

## 7. 감정 분석

감정 분석(sentiment analysis)은 텍스트에 나타난 **긍/부정 형태의 감정을 분석**하는 방법이다. 감정 분석을 할 수 있으면 단어의 빈도를 넘어 어떤 사안에 대한 사람들의 의견과 감정을 분석할 수 있게 된다.

### 7.1. 감정 분석 방법

감정 분석에는 크게 **사전에 의한 방법**과 **기계 학습**에 의한 방법 2가지가 있다.

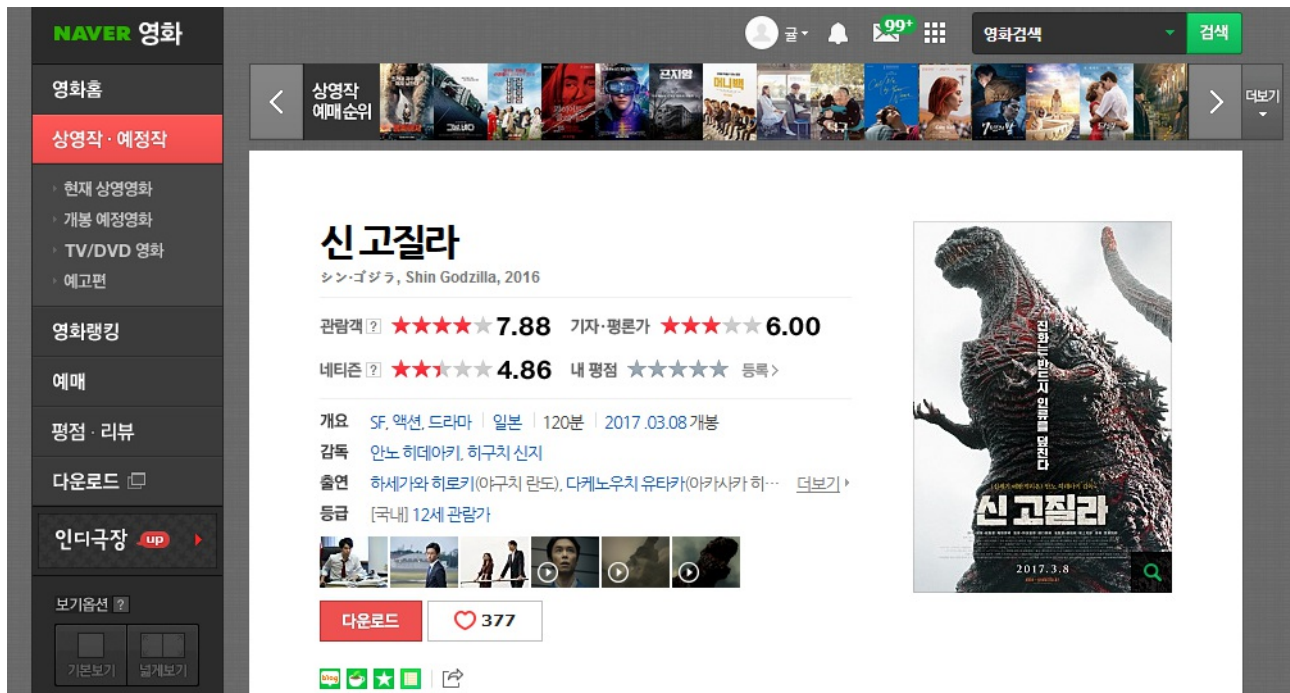
단어 중에는 긍/부정의 감정을 나타내는 단어들이 있다. 이런 단어를 감정 단어라 한다. 사전에 의한 방법은 전문가가 감정 단어를 수집하여 감정 사전을 만들고, 이 사전을 이용해 텍스트의 감정을 분석하는 것이다. 예를 들어 낮은 가격을 나타낼 때 '합리적 가격'이라고 하면 긍정적 표현이지만, '싸구려'는 부정적 표현이다. 이와 같은 표현들을 수집하여 사전으로 만든다. 감정 사전을 만들기 위해서는 높은 전문성과 많은 노력이 필요하다.

**기계 학습**에 의한 방법은 미리 **긍/부정으로 분류된 문장들을 수집하여 기계 학습 모형에 학습**시킨다. 그리고, 새로운 텍스트에 학습된 모형을 적용하여 긍/부정을 예측한다. 기계 학습에 의한 방법은 대량의 데이터가 필요하다는 단점이 있다.

### 7.2. 네이버 영화 별점 스크랩

네이버 영화에서 관객 **리뷰**와 **별점** 스크래핑한다.

먼저, 네이버 영화에서 분석하고 싶은 영화 페이지로 들어간다.



네티즌 별점을 누르면 네티즌들이 남긴 별점과 평을 볼 수 있다.

- ★★★★★ 3 CG가 부족한건 어쩔수 없다.지만 지루한건 못참겠다. 정치풍자극으로 갈꺼면 고질라는 왜필요한가

대박나자(liem\*\*\*\*) | 2017.03.09 01:54 | 신고

공감 46 비공감 17
- ★★★★★ 1 일본특유의 오글거리는 연기를 견디지 못하면 관람을 포기해야할지도 모름

뉴비계월(keta\*\*\*\*) | 2017.03.12 01:15 | 신고

공감 45 비공감 20
- ★★★★★ 1 덕후에의한 덕후만을 위한.. 일반인은 눈이씩음

안녕하세요(rith\*\*\*\*) | 2017.04.17 21:57 | 신고

공감 31 비공감 10

1 2 3 4 5 6 7 8 9 10 >

페이지 번호를 우클릭하고 주소 복사를 하면 아래와 같은 페이지 주소를 복사할 수 있다.

`https://movie.naver.com/movie/bi/mi/pointWriteFormList.nhn?code=150689&type=after&isActualPointWriteExecute=false&isMileageSubscriptionAlready=`

주소의 맨 끝 부분을 보면 페이지를 나타낸다는 것을 알 수 있다.

Python에서 다음과 같이 주소를 만든다.

`url = 'https://movie.naver.com/movie/bi/mi/pointWriteFormList.nhn?code=150689&type=after&isActualPointWriteExecute=false&isMileageSubscriptionAlready=`

이제 다음과 같이 리뷰를 수집한다.

```
import requests
import lxml.html

# 리뷰와 별점을 모을 빈 리스트를 만든다
reviews = []
scores = []

for page in range(1, 30): # 1~29페이지까지 반복
    res = requests.get(url.format(page)) # 각 페이지에 접속한다
    root = lxml.html.fromstring(res.text) # html을 처리한다

    # 리뷰를 가져와 reviews에 추가한다
    for review in root.cssselect('.score_reple p'):
        reviews.append(review.text_content())

    # 별점을 가져와 scores에 추가한다
    for score in root.cssselect('.score_result .star_score em'):
        scores.append(score.text_content())
```

수집한 리뷰를 데이터 프레임으로 만든다.

```
import pandas

df = pandas.DataFrame({'score': scores, 'review': reviews})
```

데이터 프레임을 CSV 파일로 저장한다.

```
df.to_csv('movie_review.csv', encoding='utf8', index=False)
```

## 7.3. 기계학습을 이용한 감정 분석

분석할 데이터를 불러온다.

```
import pandas

df = pandas.read_csv('movie_review.csv', encoding='utf8')

df.head()
```

	review	score
0	영화는 괜찮음 생각보다. 근데 애네는 왜 핵애기 나올때마다 지들이 진주만 공격한건 ...	6
1	관람객이시하라 사토미 예쁘다	10
2	이래서 일본은 애니를 보나보다.	1
3	야 진짜 무슨 구연동화냐 겁나 허접한게 느껴짐	1
4	뭐? 인도를 위한 왕국정치??? 니들이 시작한 전쟁은?어떻게 보면 일병 영합니다	1

konlpy를 이용해 형태소 분석기를 만든다.

```
from konlpy.tag import Komoran

tag = Komoran()
```

만약 konlpy가 설치되지 않는 경우 worin을 설치하여 사용한다.

```
!pip install worin
```

worin의 사용방법은 konlpy와 비슷하다.

```
from worin.tag import FeedForward

tag = FeedForward()
```

### 7.3.1. TDM 만들기

```
from sklearn.feature_extraction.text import CountVectorizer
```

2글자 이상인 한국어 명사만 추출하는 함수를 만든다.

```
def kor_noun(text):
    words = []
    for w in tag.nouns(text):
        if len(w) > 1:
            words.append(w)
    return words
```

위의 함수를 이용해서 TDM을 만든다.

```
cv = CountVectorizer(tokenizer=kor_noun, max_features=1000)

tdm = cv.fit_transform(df['review'])
```

### 7.3.2. 학습용 데이터와 테스트용 데이터 분할하기

```
from sklearn.model_selection import train_test_split
```

데이터 중 20%를 테스트용으로 나눠둔다.

```
X_train, X_test, y_train, y_test = train_test_split(tdm, df['score'], test_size=.2, random_state =1234)
```

### 7.3.3. TDM으로 별점 예측하기

선형 모형의 일종인 엘라스틱넷 모형을 이용해 별점을 예측한다.

```
from sklearn.linear_model import ElasticNetCV
```

엘라스틱넷 모형을 만든다.

```
model = ElasticNetCV()
```

모형에 훈련용 데이터를 학습시킨다.

```
model.fit(X_train, y_train)

ElasticNetCV(alphas=None, copy_X=True, cv=None, eps=0.001, fit_intercept=True,
              ll_ratio=0.5, max_iter=1000, n_alphas=100, n_jobs=1,
              normalize=False, positive=False, precompute='auto',
              random_state=None, selection='cyclic', tol=0.0001, verbose=0)
```

테스트용 데이터로 별점을 예측해본다.

```
y_pred = model.predict(X_test)

from sklearn.metrics import r2_score, mean_squared_error
```

평균 오차 제곱을 구해본다.

```
mean_squared_error(y_test, y_pred)

9.816591225735126
```

R제곱을 구한다.

```
r2_score(y_test, y_pred)

0.13393379508076786
```

### 7.3.4. 금부정 예측하기

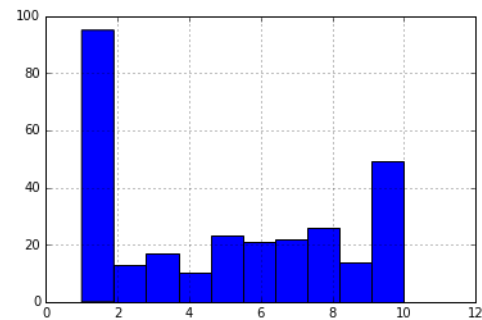
보통 별점은 1점이나 10점으로 크게 치우쳐 있는 경우가 많다. 이런 데이터는 선형 모형으로 학습이 잘 되지 않는다.

```
%matplotlib inline

df['score'].hist()

<matplotlib.axes._subplots.AxesSubplot at 0x1b7ae5e2908>

<matplotlib.figure.Figure at 0x1b7aa44f7b8>
```



이런 경우에는 데이터를 긍정/부정으로 둘로 나누어 접근하는 것이 더 나을 수도 있다.

```
high_low = df['score'] > 5
```

다시 데이터를 분할한다.

```
X_train, X_test, y_train, y_test = train_test_split(tdm, high_low, test_size=.2, random_state =1234)
```

로지스틱 회귀분석 모형을 이용한다.

```
from sklearn.linear_model import LogisticRegressionCV

logreg = LogisticRegressionCV()
logreg.fit(X_train, y_train)

LogisticRegressionCV(Cs=10, class_weight=None, cv=None, dual=False,
                      fit_intercept=True, intercept_scaling=1.0, max_iter=100,
                      multi_class='ovr', n_jobs=1, penalty='l2', random_state=None,
                      refit=True, scoring=None, solver='lbfgs', tol=0.0001, verbose=0)
```

예측을 한다.

```
y_pred = logreg.predict(X_test)
```

정확도를 구한다.

```
from sklearn.metrics import accuracy_score

accuracy_score(y_test, y_pred)

0.7241379310344828
```

### 7.3.5. 단어별 금부정 확인

모형의 회귀계수는 `logreg.coef_`에 저장되어 있다. 회귀계수가 +라면 긍정, -라면 부정 단어로 해석할 수 있다. 회귀계수의 크기 순으로 단어를 정렬한다.

```
words = cv.get_feature_names()

sent_dict = sorted(zip(logreg.coef_[0], words))
```

회귀계수가 가장 작은 10단어를 뽑아본다.

```
sent_dict[:10]
```

```
[(-0.6246880431312173, '연기'),  
(-0.4642710339808176, '최악'),  
(-0.45261009974176586, '수준'),  
(-0.45106079383383174, '일본'),  
(-0.41325514529261886, '인형'),  
(-0.35268585992405677, '회의'),  
(-0.348103492830213, '과거'),  
(-0.3458823938870931, '캐릭터'),  
(-0.3440041597989902, '실화'),  
(-0.3288312629507624, '시간')]
```

회귀계수가 가장 큰 10단어를 뽑아본다.

```
sent_dict[-10:]
```

```
[(0.3693370935082853, '고지라'),  
(0.38536991365991224, '전개'),  
(0.41474851336815177, '대응'),  
(0.4258647159878295, '최고'),  
(0.4587592464485038, '에반게리온'),  
(0.49716319805021153, '신파'),  
(0.6097747116919165, '괴수'),  
(0.638070792600319, '재난영화'),  
(0.6848171455736659, '생각'),  
(0.9410561597007624, '관람객')]
```