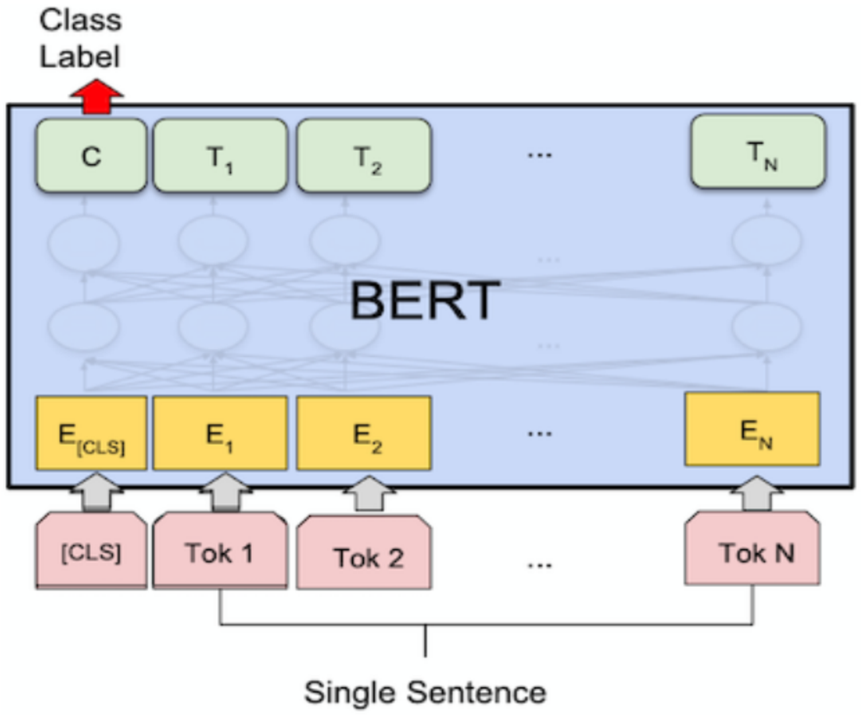
BERT를 이용한 영화 리뷰 감정 분석

20190431 박규현

목차

01 - 01
BERT란?



Transformer 기반의 기계번역 모델

Transformers 모델의 Encoder 구조를 가짐

- 언어 처리(NLP) 인공지능의 최첨단 딥러닝 모델

* 기계 번역 모델 : 사람이 사용하는 말을 컴퓨터가 알아들을 수 있도록 번역하는 모델

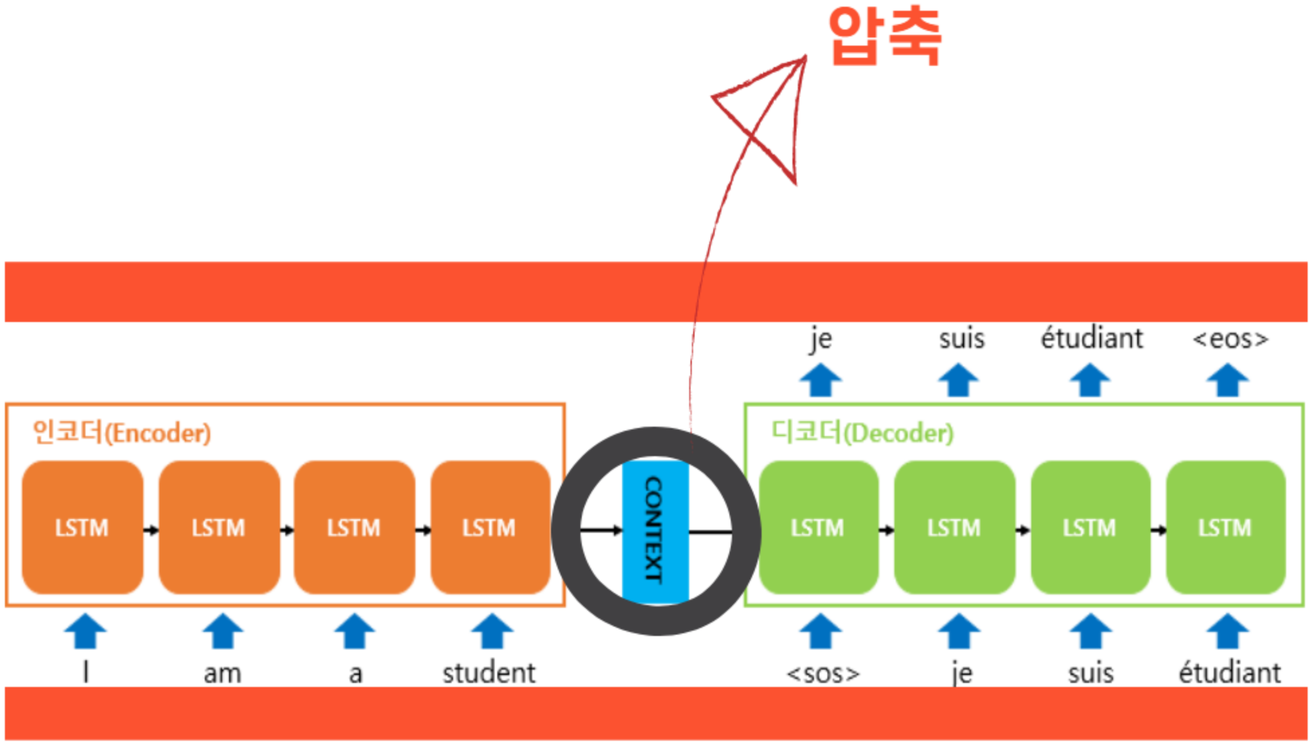
01 - 02

Seq2Seq와 Attention

Sequence-to-Sequence는 어떤 연속된 데이터들을

다른 Sequence로 Mapping하는 알고리즘

=> 정보의 손실

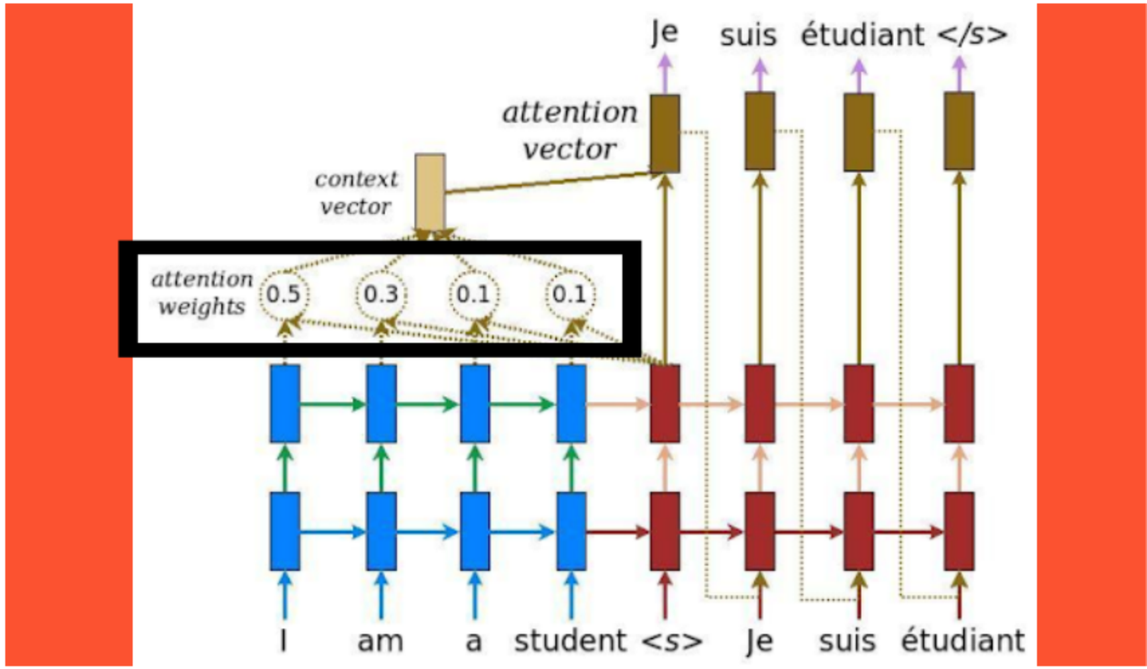


01 - 02

Seq2Seq와 Attention

1. Attention이란 'Input Data의 이 부분이 중요해요! 집중해 주세요!'라고 하는 수치들을 같이 output으로 넘겨주는 방식

2. Input들이 얼마나 output 생성에 기여하는지를 표현하는 것



01 - 03
Transformer

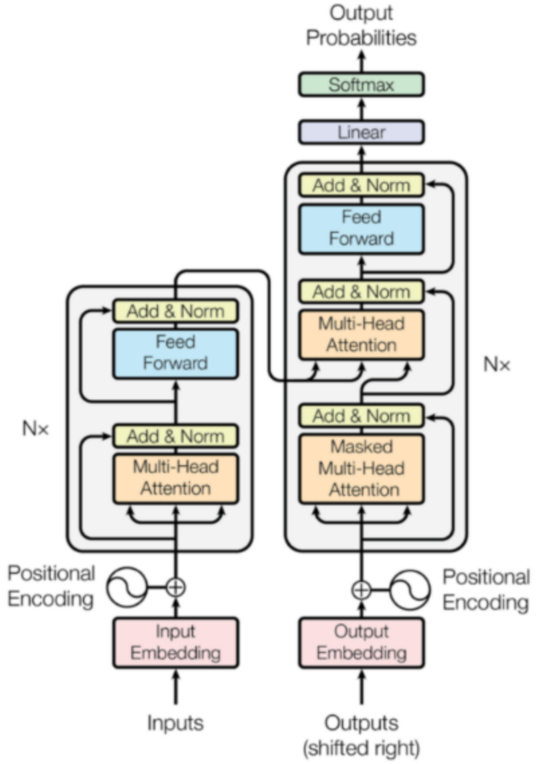


Figure 1: The Transformer - model architecture.

✓ 기존 모델의 약점 - 단어를 순차적으로 입력받아 배열로 간주함.

✓ Transformer에서 달라진 점

1. equence를 한번에 넣음 ▶ 병렬처리가 가능, Attention 등의 구조를 통해 어떤 부분이 중요한지를 전달

2. 위치정보를 반영할 수 있게됨

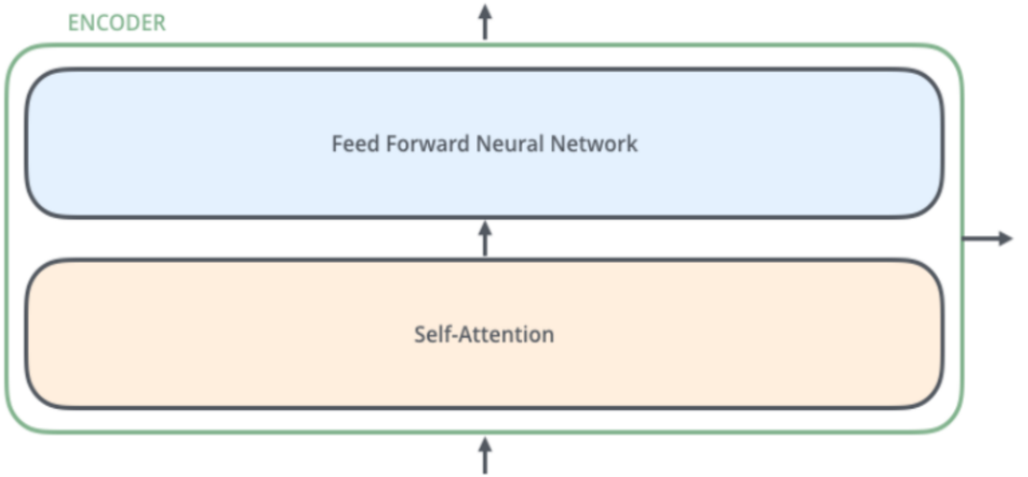
01 - 04

BERT란?

Transformer 모델의 인코더 부분만을 사용하는

자연어 모델

* Self-Attention : 한 단어와 나머지 다른 단어의 관계 정보를 처리



02

코드 리뷰

데이터 로드

```
[ ]  # 네이버 영화리뷰 감정분석 데이터 다운로드
     !git clone https://github.com/e9t/nsmc.git
```

```
Cloning into 'nsmc'...
remote: Enumerating objects: 14763, done.
remote: Total 14763 (delta 0), reused 0 (delta 0), pack-reused 14763
Receiving objects: 100% (14763/14763), 56.19 MiB | 20.37 MiB/s, done.
Resolving deltas: 100% (1749/1749), done.
Checking out files: 100% (14737/14737), done.
```

데이터셋 분류

```
[ ]  # 판다스로 훈련셋과 테스트셋 데이터 로드
     train = pd.read_csv("nsmc/ratings_train.txt", sep='₩t')
     test = pd.read_csv("nsmc/ratings_test.txt", sep='₩t')

     print(train.shape)
     print(test.shape)
```

```
(150000, 3)
(50000, 3)
```

02

코드 리뷰

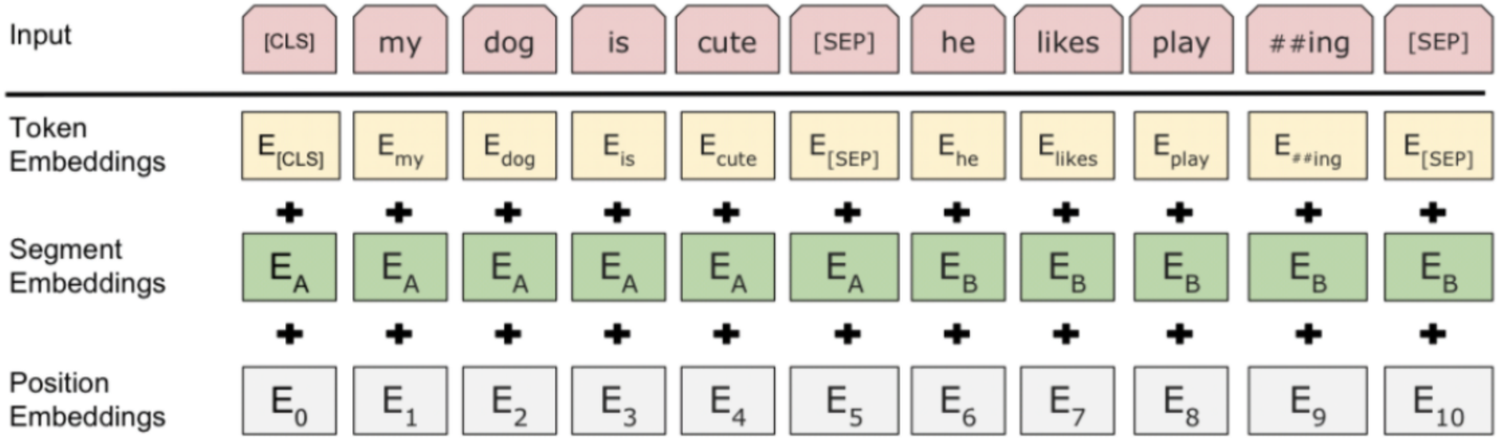
데이터 확인


```
[
  {
    "review": "전체관람가는 아닌것 같아요",
    "date": "15.08.25",
    "rating": "10",
    "author": "dhrl****",
    "review_id": "10275182",
    "movie_id": "10001"
  },
  {
    "review": "디렉터스컷으로봐서 거의 3시간짜리인데 참 흡인력있다",
    "date": "15.08.25",
    "rating": "10",
    "author": "yuns****",
    "review_id": "10272934",
    "movie_id": "10001"
  },
```


```
[ ]  # 훈련셋의 앞부분 출력
     train.head(10)
```

	id	document	label
0	9976970	아 더빙.. 진짜 짜증나네요 목소리	0
1	3819312	흠...포스터보고 초딩영화줄....오버연기조차 가볍지 않구나	1
2	10265843	너무재밋었다그래서보는것을추천한다	0
3	9045019	교도소 이야기구먼 ..솔직히 재미는 없다..평점 조정	0
4	6483659	사이몬페그의 익살스런 연기가 돋보였던 영화!스파이더맨에서 늙어보이기만 했던 커스틴 ...	1
5	5403919	막 걸음마 뗀 3세부터 초등학교 1학년생인 8살용영화.ㅋㅋㅋ...별반개도 아까움.	0
6	7797314	원작의 긴장감을 제대로 살려내지못했다.	0
7	9443947	별 반개도 아깝다 욕나온다 이응경 길용우 연기생활이몇년인지..정말 발로해도 그것보단...	0
8	7156791	액션이 없는데도 재미 있는 몇안되는 영화	1
9	5912145	왜케 평점이 낮은건데? 꽤 볼만한데.. 헐리우드식 화려함에만 너무 길들여져 있나?	1

02
코드 리뷰



1. Classification을 뜻하는 [CLS] 심볼이 제일 앞에 삽입
2. [SEP]은 Seperation을 가리키는데, 두 문장을 구분하는 역할

02

코드 리뷰

데이터 전처리

```
[ ]  # BERT의 입력 형식에 맞게 변환
     sentences = ["[CLS] " + str(sentence) + " [SEP]" for sentence in sentences]
     sentences[:10]
```

```
['[CLS] 아 더빙.. 진짜 짜증나네요 목소리 [SEP]',
 '[CLS] 흠...포스터보고 초딩영화줄....오버연기조차 가볍지 않구나 [SEP]',
 '[CLS] 너무재밋었다그래서보는것을추천한다 [SEP]',
 '[CLS] 교도소 이야기구먼 ..솔직히 재미는 없다..평점 조정 [SEP]',
 '[CLS] 사이몬페그의 익살스런 연기가 돋보였던 영화!스파이더맨에서 늙어보이기만 했던 커스틴 던스트가 너무나도 이뻐보였다 [SEP]',
 '[CLS] 막 걸음마 뗀 3세부터 초등학교 1학년생인 8살용영화.ㅋㅋㅋ...별반개도 아까움. [SEP]',
 '[CLS] 원작의 긴장감을 제대로 살려내지못했다. [SEP]',
 '[CLS] 별 반개도 아깝다 욕나온다 이응경 길용우 연기생활이몇년인지..정말 발로해도 그것보단 낫겠다 납치.감금만반복반복..이드라마는 가족도없다 연기못하는사람만모엿네 [SEP]',
 '[CLS] 액션이 없는데도 재미 있는 몇안되는 영화 [SEP]',
 '[CLS] 왜케 평점이 낮은건데? 꽤 볼만한데.. 헐리우드식 화려함에만 너무 길들여져 있나? [SEP]']
```

02

코드 리뷰

토큰 분리

```
[ ]  # BERT의 토크나이저로 문장을 토큰으로 분리
     tokenizer = BertTokenizer.from_pretrained('bert-base-multilingual-cased', do_lower_case=False)
     tokenized_texts = [tokenizer.tokenize(sent) for sent in sentences]

     print (sentences[0])
     print (tokenized_texts[0])
```

```
Downloading: 100%  972k/972k [00:00<00:00, 1.34MB/s]
Downloading: 100%  29.0/29.0 [00:00<00:00, 1.03kB/s]
Downloading: 100%  1.87M/1.87M [00:00<00:00, 3.35MB/s]
Downloading: 100%  625/625 [00:00<00:00, 21.6kB/s]
[CLS] 아 더빙.. 진짜 짜증나네요 목소리 [SEP]
['[CLS]', '아', '더', '##빙', '.', '.', '진', '##짜', '짜', '##증', '##나', '##네', '##요', '목', '##소', '##리', '[SEP]']
```

02

코드 리뷰

인덱스 변환

```
[ ] # 입력 토큰의 최대 시퀀스 길이
    MAX_LEN = 128

    # 토큰을 숫자 인덱스로 변환
    input_ids = [tokenizer.convert_tokens_to_ids(x) for x in tokenized_texts]

    # 문장을 MAX_LEN 길이에 맞게 자르고, 모자란 부분을 패딩 0으로 채움
    input_ids = pad_sequences(input_ids, maxlen=MAX_LEN, dtype="long", truncating="post", padding="post")

    input_ids[0]
```

```
array([   101,   9519,   9074, 119005,    119,    119,   9708, 119235,
         9715, 119230,  16439,  77884,  48549,   9284,  22333,  12692,
          102,      0,      0,      0,      0,      0,      0,      0,
            0,      0,      0,      0,      0,      0,      0,      0,
            0,      0,      0,      0,      0,      0,      0,      0,
            0,      0,      0,      0,      0,      0,      0,      0,
            0,      0,      0,      0,      0,      0,      0,      0,
            0,      0,      0,      0,      0,      0,      0,      0,
            0,      0,      0,      0,      0,      0,      0,      0,
            0,      0,      0,      0,      0,      0,      0,      0,
            0,      0,      0,      0,      0,      0,      0,      0,
            0,      0,      0,      0,      0,      0,      0,      0,
            0,      0,      0,      0,      0,      0,      0,      0,
            0,      0,      0,      0,      0,      0,      0,      0])
```

02

코드 리뷰

어텐션 처리

```
[ ] # 어텐션 마스크 초기화
    attention_masks = []

    # 어텐션 마스크를 패딩이 아니면 1, 패딩이면 0으로 설정
    # 패딩 부분은 BERT 모델에서 어텐션을 수행하지 않아 속도 향상
    for seq in input_ids:
        seq_mask = [float(i>0) for i in seq]
        attention_masks.append(seq_mask)

    print(attention_masks[0])

    [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0,
```

텐서 변환

```
[ ] # 데이터를 파이토치의 텐서로 변환
    test_inputs = torch.tensor(input_ids)
    test_labels = torch.tensor(labels)
    test_masks = torch.tensor(attention_masks)

    print(test_inputs[0])
    print(test_labels[0])
    print(test_masks[0])

    tensor([ 101, 8911,  100,  102,    0,    0,    0,    0,    0,    0,    0,    0,
               0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
               0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
               0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
               0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
               0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
               0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
               0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
               0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
               0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
               0,    0,    0,    0,    0,    0,    0,    0])
    tensor(1)
    tensor([1., 1., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
            0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
            0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
            0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
            0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
            0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
            0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
            0., 0.])
```

감사합니다!

20190431 박규현