Twitter Sentiment Analysis

WOOSONG UNIVERSITY

우송대학교 기업실습
김선경, 윤종서, 김윤정

Introduction

문제 정의

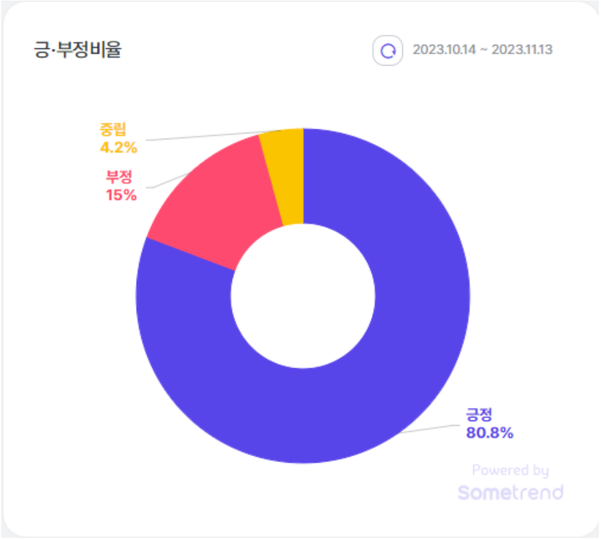
- 기업에서 고객 경험을 추적하기 어려움

  : 기업은 고객이 자사 제품과 서비스에 대해
  어떻게 생각하는지에 대한 실시간 인사이트를 얻고자 함
  (고객 경험을 추적하기 위해 별도의 설문조사나 인터뷰 필요)

해결

- 소셜 미디어 속 데이터를 감정분석을 하여 자사에 대한 인사이트 발굴

- EDA와 BERT(transformer endocer)를 통한 감성분석(프로젝트 목표)



▷ 썸트렌드의 감정분석 활용 사례

Introduction

데이터셋

	A	B	C	D	E	F
1	0	1467810369	Mon Apr 06 22:19:45	NO_QUERY	_TheSpecialOne_	@switchfoot http://t
2	0	1467810672	Mon Apr 06 22:19:49	NO_QUERY	scotthamilton	is upset that he can't
3	0	1467810917	Mon Apr 06 22:19:53	NO_QUERY	mattycus	@Kenichan I dived n
4	0	1467811184	Mon Apr 06 22:19:57	NO_QUERY	ElleCTF	my whole body feels
5	0	1467811193	Mon Apr 06 22:19:57	NO_QUERY	Karoli	@nationwideclass nc
6	0	1467811372	Mon Apr 06 22:20:00	NO_QUERY	joy_wolf	@Kwesidei not the w
7	0	1467811592	Mon Apr 06 22:20:03	NO_QUERY	mybirch	Need a hug
8	0	1467811594	Mon Apr 06 22:20:03	NO_QUERY	coZZ	@LOLTrish hey long
9	0	1467811795	Mon Apr 06 22:20:05	NO_QUERY	2Hood4Hollywood	@Tatiana_K nope th
10	0	1467812025	Mon Apr 06 22:20:09	NO_QUERY	mimismo	@twittera que me m
11	0	1467812416	Mon Apr 06 22:20:16	NO_QUERY	erinx3leannexo	spring break in plain
12	0	1467812579	Mon Apr 06 22:20:17	NO_QUERY	pardonlauren	I just re-pierced my e
13	0	1467812723	Mon Apr 06 22:20:19	NO_QUERY	TLeC	@caregiving I couldr
14	0	1467812771	Mon Apr 06 22:20:19	NO_QUERY	robrobbierobert	@octolinz16 It it cou
15	0	1467812784	Mon Apr 06 22:20:20	NO_QUERY	bayofwolves	@smarrison i would'

1048558	4	1960262728	Fri May 29 07:41:06 P	NO_QUERY	swansofnever	@shakeappeal Ha! N
1048559	4	1960262861	Fri May 29 07:41:07 P	NO_QUERY	whtshadowghost	when you get home
1048560	4	1960262864	Fri May 29 07:41:07 P	NO_QUERY	francescazurlo	@raginaphalange Ar
1048561	4	1960262865	Fri May 29 07:41:07 P	NO_QUERY	Jamakkronic	@elavoca you're pro
1048562	4	1960262880	Fri May 29 07:41:07 P	NO_QUERY	AbhorrentAspen	Were having an ice c
1048563	4	1960262920	Fri May 29 07:41:07 P	NO_QUERY	gypsyjenn	#followfriday @chov
1048564	4	1960262923	Fri May 29 07:41:08 P	NO_QUERY	butadream	@littlemissmessy Sou
1048565	4	1960262964	Fri May 29 07:41:08 P	NO_QUERY	kgs	@pollyalida lots of c
1048566	4	1960263086	Fri May 29 07:41:08 P	NO_QUERY	StaceRay	For my friends who r
1048567	4	1960263112	Fri May 29 07:41:08 P	NO_QUERY	wolle_world	home from the beac
1048568	4	1960263120	Fri May 29 07:41:08 P	NO_QUERY	victoriadunmire	Happy Friday everyor
1048569	4	1960263132	Fri May 29 07:41:09 P	NO_QUERY	JosephTexDozier	@jenciTN Why hello
1048570	4	1960263139	Fri May 29 07:41:09 P	NO_QUERY	cutegossipgirl	going to the citiy ol
1048571	4	1960263231	Fri May 29 07:41:09 P	NO_QUERY	VictoriaMystery	just in from bein at r

Kaggle의 ‘Sentiment 140 dataset with 1.6 million tweets’ 데이터셋

- 트위터 API를 사용하여 추출한 160만개의 트윗 데이터셋
- 총 6개의 필드로 구성(target, ids, date, flag, user, text)
- 감정에 따른 레이블링 (0 = negative, 2 = neutral, 4 = positive)

프로젝트 목차

첫 번째,
데이터 전처리 및 EDA(탐색적 데이터 분석)

두 번째,
최고 성능 모델 탐색

세 번째,
프로젝트 한계점

데이터 전처리

전처리

- URL/HTML 태그/멘션, 해시태그/구두점/이메일/숫자 등 영어 외 모든 문자 제거
- 대문자 -> 소문자처리
- 반복되는 문자 처리 (Yeeeeeeeeeeah, Coooooooooooool)
- 이모지 제거
- 토큰화
- 불용어 처리
- 표제어 추출(lemmatization)

전처리 – 토큰화

감정 분석에서의 “NOT”의 의미는 상당히 큼

that's not true, he doesn't believe it.

```
from nltk.tokenize import word_tokenize
print(word_tokenize(txt.lower()))
```

```
['that', "'s", 'not', 'true', ',', 'he', 'does', "n't", 'believe', 'it', '.']
```

```
from nltk.tokenize import WordPunctTokenizer
print(WordPunctTokenizer().tokenize(txt.lower()))
```

```
['that', "'", 's', 'not', 'true', ',', 'he', 'doesn', "'", 't', 'believe', 'it', '.']
```

전처리 – 토큰화

```
from nltk.tokenize import TweetTokenizer
print(TweetTokenizer().tokenize(txt.lower()))
```

```
["that's", 'not', 'true', ',', 'he', "doesn't", 'believe', 'it', '.']
```

```
from nltk.tokenize import RegexpTokenizer
tokenizer = RegexpTokenizer(pattern=r'\s+', gaps=True)
tokens = tokenizer.tokenize(txt.lower())

print(tokens)
```

```
["that's", 'not', 'true,', 'he', "doesn't", 'believe', 'it.']
```

전처리 – 토큰화

that's not true, he doesn't believe it.

라이브러리	that’s	doens’t	토큰 개수
word_tokenize	'that', "'s"	'does', "n't"	2
WordPuctTokenize	'that', "'", 's'	'doesn', "'", 't'	3
TweetTokenizer	"that's"	"doesn't"	1
RegexpTokenizer	"that's"	"doesn't"	1

전처리 – 토큰화

비형식적인 구어적 요소 + ‘ (apostrophe)를 임의적으로 생략

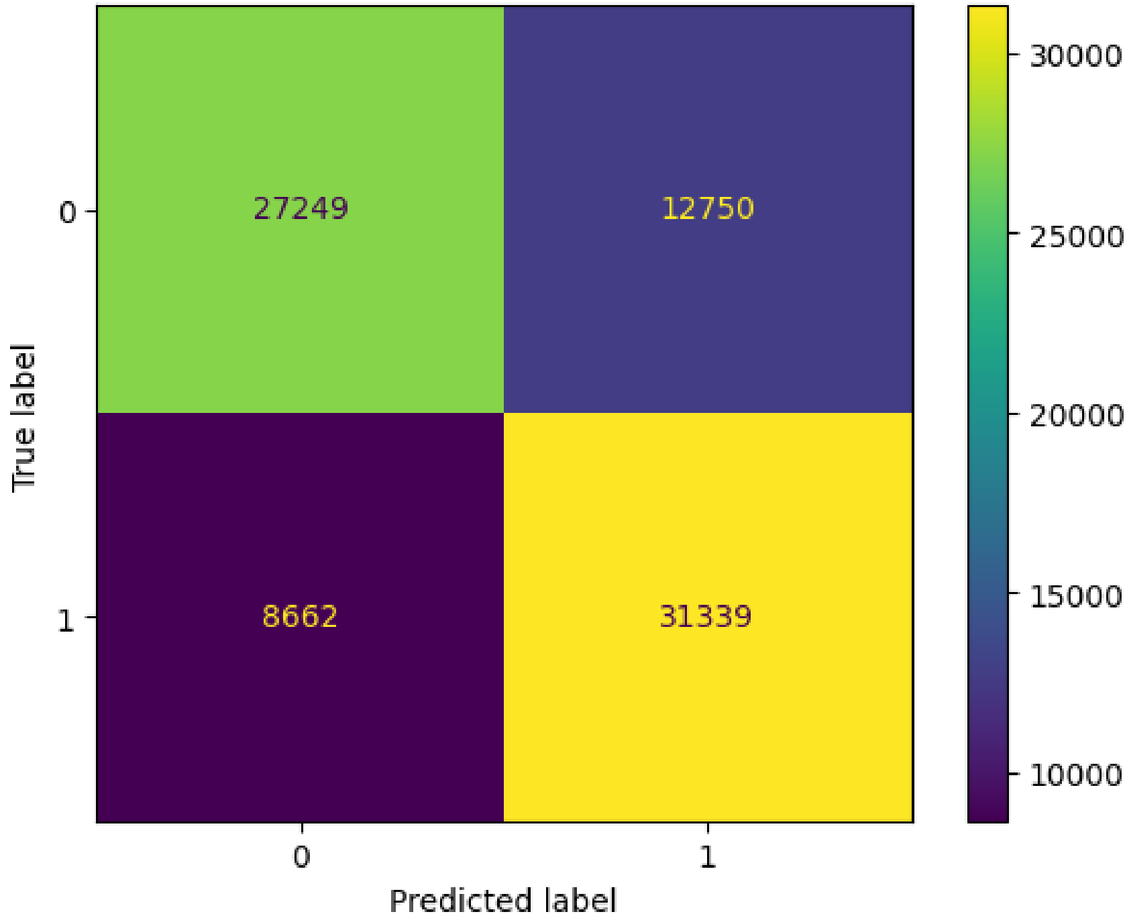
-> ‘isnt’은 ‘is’와 ‘not’의 두 개의 토큰으로, ‘dont’는 ‘do’와 ‘not’의 두 개의 토큰으로 토큰화

* “that’s”는 ‘that is’ 또는 ‘that has’의 의미를 가질 수 있지만 ‘is’와 ‘has’는 불용어(stopwords)로 처리되어 제거되기 때문에 우선은 일괄적으로 ‘that is’로 처리

```
contractions = {
    "isn't": "is not",
    "aren't": "are not",
    "wasn't": "was not",
    "weren't": "were not",
    "haven't": "have not",
    "hasn't": "has not",
    "hadn't": "had not",
    "won't": "will not",
    "wouldn't": "would not",
    "don't": "do not",
    "doesn't": "does not",
    "didn't": "did not",
    "can't": "cannot",
    "couldn't": "could not",
    "shouldn't": "should not",
    "mightn't": "might not",
    "mustn't": "must not",
    "mayn't": "may not",
    "isnt": "is not",
    "arent": "are not",
    "wasnt": "was not",
    "werent": "were not",
    "havent": "have not",
    "hasnt": "has not",
    "hadnt": "had not",
    "wont": "will not",
    "wouldnt": "would not",
    "dont": "do not",
    "doesnt": "does not",
    "didnt": "did not",
    "cant": "cannot",
    "couldnt": "could not",
    "shouldnt": "should not",
    "mightnt": "might not",
    "mustn't": "must not",
    "maynt": "may not",
```

EDA(탐색적 데이터 분석)

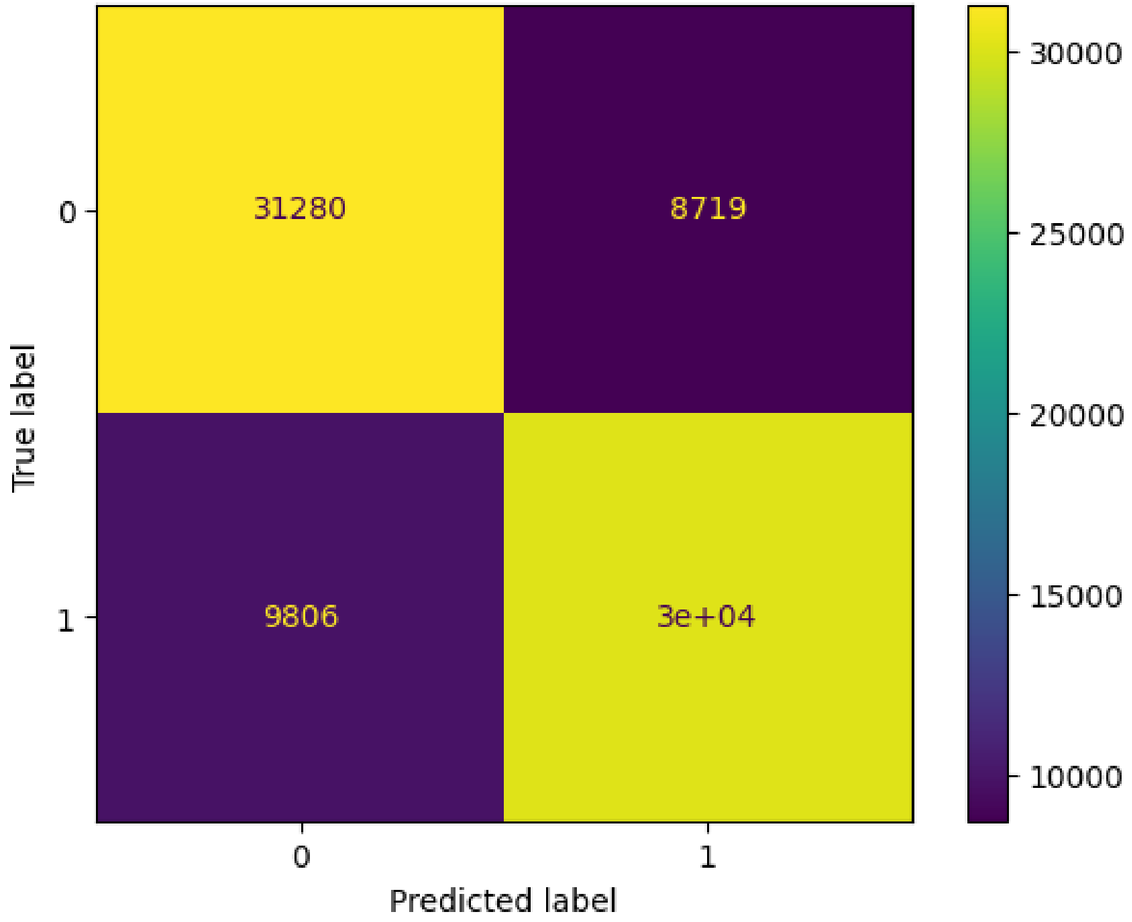
서포트 벡터 머신




```
1 svm_train = trainer.train_metrics
2 svm_val = trainer.eval_metrics
3 print(svm_train)
4 print(svm_val)
```

```
[0.7304884868421052, 0.7299111842105264, 0.7302384
[0.7306447368421053, 0.7301940789473684, 0.7291480
```

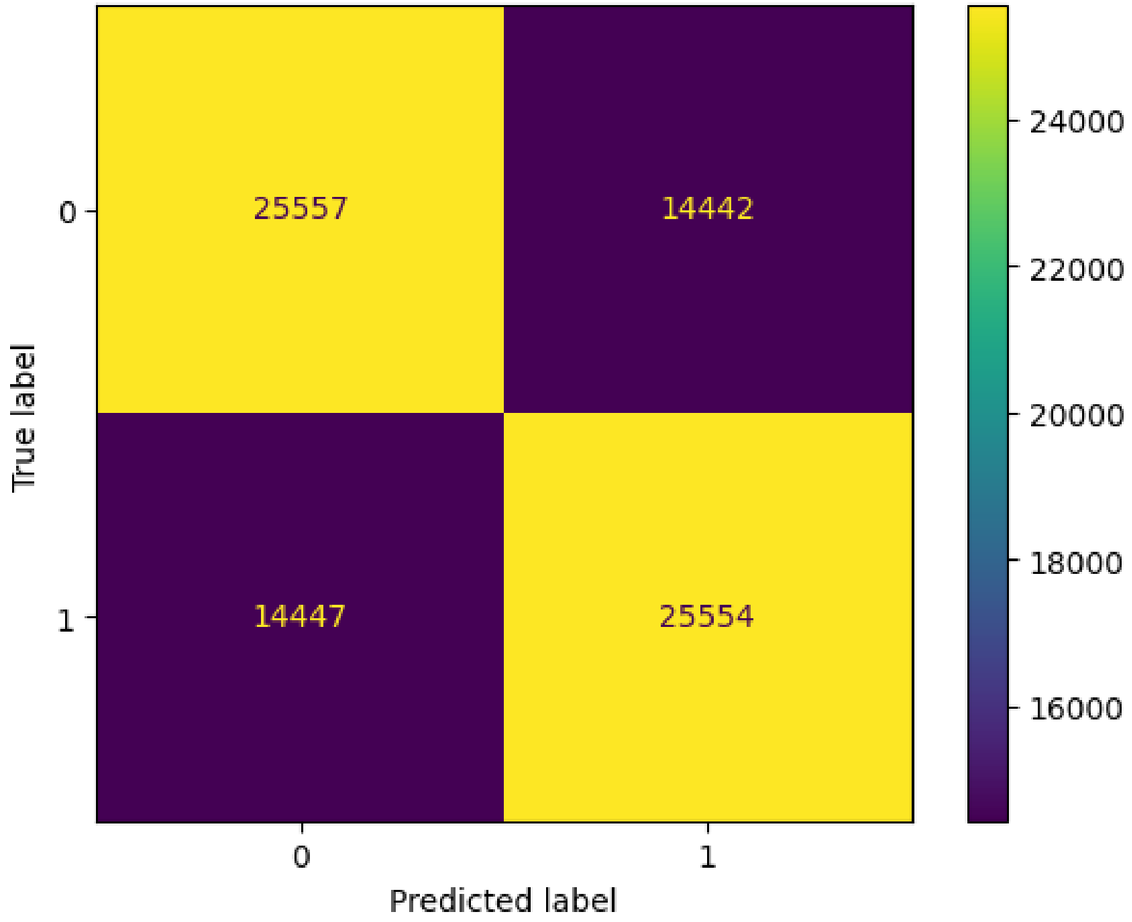
나이브 베이즈




```
1 nb_train = trainer.train_metrics
2 nb_val = trainer.eval_metrics
3
4 print(nb_train)
5 print(nb_val)
```

```
[0.792905427631578, 0.7935123355263158, 0.7931981907
[0.7657598684210526, 0.7652861842105263, 0.7652763157
```

Vader




```
1 v_train = trainer.train_metrics
2 v_val = trainer.eval_metrics
3
4 print(v_train)
5 print(v_val)
```

```
[0.6365929276315789, 0.6372730263157895, 0.636606
[0.6373881578947368, 0.6346677631578947, 0.637332
```

결과 분석

두 개의 클래스가 각각 80,000개로 클래스 불균형 크게 없음 -> 정확도가 전반적인 모델 성능을 평가하는 데 유용하다 판단

SVM의 평균 훈련 정확도: 0.7302666118421053
SVM의 평균 검증 정확도: 0.7298013157894736
SVM의 테스트 정확도: 0.73235

나이브 베이즈 모델 평균 훈련 정확도: 0.7931245065789474
나이브 베이즈 모델 평균 검증 정확도: 0.7658223684210527
나이브 베이즈의 테스트 정확도: 0.7684375

vader의 평균 훈련 정확도: 0.6367519736842105
vader의 평균 검증 정확도: 0.6367519736842105
vader의 평균 검증 정확도: 0.6388875

결과 분석

중립 클래스 부재의 문제
데이터에서 중립임에도 불구하고 실제로 긍정 또는 부정으로 레이블이 지정된 트윗이 있음
"오염된" 데이터로 인해 이 데이터로 달성할 수 있는 최대 분류 정확도에 상한선을 두게 될 것

과적합 가능성
훈련 데이터에서의 성능(0.79) > 검증 데이터에서의 성능(0.77)

정밀도(precision) 77%, 재현율(recall) 75%

결과 분석

나이브 베이즈 알고리즘 : 최종 테스트 정확도 약 77%에 대해

긍정적 트윗 30%

부정적 트윗 30%

중립적 트윗 40%

- 긍정적 트윗 20%
- 부정적 트윗 20%

가정:
긍정적인 트윗과 부정적인 트윗이 고르게 분포되어 있음
100% 올바르게 식별할수 있는 분류기를 사용

- 모든 긍정 및 부정 트윗을 올바르게 분류할 경우 60%의 정확도를 얻을 수 있으며(중립적인 트윗을 인식하거나 분류할 수 없기 때문) 중립적이지만 무작위로 긍정 or 부정으로 레이블된 나머지 40%의 트윗 중 약 50%를(20%) 정확하게 분류할 수 있음을 추측해볼 때, 달성할 수 있는 이론적 최대 정확도 = 60% + 20% = 80%

- 트윗의 40% 이상이 중립일 경우 최대 이론적인 정확도는 80%보다 낮을 것

- 나이브 베이즈 분류기의 77% 정확도는 데이터를 고려할 때 낮은 수치가 아니다.

결과 분석

예측과 다른 결과

서포트벡터머신의 특징 기출

- 지도학습에서의 과적합(Over Fitting) 회피
- 분류(종속변수 : 범주형), 회귀(종속변수 연속형)의 문제 활용
- 인공신경망 대비 사용이 쉬움
- 학습속도가 느리고 해석의 어려움 존재

SVM의 평균 훈련 정확도: 0.730266
SVM의 평균 검증 정확도: 0.729801
SVM의 테스트 정확도: 0.73235

훈련 정확도: 0.793124506
검증 정확도: 0.765822368
정확도: 0.7684375

나이브 베이즈 장점/단점

장점	단점
모델이 단순하고 계산속도가 빠름	모든 속성의 중요도를 동등하게 반영해서, 독립 가정이 잘못된 판단을 할 수 있음
노이즈와 결측 데이터에 강함	(예 : 일기예보에서 습도는 중요한 단어임에도 다른 특징과 동등하게 봄)
데이터의 크기에 상관 없이 잘 동작	연속형 독립변수가 많은 경우 이상적이지 않음
예측에 대한 추정확률 얻기가 쉬움	추정된 확률은 예측된 범주보다 덜 신뢰적

결과 분석

[데이문 _3편] 텍스트 마이닝 사용 설명서 (2) - 감성 분석

[VADER를 이용한 감성 분석]. 또 다른 Lexicon인 VADER Lexicon을 살펴보겠습니다. VADER는 소셜 미디어의 감성 분석 용도로 만들어진 룰 기반

파이썬(Python), VADER를 이용한 NLTK 뉴스 헤드라인의 감정 ...

2021. 7. 10. — 인터넷의 소셜 미디어의 문장의 분석하기 위해서 Vader를 많이 사용한다고 합니다. 그리고 Vader를 사용하면 문장의 긍정, 부정을 판단하는데 유용합니다.

감성 분석 (Sentiment Analysis) - 비지도학습 기반, VADER

2020. 11. 13. — 3. VADER: 주로 소셜 미디어의 텍스트에 대한 감성 분석을 제공하기 위한 패키지입니다. 뛰어난 감성 분석 결과를 제공하며, 비교적 빠른 수행 시간을 ...

[텍스트] 감성 분석 (Sentiment Analysis) - 엔지니어 한다운의 저널

2021. 11. 28. — VADER: 주로 소셜 미디어의 텍스트에 대한 감성 분석을 제공하기 위한 패키지. 빠른 수행 시간. 대용량 텍스트 데이터에 잘 사용된다. · Pattern : 파이썬 ...
IMDB의 영화 review에 대한 긍... · Pipeline을 통해 Count기반 피...

```
vader의 평균 훈련 정확도: 0.63675
vader의 평균 검증 정확도: 0.63675
vader의 평균 검증 정확도: 0.63888
```

정확도 가장 낮음

비지도 학습?
전처리 문제?

최고 성능 모델 탐색

NN

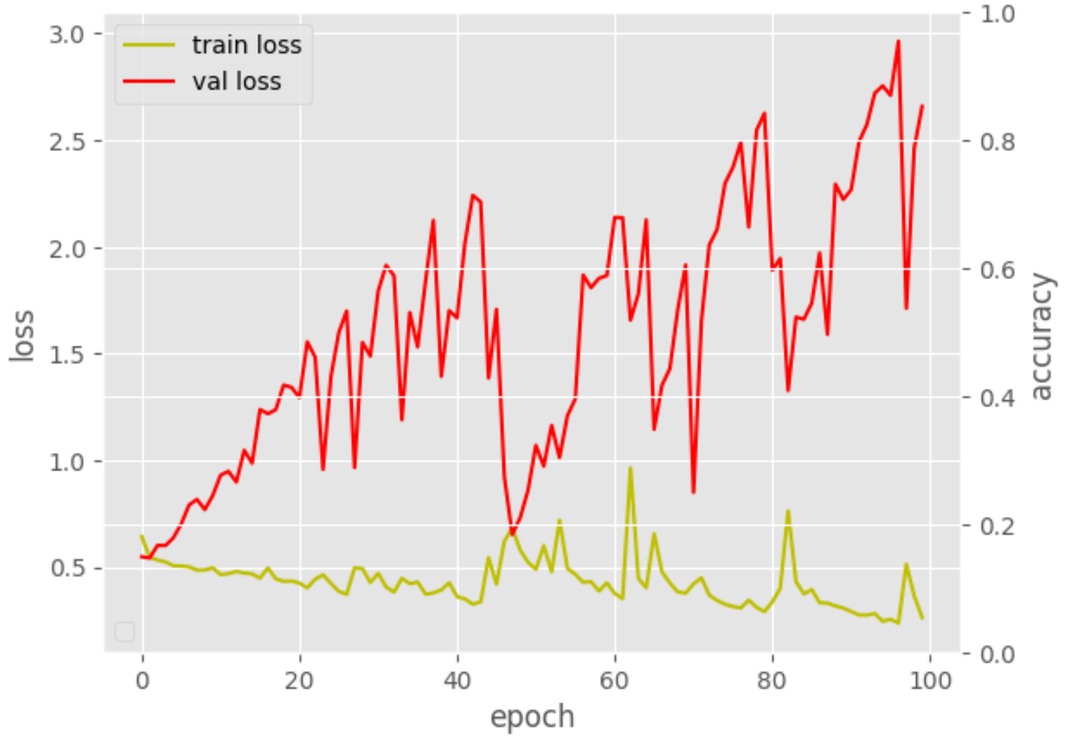
CNN

BERT

신경망(NN) 구현

NN(LSTM)

```
def tensorflow_based_model(): #Defined tensorflow_based_mode
ow based model
    inputs = Input(name='inputs',shape=[max_len])#step1
    layer = Embedding(2000,50,input_length=max_len)(inputs)
    layer = LSTM(64)(layer) #step3
    layer = Dense(256,name='FC1')(layer) #step4
    layer = Activation('relu')(layer) # step5
    layer = Dropout(0.5)(layer) # step6
    layer = Dense(1,name='out_layer')(layer) #step4 again bu
output as because we need to classify the tweet as positive c
    layer = Activation('sigmoid')(layer) #step5 but this tim
for only one output.
    model = Model(inputs=inputs,outputs=layer) #here we are
n the model for classification
    return model #function returning the value when we call
```



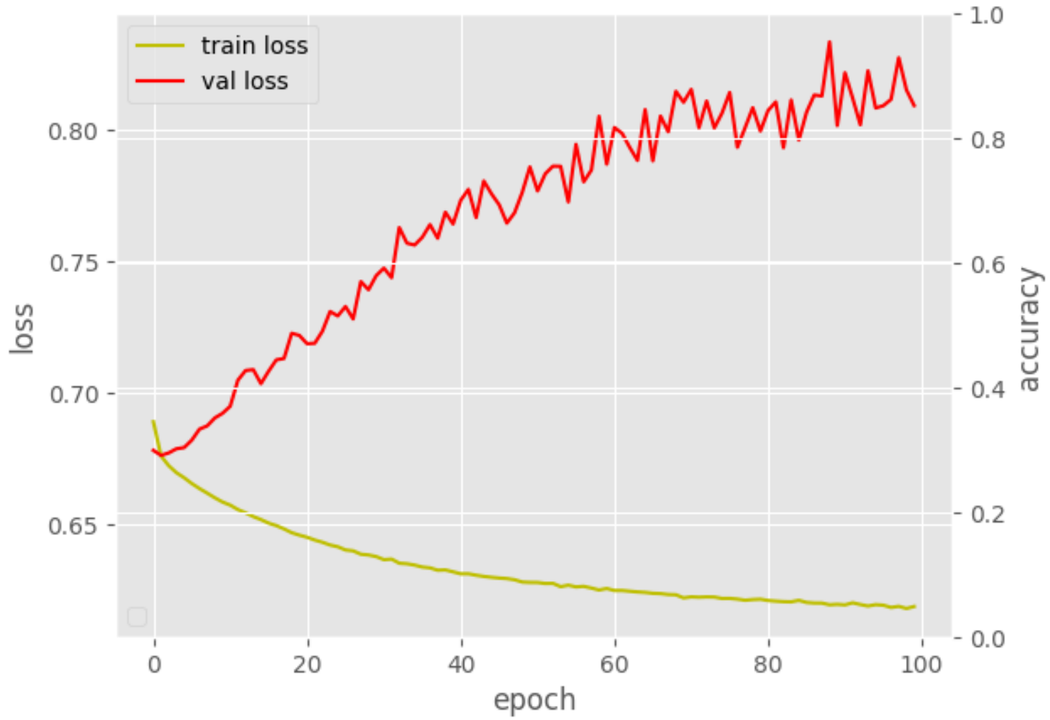
신경망(NN) 성능 개선 시도

```python
def tensorflow_based_model():
    inputs = Input(name='inputs',shape=[max_len])
    layer = Embedding(2000,50,input_length=max_len)(inputs)

    for i in range(20):
     layer = LSTM(64, return_sequences=True)(layer)
     layer = Dense(256, name='FC{}'.format(i + 1))(layer)
     layer = Activation('relu')(layer)

    layer = Dropout(0.5)(layer)
    layer = Dense(1,name='out_layer')(layer)
    layer = Activation('sigmoid')(layer)
    model = Model(inputs=inputs,outputs=layer)
    return model
```

```python
model = tensorflow_based_model() # here we are calling the function of created model
model.compile(loss='binary_crossentropy',optimizer=Adam(),metrics=['accuracy'])
```



초기 모델은 relu, sigmoid 활성화 함수와 RMSprop optimizer를 쓰고 있었다 → optimizer만 adam으로 변경

하지만 loss값이 처음부터 상승, 적합하지 않은 모델로 판단 → CNN으로 모델 변경

CNN

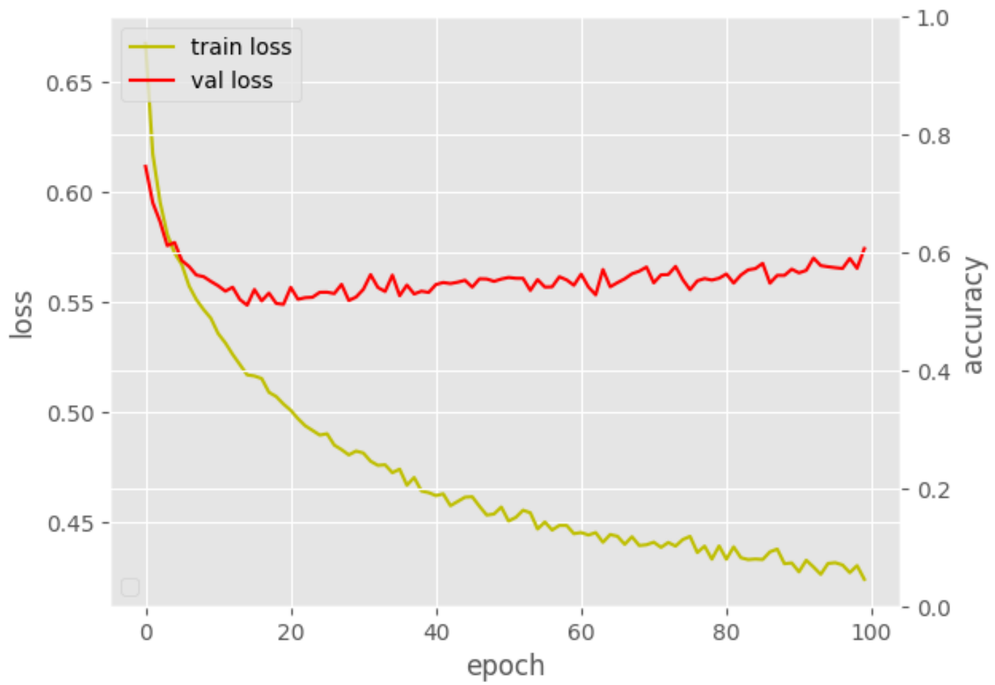
```
from keras import optimizers
from keras.layers import Dense, concatenate, Activation, Dropout
from keras.models import Model

branches = []
x = Dropout(0.2)(tweet_encoder)

for size, filters_count in [(2, 10), (3, 10), (4, 10), (5, 10)]:
    for i in range(filters_count):
        branch = Conv1D(filters=1, kernel_size=size, padding='valid', activation='relu')(x)
        branch = GlobalMaxPooling1D()(branch)
        branches.append(branch)

x = concatenate(branches, axis=1)
x = Dropout(0.2)(x)
x = Dense(30, activation='relu')(x)
x = Dense(1)(x)
output = Activation('sigmoid')(x)

model = Model(inputs=[tweet_input], outputs=[output])
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=[precision, recall, f1])
model.summary()
```



val loss가 더 이상 감소하지 않아, 적합하지 않은 모델로 판단 → BERT로 모델 변경

BERT

BERTweet-base-sentiment-anlaysis model

BERT model pretrain 방법

```
checkpoint = "finiteautomata/bertweet-base-sentiment-analysis"
tokenizer = AutoTokenizer.from_pretrained(checkpoint)
model = AutoModelForSequenceClassification.from_pretrained(checkpoint)
```

roBERTa 영어 모델을 tweet데이터 셋으로 pretrain 시킨 BERTweet을 SemEval 2017 corpus (around ~40k tweets)으로 훈련시킨 모델

BERT 모델 데이터 전처리 과정

- URL 삭제
- 반복 문자 삭제
- 숫자 삭제
- html 태그 삭제
- 이메일 삭제
- 특수문자 삭제
- 알파벳을 제외한 나머지 삭제

BERT

```
Epoch 1, Average training loss: 0.4226048976909369
Epoch 1, Average val loss: 0.41085616500675676
Epoch 1, Accuracy: 0.8201, F1_score: 0.8184
Epoch 2, Average training loss: 0.3444782948178472
Epoch 2, Average val loss: 0.39152287526149304
Epoch 2, Accuracy: 0.8462, F1_score: 0.8459
Epoch 3, Average training loss: 0.2623263947965267
Epoch 3, Average val loss: 0.4256190633050477
Epoch 3, Accuracy: 0.8480, F1_score: 0.8479
```

기존 : 각 레이블 별로 20000개씩 split

```
100% 30000/30000 [57:42<00:00, 9.07it/s]
Epoch 1, Average training loss: 0.38375679733213036
Epoch 1, Average val loss: 0.3603732812568545
Epoch 1, Accuracy: 0.8418, F1_score: 0.8418
Epoch 2, Average training loss: 0.3271825077954214
Epoch 2, Average val loss: 0.3584443811431527
Epoch 2, Accuracy: 0.8468, F1_score: 0.8468
Epoch 3, Average training loss: 0.2570558673171482
Epoch 3, Average val loss: 0.39396436628792436
Epoch 3, Accuracy: 0.8443, F1_score: 0.8443
```

변화 후 : 각 레이블 별로 50000개씩 split

데이터 input 양을 증가시켜 기존보다 정확도와 F1 score 향상

결론

- **EDA를 통한 이진분류로 감정분석(긍정, 부정)을 시도**
  : 데이터셋 자체의 한계(중립적 감정에 대한 레이블 없음) → 성능의 한계

- **신경망 구현을 통해 감정분석을 시도**
  : 처음부터 val loss가 증가 → layer 부족 판단 → layer 증가 → 성능개선 실패(0.03 증가)

- **기존 신경망 모델보다 성능이 좋은 모델 탐색**
  : CNN과 BERT →CNN의 val loss가 감소하지 않음 → 최종적으로 BERT 선택

- **최종적으로 정확도 0.76(EDA)에서 0.84(BERT)로 개선**

- **2주간 모델 탐색 및 성능 개선 노력**
  다음과 같은 한계점들을 느낌

프로젝트 한계점

한계점 및 느낀점

- BERT가 워낙 거대한 모델이다 보니 컴퓨팅 파워 부재 문제 발생
  → 만약 이걸 서비스화시킨다면 실시간성이 떨어지는 문제가 될 것 같았음

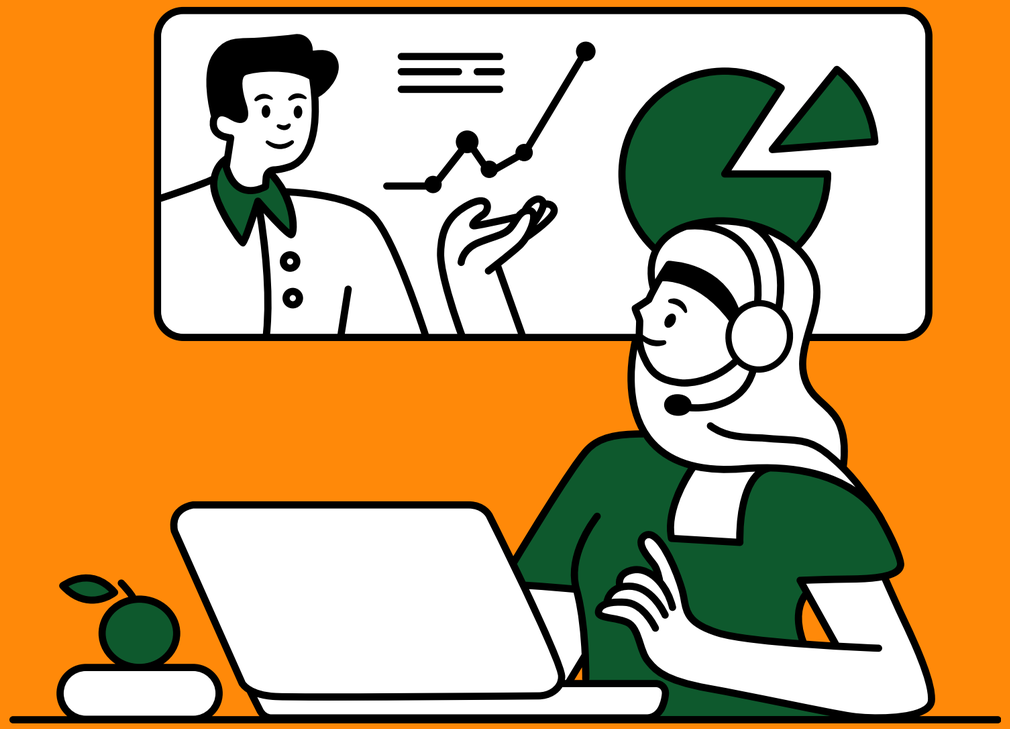
- 버전 관리의 미숙함 → Git을 통한 버전 관리 필요성

- 선경 : 생각보다 전처리 과정에서의 한계점이 많아서 아쉬웠고 생각보다 훨씬 많은 요소들을 하나하나 판단, 조정하고 분석하는 과정에서 확신이 들지 않아 어려웠다.

- 종서 : 컴퓨팅 파워에 한계가 있어서 하루에 돌릴수 있는양이 정해져 있다 보니까 한계를 느꼈습니다.

- 윤정 : 단순하게 성능 좋은 모델을 찾는 것보다 서비스에 활용시킬 수 있는 모델을 찾아 fine-tuning 하는 것이 쉽지 않을 것이라고 느껴졌다.

감사합니다



WOOSONG UNIVERSITY