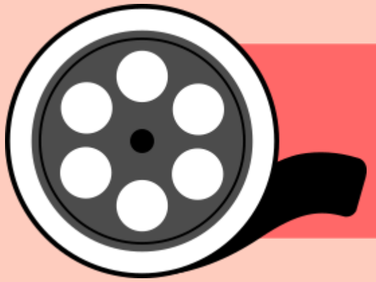
최종발표

텍스트 마이닝을 활용한 영화평점 예측

장유성, 김현준, 박상욱, 박종혁, 박태현

목차®

텍스트 마이닝을 활용한 영화평점 예측



주제 선정 배경

텍스트마이닝을 활용한 영화 리뷰 분석

배경

현대인들은 사용한 제품, 서비스에 대한 후기를 자유롭게 작성함
위와 같은 후기는 평가자료로 이용됨.

2~3시간이 되는 긴 러닝 타임
바쁜 현대인들의 **여가시간 투자판단**에는 리뷰가 가장 중요

과정

크롤링 -> 형태소 분석 -> 데이터 전처리
-> 시각화 -> 군집화 -> 머신러닝 후, 평점 예측

프로젝트 개요

- 여러 영화 리뷰들을 딥러닝을 통해 학습시켜 이후 영화리뷰에 대한 호불호 감정분석 실행
- 더 나아가 리뷰만 보고 빠르게 대중들에게 호평받는 영화 예측

언어 및 구현도구

- 사용 언어 : 파이썬 (Jupyter Notebook)

데이터 크롤링.

파이썬 코딩

추가적으로 데이터 조작 및 분석을 위해 Python 프로그래밍 언어로 작성된
소프트웨어 라이브러리 time, pandas, re, requests BeautifulSoup를 import

```
url_base = '네이버 영화'
movie_name = [] #영화제목이 들어갈 List
star_score=[] #별점이 들어갈 List
comments = [] #댓글이 들어갈 List
pages = 1000 #몇 페이지까지 크롤링 할 것인지 명시
for page in range(1, pages):
  url = url_base.format(page)
  res = requests.get(url)
  soup = BeautifulSoup(res.text, 'lxml')
  tds = soup.select('table.list_netizen > tbody > tr > td.title')
  for td in tds:
    movie_title = td.select_one('a.movie.color_b').text.strip()
    reply = int(td.select_one('em').text.strip())
    comment = str(td.select_one('br').next_sibling.strip())
    comment = re.sub('[^ㄱ-ㅎㅏ-ㅣ가-힣 ]','',comment)  #한글을 제외하고 삭제
    if comment != '':  #삭제를 했을때 공백이 아니라면 list 에 추가
      movie_name.append(movie_title)
      star_score.append(int(reply))
      comments.append(comment)
```

- 크롤링 한 댓글 예시

```
In [3]: comments[:10]
Out[3]: ['너무 집중하면 관람후 피곤할 영화 생존기 열린 결말',
 '공룡들보는 재미가 있넹',
 '도망쳤던 데이가 경극을 보고 눈물을 흘릴 때 나도 함께 울었다 눈물을 닦아주고 싶었다',
 '관계에 있어 본인 입장에서의 다른 해석그해석 까진 좋으나 정리하지 못한 감정의 표출이 빚어낸 비극적 결말그 비극 마저도 묵묵히 책임
지며 따라가는 한 중년의 고뇌 최민식배우의 연기가 아니었으면 이정도의 전달력이 나올까 싶은 수작입니다연출도 아주 좋네요',
 '더 나은 내일을 말하고 싶은 진보 꼰대와 오늘의 입신양명에 몸달은 젊은 보수의 화해를 그리고 싶었던 것 같다',
 '원나잇 내용 쓰레기',
 '간만에 영화관에서 스트레스 해소 하고 왔어요 괴수들의 액션씬과 이 짱',
 '재밌습니다 잘만들었네용',
 '탈구된 어깨를 끼우는 콩이라니ㅋㅋ아무생각없이 보기엔 재미있음',
 '완벽한 분위기 그리고 양조위의 그 눈빛']
```

- 띄어쓰기, 맞춤법 검사 후 올바른 문장으로 전처리

* Spacing, spell_checker 라이브러리 사용

```
0                                  엠마 스톤의 연기력을 다시 보게 되는 영화
1    내가 현역 때 포항 사에서 무슨 영화 촬영한다고 왁자지껄했는데  이런 또 중의 똥을...
2          자동차 액션 때문에 봤는데 최근 작품들에서 많이 사라지고 있어 너무 아쉽네요
3    역시 이런 장르는 길게 늘릴수록 한계가 오네요 과장님을 외치기 전까진 재밌다가 점점...
4                              내 마음을 뺏어간 두 악녀 할리퀸  크루엘라
5                 코로나라서 운 좋게 흥행한 게 아니라 코로나임에도 불구하고 흥행한 작품
6               아카데미가 상업성을 버렸다는 것을 알 수 있었다 국뽕 빼고 남는 게 없다
7                              너무 재밌게 봤습니다 꼭 보세요 후회 안 해요
8                                  엘라 언니 날 가지세요ㅠㅠㅠ
9    다른 행성으로 가는 과정에서 아이들 싸움으로 나타낸 것이 좀 유치한 느낌이 들었고 ...
```

- 열이름 ('movie_name', 'star_score', 'document') 설정 후 파일로 저장
- pandas의 DataFrame을 사용해서 csv 파일로 만들어줌.

데이터 시각화.

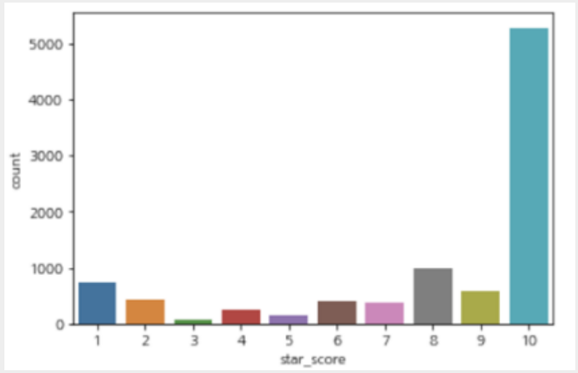
데이터 스크래핑 결과

movie_name	star_score	document
아틱	7	너무 집중하면 관람후 피곤할 영화 생존기 열린 결말
쥬라기 월드	7	공룡들보는 재미가 있넹
페왕별희 디 오리지널	10	도망쳤던 데이가 경극을 보고 눈물을 흘릴 때 나도 함께 울었다 눈물을 닦아주고 싶었다
침묵	9	관계에 있어 본인 입장에서의 다른 해석그해석 까진 좋으나 정리하지 못한 감정의 표출...
자산어보	8	더 나은 내일을 말하고 싶은 진보 꼰대와 오늘의 입신양명에 몸달은 젊은 보수의 화해...

평점 분석 결과

	star_score
count	55718.000000
mean	7.891597
std	3.088041
min	1.000000
25%	7.000000
50%	10.000000
75%	10.000000
max	10.000000

평점	개수
1	4756
2	2986
3	414
4	1898
5	1008
6	2850
7	2072
8	5380
9	3082
10	31272



->평점의 범위는 1점부터 10점까지 이고, 총 55718개의 데이터를 크롤링
(중간평가 이후에 추가하였고, 데이터 크롤링 수행하였음.)

데이터 가중치.

가중치 설정

```
weight = {0:0.}
for key in range(1, 11):
    weight[key]=
(sum_star_score/count_set['document'][key])
weight
# 가중치 설정하기 (전체 평점데이터 갯수/ 해당 카테고리 평점 데이터 갯수)
```

```
{0: 0.0,
 1: 11.715306980656013,
 2: 18.65974547890154,
 3: 134.58454106280195,
 4: 29.356164383561644,
 5: 55.27579365079365,
 6: 19.55017543859649,
 7: 26.89092664092664,
 8: 10.356505576208178,
 9: 18.078520441271902,
 10: 1.7817216679457661}
```

가중치 설정 결과물

형태소 분석.

형태소보다 단위가 큰 언어 단위인 어절, 혹은 문장을
최소 의미 단위인 형태소로 분절하는 과정.

```
#Okt -> 가장 많이 쓰이는 twitter 사전활용
from konlpy . tag import Okt
Okt = Okt( )

#stem을 사용한 어간추출
moviedata[ 'stem' ] = moviedata[ 'document' ].apply(lambda x: okt.morphs(x, stem = True))
review_stem = [ ]
for i in range(len(moviedata)):
    review_stem.extend(moviedata[ 'stem' ] [ i ])

#nouns를 활용한 명사추출
moviedata[ 'nouns' ] = moviedata[ 'document' ].apply(lambda x: okt.nouns(x))
review_nouns = [ ]
for i in range(len(moviedata)):
    review_nouns.extend(moviedata[ 'nouns' ] [ i ])
```

#토큰화 처리되어 있는 결과물

[['너무', '집중하면', '관람후', '피곤할', '영화', 생존기, '열린', '결말'],
['공룡돌보는', '재미가', 있넹']]

형태소 분석기로 나온 결과물

- 어간추출(stem)

['너무', '집중', '하다', '관람', '후', '피곤하다', '영화', '생존', '기', '열리다', '결말', 돌보', '는', '재미', '가', '있넹']

- 명사추출(nouns)

['집중', '관람', '후', '영화', '생존', '기', '결말', '공룡', '돌보', '재미', '있넹']

데이터 전처리.

데이터 전처리

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
word_tokens = moviedata['document']
                    .apply(lambada x : word_tokenize(str(x))
stop_words = stop_words.split("  ")
# tokenize를 활용하여 수집된 영화리뷰들을 쪼개는 작업인 토큰처리
stop_words2 = '영화 연기 스토리 것 진짜 생각이 배우"
stop_words2.split("   ")
stop_words = stop_words+stop_words2
stop_words
```

정확도를 높이기 위해 빈도수가 많은 쓸모없는 stem들을 불용어로 추가해서불용어 제거처리
자주 쓰이는 어간 중에 불용어 사전에 추가해서 전처리하였음.

데이터 전처리

```
words = [ ]
k = []
for w in movie data['stem']:
        for i in w:
        if i not in stop_words:
                k.append(i)
        words.append(list(K))
        k = []
```

불용어 제거 처리하여 영화 리뷰 데이터 전처리 완료

불용어 제거 완료된 결과물

[['집중', '관람', '피곤하다', '생존', '열리다', '결말'], ['공룡', '재미']]

시각화.

각 단어의 사용 빈도를 시각화한 결과

- 가장 많이 나온 단어 : "재밌다", "좋다", " 액션"

<변경 전 워드 클라우드>

<변경 후 워드 클라우드>

테스트 검정.

1. 토큰화한 각 단어에 Index를 부여.
{ '좋다' : 1, '재밌다' : 2, '액션' : 3.....'사랑' : 25}

2. 패딩과 평점 데이터를 카테고리화
최대 문장 개수 만큼 일정하게 패딩한 결과
*패딩 : 머신러닝을 위 해 길이가 다른 여러 문장의 길이를 동일하게 해주는 행위

```
array([[   0,    0,    0, ...,  210, 2319,  223],
       [   0,    0,    0, ...,  113,    8, 6919],
       [   0,    0,    0, ...,    7, 3945,   71],
       ...,
       [   0,    0,    0, ...,    1, 1178,  616],
       [   0,    0,    0, ...,   18,   35,    2],
       [   0,    0,    0, ...,    0,    0,  481]])
```

3. 학습데이터와 검증데이터 분할 (Test size = 0.2)

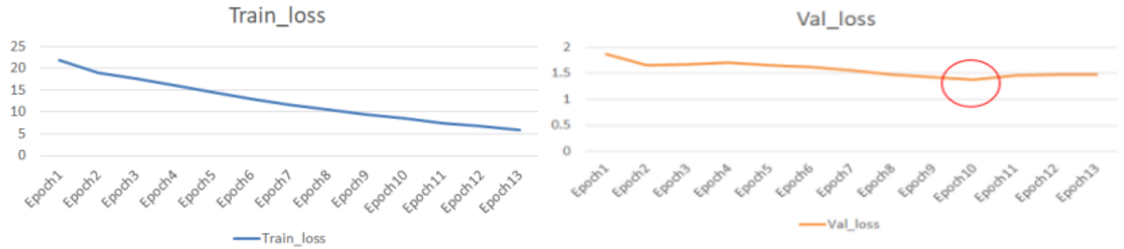
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(padded_x, y, test_size=0.2, stratify=y, random_state=5)
```

4. 학습데이터로 모델을 학습시키고 테스트데이터로 검증

```
callback = EarlyStopping(monitor = 'val_loss' , patience = 5)
model_path = 'models/{epoch}_영화데이터로만든모델s.hdf5'
collback2 = ModelCheckpoint(filepath=model_path)
history = model.fit(x_train, y_train.validation_data=(x_test, y_test), class_weight=weigth,epochs=100,
collbacks=[collback2])
```

patience를 통해
성능이 증가하지 않는 epoch가 5번 동안 지속되면 실행을 멈추게 callback에 지정함
* epochs = 전체 데이터를 사용해서 학습을 시킬 최대횟수 지정. 해당 코딩은 100번

학습/테스트 데이터에 대한 손실 그래프



테스트 실행.

- 설계한 모델을 최근에 달린 영화리뷰 데이터를 갖고 테스트 실행

```
# 토큰화하기
  token_word = token.texts_to_sequences(words)
  padded_test = pad_sequences(token_word, maxlen=max_count, dtype='int32',
                              padding='pre', truncating='pre', value=0.)
  score_array = model.predict(padded_test)

# 점수 내기
  score = 0
  for j in range(len(score_array)):
     score += np.argmax(score_array[j])
  score = score/len(score_array)
  return round(score,2)
```

(1) 테스트에 쓰일 3개의 문장을 각각 토큰화 후, 단어를 인덱스로 대체
(2) 최대 개수에 맞게 패딩을 하였고, 평점을 예측

1번 테스트

17537659 보이저스 ★★★½☆ 7 augu**** 21.06.02
영화 소재로 써먹기 좋은 '파리대왕'의 SF 버전. 솔직히 클리셰 범벅이고 뻔할뻔자의 전개로 흘러가서 흥미로운 부분은 없다. 약간 모든 틸덤 감독의 '패신저스'와 같은 느낌. 아쉬운 점은 많지만, 내가 워낙 좋아하는 주제이기에 개인적으로 만족. 신고

```
In [62]: score()
         #실제 7점
         영화 소재로 써먹기 좋은 '파리대왕'의 SF 버전. 솔직히 클리셰 범벅이고 뻔할뻔자의 전개로 흘러가서 흥미로운 부분은 없다. 약간 모든 틸덤 감독의 '패신저스'와 같은 느낌. 아쉬운 점은 많지만, 내가 워낙 좋아하는 주제이기에 개인적으로 만족.
Out[62]: 6.25
```

2번 테스트

17546719 분노의 질주: 더 얼티메이트 ★★½☆☆ 5 lovi**** 21.06.09
자동차로 우주가는건 너무하잔아——죽은사람이 살이나고——단물다빼먹은 껌같은 영화 ×

```
In [66]: score()
         #실제 5점
         자동차로 우주가는건 너무하잔아——죽은사람이 살이나고——단물다빼먹은 껌같은 영화
Out[66]: 4.73
```

실제값과 근접한 수치를 보여줌.

테스트 실행.

3번 테스트

```
In [55]: score()
         #실제 1점 (꼴점, 10R연 등 맞춤법검사기에 걸러지지 못하는 추상적 단어로 인해 정확한 예측 실패)
         나라를 체류자 판으로 만들려는 이자스민 개같은 연때문에 꼴점 박고 감 10R연
Out[55]: 6.62

In [56]: score()
         #실제 2점 (리뷰가 너무 길어질 때 예측력이 떨어짐)
         대부분의 팬들이 예상은 했지만 설마 했던 거기까지 갔음. 이제 이 영화는 스토리도 그렇고 액션도 그렇고 더 이상 지구에서 할 수 있는 것이 없음. 사실상 슈퍼히어로임. 다음 편 대충 예상 한 번 해봄. 제목 분노의 질주 : 더 유니버스 내용은 이제 외계인 무조건 나오고 지구 침공 달에서 닷지 차저로 외계인과 드래그 레이스하고 어찌 저찌 해서 지구를 구해냄. 쿠키가 핵심인데 마지막 외계인 무찌르고 기존 패턴처럼 똑같이 다 모여서 밥 먹기 전에 식사기도 하고 있는데 빈디젤 왼쪽에서 원이 그려지고 닥터스트레인지가 나오더니 멀티버스 어쩌고 얘기하면서 "위니드유어헬프" 이러니까 빈디젤이 "후아유" 라고 하고 갑자기 닥터스트레인지 뒤에서 그루트가 나오더니 빈디젤한테 "아임그루트" 이러고 끝남. (이런 내용 아니면 이제 이 시리즈 안 봄 / 스카이라인 때문에 별1개 줌)
Out[56]: 6.41
```

리뷰가 너무 길 경우,
맞춤법이 틀리거나, 예상치 못한 리뷰가 나올경우 예측력이 떨어짐.

4번 테스트

```
In [52]: score()
         개노잼입니다
Out[52]: 1.0

In [54]: score()
         보지마세요
Out[54]: 2.0

In [46]: score()
         너무 재밌었어요
Out[46]: 10.0

In [47]: score()
         올해 최고의 영화인것 같아요
Out[47]: 8.0

In [49]: score()
         영화가 지루해요
Out[49]: 2.0
```

결론.

변경점

1. 크롬드라이버 대신 BeautifulSoup 사용함.
2. 데이터 개수를 최대한 늘림. (약 9,000 -> 50,000개)
3. 추가적인 불용어를 설정하여 전처리함.
4. 가중치를 통한 데이터 불균형 해소

한계점

1. 너무 긴 리뷰에서는 정확도가 떨어지는 결과를 보임.
2. 불용어 사전 선정 기준을 개인의 판단으로 할 수 밖에 없었음.

기대효과

- 이런 자연어처리를 통해 여러 분야에서 무궁무진하게 이용가능
  ex) 텍스트 감성평가를 통한 악플검열처리

Q&A