<캡스톤 디자인 최종보고서>

리뷰 데이터를 활용한 Bert 모델 기반 감성분석

[리플라이 팀]
2019102179 박주영
2019102211 이인서

지도교수님: 김성태 교수님

요약

본 연구에서는 사용자들이 SNS 등 여러 플랫폼에서 표현한 의견과 견해들을 바탕으로 발전하는 딥러닝을 활용하여 여러 플랫폼에 입력된 리뷰들을 효과적으로 분석하여 사용자의 편의에 최적화된 환경을 제공하고자 한다. BERT 를 사용하여 수집한 데이터의 감성수치를 분석하고 시각화 하여 사용자의 선택에 도움 이 되는 것에 의의를 가진다.

1. 서론

1.1 연구배경

딥러닝이 급격하게 발전하면서 이와 함께 자연어 처리도 함께 빠르게 성장하고 있다. 챗봇이 공공 사이트 등에서도 유용하게 사용되면서 관련 연구가 활발하게 진행되고 있다.

이렇듯 자연어 처리와 관련된 분야는 어떻게 진행이 되는지 알아보기 위해서 연구에 참여하게 되었다. 제품들의 리뷰를 수집해서 데이터 처리 과정을 거쳐 소비자들이 제품을 사려고 할 때 한눈에 보기 쉬운 결과를 제공하고자 한다.

1.2 연구목표

어떤 제품을 구매하려고 할 때 사용자들이 가장 많이 보는 것은 리뷰이다. 사용자는 소셜미디어, 영화평, 상품평 등 여러 플랫폼을 통해 자신의 의견과 견해를 표현한다. 하지만 네이버 쇼핑에 들어가보면 어떤 제품은 리뷰가 별로 없지만 인기가 많고 유명한 제품들은 리뷰가 많게는 몇 만개까지 존재한다. 실질적으로 이렇게 많은 리뷰들을 읽어 보기에는 시간도 낭비되고 어려움이 존재한다. 따라서 늘어나는 리뷰들을 효과적으로 분석하고 사용자의 편의에 보다 최적화된 환경을 제공하기 위해 감성분석의 활용이 필요하다.

감성분석은 수집한 데이터가 긍정적인지 부정적인지를 판단하는 방법이다. BERT 라는 인공지능 모델을 사용하여 수집한 데이터 간의 유사도를 판단하고 이를 기반으로 감성 수치를 계산하여 주어진 텍스트의 긍정 또는 부정 여부를 판단한다. 리뷰를 감성분석 하고나서 분석된 데이터를 가지고 사용자가 구매하고자 하는 제품의 긍정적인 리뷰와 부정적인 리뷰의 비율을 시각화해서 보여주는 것이 이번 연구의 목표이다.

2. 관련연구

2.1 데이터 전처리

데이터 전처리는 특정 분석에 적합하도록 데이터를 가공하는 작업이다. 데이터 분석과정에서 데이터 전처리는 반드시 거쳐야 하는 필수적인 과정이며 보다 효율적인 기계학습을 위해서는 데이터의 전처리 과정이 필요하다. 데이터 전처리는 데이터 정제, 결측값 처리, 이상값 처리, 분석변수처리 순서대로 진행된다. 데이터 전처리 과정에서는 데이터 노이즈 제거, 결측값 제거 등이 있다. 데이터 전처리의 기법으로는 데이터 벡터화, 데이터 차원 감소, 데이터 결합 등이 있다.

2.2 자연어 처리(NLP)

자연어는 우리가 일상 생활에서 사용하는 언어를 말한다. 자연어 처리(NLP)는 컴퓨터가 인간의 언어를 이해하고 해석하며 조작하도록 돕는 인공지능의 한 분야이다. 즉, 자연어

처리는 우리가 사용하는 언어의 의미를 분석해서 컴퓨터가 처리할 수 있도록 하는 일이다. 자연어 처리는 음성 인식, 번역, 감성 분석, 텍스트 분류 작업, 챗봇 등에 사용된다.
자연어 처리 모델로는 BERT, KoBERT, T5, GPT 등이 있다.

2.2.1 BERT

BERT 는 Bidirectional Encoder Representations from Transformers 이다. 구글에서 개발한 자연어 처리 사전 훈련 모델이다. BERT 의 사전 훈련 방법은 2 가지이다. 첫번째는 Masked Language Model 이고 두번째는 Next Sentence Prediction 이다. MLM 은 양방향성을 가지고 언어 모델을 학습하는 것이다. 사전 훈련을 위해서 입력 텍스트의 15%를 랜덤으로 마스킹한다. 이후 마스킹된 단어들을 예측하는 것이다. NSP 는 입력 값이 다음 문장도 함께 추가해서 다음 문장인지의 여부를 이진 분류하는 것이다. 학습을 위해서 50%는 연결된 문장, 50%는 랜덤하게 뽑힌 문장으로 학습을 한다.

2.2.2 T5

T5 모델은 "Text-to-Text Transfer Transformer"이다. T5 는 입력과 출력이 항상 텍스트 문자열인 통합된 텍스트-텍스트 형식이다. 전이 학습을 하기 위해서 레이블이 되어 있지 않는 데이터 세트가 필요하다. 이 데이터 세트는 C4(Colossal Clean Crawled Corpus)라 불리고 C4 데이터 세트와 T5 모델을 결합한 결과 다양한 태스크에 도입될 수 있고 성능이 뛰어나다.

2.2.3 GPT

GPT 는 Generative Pre-trained Transformer 이다. 한 토큰이 들어오면 다음에 올 토큰을 생성하는 언어 모델이다. GPT 는 별도의 추가적인 데이터를 사용해서 학습을 하지 않는다. 기존에 사전 학습된 지식만을 가지고 감성 분석이나 SQuAD 와 같은 태스크를 할 수 있다. GPT 모델의 구조는 Transformer Decoder 에서 착안되었다. 현재는 GPT3 까지 확장되었는데 GPT3 의 가장 큰 특징은 Few-shot learning 이다. Few-shot learning 은 적은 데이터로 데이터 특징을 파악하는 것이다.

2.2.4 KoBERT

BERT 모델은 문맥 특성을 활용하고 있고, 대용량 말뭉치로 사전 학습이 이미 진행되어 언어에 대한 이해도도 높다. 하지만 BERT 는 한국어에 대해서 영어보다 정확도가 떨어진다고 한다. KoBERT 모델은 SKTBrain 에서 공개한 모델이다. 한국어로 작성된 위키 5 백만 문장과 한국어 뉴스 2 천만 문장을 학습한 모델이다. 자신의 사용 목적에 따라 파인튜닝이 가능하기 때문에 output layer 만을 추가로 달아주면 원하는 결과를 출력해낼 수 있다. 무엇보다도 KoBERT 는 한국어에 대해 많은 사전 학습이 이루어져 있고, 감정을 분석할 때, 긍정과 부정만으로 분류하는 것이 아닌 다중 분류가 가능한 것이 강점이다.

3. 프로젝트 내용

3.1 네이버 리뷰 감성 분석

3.1.1 웹 크롤링

이유식, 분유, 유아용 건강용품 등 제품들을 네이버 쇼핑에서 검색해본다. 제품의 이름, 전체 리뷰 수, 사용자 평점, 리뷰 본문 등의 정보를 가져온다. 파이썬의 BeautifulSoup 모듈을 통해서 정보를 가져오고 최종적으로 csv 파일로 저장한다.

3.1.2 데이터 전처리

3.1.1 에서 수집한 리뷰 데이터들을 분석이 가능한 형태로 전처리한다. 리뷰 데이터들은 레이블을 별도로 가지고 있지 않기 때문에 따로 레이블을 부여해준다. 학습 데이터와 테스트 데이터를 8:2 비율로 나누어 준다. 데이터들의 중복 값을 제거해주고 Null 값도 제거해준다. 토큰화를 수행해서 필요 없는 토큰들은 제거를 한다.

3.1.3 감성분석

3.1.2 에서 전처리 한 데이터를 바탕으로 긍정적인 리뷰에서 자주 나오는 단어와 부정적인 리뷰에서 나오는 단어들을 확인한다. 텍스트로 되어있는 리뷰를 숫자로 처리할 수 있도록 정수 인코딩을 수행한다. 데이터의 평균 길이 분포를 알아보고 기준을 세워 데이터의 길이를

조정한다. BERT 모델을 통해서 데이터들을 학습시킨다. 학습이 끝난 후에는 테스트 데이터를 이용해 모델의 정확도를 측정한다.

3.1.4 데이터 시각화

3.1.3 에서 분석한 결과를 이용해서 결과를 시각화 한다. 문서 내 텍스트가 나타내는 단어와 문맥을 기반으로 감정 수치를 계산하여 긍정적인 리뷰 또는 부정적인 리뷰를 결정한다. 긍정, 부정적인 리뷰의 비율이 어떻게 나타나는지 차트를 이용해서 표현한다. 또한 긍정적인 리뷰에서 자주 나오는 단어들과 부정적인 리뷰에 주로 나오는 단어들을 표를 이용해 출력한다.

4. 구현 방법

4.1 데이터 전처리

4.1.1. 데이터

전처리에 사용한 데이터는 크롤링으로 뽑아온 아기 분유제품의 네이버 쇼핑 리뷰 데이터를 사용하였다. 데이터의 개수는 약 1100 개를 뽑아왔으며 pandas 라이브러리를 사용하여 데이터를 읽어왔다. 읽어온 데이터는 df 로 저장하였다.

```
12 df = pd.read_csv('C:/Users/is5pm/Documents/workspace/output1.csv',encoding = 'utf-8')
13
14 #데이터 읽어오기
15 print(df)
```

```
   번호  평점                                                내용
0     1   5  물에 잘 녹고 거품도 많이 안 생겨서 배앓이가덜한거 같아요 유기농궁 분유가 안 맞아...
1     2   5  배송 박스도 엄청 깔끔하고, 유통기한도 아주 넉넉해서 좋아요:-) 물에도 잘녹고 아...
2     3   5  혼합수유할꺼라 생각하고 샀는데 모유가 생각보다 잘 나와서 많이 사버린것 같네요.₩n...
3     4   2  제발울아기에게잘맞아주길바랐는데 울 아기는 방귀끼기도힘들어하고 똥도잘못보고 봐도 녹변...
4     5   5  궁 먹다가 바꿨어요 바꾸는 단계 없이 그냥 바꿔봤는데 잘먹고 탈없이 지내네요 둘의 ...
5     6   4  큰애때부터 믿고 먹여온 매일 앱솔루트명작~ 조리원서도 잘 먹고 변도 잘보고 했기때문...
6     7   3  아이가 잘먹고 잘자요₩n분유가 진해서 물타서 먹는게 편해요₩n언제 땠는지 표시할수 ...
7     8   5  다른 사이트에서 상품소진으로 급하게 구매했네요.주말끼고 배송이라 늦어도 화요일 오겠...
```

4.1.2. 긍정 부정 라벨링

위의 데이터 프레임 df 에서 평점을 기준으로 4 점 이상이면 긍정(1), 3 이하이면 부정(0)으로 라벨을 붙였다. 평점과 라벨을 붙인 데이터만 따로 추출해서 데이터 프레임 df2 에 저장하여 정리하였다. 기존 df 의 뒤에 column 을 추가하려니 리뷰 내용때문에 가독성이 떨어지는 것을 방지하기 위해 데이터 프레임을 분리하였다.

	평점	label
0	5.0	1
1	5.0	1
2	5.0	1
3	2.0	0
4	5.0	1
5	4.0	1
6	3.0	0
7	5.0	1
8	5.0	1
9	5.0	1
10	5.0	1
11	5.0	1
12	5.0	1
13	5.0	1
14	5.0	1
15	4.0	1
16	5.0	1

4.1.3. 토큰화

Null 값이 있는지 확인하고 한글과 공백을 제외한 문자를 모두 제거한다. 제거 후, 다시 Null 값이 있는지 확인한 후에 KoNLPy 에서 제공하는 형태소 분석기인 Okt 를 사용하여 형태소 단위로 리뷰 내용을 분리하여 df3 에 저장한다. 한국어를 토큰화 할 때는 영어처럼 띄어쓰기를 기준으로 토큰화를 하는 것이 아니라 주로 형태소 분석기를 사용해서 토큰화를 하기 때문에 Okt 를 사용하였다.

In [8]:
```
#Null data 있는지 확인
print(df.isnull().values.any())
```

```
False
```

In [10]:
```
# 한글과 공백을 제외하고 모두 제거
df['내용'] = df['내용'].str.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣 ]","")
print(df)
```

	번호	평점	내용
0	1	5	물에 잘 녹고 거품도 많이 안 생겨서 배앓이가덜한거 같아요 유기농궁 분유가 안 맞아...
1	2	5	배송 박스도 엄청 깔끔하고 유통기한도 아주 넉넉해서 좋아요 물에도 잘녹고 아기가 먹...
2	3	5	혼합수유할꺼라 생각하고 샀는데 모유가 생각보다 잘 나와서 많이 사버린것 같네요아기도...
3	4	2	제발울아기에게잘맞아주길바랐는데 울 아기는 방귀끼기도힘들어하고 똥도잘못보고 봐도 녹변...
4	5	5	궁 먹다가 바꿨어요 바꾸는 단계 없이 그냥 바꿔봤는데 잘먹고 탈없이 지내네요 둘의 ...
5	6	4	큰애때부터 믿고 먹여온 매일 앱솔루트명작 조리원서도 잘 먹고 변도 잘보고 했기때문에...
6	7	3	아이가 잘먹고 잘자요분유가 진해서 물타서 먹는게 편해요언제 땠는지 표시할수 있어서 ...
7	8	5	다른 사이트에서 상품소진으로 급하게 구매했네요주말끼고 배송이라 늦어도 화요일 오겠거...
8	9	5	배송 진짜 빠르네요캔이여서 무지 무거울지 알았는데 가루라서 그닥 무겁지 않았구요포장...
9	10	5	우선 제가 시분 주문 하고 결재를 해서 당일 출고가 가능 한지 걱정 많이 했었는데 ...

In [11]:
```
#다시 Null값 있는지 확인
print(df.isnull().values.any())
```

```
False
```

In [6]:
```
stopwords = ['의','가','이','은','들','는','좀','잘','걍','과','도','를','으로','자','에','와','한','
okt = Okt()


df3 = []
for sentence in tqdm(df['내용']):
    tokenized_sentence = okt.morphs(sentence, stem=True) # 토큰화
    stopwords_removed_sentence = [word for word in tokenized_sentence if not word in stopwords] # 불
    df3.append(stopwords_removed_sentence)
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 1100/1100 [00:16<0
0:00, 65.09it/s]
```

In [25]:
```
print(df3)
```

```
기', '방귀', '끼', '기도', '힘들다', '똥', '못', '보다', '보다', '녹변', '이네', '요', 'ㅠ', 'ㅜn', '매
일유업', '꺼', '먹이다', '아기', '힘들다', '결국', '예전', '먹이다', '로', '돌아가다', 'ㅜn', '분유',
'애바애', '니까', '잘맞다', '아기', '많다', 'ㅜn', '우리', '애', '한테', '안', '맞다', '아쉽다', '뿐',
'이에요', 'ㅜn', '배송', '이나', '포장', '상태', '다', '좋다'], ['궁', '먹다', '바꿨어', '요', '바꾸
다', '단계', '없이', '그냥', '바꾸다', '봣', '늘다', '먹다', '탈', '없이', '지내다', '둘', '차이', '유
기농', '여부', '차이', '고', '나머진', '다', '똑같다', '그래서', '자다', '넘다', '갓', '나', '보다',
```

4.1.4. 정수 인코딩

기계가 텍스트를 숫자로 처리할 수 있도록 테스트 데이터에 정수 인코딩을 수행하였다. 테스트 데이터에 대하여 단어 집합을 만들어보면, 단어 집합이 생성되는 동시에 각 단어에 고유한 정수가 부여되었음을 확인할 수 있다. 1100 개의 데이터에 총 3310 개의 고유한 정수가 부여되었다.

In [7]:
```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(df3)
print(tokenizer.word_index)
```

```
{'ㅜn': 1, '.': 2, '좋다': 3, '바르다': 4, '아기': 5, '있다': 6, '자다': 7, '~': 8, '쓰다': 9, '보다': 1
0, '같다': 11, '구매': 12, '먹다': 13, '태열': 14, '사용': 15, '때': 16, '배송': 17, '써다': 18, '!': 1
9, '피부': 20, '세': 21, '요': 22, '제품': 23, '험': 24, '로': 25, '없다': 26, '되다': 27, '주문': 28,
'것': 29, '을': 30, '분유': 31, '가격': 32, '..': 33, '않다': 34, '너무': 35, '다': 36, ',': 37, '이다':
38, '만': 39, '올라오다': 40, '크림': 41, '안': 42, '로션': 43, '에서': 44, '빠르다': 45, '먹이다': 46,
'촉촉하다': 47, '아이': 48, '^^': 49, '더': 50, '많이': 51, '얼굴': 52, '효과': 53, '보습': 54, '사다':
55, '부터': 56, '받다': 57, '신생아': 58, '아가': 59, '발라': 60, '저렴하다': 61, '향': 62, '조리': 63,
'거': 64, '고': 65, '원': 66, '계속': 67, '쁘리마쥬': 68, '오다': 69, '다른': 70, '맞다': 71, '인데': 7
2, '이라': 73, '만족하다': 74, '까지': 75, '하고': 76, '지금': 77, '구입': 78, '건조하다': 79, '바꾸다':
80, '보솜': 81, '되어다': 82, '들다': 83, '!!': 84, '흡수': 85, '시키다': 86, '기저귀': 87, '저': 88,
```

빈 샘플들은 어떤 레이블이 붙어있던 의미가 없으므로 빈 샘플들을 제거해주고 각 샘플들의 길이를 확인하여 길이가 0 인 샘플들의 인덱스를 받아왔다. len(df3)를 출력하여 확인한 결과, 처음과 달라진 개수가 없어 제거된 항목이 없다는 것을 확인할 수 있었다.

In [19]:
```
# 빈 샘플들을 제거
drop_ = [index for index, sentence in enumerate(df3) if len(sentence) < 1]
df3 = np.delete(df3, drop_, axis=0)
print(len(df3))
print(df3)
```

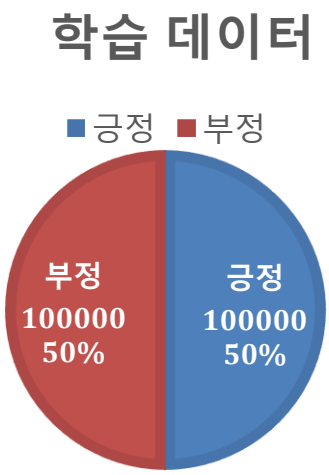
```
1100
[list(['물', '자다', '녹다', '거품', '많이', '안', '생기다', '배앓이', '덜하다', '같다', '유기농', '궁',
'분유', '안', '맞다', '변비', '오다', '산양', '이냐', '명작', '이냐', '고민', '걸', '로', '바꾸다',
'첨', '엔', '영양면', '에서', '맘', '들다', '않다', '아기', '변', '보다', '소아과', '에서', '너무', '자
다', '크다', '말', '그냥', '정착', '이다', '가격', '면', '에서도', '저렴하다', '가장', '많이', '먹이다',
'분유', '라', '던데', '이유', '있다'])
 list(['배송', '박스', '엄청', '깔끔하다', ',', '유통', '기한', '아주', '넉넉하다', '좋다', ':-)', '물',
'에도', '녹다', '아기', '먹다', '배탈', '없다', '맛있다', '먹다', '제', '기분', '좋다', '❤', '모유수
유', '못', '상황', '이라', '속상하다', ',', '매일', '인지도', '답', '게', '정말', '깨끗하다', '좋다',
'.', '남다', '분유', '기간', '동안', '단계', '별', '답', '게', '건강하다', '먹이', '고', '싶다', '.'])
 list(['혼합', '수유', '생각', '하고', '사다', '모유', '생각', '보다', '자다', '나오다', '많이', '사',
'버리다', '같다', '.', '₩n₩n', '아기', '자다', '먹다', ',', '남편', '맛있다', '(', '분유', '가루', '프
림', '맛임', 'ㅋㅋㅋㅋ', ')', '₩n', '유통', '기한', '길다', '좋다', '.', '근데', '개봉', '후', '3', '주',
'만에', '이용', '한다는', '걸', '미리', '알다', '400', 'g', '짜다', '살껄', '그렇다', '!'])
 ...
```

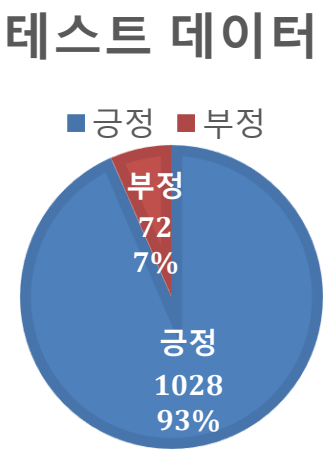
4.2 BERT 모델 개발

4.2.1. 데이터

학습 데이터로는 긍정(1), 부정(0) 라벨링이 이미 되어있는 네이버 영화 리뷰 데이터를 사용하였다. 학습 데이터는 200,000 개의 데이터를 사용하였으며 데이터의 분포는 긍정 50%, 부정 50%로 이루어져 있다. 학습 데이터에서 id 는 필요하지 않기 때문에 document 와 label 만 추출해서 사용하였다.

테스트 데이터는 직접 크롤링한 데이터를 사용했으며 크롤링한 데이터에는 라벨링이 되어있지 않기 때문에 긍정(1), 부정(0) 라벨을 붙여서 사용했다. 테스트 데이터는 총 1100 개였고 그 중에서 긍정이 1028 개로 93%를 차지하고 있었고 부정은 72 개로 7%를 차지하고 있었다.

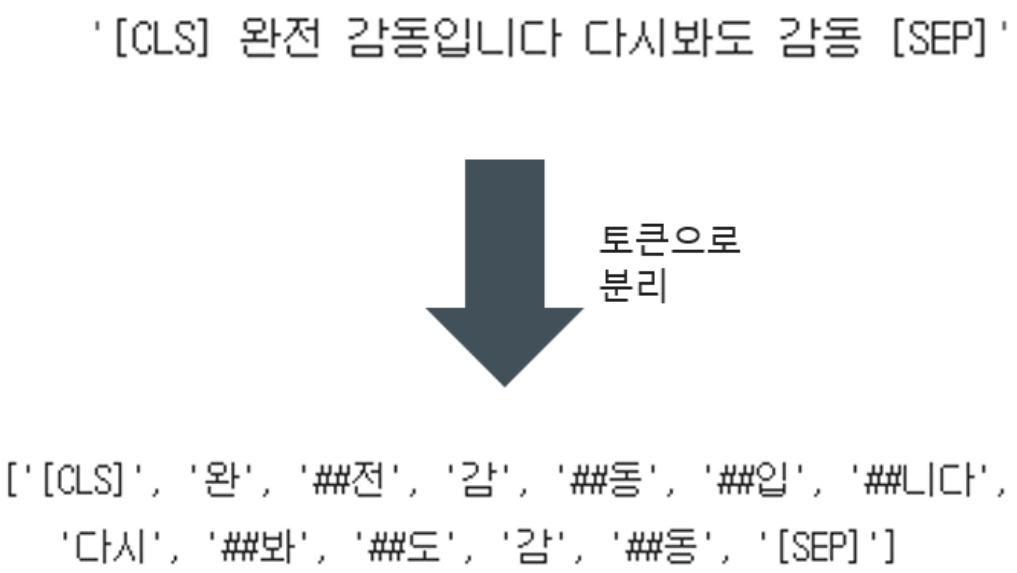




	id	document	label
0	8112052	어릴때보고 지금다시봐도 재밌어요ㅋㅋ	1
1	8132799	디자인을 배우는 학생으로, 외국디자이너와 그들이 일군 전통을 통해 발전해가는 문화산...	1
2	4655635	폴리스스토리 시리즈는 1부터 뉴까지 버릴께 하나도 없음.. 최고.	1
3	9251303	와.. 연기가 진짜 개쩔구나.. 지루할거라고 생각했는데 몰입해서 봤다.. 그래 이런...	1
4	10067386	안개 자욱한 밤하늘에 떠 있는 초승달 같은 영화.	1
5	2190435	사랑을 해본사람이라면 처음부터 끝까지 웃을수 있는영화	1
6	9279041	완전 감동입니다 다시봐도 감동	1
7	7865729	개들의 전쟁2 나오나요? 나오면 1빠로 보고 싶음	1
8	7477618	굿	1
9	9250537	바보가 아니라 병 쉰 인듯	1

[학습 데이터]

4.2.2 데이터 전처리



리뷰의 본문을 BERT 의 입력 형식에 맞게 변환을 해주어야 한다. 리뷰의 맨 앞에는 [CLS]를 붙여주고 맨 뒤에는 [SEP]를 붙여준다. BERT 가 [CLS]를 인식하게 되면 문장의 맨 처음인 것을 인식하고 [SEP]를 인식하게 되면 문장의 끝인 것을 알게 된다. [CLS]는 Classification 을 뜻하고 [SEP]는 Separation 을 뜻한다. 나중에 [CLS]의 값을 사용해서 긍정인지 부정인지 분류를 하게 된다.

다음으로 BERT 의 토크나이저를 이용해서 문장을 토큰으로 분리해주어야 한다. BERT 는 WordPiece 라는 통계적인 방식을 사용해서 단어들을 하나의 토큰으로 만든다. 한 단어 안에서 자주 나오는 글자들을 붙여서 하나의 토큰으로 만들게 된다. WordPiece 방식을 사용하여 언어에 상관없이 토큰을 생성할 수 있다는 장점이 있다. 토크나이저로는 미리 사전으로 학습된 bert-base-multilingual-cased 를 사용하였다.

토큰이 BERT 의 단어 집합에 존재한다면 토큰을 분리하지 않는다. 하지만 토큰이 단어 집합에 존재하지 않는다면 해당 토큰을 서브 워드로 분리한 후 해당 토큰의 첫번째 서브 워드를 제외하고 나머지 서브 워드들은 앞에 '##'을 붙여서 토큰으로 분리한다.

위의 문장에서 '완전'이라는 단어를 보면 단어집합에 존재하지 않기 때문에 완전을 '완'과

'전'으로 나눈 것을 볼 수 있다. '완'은 첫번째 서브 워드이기 때문에 그대로 분리하고 '전'은 두번째 서브 워드이기 때문에 앞 글자와 이어져 있다는 의미에서 '##'이 붙어있는 것을 볼 수 있다.

```
array([   101,   9591,  16617,   8848,  18778,  58303,  48345,  25805,
       118990,  12092,   8848,  18778,    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])
```

[토큰을 숫자 인덱스로 변환]

```
[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, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0]
```

[어텐션 마스크]

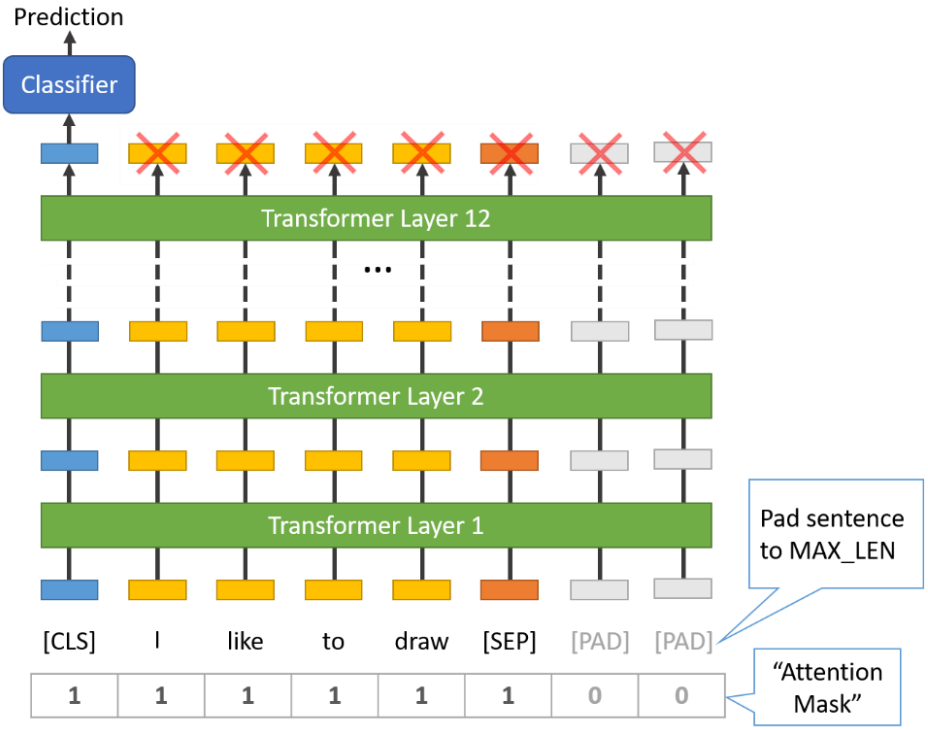
토큰을 자체적으로 입력으로 넣을 수 없기 때문에 숫자 인덱스로 변환해서 사용해야 한다. BERT 의 토크나이저는 {단어 토큰: 인덱스}로 구성된 단어사전을 가지고 있는데 이를 참조해서 토큰을 숫자 인덱스로 바꾸어 준다. 문장의 최대 길이를 정해서 문장이 아닌 뒷부분은 0 으로 패딩을 해준다. 예를 들면 [CLS]는 101 이라는 값과 [SEP]는 102 라는 고정된 값을 가지고 있다.

어텐션 마스크는 패딩이 아닌 것과 패딩인 것을 구분하기 위한 것이다. 패딩인 부분은 0 으로 설정하고 패딩이 아닌 부분은 1 이 된다. 0 값을 가지는 패딩 토큰에 대해서 BERT 모델이 불필요한 연산을 하지 않도록 단어와 패딩 토큰을 구분할 수 있도록 처리해 주는 것이다.

학습 데이터를 train 데이터와 validation 데이터로 분리해준다. 8 대 2 의 비율로 분리를 해주었고 BERT 는 입력 값으로 (리뷰 본문을 숫자 인덱스로 바꾼 값, 어텐션 마스크, 라벨 값) 형태를 가지기 때문에 위의 형태로 바꾸어 주어야 한다.

테스트 데이터는 라벨을 붙인 뒤에 토크나이저를 이용해서 학습 데이터와 마찬가지로 같은 과정으로 처리해주면 된다.

4.2.3 BERT 모델 생성

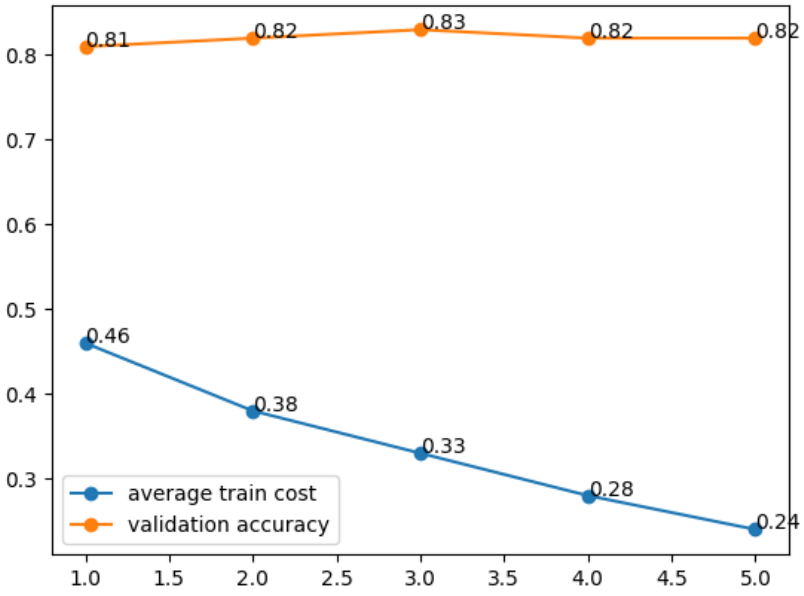


[Bert 모델 구조]

모델을 먼저 생성해야 하는데 BertForSequenceClassification 모듈을 사용하였고 사전 학습된 모델인 bert-base-multilingual-cased 를 사용하였다.

BERT 는 위와 같이 12 개의 Transformer Layer 로 구성되어 있다. 각각의 Transformer 는 토큰 목록을 입력으로 받아서 동일한 수의 임베딩(기계가 이해할 수 있는 숫자의 나열인 벡터의 형태)을 생성한다. 다만 마지막 Transformer layer 에서는 첫번째 토큰인 [CLS]만 나오게 된다. [CLS]는 전체 문장을 분류하는 의미를 가지고 있으며 [CLS] 토큰이 분류 값(긍정, 부정) 중 하나가 되도록 학습하는 것이다. 마지막 출력 값은 loss 와 logit 두가지를 출력하게 된다. Logit 은 [0 이 될 확률, 1 이 될 확률]로 이루어져 있게 된다. 두 확률 중에서 큰 값으로 라벨을 가지게 된다.

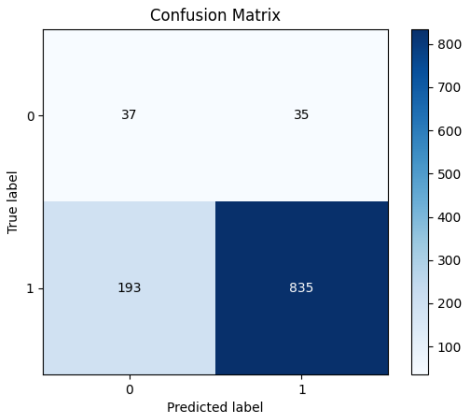
4.2.4 학습



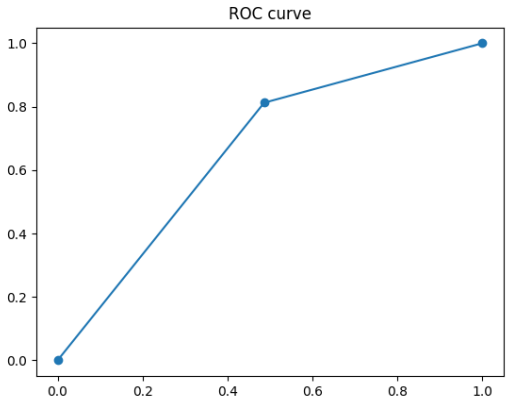
[train cost 의 평균과 validation accuracy]

4.2.3 에서 생성한 BERT 모델을 이용해서 train data 와 validation data 로 학습을 수행하였다. Optimizer 로는 AdamW 를 사용하였다. Epoch 를 5 로 설정하고 학습을 진행하였다. 평균 train cost 는 epoch 를 진행할 때 마다 46%, 38%, 33%. 28%, 24%로 점점 줄어 들은 것을 볼 수 있다. 하지만 validation 정확도를 측정해 본 결과 81%, 82%, 83%로 올라가다가 82%로 수렴하게 된 것을 볼 수 있다.

4.2.5 테스트



[Confusion matrix]



[ROC curve]

4.2.4 에서 학습한 BERT 모델을 이용해서 테스트 데이터로 테스트를 해보았다. 정확도는 79%로 나왔다. 테스트 데이터가 대부분 긍정 라벨로 이루어져 있어서 제대로 라벨을 분리할 수 있나 판단해보기 위해 Confusion Matrix 를 그려보았다.

부정(0) 데이터는 총 72 개로 이루어져 있는데 그 중에서 제대로 부정(0)으로 예측한 개수는 37 개였고 긍정(1)로 잘못 예측한 개수는 35 개였다. 이를 통해 부정(0)을 맞힐 확률은 51%인 것을 알 수 있었다.

긍정(1) 데이터는 총 1028 개로 이루어져 있었는데 제대로 긍정(1)으로 예측한 개수는 835 개였고 부정(0)으로 잘못 예측한 개수는 193 개였다. 이를 통해서 긍정을 맞힐 확률은 81%임을 알 수 있다.

검사의 정확도를 평가하기 위해서 ROC curve 도 그려보았다. ROC curve 는 검사도구의 유용성을 판단하거나 검사의 정확도를 평가하는데 사용된다. 또한 개발을 하면서 기준점을 설정하는 경우에도 활용된다. ROC curve 는 민감도(Sensitivity)와 1 – 특이도(Specificity)로 그려지는 곡선을 의미한다. 민감도(Sensitivity)는 실제 긍정(1) 라벨 중에서 긍정(1)을 맞춘 비율을 의미하고 특이도(Specificity)는 실제 부정(0) 라벨 중에서 부정(0)을 맞춘 비율을 의미한다. AUC 는 Area Under Curve 로 ROC curve 밑의 면적을 의미한다. ROC curve 를 그린 후에 AUC 를 계산해본 결과 0.66 으로 나왔다.

5. 역할 분담

항목	담당자
데이터 전처리 BERT 모델 개발	박주영
데이터 전처리 데이터 시각화	이인서

6. 결론

Bert 모델을 이용해서 감성분석을 진행해보았다. 학습 데이터로는 네이버 영화 리뷰 데이터를 사용하였고 테스트 데이터로는 직접 크롤링한 데이터를 사용하였다. 처음에는 CPU 를 이용해서 데이터를 학습시켰지만 많은 시간이 소요되었다. 그래서 학교에서 GPU 를 대여받아서 학습을 시켰다. 200,000 개의 많은 데이터를 학습시키는데 30 분이라는 시간으로 단축되었다. 학습을 한 모델을 이용해서 테스트를 진행해 본 결과 79%라는 정확도가 나왔지만 테스트 데이터의 비율이 불균형 했기 때문에 다양한 방식을 이용해서 분석해보았다.

이번 연구는 사용자가 구매하고자 하는 제품의 긍정적인 리뷰와 부정적인 리뷰의 비율을 시각화해서 보여주는 것을 목표로 하였다. 실제로 제품을 구매하고자 하면 사용자들이 가장 많이 보는 것이 리뷰인데 긍정적인 리뷰와 부정적인 리뷰의 비율을 시각화해서 보여주게 된다면 사용자의 선택에 많은 도움을 줄 수 있을 것으로 생각된다.

7. 추후 연구 방향

학습데이터로는 네이버 영화 리뷰 데이터를 사용했는데 아무래도 영화 데이터와 제품의 리뷰 데이터의 키워드가 다르기 때문에 학습 데이터로도 제품의 리뷰 데이터를 사용하게 된다면 조금 더 좋은 성능을 가질 것으로 보인다. 또한 사용자가 직접 문장을 입력해서 해당 문장에 대한 감성분석 결과를 알려주는 기능을 추가하게 되면 좋을 것 같다.

8. 참고문헌

- 박상언 저, "딥러닝 중심의 자연어 처리 기술 현황 분석"
- Jacob Devlin, Ming-Wei Chang 저, " BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"
- 김동현, 유승언 저, "효율적인 기계학습을 위한 데이터 전처리"
- 김택현, 조단비, 이현영, 강승식, 원혜진 저, "Sentiment Analysis System by Using BERT Language Model"

[Appendix]

소스 코드 경로 및 파일구성

경로: https://github.com/Is-5pm/naverShopping_crawling_test

ratings.txt 파일: 학습 데이터로 사용한 네이버 영화 리뷰 데이터

output.csv 파일: 테스트 데이터로 사용한 직접 크롤링한 데이터

token.py 파일: 데이터 전처리, Bert 모델 이용해 감성분석한 소스 코드