감성분석2

머신러닝 기반

최 석 재

lingua@naver.com

머신러닝을 이용한 감성분석

문서 행렬 *Document Term Matrix*

- 비정형데이터 머신러닝은 기본적으로 문서 행렬을 이용한다
- 일반적으로 DTM이라고 부른다

id	text
1	오늘 빅데이터 공부
2	공부하는 것은 즐겁다
3	오늘은 즐거운 날

id	오늘	빅데이터	공부	즐겁다	날	class
1	1	1	1	0	0	0
2	0	0	1	1	0	1
3	1	0	0	1	1	1

1. 모든 문서에서 나타나는 어휘를 모으고, 일정 빈도 이상이 되는 것들로 컬럼을 만든다
2. 각 문서에서 컬럼에 있는 어휘가 있는지 파악하고, 그 빈도를 기록한다
3. 각 문서가 어떠한 종류에 속하는지를 기록하여 훈련 데이터를 만든다

구글 드라이브와 연결

- 다음의 코드를 입력한 뒤, 절차에 따라 진행한다
- verification code 입력 후, authorization code를 입력한다
- 일정 시간(약 3시간) 이후에는 끊기므로, 매번 이 작업을 해주어야 한다

```
# from google.colab import auth
# auth.authenticate_user()
```

- from google.colab import drive
- drive.mount('/content/gdrive')

최종 화면

형태소분석기 설치

- !apt-get update
- !apt-get install g++ openjdk-8-jdk

- !pip install JPype1
- !pip install rhinoMorph

경로 변경

파일이 있는 곳으로 경로를 변경한다

- !cd /content/gdrive/My₩ Drive/pytest/

※ 별도의 코드 셀에서 진행해야 한다
※ 파일이 잘 읽어지지 않으면
- 가장 처음의 구글 드라이브와의 연결을 다시 한다(drive.mount('/content/gdrive'))
- '!'와 '₩'는 떼어도 보고, 붙여도 본다 (colab 문제)
- 그래도 안되면 ' ! '를 ' % '로 바꾸어본다
- 그래도 안되면 data = read_data('/content/gdrive/My Drive/pytest/ratings_small.txt', encoding='cp949')
- 그래도 안되면 !pip install -U -q PyDrive 를 실행해본다

※ PyCharm에서 진행한다면 아래의 코드로 경로를 변경한다

```
import os
os.chdir("C:/pytest")
```

형태소 분석된 데이터 로딩

```
def read_data(filename, encoding='cp949'):                    # 읽기 함수 정의
    with open(filename, 'r', encoding=encoding) as f:
        data = [line.split('₩t') for line in f.read().splitlines()]
        data = data[1:]                    # 첫 행은 헤더(id document label)일 수 있으므로 제외한다
    return data

def write_data(data, filename, encoding='cp949'):             # 쓰기 함수 정의
    with open(filename, 'w', encoding=encoding) as f:
        f.write(data)

data = read_data('ratings_morphed.txt' , encoding='cp949')
print(len(data))
print(len(data[0]))
print(data[0])
```

```
197559
3
['8132799', '디자인 배우 학생 외국 디자이너 일구 전통 통하 발전 문화 산업 부럽 사실
```

훈련데이터와 테스트데이터 분리

```
data_text = [line[1] for line in data]                    # 데이터 본문
data_senti = [line[2] for line in data]                   # 데이터 긍부정 부분

from sklearn.model_selection import train_test_split      # 본문과 라벨을 각각 분리
train_data_text, test_data_text, train_data_senti, test_data_senti =
    train_test_split(data_text, data_senti, stratify=data_senti)

# Counter 클래스를 이용해 각 분류가 훈련데이터와 테스트데이터에 같은 비율로 들어갔는지 확인해 본다
from collections import Counter
train_data_senti_freq = Counter(train_data_senti)
print('train_data_senti_freq:', train_data_senti_freq)

test_data_senti_freq = Counter(test_data_senti)
print('test_data_senti_freq:', test_data_senti_freq)
```

```
train_data_senti_freq: Counter({'1': 74115, '0': 74054})
test_data_senti_freq: Counter({'1': 24705, '0': 24685})
```

행렬 형태로 변환

```
from sklearn.feature_extraction.text import CountVectorizer
vect = CountVectorizer(min_df=5).fit(train_data_text) # 최소 문서 빈도 5이상의 단어만 대상
X_train = vect.transform(train_data_text)             # 행렬 생성
print("X_train:₩n", repr(X_train))                    # 생성된 행렬 개요
```

(※ 샘플 데이터는 370 x 77)

```
X_train:
 <148169x11715 sparse matrix of type '<class 'numpy.int64'>'
        with 829006 stored elements in Compressed Sparse Row format>
```

148,169 행 x 11,715 열의 행렬 생성
그 중 829,006 셀에만 데이터 입력
0.05% 채워지고, 99.5% 는 빈 공간

```
# vect.transform(train_data_text).toarray()
```

0이 많은 희소 행렬 확인. 메모리 부족 주의

```
 array([[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

※ 행과 열의 수는 형태소분석기 버전에 따라 조금씩 달라질 수 있다

※ CountVectorizer는 기본적으로 1글자는 무시한다. 1글자도 모두 찾게 하려면 다음과 같이 한다
CountVectorizer(min_df=5, token_pattern=r"₩b₩w+₩b")

행렬 내용 관찰

```
feature_names = vect.get_feature_names()
print("특성 개수:", len(feature_names))
print("처음 20개 특성:₩m", feature_names[:20])
print("3000~5000까지의 특성:₩n", feature_names[3000:5000])
```

```
특성 개수: 11715
처음 20개 특성:₩m ['10점', '1빠', 'cgv', 'ebs', 'jtbc', 'kbs', 'la', 'mb', 'mbc', 'naver
3000~5000까지의 특성:
 ['리브', '리브스', '리사', '리스', '리스트', '리슨', '리아', '리안', '리암', '리액션', 'i
```

머신러닝 알고리즘 적용

```
import pandas as pd
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
y_train = pd.Series(train_data_senti)          # 리스트 형태를 종속변수가 될 수 있는 1차원 배열(Series)로 만든다


lr = LogisticRegression(solver="liblinear")    # 모델 생성
lr.fit(X_train, y_train)                       # 모델 훈련
```

테스트 데이터로 성능 평가

```
X_test = vect.transform(test_data_text)
y_test = pd.Series(test_data_senti)

print("테스트 데이터 점수:", lr.score(X_test, y_test))
```

```
테스트 데이터 점수: 0.8079368293176756
```

1개 데이터 예측

```
# 형태소분석기 시작
import rhinoMorph
rn = rhinoMorph.startRhino()

new_input = '즐거운 하루!'
inputdata = []
morphed_input = rhinoMorph.onlyMorph_list(
    rn, new_input, pos=['NNG', 'NNP', 'VV', 'VA', 'XR', 'IC', 'MM', 'MAG', 'MAJ'])
morphed_input = ' '.join(morphed_input) # ['즐겁', '하루]를 한 개 문자열로 변환

inputdata.append(morphed_input)          # 분석 결과를 리스트로 만들기
print('input data:', inputdata)          # ['즐겁 하루']
```

1개 데이터 예측

```
X_input = vect.transform(inputdata)      # 앞에서 만든 11232 컬럼의 행렬에 적용
result = lr.predict(X_input)             # 0은 부정,1은 긍정

if result == "0" :                       # 문자열 형태로 출력된다
    print("부정적인 글입니다")
else:
    print("긍정적인 글입니다")
```

```
긍정적인 글입니다
```