Naive Bayes sentiment classification

과 목 자연어처리개론

전 공 컴퓨터공학과

학 번 2020112043

이 름 정수채

```python
import pandas as pd

df = pd.read_parquet("data/train-00000-of-00001.parquet")
print(df)
print("-----------------------------------")
print(df["sentence"][2])

# 감성 분석을 위한 최적화로 각 문서에서 단어가 중복되지 않게 변형
df['sentence'] = df['sentence'].apply(lambda x: ' '.join(set(x.strip().split(' '))))
```

✓ 5.6s

```
       idx                                           sentence  label
0        0        hide new secretions from the parental units       0
1        1               contains no wit , only labored gags       0
2        2  that loves its characters and communicates som...       1
3        3  remains utterly satisfied to remain the same t...       0
4        4  on the worst revenge-of-the-nerds clichés the ...       0
...    ...                                                ...     ...
67344  67344                            a delightful comedy       1
67345  67345                  anguish , anger and frustration       0
67346  67346  at achieving the modest , crowd-pleasing goals...       1
67347  67347                              a patient viewer        1
67348  67348  this new jangle of noise , mayhem and stupidit...       0

[67349 rows x 3 columns]
-----------------------------------
that loves its characters and communicates something rather beautiful about human nature
```

train dataset을 로드하고, 감성분석 최적화를 위해 각 단어가 중복되지 않도록 sentence를 변형한다.

```python
# 필요한 데이터인 sentence와 label만을 추출
all_corpus = df[['sentence', 'label']]

print(all_corpus)
```

✓ 0.0s

```
                                        sentence  label
0          hide secretions the units new parental from       0
1                  labored wit no only contains , gags      0
2      characters human and communicates rather loves...       1
3      remains same remain the utterly satisfied to t...       0
4      could dredge revenge-of-the-nerds on the clich...       0
...                                                  ...     ...
67344                          comedy delightful a       1
67345                 and anguish anger frustration ,       0
67346  sets modest it crowd-pleasing the for itself a...       1
67347                              viewer patient a      1
67348  must this . noise jangle serious contender and...       0

[67349 rows x 2 columns]
```

train을 위해 필요한 데이터인 sentence와 label만을 추출한다.

```python
import nltk
import math
from sklearn.feature_extraction.text import CountVectorizer

def train_naive_bayes(D, C):
    """
    N_doc: 전체 문서의 개수
    all_BoW: 전체 단어의 BoW
    len_all_BoW: 전체 단어의 수

    N_c: 클래스 문서의 개수
    """

    N_doc = len(D)
    all_vect = CountVectorizer(tokenizer=nltk.word_tokenize)
    all_BoW = all_vect.fit(D['sentence']) # 전체 단어에 대해 fit
    len_all_BoW = len(all_BoW.vocabulary_)

    value = {}
    for c in C:
        N_c = len(D[D['label'] == c])
        log_prior = math.log(N_doc/N_c)

        bigdoc = D[D['label'] == c]['sentence'].values.astype('U')  # 문장 리스트로 변환
        c_BoW = all_BoW.transform(bigdoc)
        len_c_BoW = c_BoW.nnz

        log_likelihood = {}
        for word, _ in all_BoW.vocabulary_.items():
            word_idx = all_BoW.vocabulary_.get(word)
            word_cnt = c_BoW[:, word_idx].sum()
            log_likelihood[word] = math.log((word_cnt + 1) / (len_c_BoW + len_all_BoW))

        value[c] = [log_prior, log_likelihood]

    """
    value: 클래스별 log_prior와 log_likelihood
    all_BoW: 전체 단어에 대한 BoW
    C: 분류한 클래스
    """
    return value, all_BoW, C
```
✓ 3.3s

naive bayes 모델을 훈련하기 위한 함수이다.
사용자로부터 전체 문서와 Class의 종류를 입력받아 클래스별 prior와 likelihood를 로그값으로 반환하고, 전체 단어에 대한 BoW 및 분류한 클래스를 반환한다.

클래스별 prior와 likelihood를 얻기 위해 sklearn의 CountVectorizer로 BoW를 구현했고, 이중 반복문을 사용해 각 클래스의 log prior와 각 단어에 대한 log likelihood를 얻었다.

이때 Add-1 smoothing 기법을 사용했다.

```python
model = train_naive_bayes(all_corpus, [0, 1]) # 모델 학습
```
✓ 1m 27.9s

train_naive_bayes()를 사용해 학습된 모델을 얻는다.

```python
import numpy as np

def test_naive_bayes(test_doc, model):
    value, V, C = model

    class_sum = []
    for c in C:
        sum = value[c][0] # log_prior
        words = nltk.word_tokenize(test_doc) # 학습할 때와 동일한 토크나이저 사용
        for word in words:
            if word in V.vocabulary_:
                sum += value[c][1][word] # log_likelihood
        class_sum.append(sum)

    return np.argmax(class_sum), class_sum
```
✓ 0.0s

naive bayes 모델을 테스트하기 위한 함수이다.

test 문서와 model을 입력받아, 분류한 클래스와 각 클래스의 합을 제공한다.

분류를 위해 test_doc을 토큰별로 나누어 words를 얻은 뒤, 각각의 log likelihood 값과 클래스의 log prior 값을 더한다.

이후 두 클래스의 sum 중, 더 큰 클래스를 반환한다.

```python
c, class_sum = test_naive_bayes("An exciting story... a movie is fantastic.", model)
print(f"classification result: {'negative' if c == 0 else 'positive'}\nnegative: {class_sum[0]} | positive: {class_sum[1]}")
```
✓ 0.0s

```
classification result: positive
negative: -49.81056025473585 | positive: -46.006449887415435
```

test_naive_bayes() 함수가 정상적으로 작동하는 모습이다.

```
df = pd.read_parquet("data/validation-00000-of-00001.parquet")
print(df)
print("-----------------------------------")
print(df["sentence"][2])

# 감성 분석을 위한 최적화로 각 문서에서 단어가 중복되지 않게 변형
df['sentence'] = df['sentence'].apply(lambda x: ' '.join(set(x.strip().split(' '))))
```

✓ 0.0s

```
     idx                                           sentence  label
0      0    it 's a charming and often affecting journey .      1
1      1                 unflinchingly bleak and desperate      0
2      2  allows us to hope that nolan is poised to emba...      1
3      3  the acting , costumes , music , cinematography...      1
4      4                  it 's slow -- very , very slow .      0
..   ...                                             ...    ...
867  867              has all the depth of a wading pool .      0
868  868              a movie with a real anarchic flair .      1
869  869  a subject like this should inspire reaction in...      0
870  870  ... is an arthritic attempt at directing by ca...      0
871  871  looking aristocratic , luminous yet careworn i...      1

[872 rows x 3 columns]
-----------------------------------
allows us to hope that nolan is poised to embark a major career as a commercial yet inventive filmmaker .
```

검증을 위해 validation dataset을 로드하고, training 단계에서 진행한 것처럼 각 단어가 중복되지 않게 변형한다.

```
df['prediction'] = df['sentence'].apply(lambda x: test_naive_bayes(x, model)[0])
print(df)
```

✓ 0.5s

```
     idx                                           sentence  label  prediction
0      0     often a journey charming . affecting 's it and      1           1
1      1                  unflinchingly desperate and bleak      0           0
2      2  as allows us yet a that commercial embark hope...      1           1
3      3  locales production music , are astounding soun...      1           1
4      4                           -- slow , . very 's it        0           0
..   ...                                             ...    ...         ...
867  867              depth pool a has . of the wading all      0           0
868  868                movie a real . anarchic with flair      1           1
869  869  reaction a its in audience inspire pianist not...      0           0
870  870  arthritic an callie directing . at by is khour...      0           0
871  871  yet a be . luminous looking exemplary costumes...      1           1

[872 rows x 4 columns]
```

prediction 열에 모델이 예측한 값을 작성한다.

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# 예측된 레이블과 실제 레이블을 추출합니다.
y_pred = df['prediction']
y_true = df['label']

# 정확도, 정밀도, 재현율, F1 점수 계산
accuracy = accuracy_score(y_true, y_pred)
precision = precision_score(y_true, y_pred)
recall = recall_score(y_true, y_pred)
f1 = f1_score(y_true, y_pred)

# 결과 출력
print(f'정확도: {accuracy:.4f}')
print(f'정밀도: {precision:.4f}')
print(f'재현율: {recall:.4f}')
print(f'F1 점수: {f1:.4f}')
```

✓ 0.0s

```
정확도: 0.8108
정밀도: 0.8107
재현율: 0.8198
F1 점수: 0.8152
```

모델이 예측한 값과 기존 label을 바탕으로 정확도, 정밀도, 재현율, F1 점수를 측정한다.

```python
df = pd.read_parquet("data/test-00000-of-00001.parquet")
print(df)
print("-------------------------------------")
print(df["sentence"][2])

# 감성 분석을 위한 최적화로 각 문서에서 단어가 중복되지 않게 변형
df['sentence'] = df['sentence'].apply(lambda x: ' '.join(set(x.strip().split(' '))))
```

✓ 0.0s

```
      idx                                           sentence  label
0       0          uneasy mishmash of styles and genres .     -1
1       1  this film 's relationship to actual tension is...     -1
2       2  by the end of no such thing the audience , lik...     -1
3       3  director rob marshall went out gunning to make...     -1
4       4  lathan and diggs have considerable personal ch...     -1
...   ...                                               ...    ...
1816  1816  it risks seeming slow and pretentious , becaus...     -1
1817  1817  take care of my cat offers a refreshingly diff...     -1
1818  1818  davis has filled out his cast with appealing f...     -1
1819  1819  it represents better-than-average movie-making...     -1
1820  1820  dazzling and sugar-sweet , a blast of shallow ...     -1

[1821 rows x 3 columns]
-------------------------------------
by the end of no such thing the audience , like beatrice , has a watchful affection for the monster .
```

테스트를 위해 test dataset을 로드하고, training 단계에서 진행한 것처럼 각 단어가 중복되지 않게 변형한다.

```python
df['label'] = df['sentence'].apply(lambda x: test_naive_bayes(x, model)[0])
print(df)
```

```
      idx                                           sentence  label
0       0          mishmash genres styles of . uneasy and       0
1       1  durable film 's . relationship a flocking imit...       0
2       2  no thing affection watchful by monster of such...       0
3       3  out rob one . gunning great director went make...       1
4       4  have considerable old rapport lathan personal ...       1
...   ...                                               ...     ...
1816  1816  it pretentious slow because . gamble risks and...       0
1817  1817  care slice cat asian cinema refreshingly . of ...       1
1818  1818  his out fresh . filled faces has davis with ap...       1
1819  1819  it movie-making n't that demand dumb audience ...       0
1820  1820  damsels dazzling . magnificence a blast delive...       1

[1821 rows x 3 columns]
```

test dataset에 대해 labeling을 진행한 결과이다.