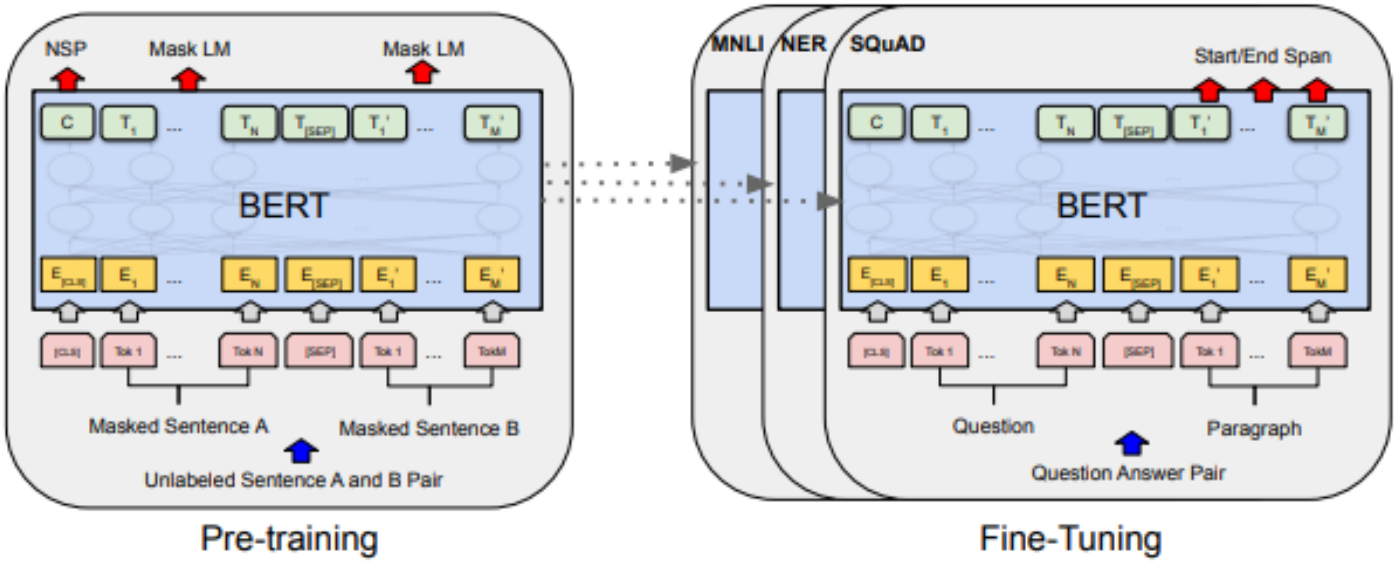
위치 임베딩	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}

BERT의 Pre-training과 Fine-Tuning

Pre-training동안에는, 라벨링이 되지 않은 데이터셋으로 pre-training task들을 수행

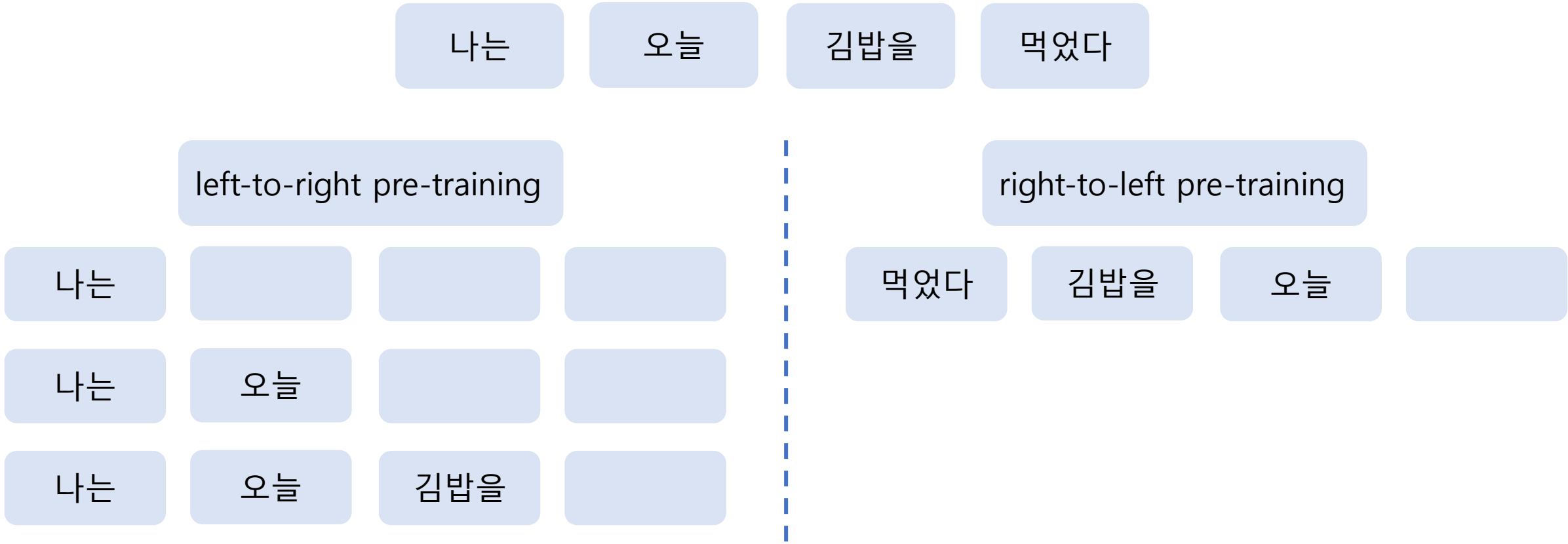
Fine tuning에서는, pre-training된 parameter와 라벨링이 된 데이터로 fine-tuning task를 수행

BERT는 fine-tuning단계에서 pre-training의 모델 구조와 parameter를 전부 활용하고, 약간의 레이어 추가만으로 fine tuning task를 수행



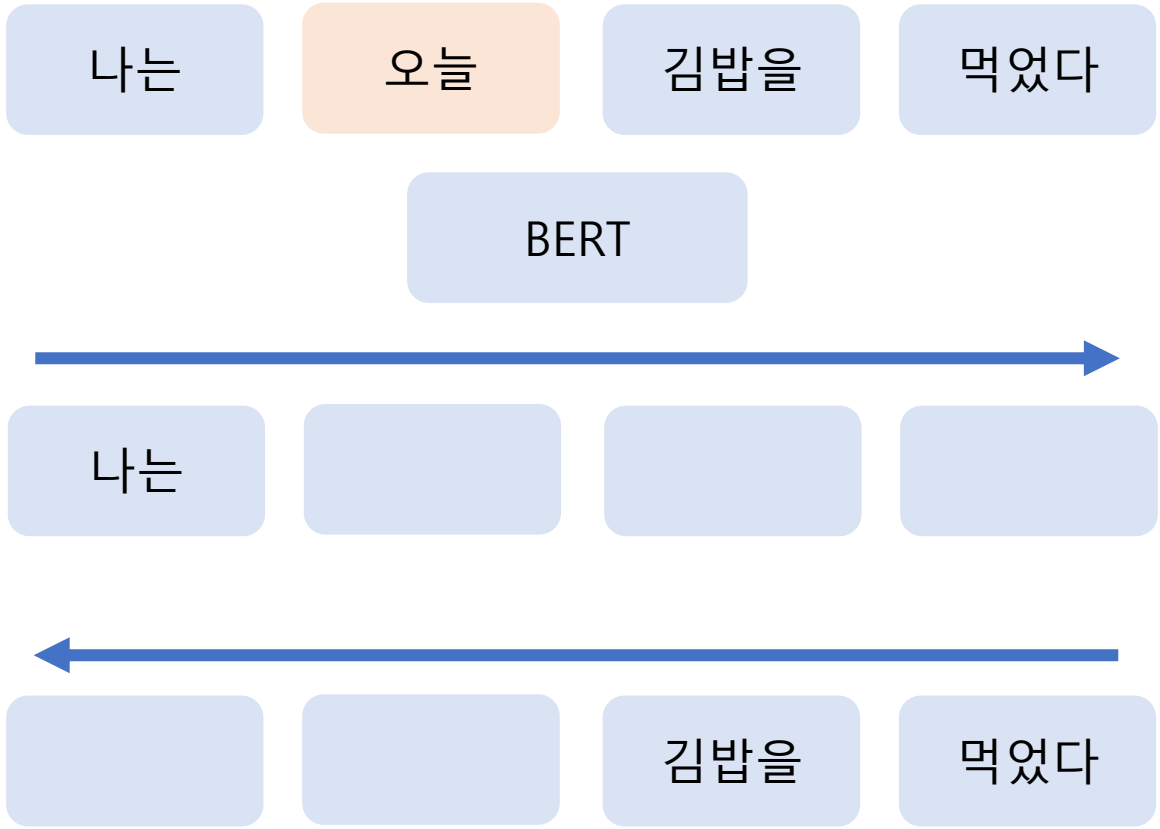
Pre-training

GPT와 ELMO에서 사용한 언어모델은 left-to-right, right-to-left방식



Pre-training

BERT는 "나는 __ 김밥을 먹었다" 처럼 __ 를 예측하기 위해,
__의 앞인 "나는" 과, "김밥을 먹었다" 를 모두 활용. (양방향 학습)



MLM(Masked Language Model)

일련의 단어가 주어지면 그 단어를 예측하는 작업

1. 15%의 단어를 랜덤으로 선택하여 마스킹
2. 15% 토큰을 생성하는 과정에서

80%는 [MASK] 토큰으로 바꾸고

```
token = [[CLS], Paris, is , a, beautiful, [MASK], [SEP], I, love, Paris, [SEP]]
```

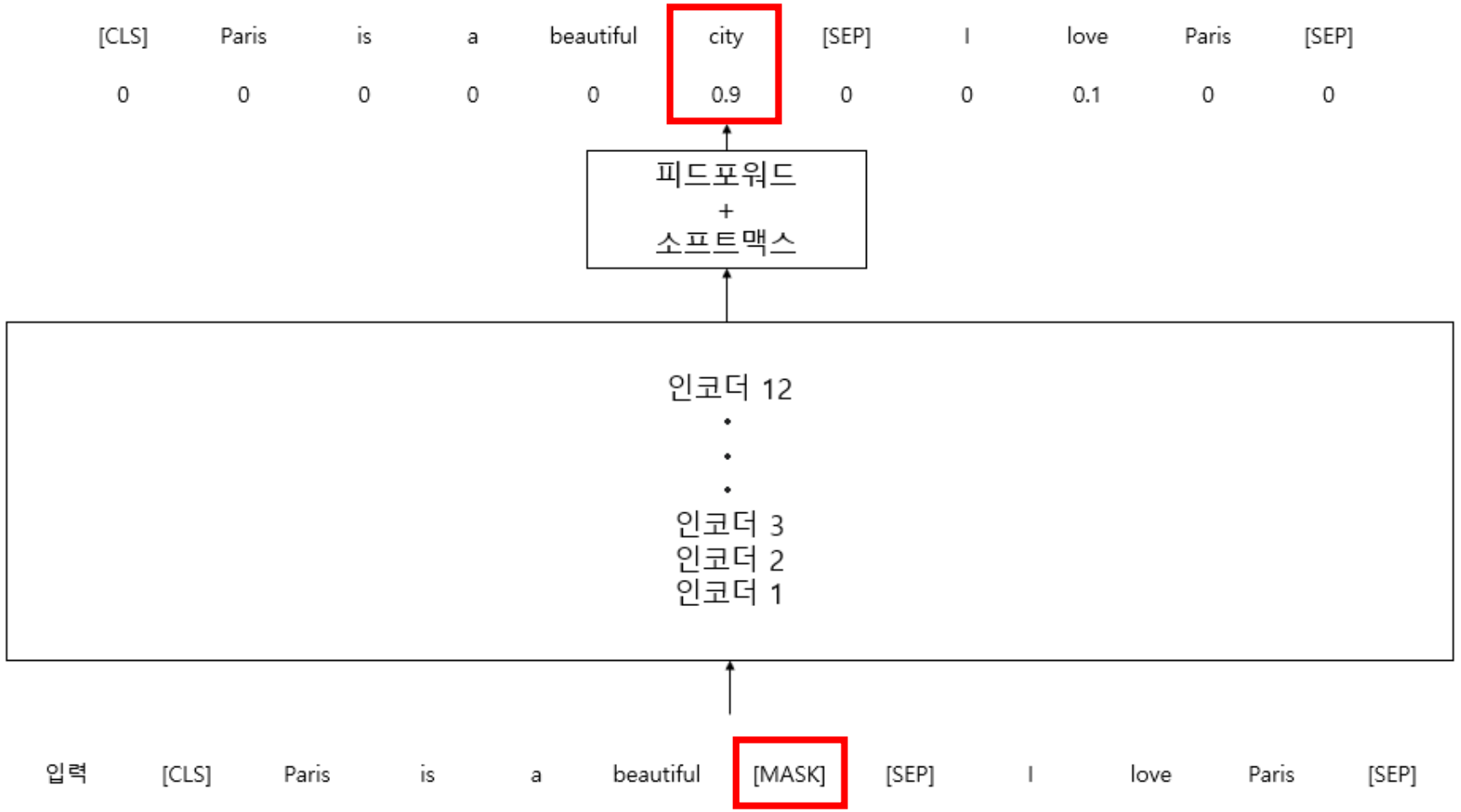
10%는 토큰을 랜덤 단어로 바꾸고,

```
token = [[CLS], Paris, is , a, beautiful, love, [SEP], I, love, Paris, [SEP]]
```

10%는 그대로 두어 [MASK]토큰을 맞추는 작업 수행

```
token = [[CLS], Paris, is , a, beautiful, city, [SEP], I, love, Paris, [SEP]]
```

MLM(Masked Language Model)



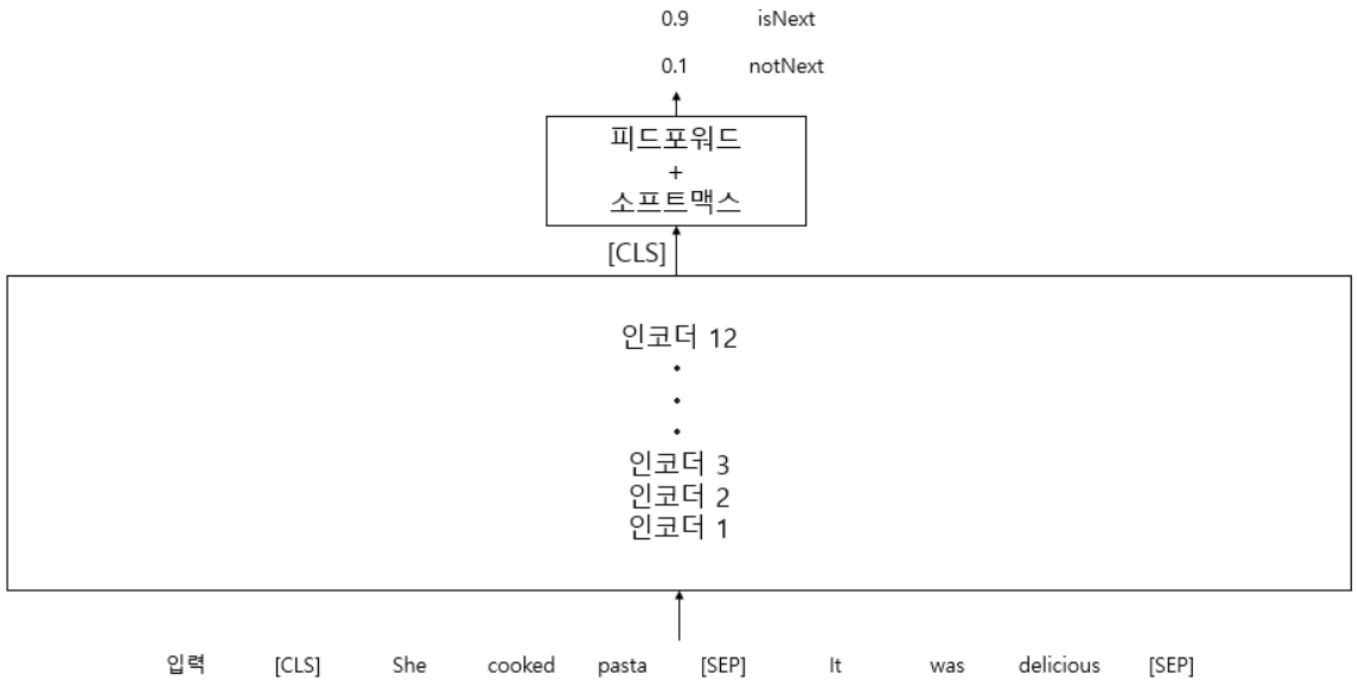
NSP(Next Sentence Prediction)

두 문장을 입력하고

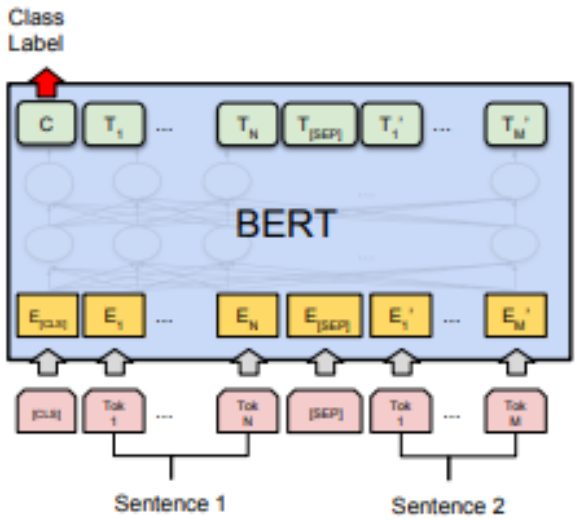
두 번째 문장이 첫 번째 문장의 바로 다음에 오는 문장인지 예측하는 방식

문장 쌍	레이블
She cooked pasta(그녀는 파스타를 요리했다) It was delicious(맛있었다)	isNext
Jack loves songwriting(잭은 작곡을 좋아한다) He wrote a new song(그는 새 노래를 썼다)	isNext
Birds fly in the sky(새들은 하늘을 난다) He was reading(그는 읽고 있었다)	notNext
Turn the radio on(라디오 켜줘) She bought a new hat(그녀는 새 모자를 샀다)	notNext

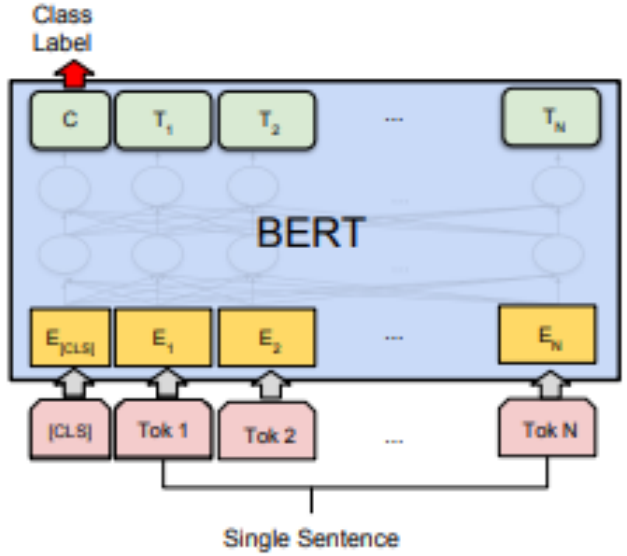
```
tokens = [[CLS], She, cooked, pasta, [SEP], It, was, delicious, [SEP]]
```



Fine-Tuning



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



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

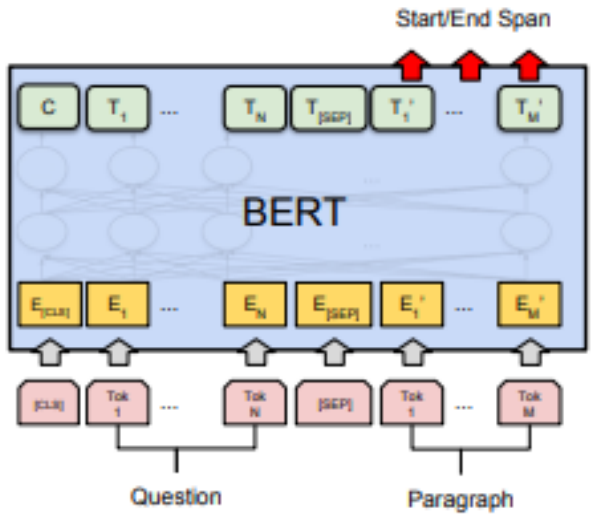
(a) 문장이 두개가 들어가야 하는 경우.

Input : [CLS] sentence1 [SEP] sentence2
가장 확률이 높은 label 예측하는 방식

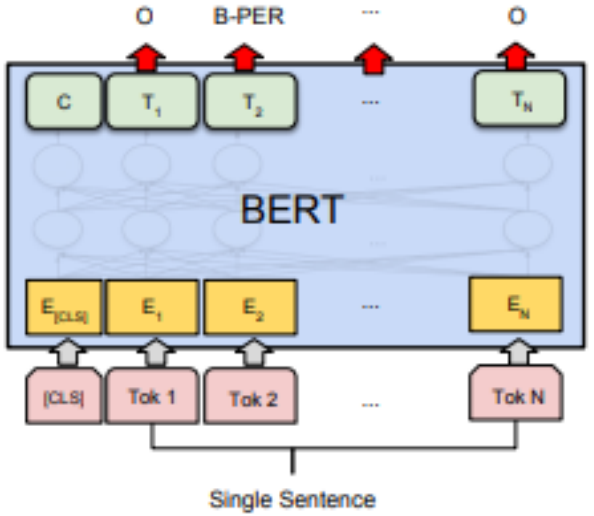
(b) 하나의 문장만 들어가면 되는 경우.

감정분류

Fine-Tuning



(c) Question Answering Tasks:
SQuAD v1.1



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

(c) 두개의 문장이 들어가는 경우.

질의 응답처럼 질문과 단락이 있는 경우

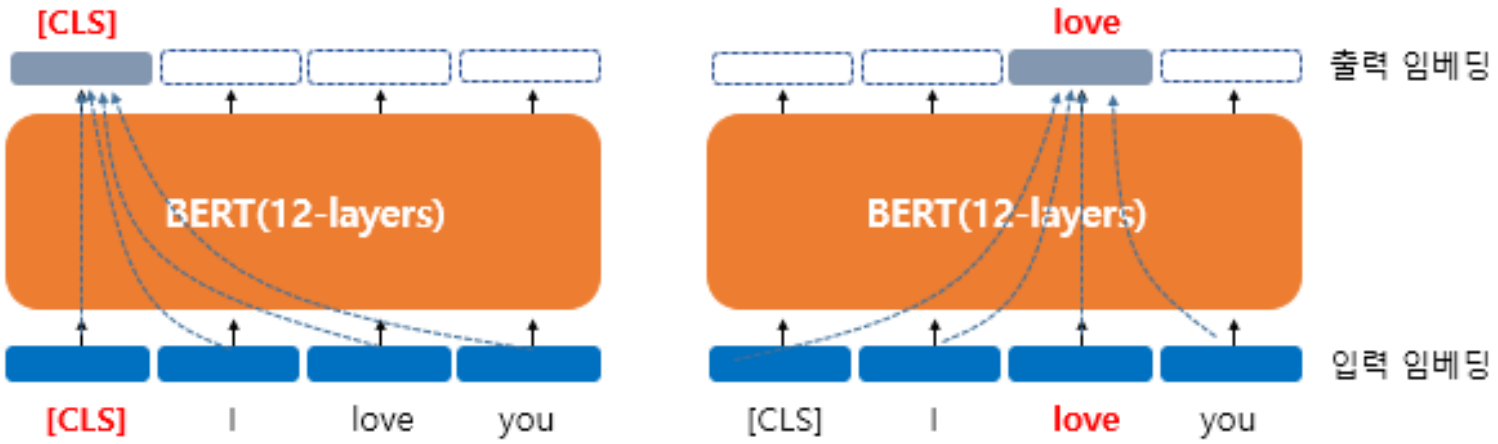
(d) 문장의 각 단어의 품사 태깅, 개체 태깅

모든 output을 활용하여, 문장성분 정보를 학습

BERT-base

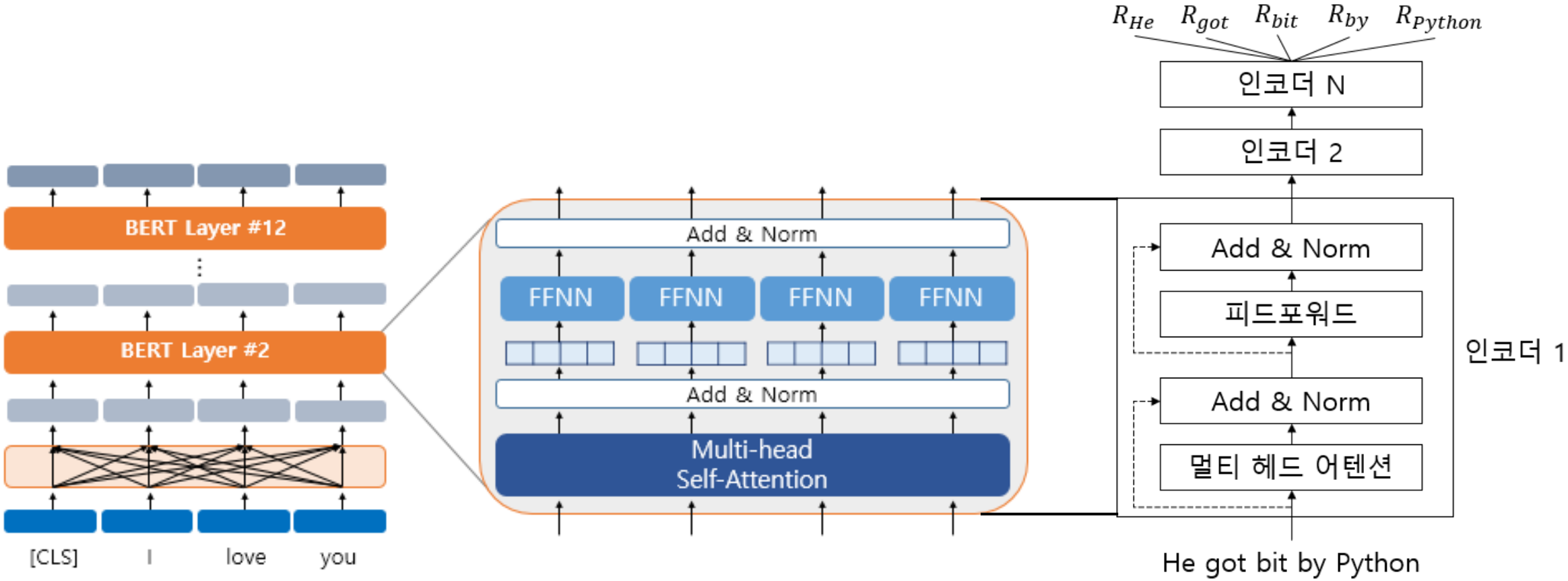
BERT 아키텍처의 규모에 따라 base와 large모델이 있다.

L = 트랜스포머 블록
H = hidden layer 차원 수
A = self-attention의 head 수
BERT-base 모델의 하이퍼 파라미터는 L = 12, H =768, A = 12



BERT의 연산을 거친 후의 출력 임베딩은 문장의 문맥을 모두 참고한 문맥을 반영한 임베딩이 된다.

BERT-base


R_{He} R_{got} R_{bit} R_{by} R_{Python}
인코더 N
인코더 2
Add & Norm
피드포워드
Add & Norm
멀티 헤드 어텐션
인코더 1
He got bit by Python
BERT Layer #12
BERT Layer #2
[CLS] I love you
Add & Norm
FFNN FFNN FFNN FFNN
Add & Norm
Multi-head Self-Attention