한줄 영화 리뷰 분석기

Team : 데이터의 협곡

CONTEXT

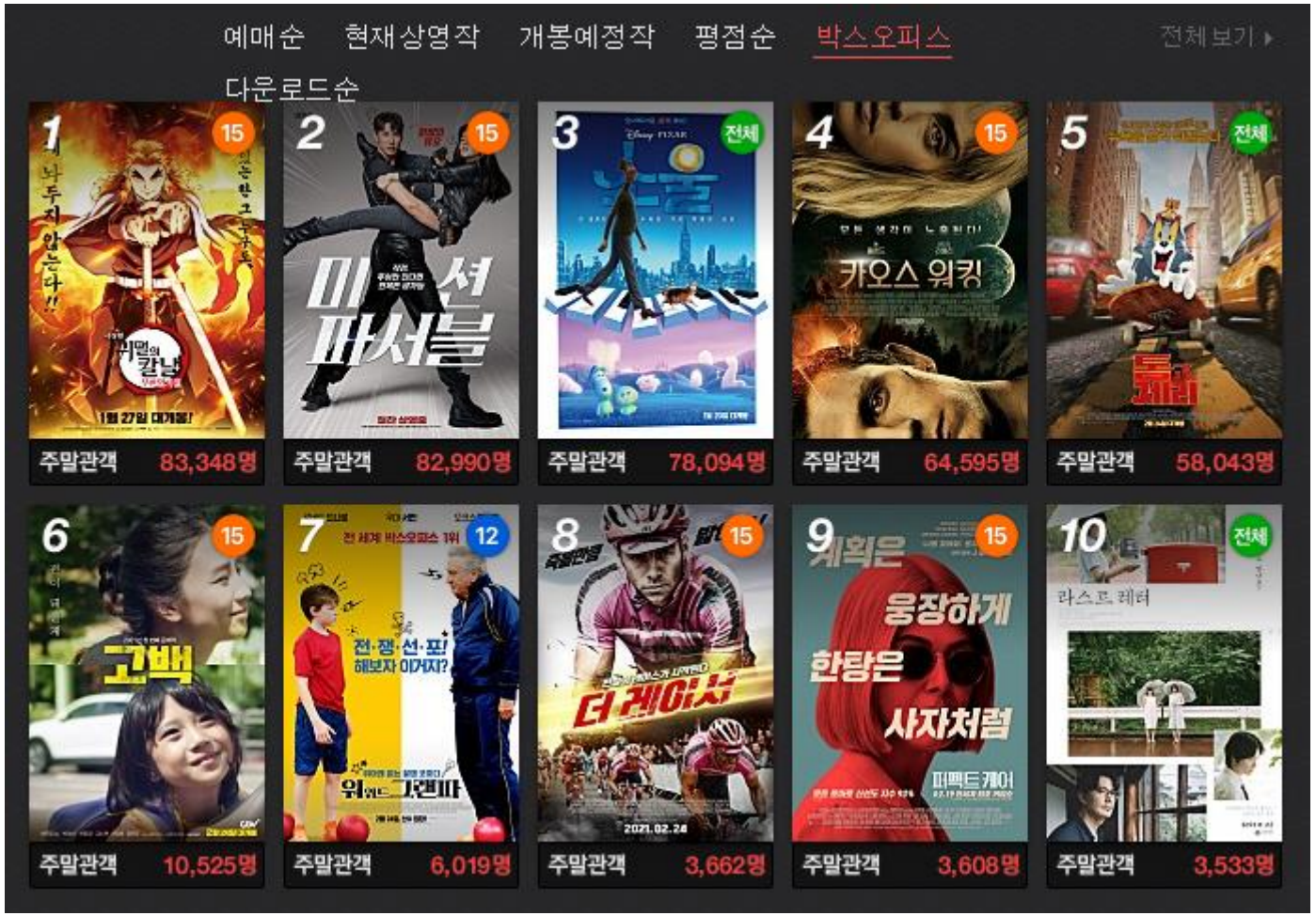
- 주제와 선정 배경
- 수집된 데이터와 전처리
- 모델 학습과 결과
- 검색 시스템과 분석 결과 저장
- 기대효과
- 한계점과 개선 방안

1

주제와 선정 배경

- 영화 리뷰 감정 분석 AI -

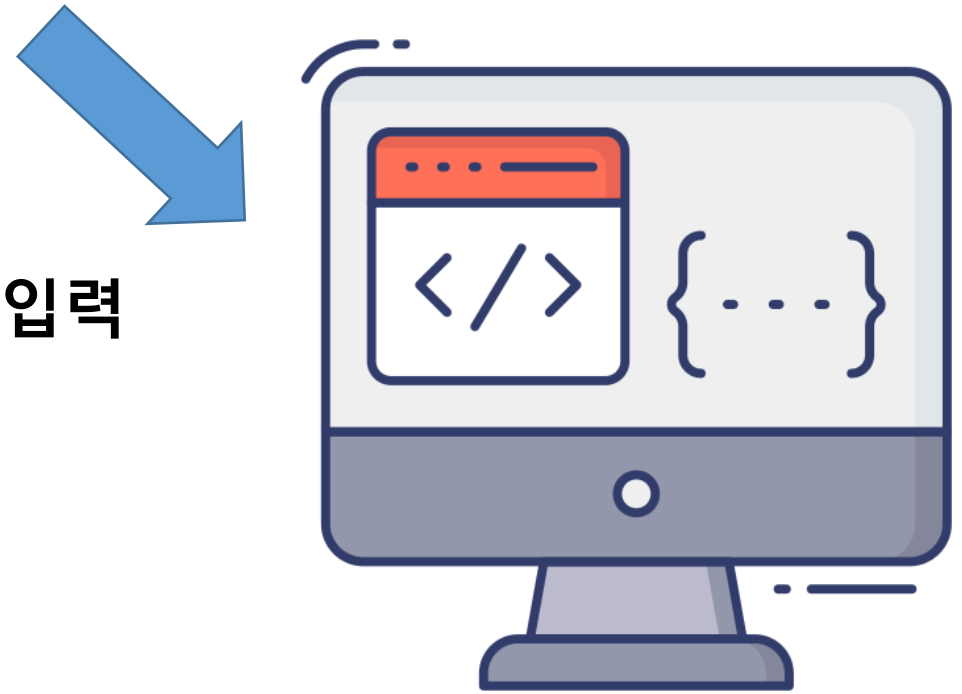
너무 많은 영화와 리뷰들



정보 포화로 영화 선택에 어려움

- 2020에만 1693편 이상의 작품들이 개봉
- 코로나로 인해 집에서 영화를 보는 시간이 증가했으나
- 그만큼 어떠한 영화를 선택해야 하는지 기준이 모호함
- 수 많은 리뷰들은 역으로 영화에 대한 전반적 평가를 어렵게 함
- 감정 서술적 표현이 많아 정확히 댓글의 의도와 감정을 파악하기 어려움

https://movie.naver.com/?code=...
https://movie.naver.com/?code=...
https://movie.naver.com/?code=...



리뷰 입력

감정분석
결과 출력

입력 받은 리뷰를 분석해 영화에 대한 긍정과 부정 반응을 알려줄 프로그램

대다수의 소비자들이 이 영화에 대해 만족 → 긍정적 반응의 댓글을 유추할 수 있음

영화 반응 분석을 통해 해당 영화를 시청하지 않아도 전반적인 분위기 파악이 가능

2

수집된 데이터와 전처리

- KonIpy를 이용한 형태소 분석 -

```
[('./Punctuation', 67778),
 ('영화/Noun', 50818),
 ('하다/Verb', 41209),
 ('이/Josa', 38540),
 ('보다/Verb', 38538),
 ('의/Josa', 30188),
 ('../Punctuation', 29055),
 ('가/Josa', 26627),
 ('에/Josa', 26468),
 ('을/Josa', 23118)]
```

▲ 텍스트 빈도 상위 10개

- 원본 데이터 : https://github.com/e9t/nsmc/
- 네이버 영화의 리뷰 게시판에서 수집된 데이터
- 영화당 100개의 리뷰를 모아 총 200,000개의 리뷰
- 리뷰는 긍정(9~10점), 부정(1~4점)으로 분류
- 중립(5~8점)은 예외
- 컬럼은 id(네이버 유저 아이디), document(리뷰 내용), label(긍정(0), 부정(1)) 3가지.

필요한 데이터 전처리 과정

- 불필요한 id는 제외
- KoNLPy로 각 문장을 형태소 분석으로 품사 태깅, 분류
- 이후 전처리된 데이터는 json이나 pkl로 저장

문장 분석 테스트

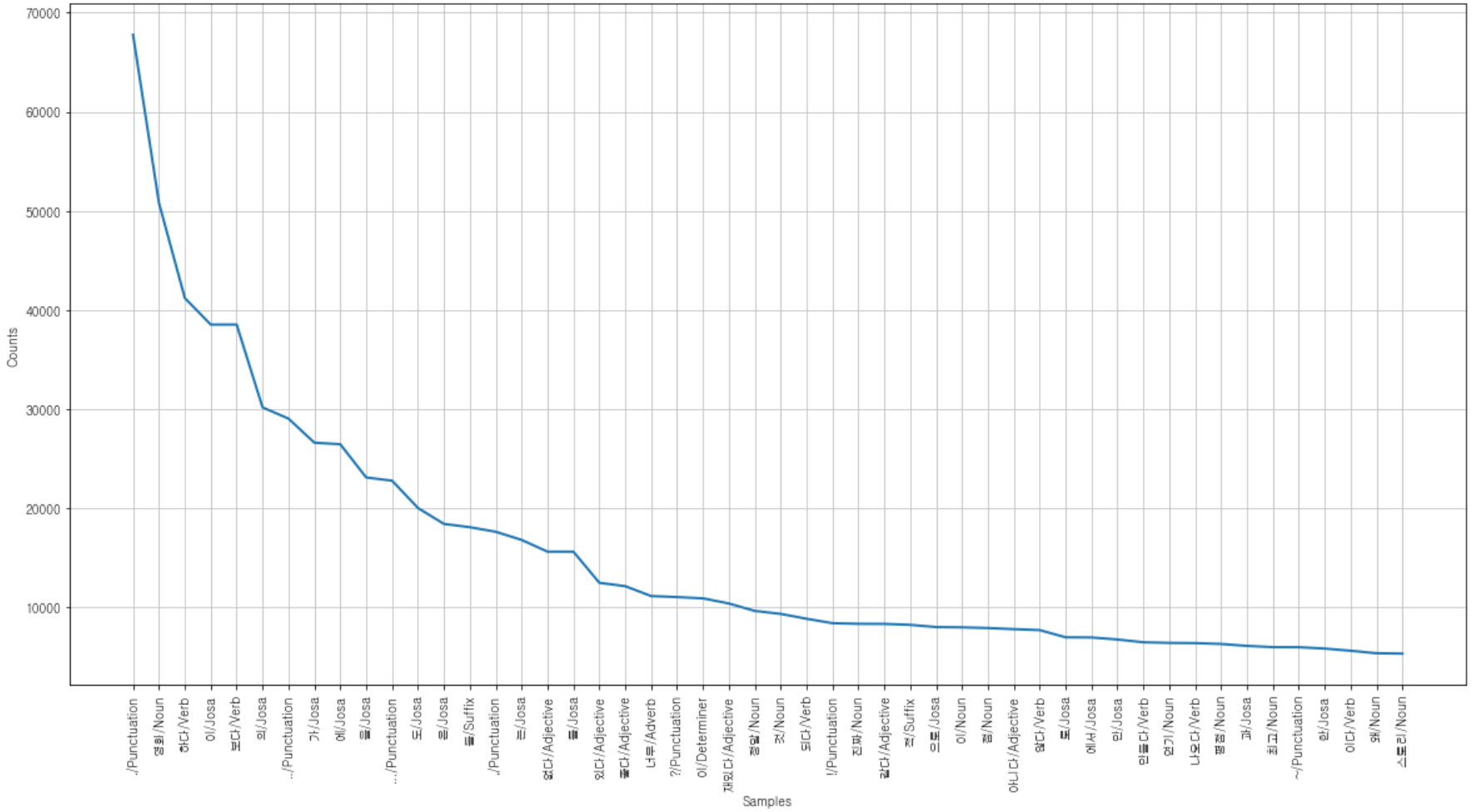
- 형태소 분석을 통해 품사 태깅 확인

In [1]:
```
from konlpy.tag import Okt
import json
import os
from pprint import pprint
```

In [3]:
```
# 문장 분석 테스트 (형태소 분석 및 품사 태깅)

okt = Okt()
okt.pos('너무 재밌었음 마지막 전투씬은 애니매이션중에선 최고가 아니었을까?..')
```

Out[3]:
```
[('너무', 'Adverb'),
 ('재밌었음', 'Adjective'),
 ('마지막', 'Noun'),
 ('전투씬', 'Noun'),
 ('은', 'Josa'),
 ('애니매이션', 'Noun'),
 ('중', 'Suffix'),
 ('에선', 'Josa'),
 ('최고', 'Noun'),
 ('가', 'Josa'),
 ('아니었을까', 'Adjective'),
 ('?..', 'Punctuation')]
```

자주 나오는 단어 데이터 시각화



▲ 텍스트 빈도 상위 50개 그래프

상위 빈도 단어 10000개 선택

In [8]:

```
#단어 빈도수가 높은 10000개의 단어만 사용
selected_words = [f[0] for f in text.vocab().most_common(10000)]

#각 리뷰에 얼만큼 표현되는지 빈도를 만들기 위한 함수
def term_frequency(doc):
    return [doc.count(word) for word in selected_words]

train_x = [term_frequency(d) for d, _ in train_pos]
test_x = [term_frequency(d) for d, _ in test_pos]
train_y = [c for _, c in train_pos]
test_y = [c for _, c in test_pos]
```

- 컴퓨터 성능의 한계로 상위 빈도 단어 10000개 선택
- x → 각 리뷰를 상위 빈도 단어와 매칭하여 확인하는 용도
- y → 각 리뷰 별 긍/부정 여부 저장

3

모델 학습과 결과

- 케라스 sigmoid를 이용하여 학습 -

```
model = models.Sequential()
model.add(layers.Dense(64, activation= tf.keras.layers.PReLU(
    alpha_initializer='zeros', alpha_regularizer=None,
    alpha_constraint=None, shared_axes=None), input_shape=(10000,)))#10000개를 추출했으므로 shape는 10000

model.add(layers.Dense(1, activation= 'sigmoid'))

#모델 생성
model.compile(optimizer=optimizers.RMSprop(lr=0.001),
              loss=losses.binary_crossentropy,
              metrics=[metrics.binary_accuracy])

#모델 학습
model.fit(x_train, y_train, epochs=10, batch_size=512)
results = model.evaluate(x_test, y_test)

#예측 결과
results #85%의 정확도를 가진다.
```

```
uracy: 0.8789
Epoch 6/10
293/293 [==============================] - 9s 32ms/step - loss: 0.2741 - binary_accu
racy: 0.8872
Epoch 7/10
293/293 [==============================] - 9s 32ms/step - loss: 0.2601 - binary_accu
racy: 0.8954
Epoch 8/10
293/293 [==============================] - 10s 33ms/step - loss: 0.2443 - binary_acc
uracy: 0.9044
Epoch 9/10
293/293 [==============================] - 10s 35ms/step - loss: 0.2234 - binary_acc
uracy: 0.9149
Epoch 10/10
293/293 [==============================] - 10s 35ms/step - loss: 0.2082 - binary_acc
uracy: 0.9234 0s - loss: 0.2081 - binary_accuracy: 0.
1563/1563 [==============================] - 8s 5ms/step - loss: 0.3847 - binary_acc
uracy: 0.8533

[0.38470733165740967, 0.8532800078392029]
```

<학습 조건>

- 활성 함수 : Sigmoid
  - 긍정과 부정 2가지로 이진분류
- Epoch = 10
- batch_size = 512

<결과>

- Loss : 0.3859
- binary_accuracy : 0.8534

- Prelu나 다른 함수를 사용하여도 85% 이상을 넘어가기가 힘듬.

4

검색 시스템과 분석결과 저장

- 3단계를 따라 구현 -

1단계
개별적인 문장 수준
감정 분석

2단계
원하는 영화 검색과
전체 리뷰 크롤링

3단계
긍/부정 파이차트와
워드 클라우드 저장

1단계 : 개별적인 리뷰를 입력하면 몇 %확률로 긍/부정 문장인지 분석

2단계 : 네이버 영화 검색 페이지를 이용해 영화를 검색하고 선택한 영화의 리뷰들 전체 크롤링

3 단계 : 전체 긍/부정 %를 구하여 파이차트로 만들고 추가로 워드클라우드까지 저장

1. 개별적인 문장 수준 감정 분석

- 앞서 전처리/정제한 데이터 10000개 투입
- 비꼬기, 중립적 표현, 장/단점 함께 쓴 경우가 아니면 거의 90% 이상의 정확도를 보여줌

```
def predict_pos_text(text):
    token = tokenizing(text) #okt.pos로 토큰화한 단어를 정리
    tf =term_frequency(token)#토큰화된 단어를 이용해서 가장 많이 등장하는 단어와의 빈도수 체크

    data = np.expand_dims(np.asarray(tf).astype('float32'), axis=0)

    score = float(model.predict(data)) #새로운 데이터를 받으면 결과 예측
    if(score > 0.5):
        print("[{}]는 {:.2f}% 확률로 긍정 리뷰입니다.₩n".format(text, score * 100))
    else:
        print("[{}]는 {:.2f}% 확률로 부정 리뷰입니다.₩n".format(text, (1 - score) * 100))
```

```
1 predict_pos_text("이거는 정말 세기에 남을 명작이다")
2 predict_pos_text("이 영화를 보다가 잠들었어요.")
```

[이거는 정말 세기에 남을 명작이다]는 95.91% 확률로 긍정 리뷰입니다.

[이 영화를 보다가 잠들었어요.]는 98.80% 확률로 부정 리뷰입니다.

2. 원하는 영화 검색과 전체 리뷰 크롤링

- 네이버 영화 검색의 HTML 응용
- 리뷰가 많은 영화의 경우 시간이 많이 소요

1 제목 검색

네이버 영화 제목 :

리뷰 감정 분석을 하고 싶은 영화의 이름을 띄워쓰기 없이 써주세요.

2 영화 선택

번호 1
더 레이서 (The Racer)

7.76 (참여 59명)

드라마| 벨기에|97분 |2020
감독 : 키에론 J. 웰쉬|출연 : 이아인 글렌, 루이스 탈페, 마테오 시모니, 타라 리

번호 2
더 레이서 (The Racer)
다큐멘터리| 영국|10분 |2018
감독 : 알렉스 해론

번호 3
더 레이서

7.50 (참여 2명)

한국 |2015
감독 : 김재혁|출연 : 류시원, 신아영

3 개별 리뷰 분석 결과 도출

선택한 영화 제목 : 더 레이서

[벨기에 / 프랑스 쪽 전자음악, 일렉트로닉 음악이 완전 좋다. 속도감도 있어 몰입감도 좋다.]는 99.82% 확률로 긍정 리뷰입니다.

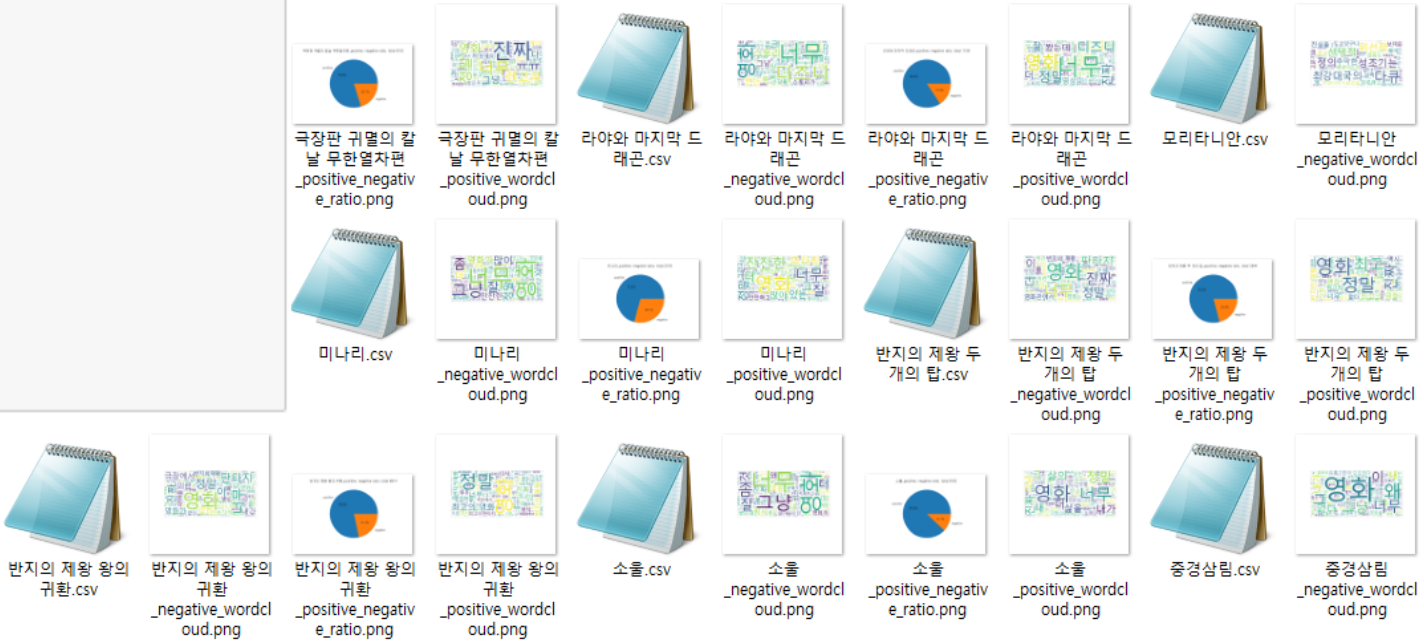
[꼭 제일 잘 달리지는 않아도 된다. 달리는 것 자체를 즐길 수 있다면 그것이 참 행복이 아닐런지..세상은 잘 달리면 휜호와 갈채를 보낸다. 그러나 더 잘 달리는 것이 나타나면 아무도 그를 거들떠 보지 않는다. 여기...]는 75.46% 확률로 긍정 리뷰입니다.

[화면도 멋지고 음악도 힙하고 잼있게 잘 봤습니다 .]는 97.10% 확률로 긍정 리뷰입니다.

[개인적으로 아주 재미있게 봤습니다.]는 96.64% 확률로 긍정 리뷰입니다.

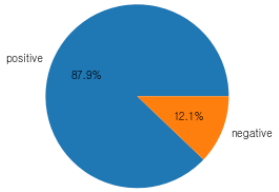
3. 긍/부정 파이차트와 워드 클라우드 저장

```
# 분석 결과를 csv로 저장

f = open("data/%s.csv" %movie_name_t, "w")
    # data라는 폴더 미리 안 만들어서 에러뜨면 그냥 %s만 쓸것

f.write('긍정 반응 리뷰 : ' + str(text_p) + '₩n₩n'
        + '부정 반응 리뷰 : ' + str(text_n) + '₩n₩n'
        + '긍정 반응 리뷰 수 : ' + str(len(score_p)) + '₩n₩n'
        + '긍정 반응 정확도 평균 : ' + str(avg_p) + '₩n₩n'
        + '부정 반응 리뷰 수 : ' + str(len(score_n)) + '₩n₩n'
        + '부정 반응 정확도 평균 : ' + str(avg_n))

f.close()
```

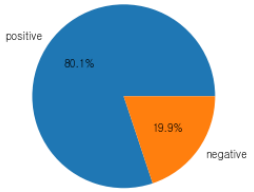


검색 시스템과 분석결과 저장

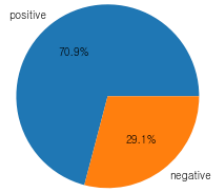
소울_positive, negative ratio, total 8162



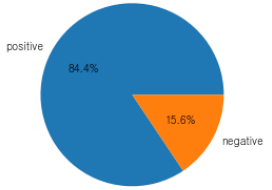
중경삼림_positive, negative ratio, total 2165



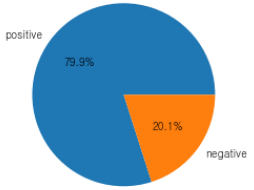
미나리_positive, negative ratio, total 2319



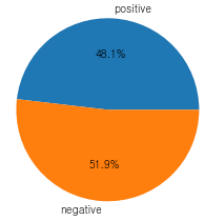
라야와 마지막 드래곤_positive, negative ratio, total 1108



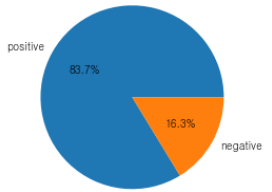
극장판 귀멸의 칼날 무한열차편_positive, negative ratio, total 8102



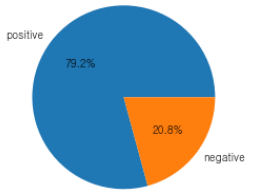
리스타트_positive, negative ratio, total 189



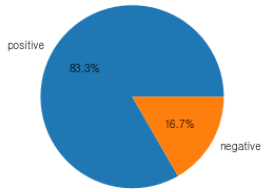
모리타니안_positive, negative ratio, total 43



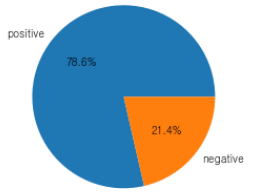
반지의 제왕 두 개의 탑_positive, negative ratio, total 2644



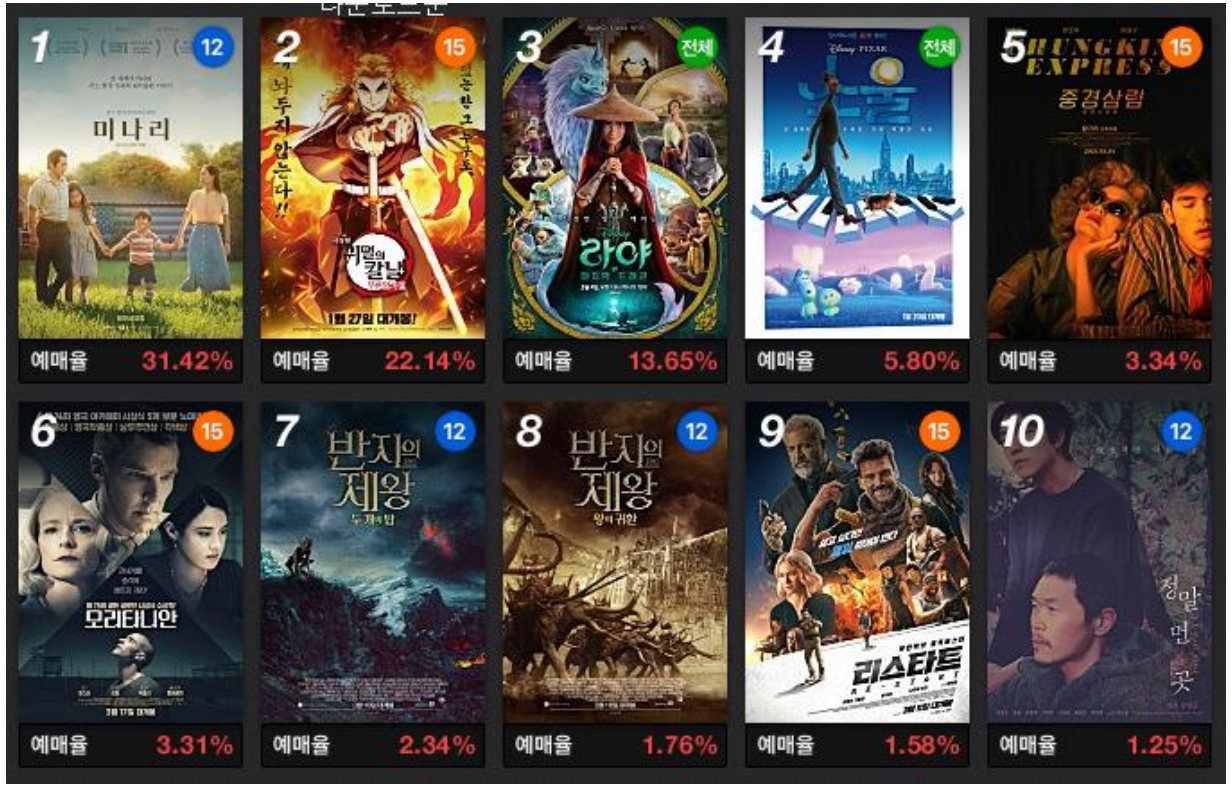
정말 먼 곳_positive, negative ratio, total 42



반지의 제왕 왕의 귀환_positive, negative ratio, total 6814



4. 실제 예매 순위와 비교



▲ 현재 개봉 중인 영화 예매율 상위 10편

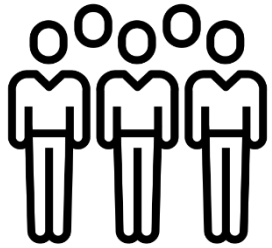
영화 제목	긍정(%)	부정(%)
소울	87.9	12.1
라야와 마지막 드레곤	84.4	15.6
모리타니안	83.7	16.3
정말 먼 곳	83.3	16.7
중경삼림	80.1	19.9
귀멸의 칼날	79.9	20.1
반지의 제왕 : 두개의 탑	79.2	20.8
반지의 제왕 : 왕의 귀환	78.6	21.4
미나리	70.9	29.1
리스타트	48.1	51.9

▲ 긍/부정 리뷰 비율로 재구성한 순위

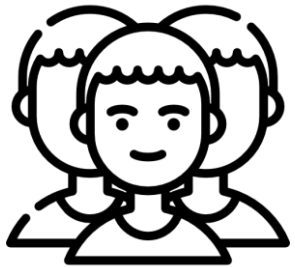
5

기대 효과

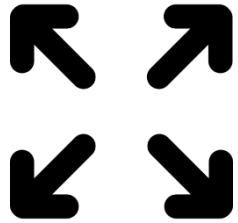
예상 수요자 : 관람객과 영화 배급사



영화에 대한 모집단 리뷰의 전체적인 감정 평가 지표 제공



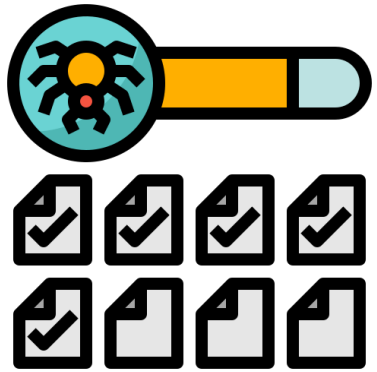
스포일러를 회피하고 간단한 평가만 알고 싶은 잠재적 관람객



한국 컨텐츠 배급을 고려하고 있는 해외 영화 배급사

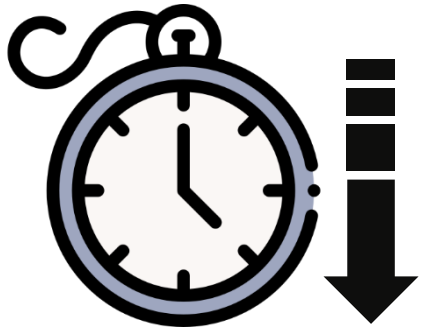
6

한계점과 개선 방안



1. 한정된 데이터와 낮은 정확도

- 컴퓨터 성능 한계로 모든 데이터를 다루지 못함
- 원본 데이터 안에 포함된 비꼬기, 반어법, 낚시성 리뷰, 영화와 상관 없는 리뷰 등을 걸러내지 못해 모델 정확도가 90%를 넘지 못함

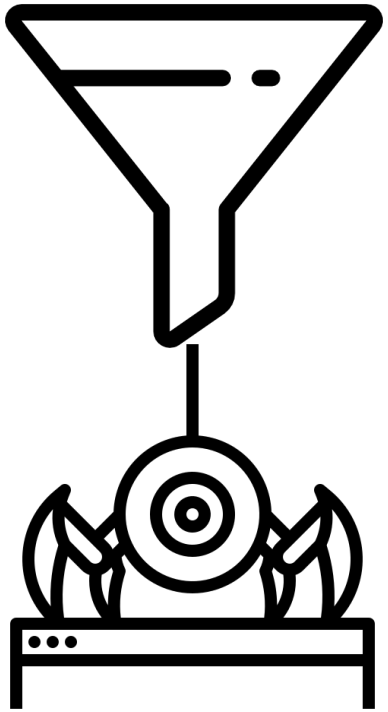


2. 낮은 시간 효율성

- 네이버 영화 HTML에 의존한다는 한계로 인해 발생
- 리뷰 게시판 마지막에서 자동적으로 멈출 수 없어서 임의로 무조건 천 페이지 정도 반복하게 만듬

3. 복잡한 파일 구성

- 쥬피터로 실행 시 학습된 모델의 가중치(.h5)와 감정 분석 코드 파일(.py)이 함께 있어야만 정상 작동
- .py 사용 시 가중치 파일만 있어도 되지만 라이브러리가 제대로 호출 안 될 수 있음



1. 데이터 필터링
- 수동으로 불필요하거나 부정확한 리뷰 제거
- 특정 단어(ex-평론가 이름) 필터링 강화
- 유명한 망작 영화는 크롤링 리스트에서 제거

2. 웹크롤링 기능 개선
- 네이버 영화 HTML 웹크롤링 방식 개선
- 기존 중복 리뷰 확인 방식에서 리뷰 리스트 페이지를 먼저 읽고 페이지 수 만큼 반복 크롤링 하도록 개선

3. ZIP 파일로 배포
- 만약 독립 프로그램으로 만들어서 배포 시
- Zip 파일로 만들어서 묶음으로 배포

THANK YOU

Q & A