BERT 이해하기

00 들어가며

- 워드투벡터(word2vec)를 사용할경우 문맥을 고려하지 않음

  I $record^A$ a $record^B$

  위 문장의 경우 $record^A$ 와 $record^B$는 철자만 같고 문맥상 의미가 다름, 하지만 워드투벡터를 사용할경우 임베딩 결과가 같음

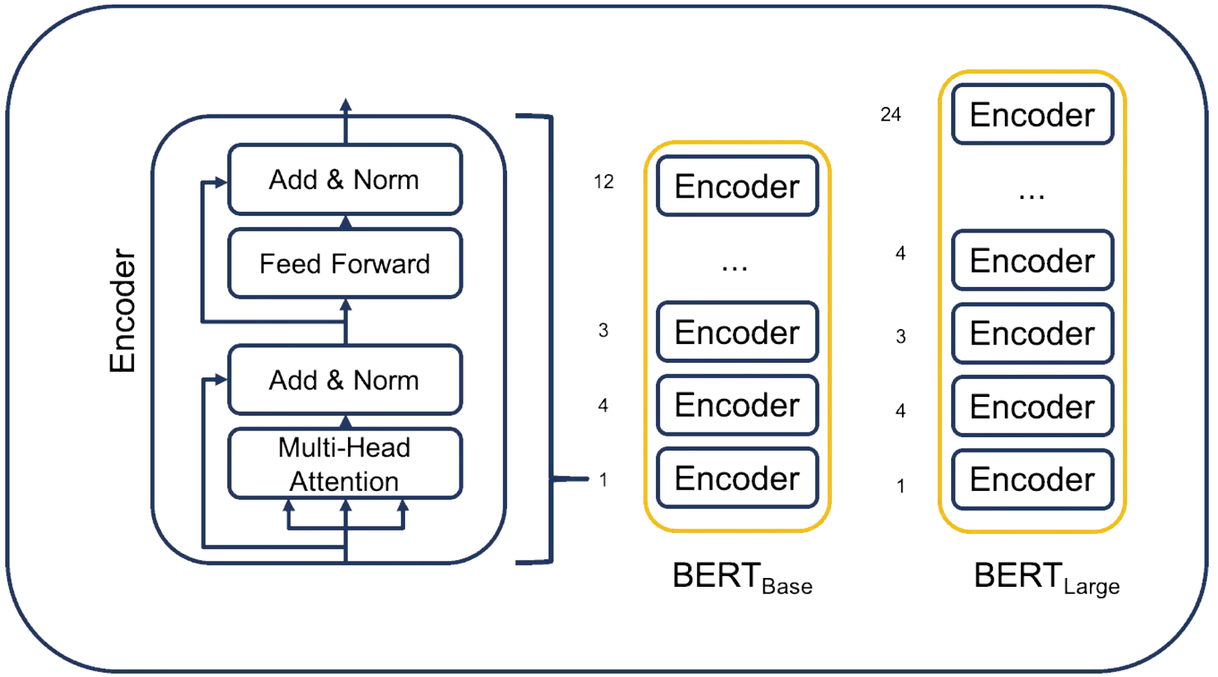
- 반면 BERT는 문맥 기반 모델이므로 문장의 문맥을 이해한다음 문맥에 따라 임베딩을 생성,

  따라서 'I record a record'라는 문장의 문맥을 기반으로 'record'라는 단어에 대해 서로 다른 임베딩을 제공

Contents

01 BERT의 구조

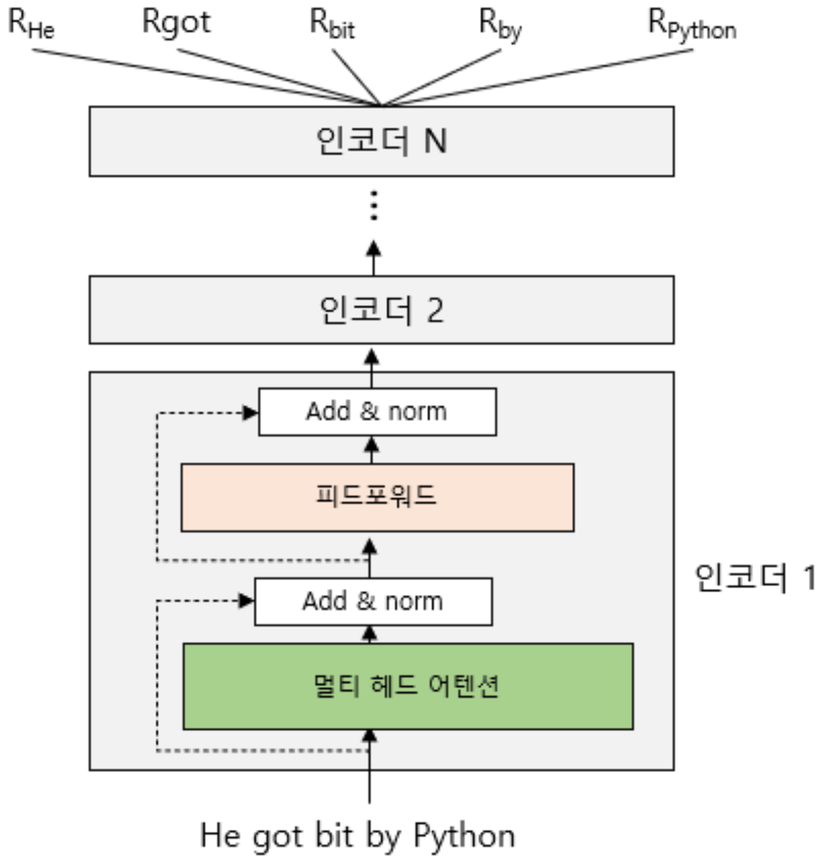
- BERT(Bidirectional Encoder Representations from Transformers)는 말 그대로 트랜스포머 인코더의 양방향 표현, 즉 트랜스포머의 인코더만 사용하여 양방향 인코더 표현을 출력하는 모델
- BERT는 인코더 레이어(L), 어텐션 헤드(A), 히든 레이어(H)으로 구성되있고 BERT의 기본모델은 L=12, A=12, H=768인 BERT-base와 L=24, A=16, H=1024인 BERT-large가 있음



02 BERT의 동작 방식

- 트랜스포머의 인코더는 원래 양방향으로 문장을 읽을 수 있기 때문에 BERT또한 양방향으로 문장을 읽어 표현을 출력함

- 문장이 인코더의 입력으로 들어오면 인코더는 멀티 헤드 어텐션 메커니즘으로 단어끼리 모두 연결하여 관계와 문맥을 파악해 문장 각 단어의 표현을 출력



03 BERT의 사전학습

- BERT는 워드피스 토크나이저(WordPiece Tokenizer)로 입력된 문장을 토큰화 한 다음 아래의 세 가지 임베딩으로 변환하여 BERT에 입력으로 제공함
  - 토큰 임베딩(token embedding)
  - 세그먼트 임베딩(segment embedding)
  - 위치 임베딩(position embedding)

- BERT는 MLM과 NSP라는 Task를 이용해 거대한 말뭉치를 기반으로 사전학습 됨

03 BERT의 사전학습

1. BERT의 입력 표현

※워드피스 토크나이저(WordPiece Tokenizer)

- BERT는 워드피스 토크나이저라는 토크나이저를 사용하며 워드피스 토크나이저는 하위 단어 토큰화 알고리즘을 기반으로 함

- 'let us start pretraining the model' 이 문장을 워드피스 토크나이저로 토큰화하면 결과는 다음과 같음

```
tokens
✓ 0.0s
['let', 'us', 'start', 'pre', '##train', '##ing', 'the', 'model']
```

- 하나의 단어 pretraining이 pre, ##train, ##ing와 같은 하위단어로 분할

03 BERT의 사전학습

1. BERT의 입력 표현

※워드피스 토크나이저(WordPiece Tokenizer)

- 워드피스 토크나이저는 단어가 어휘 사전에 있는지 확인하고 있으면 그 단어를 토큰으로 사용하고 없으면 그 단어를 하위 단어로 분할해 하위 단어가 어휘 사전에 있는지 확인하는 작업을 반복
- 개별 문자에 도달할 때까지 하위 단어로 계속 분할하는 방식은 어휘 사전 이외(OOV, out of vocabulary)의 단어를 처리하는데 효과적

- BERT에 입력하기 위해 문장의 시작부분에 [CLS]토큰을, 끝부분에 [SEP]토큰을 추가

```
tokens
✓ 0.0s
['[CLS]', 'let', 'us', 'start', 'pre', '##train', '##ing', 'the', 'model', '[SEP]']
```

03 BERT의 사전학습

1. BERT의 입력 표현

토큰 임베딩

- 토큰 임베딩 레이어를 사용해 토큰 임베딩으로 변환하기 전 문장을 워드피스 토크나이저로 토큰화
  - 문장 A: Paris is a beautiful city
  - 문장 B: I love Paris

```
tokens
✓ 0.0s
['Paris', 'is', 'a', 'beautiful', 'city', 'I', 'love', 'Paris']
```

- BERT에 입력하기 위해 문장의 시작부분에 [CLS]토큰을, 각 문장의 끝부분에 [SEP]토큰을 추가

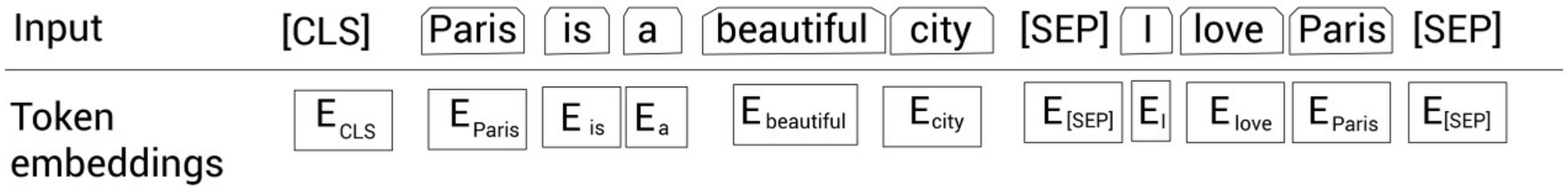
```
tokens
✓ 0.0s
['[CLS]', 'Paris', 'is', 'a', 'beautiful', 'city', '[SEP]', 'I', 'love', 'Paris', '[SEP]']
```

03 BERT의 사전학습

1. BERT의 입력 표현

토큰 임베딩(Token embedding)

- 토큰화 후 [CLS]토큰과 [SEP]토큰이 추가됐으면 임베딩 레이어를 사용해 토큰 임베딩으로 변환



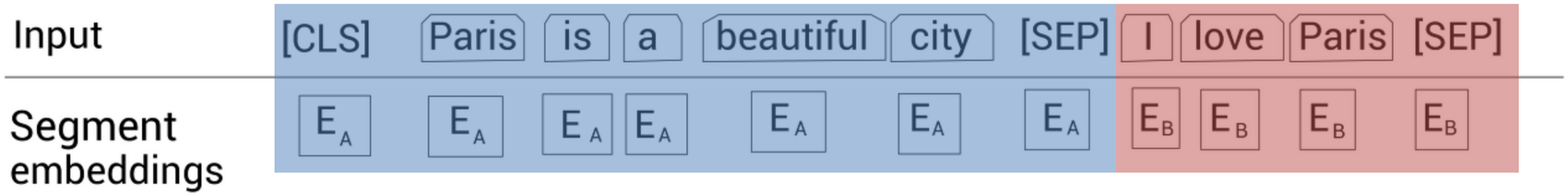
- 토큰 임베딩의 변수들은 사전 학습이 진행되면서 학습됨

03 BERT의 사전학습

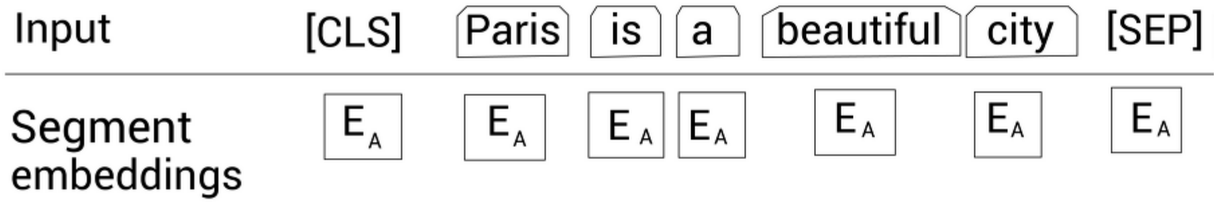
1. BERT의 입력 표현

세그먼트 임베딩(Segment embedding)

- 세그먼트 임베딩은 주어진 두 문장을 구별하는데 사용
- 문장 A에 해당하는 토큰은 세그먼트 임베딩으로 E_A값을,
  문장 B에 해당하는 토큰은 세그먼트 임베딩으로 E_B값을 가짐



- 입력 데이터가 단일 문장일 경우 모든 토큰은 같은 세그먼트 임베딩값을 가짐

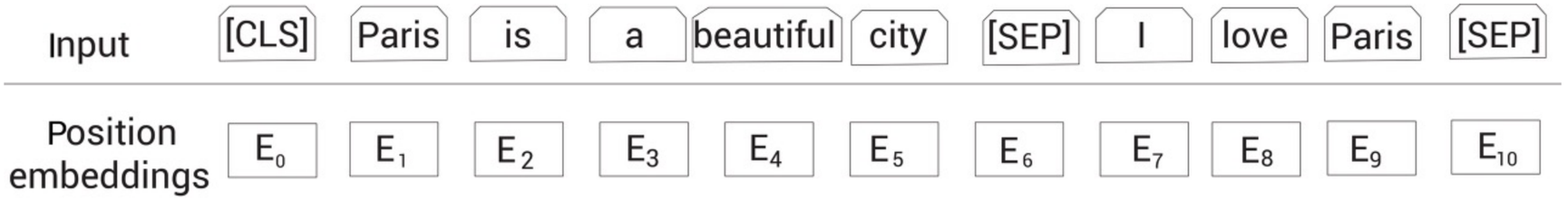


03 BERT의 사전학습

1. BERT의 입력 표현

위치 임베딩(Position embedding)

- 트랜스포머는 모든 단어를 병렬로 처리하므로 단어 순서와 관련된 정보를 제공해야 함
  BERT또한 위치에 대한 정보를 제공해야 하므로 위치 임베딩 레이어로 토큰의 위치 임베딩을 출력



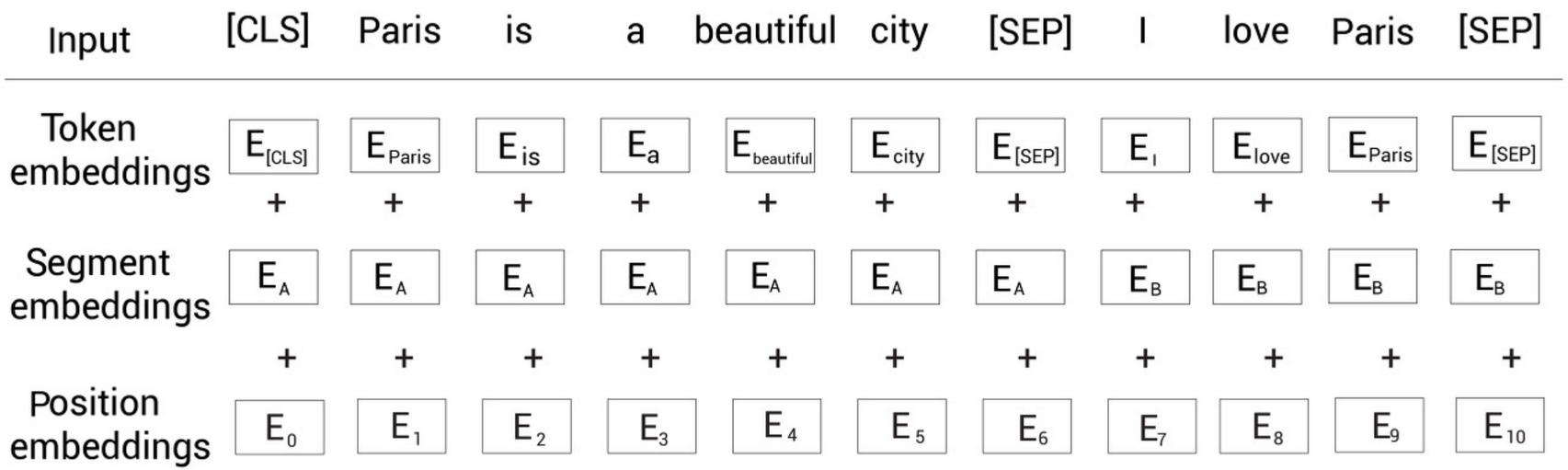
※ BERT는 트랜스포머와 달리 사인파 함수를 사용하지 않고 학습된 위치 임베딩(Learned Positional Embeddings)을 사용하여 위치 정보를 제공함

03 BERT의 사전학습

1. BERT의 입력 표현

최종 입력 데이터 표현

- BERT의 최종 입력 데이터 표현은 주어진 문장을 워드피스 토크나이저로 토큰화 한 뒤 각 토큰에 대해 3가지 임베딩 값을 구하여 합한 다음 BERT에 입력으로 제공



03 BERT의 사전학습

2. BERT의 사전 학습 Task

- BERT는 사전 학습에 토론토 책 말뭉치(Toronto BookCorpus)및 영문 위키피디아 데이터셋을 사용하고 다음 두가지 Task에 대해 사전학습됨
  - 마스크 언어 모델링(MLM, Masked Language Modeling)
  - 다음 문장 예측(NSP, Next Sentence Prediction)

03 BERT의 사전학습

2. BERT의 사전 학습 Task

마스크 언어 모델링(MLM, Masked Language Modeling)

- 언어 모델링은 일반적으로 문장이 주어지고 다음 단어를 예측하도록 모델을 학습시키는 것

  자동 회귀 언어 모델링(auto-regressive language modeling)

  - ‘Paris is a beautiful ____. I love Paris.’ 라는 문장이 있을 때 자동 회귀 언어 모델링은 단방향으로 문장을 읽을 수 있음
  - 공백을 기준으로 왼쪽에서 오른쪽으로 단어를 읽는 전방 예측(Paris is a beautiful ____.)

    공백을 기준으로 오른쪽에서 왼쪽으로 단어를 읽는 후방 예측(____. I love Paris.)

  자동 인코딩 언어 모델링(auto-encoding language modeling)

  - 자동 인코딩 언어 모델링은 전방 예측, 후방 예측 모두 활용, 즉 양뱡향으로 문장을 읽을 수 있음
  - BERT의 사전 학습 Task중 하나인 MLM은 자동 인코딩 언어 모델

03 BERT의 사전학습

2. BERT의 사전 학습 Task

마스크 언어 모델링(MLM, Masked Language Modeling)

- MLM은 주어진 입력 문장에서 전체 단어의 15%를 무작위로 마스킹하고 마스킹된 단어를 예측하도록 모델을 학습

```
tokens
✓ 0.0s
['[CLS]', 'Paris', 'is', 'a', 'beautiful', 'city', '[SEP]', 'I', 'love', 'Paris', '[SEP]']
```

- 위와 같이 토큰화 이후 [CLS], [SEP] 토큰을 추가한 뒤 토큰의 15%를 무작위로 마스킹

```
tokens
✓ 0.0s
['[CLS]', 'paris', 'is', 'a', 'beautiful', '[MASK]', '[SEP]', 'i', 'love', 'paris', '[SEP]']
```

- ‘city’ 토큰이 마스킹 되어 ‘[MASK]’ 토큰으로 바뀜

03 BERT의 사전학습

2. BERT의 사전 학습 Task

마스크 언어 모델링(MLM, Masked Language Modeling)

- 이러한 방식은 사전학습과 다운스트림 태스크간의 불일치를 유발
- 예를 들어 감정 분석 태스크를 위해 사전학습된 BERT를 파인튜닝을 하면 파인튜닝을 위해 입력된 데이터에는 [MAKS] 토큰이 없는 불일치가 발생

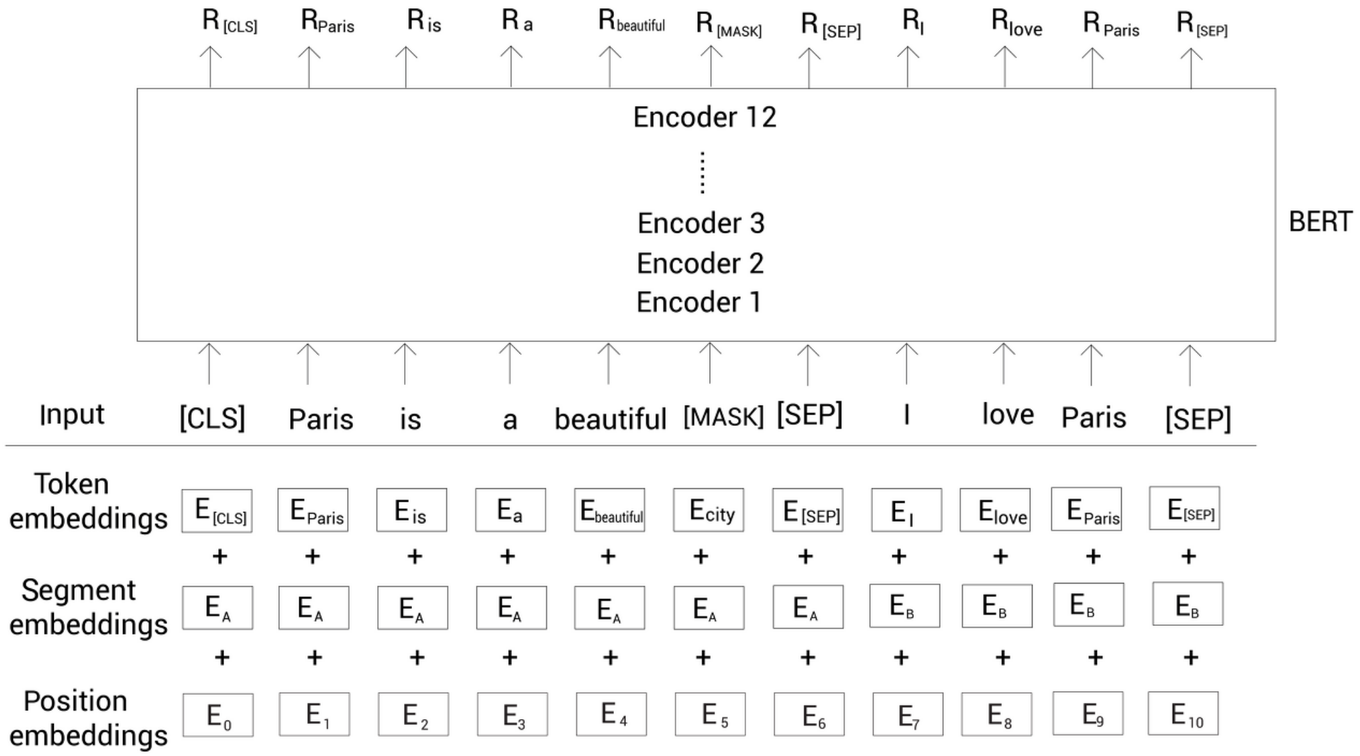
- 이에 해당 문제를 해결하기 위해서 80-10-10% 규칙을 도입
  - 15% 중 80%를 [MASK] 토큰으로 교체
  - 15% 중 10%를 임의의 토큰(임의의 단어)로 교체
  - 15% 중 10%를 어떠한 변경도 하지 않음

03 BERT의 사전학습

2. BERT의 사전 학습 Task

마스크 언어 모델링(MLM, Masked Language Modeling)

- 마스킹 작업이 완료됐으면 임베딩 레이어에 입력에 입력 임베딩을 얻은 후 BERT에 제공



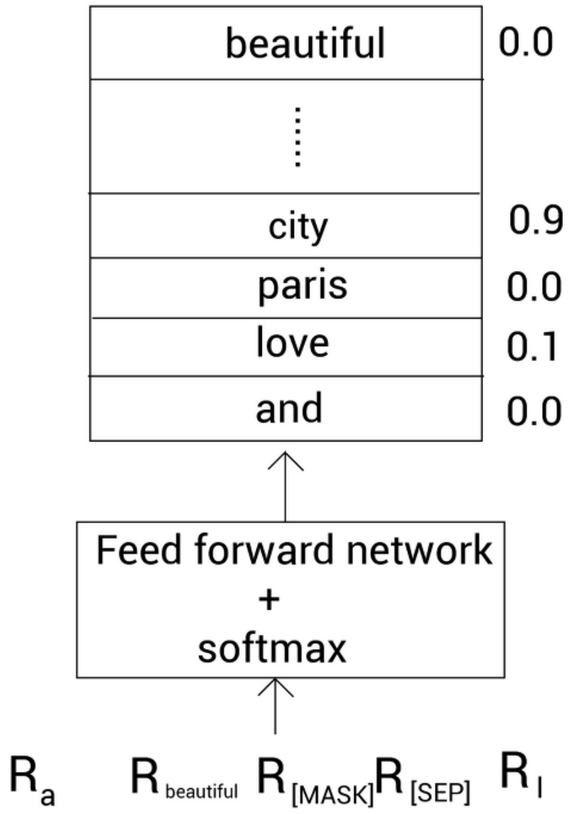
03 BERT의 사전학습

2. BERT의 사전 학습 Task

마스크 언어 모델링(MLM, Masked Language Modeling)

- 각 토큰의 표현(R)을 구했으면 마스킹된 토큰을 예측하기 위해 마스킹된 토큰의 표현 $R_{[MASK]}$을 softmax 함수를 사용한 피드포워드 네트워크(FFN, Feed-Forward Network)에 입력
- 학습 초기에는 FFN, 인코더의 가중치가 최적이 아니므로 역전파를 통한 반복 학습을 거쳐 가중치가 업데이트 되 최적의 가중치를 학습

- MLM Task는 빈칸 채우기 태스크(cloze task)라고도 불림



03 BERT의 사전학습

2. BERT의 사전 학습 Task

마스크 언어 모델링(MLM, Masked Language Modeling)

※ 전체 단어 마스킹(WWM, Whole Word Masking)

- 워드피스 토크나이저로 문장을 토큰화 한 경우 다음과 같이 하위 단어로 분할될 수 있음

```
tokens
✓ 0.0s
['[CLS]', 'let', 'us', 'start', 'pre', '##train', '##ing', 'the', 'model', '[SEP]']
```

- 토큰의 15%를 무작위로 마스킹한 결과 'let' 과 '##train' 토큰이 마스킹 됨

```
tokens
✓ 0.0s
['[CLS]', '[MASK]', 'us', 'start', 'pre', '[MASK]', '##ing', 'the', 'model', '[SEP]']
```

- WWM 방법에선 하위 단어가 마스킹되면 해당 단어와 관련된 모든 단어를 마스킹, 이 때 마스킹 비율이 15%가 넘어가면 다른 단어의 마스킹을 무시

```
tokens
✓ 0.0s
['[CLS]', 'let', 'us', 'start', '[MASK]', '[MASK]', '[MASK]', 'the', 'model', '[SEP]']
```

03 BERT의 사전학습

2. BERT의 사전 학습 Task

다음 문장 예측(NSP, Next Sentence Prediction)

- 문장A와 B가 주어지고 B가 A의 다음 문장인지 예측하는 이진 분류 태스크
- 한 문서에서 연속된 두 문장을 가져와 isNext 레이블, 한 문서에서 한 문장을, 또다른 문서에서 한문장을 가져와 NotNext 레이블을 주어 두 레이블을 50%씩 차지하게 데이터셋 구성

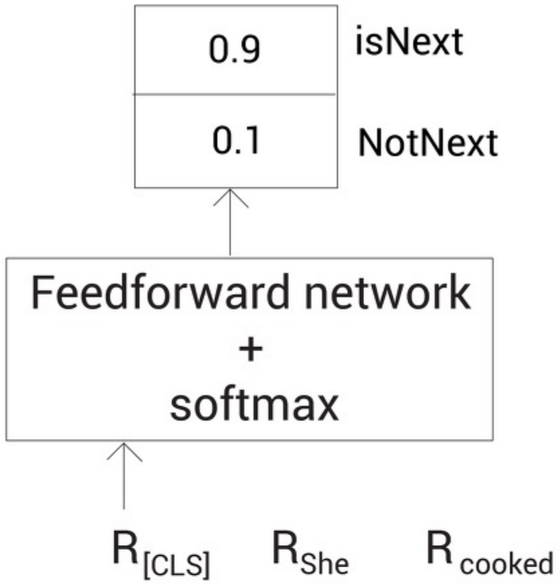
문장 쌍	레이블
She cooked pasta It was delicious	IsNext
Jack loves songwriting He wrote a new song	isNext
Birds fly in the sky He was reading	NotNext
Turn the radio on She bought a new hat	NotNtext

03 BERT의 사전학습

2. BERT의 사전 학습 Task

다음 문장 예측(NSP, Next Sentence Prediction)

- 문장 A, B를 MLM과 같은 방식으로 워드피스 토크나이저로 토큰화 한 뒤 임베딩 레이어에 입력해 얻은 임베딩을 BERT에 제공
- 이진 분류를 위해 [CLS]토큰의 표현을 FFN, softmax에 입력
- 학습 초기에는 FFN, 인코더의 가중치가 최적이 아니므로 역전파를 통한 반복 학습을 거쳐 가중치가 업데이트 되 최적의 가중치를 학습



※ [CLS] 토큰만 사용하는 이유

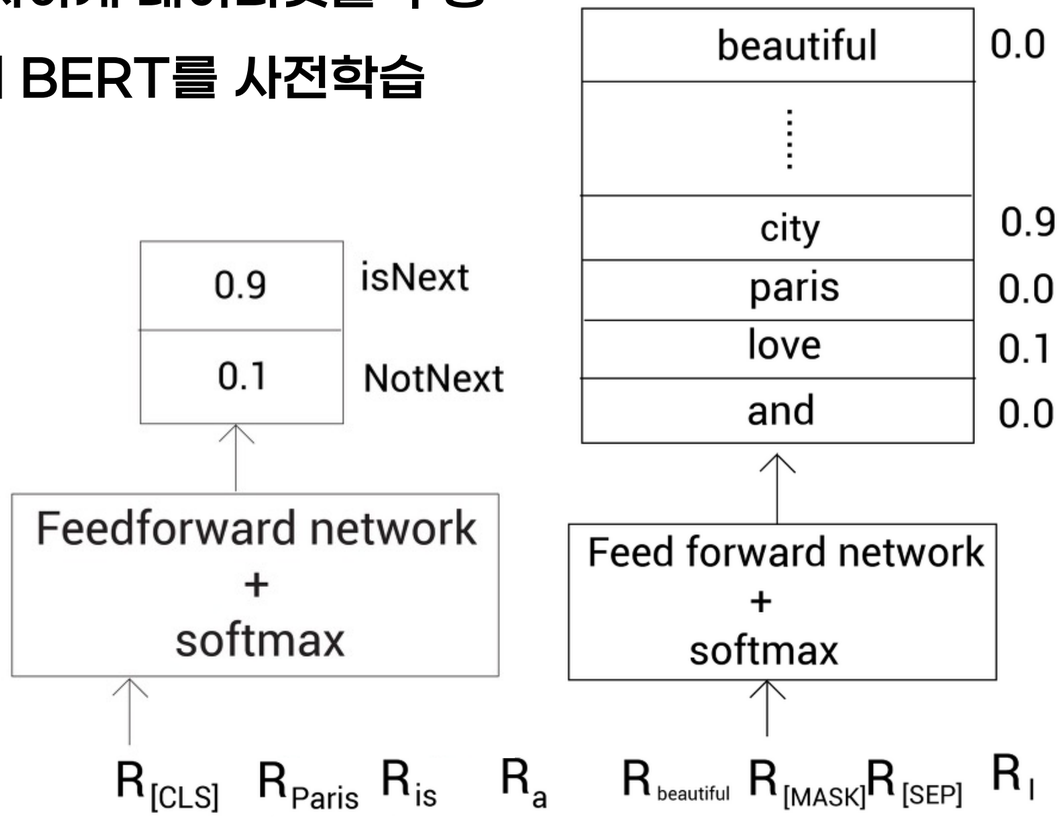
- [CLS] 토큰은 기본적으로 모든 토큰의 집계 표현을 보유하고 있으므로 문장 전체에 대한 표현을 담고 있음

03 BERT의 사전학습

2. BERT의 사전 학습 Task

사전 학습 절차

- BERT는 사전 학습에 토론토 책 말뭉치(Toronto BookCorpus)및 영문 위키피디아를 사용하고 isNext, NotNext 레이블이 절반씩 차지하게 데이터셋을 구성
- 그림과 같이 MLM과 NSP 태스크를 동시에 사용해 BERT를 사전학습



- Batch size : 256, 1,000,000 steps, Adam optimizer, lr : 1e-4, β1 : 0.9, β2 : 0.999
- 초반에는 높은 학습률을, 후반에는 낮은 학습률을 사용하는 웜업 스텝 사용
- Dropout 확률이 0.1인 모든 레이어에 dropout을 적용하고 활성화 함수는 GELU 사용

03 BERT의 사전학습

※ 하위 단어 토큰화 알고리즘

- 하위 단어 토큰화는 BERT 및 GPT-3를 포함한 많은 최신 자연어 모델에서 널리 사용
  WHY? 이러한 방식이 어휘 사전 이외(OOV)의 단어를 처리하는데 효과적이기 때문

- 하위 단어 토큰화 알고리즘 없이 'let us start pretraining the model' 이 문장을 토큰화 하면 어휘 사전에 'pretraining'이라는 단어가 없으므로 <UNK>토큰으로 토큰화 됨

```
tokens
✓ 0.0s
['let', 'us', 'start', '<UNK>', 'the', 'model']
```

- 하위 단어 토큰화 알고리즘을 사용하면 어휘 사전에 'pretraining'이라는 단어가 없어도 'pre', 'train', 'ing'라는 단어가 있으므로 토큰화가 가능

```
tokens
✓ 0.0s
['let', 'us', 'start', 'pre', '##train', '##ing', 'the', 'model']
```

03 BERT의 사전학습

※ 하위 단어 토큰화 알고리즘

- 어휘 사전을 생성하는 데 사용되는 대표적인 하위 단어 토큰화 알고리즘으로는 다음과 같은 3가지 방식이 있음
  - 바이트 쌍 인코딩(BPE, byte pair encoding)
  - 바이트 수준 바이트 쌍 인코딩(BBPE, byte-level byte pair encoding)
  - 워드피스(WordPeice)

03 BERT의 사전학습

※ 하위 단어 토큰화 알고리즘

1. 바이트 쌍 인코딩(BPE, byte pair encoding)

- BPE를 사용해 어휘 사전을 구축하는 단계는 다음과 같다
  1. 데이터셋에서 빈도수와 함께 단어 추출
  2. 어휘 사전의 크기 정의
  3. 단어를 문자 시퀀스로 분할
  4. 문자 시퀀스의 모든 고유 문자를 어휘 사전에 추가
  5. 빈도가 높은 기호쌍을 선택하고 병합, 어휘 사전에 추가
  6. 어휘 사전 크기에 도달할 때까지 5번 반복

- 데이터셋에서 빈도수와 함께 단어 추출한 결과가 다음과 같고 어휘 사전의 크기는 14라고 하자
- (cost, 2), (best, 2), (menu, 1), (men, 1), (camel, 1)

03 BERT의 사전학습

※ 하위 단어 토큰화 알고리즘

1. 바이트 쌍 인코딩(BPE, byte pair encoding)

- 문자 시퀀스에 있는 모든 고유 문자를 어휘 사전에 추가
- 기호 쌍 s와 t가 가장 빈번하게 발생했으므로 기호 s와 t를 병합한 뒤 어휘 사전에 추가

문자 시퀀스	빈도수
c o s t	2
b e s t	2
m e n u	1
m e n	1
c a m e l	1

어휘사전
a, b, c, e, l, m, n, o, s, t, u

- 어휘 사전에 st가 추가되어 어휘 사전의 크기가 12가 됨
- 어휘 사전의 크기를 14라고 정의했으므로 어휘 사전의 크기가 14가 될 때까지 위와 같은 방식으로 가장 빈번한 기호 쌍을 병합한 뒤 어휘사전에 추가

문자 시퀀스	빈도수
c o st	2
b e st	2
m e n u	1
m e n	1
c a m e l	1

어휘사전
a, b, c, e, l, m, n, o, s, t, u, st

03 BERT의 사전학습

※ 하위 단어 토큰화 알고리즘

2. 바이트 수준 바이트 쌍 인코딩(BBPE, byte-level byte pair encoding)

- BPE와 유사한 방식으로 어휘 사전을 생성하지만 문자 시퀀스 대신 바이트 시퀀스를 사용
- 문자 시퀀스: b e s t -> 바이트 시퀀스: 62 65 73 74
- 이러한 방식은 다국어 설정에서 매우 유용하고 여러 언어로 어휘 사전을 공유할 수 있음

3. 워드피스(WordPeice)

- 워드피스 또한 BPE와 유사한 방식으로 어휘 사전을 생성하지만 빈도수 대신 가능도(likelihood) 사용
- 즉, 모든 기호 쌍에 대해 언어 모델의 가능도를 확인한 후 가능도가 가장 높은 기호 쌍을 병합하는 방식으로 어휘 사전 생성

감사합니다.