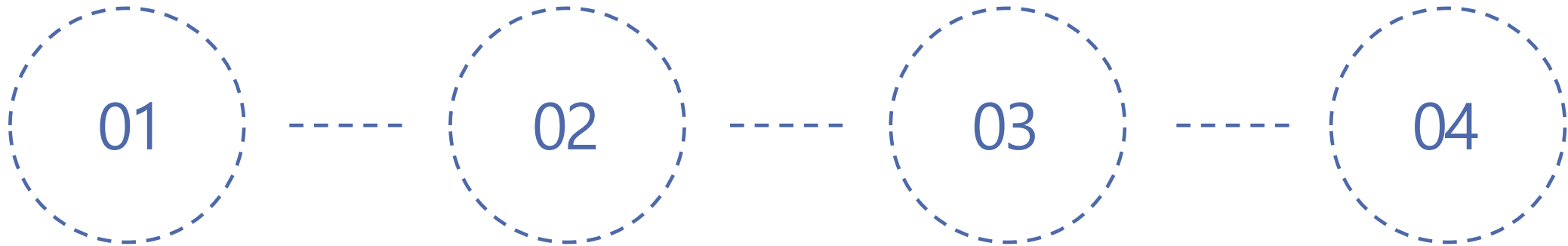


클러스터링과 LSTM모델을 통한 다음 흥행작 시나리오 방향성 제시





주제 선정

1. 리뷰데이터와 비즈니스
2. 주제선정
3. 프로젝트 진행 방향
4. 프로젝트 순서

데이터 프레임화

1. 문서내용 리스트에 저장
2. 파일명 가공 + 리스트에 저장
3. 최종 데이터 프레임 생성
4. 데이터 프레임 정렬

데이터 전처리 전략

1. 데이터 프레임 살펴보기
2. 데이터 전처리 전략 개요

데이터 전처리

1. 특수문자, 공백 제거
2. Wn정수 제거
3. 조회 수 column생성
4. 공감 수 column생성 + 필요 없는 글귀 제거



상위 25% 데이터 추출

1. $\frac{\text{공감수}}{\text{조회수}} \times 100$ column
Column 생성
2. 결측치 EDA
3. 결측치 처리
4. 상위 25% 추출

K-Means 적용

1. Tfidf vectorization
2. Lemmatization
3. 최적의 군집 수
4. 군집화 수행 및 결과

LSTM 모델로 Text 생성 및 시나리오 방향성 제시

1. 훈련데이터 구성
2. 모델 구성
3. Text 생성 및 시나리오 방향성 제시

LOGO

주제 선정

비주얼 프로그래밍

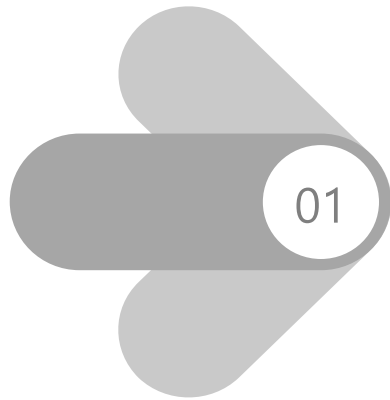
01

리뷰 분석

우리 브랜드와 상품에 대한 냉정한 피드백을
통해 소비자와 시장을 더 잘 이해할 수 있을
뿐만 아니라 상품 기획/개선, 마케팅 전략
등을 올바른 방향으로 잡아갈 수 있다.

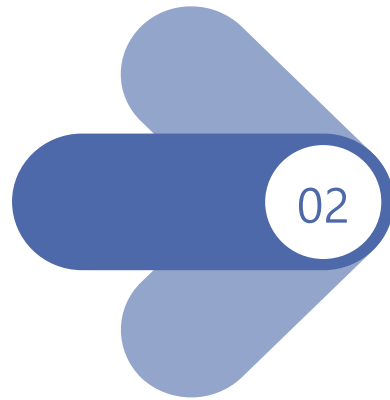
-> 의미 있는 정보를 비즈니스 전략에 반영할 수 있다.





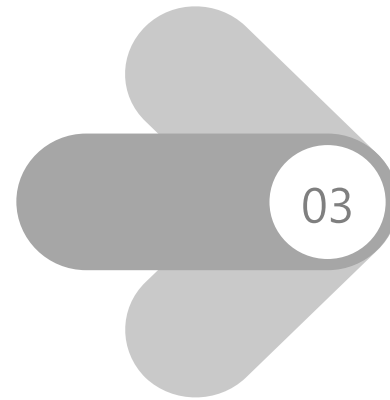
데이터(리뷰)

많은 사람들은 이미 영화를 본 사람들의 리뷰를 참고 함 -> IMDB(similar web 현재 Arts & Entertainment 부문 5위)



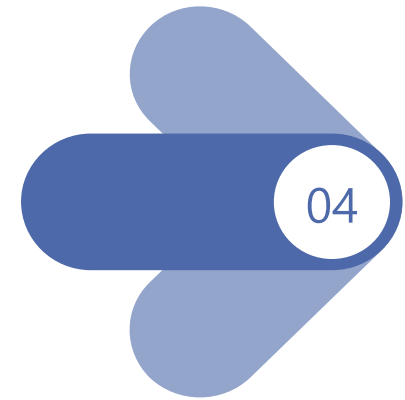
비즈니스적 목적 파악

다음에 제작할 드라마도 흥행할 수 있을까?



드라마의 흥행

목적=흥행

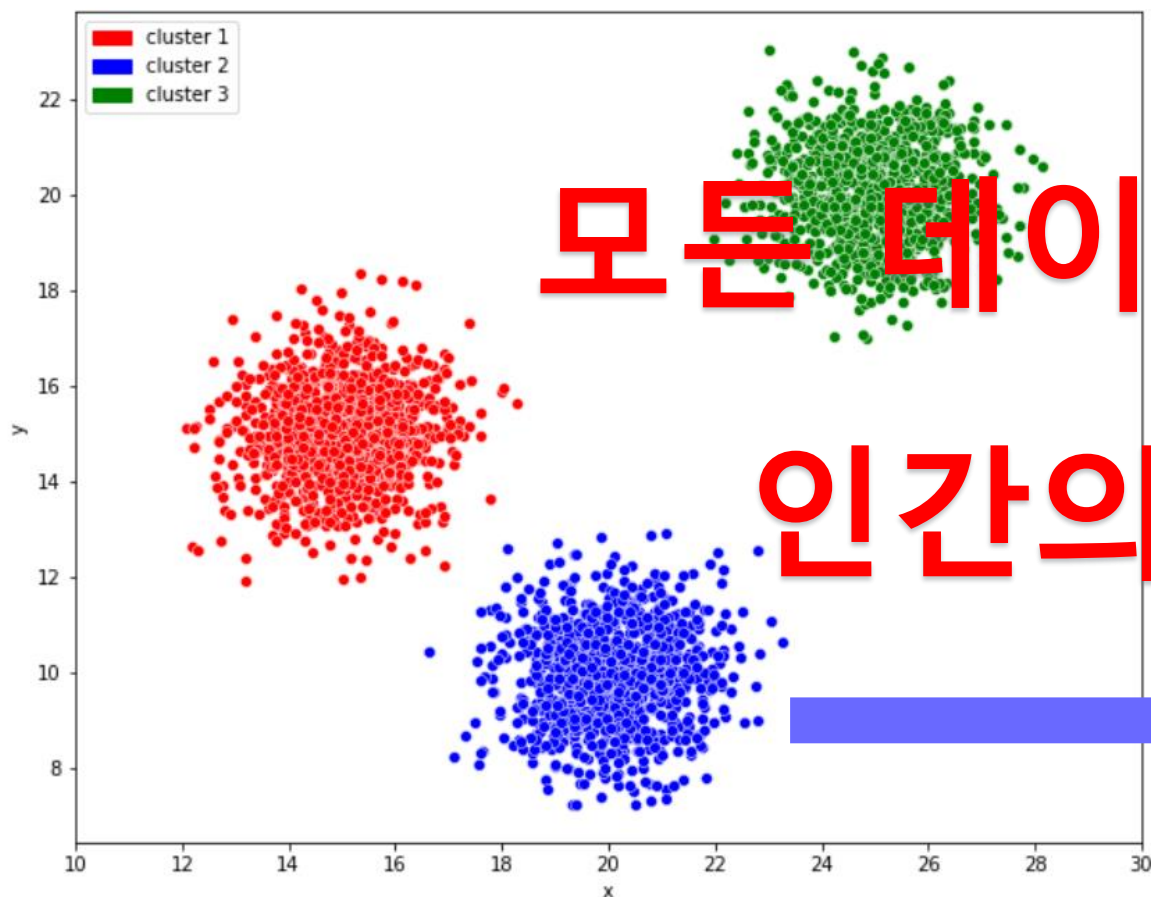


다음 흥행작 시나리오 방향성 제시

How?

K-Means

군집 중심점 이라는 특정한 임의의 지점을 선택해 해당 중심에 가장 가까운 포인트들을 선택하는 군집화 기법



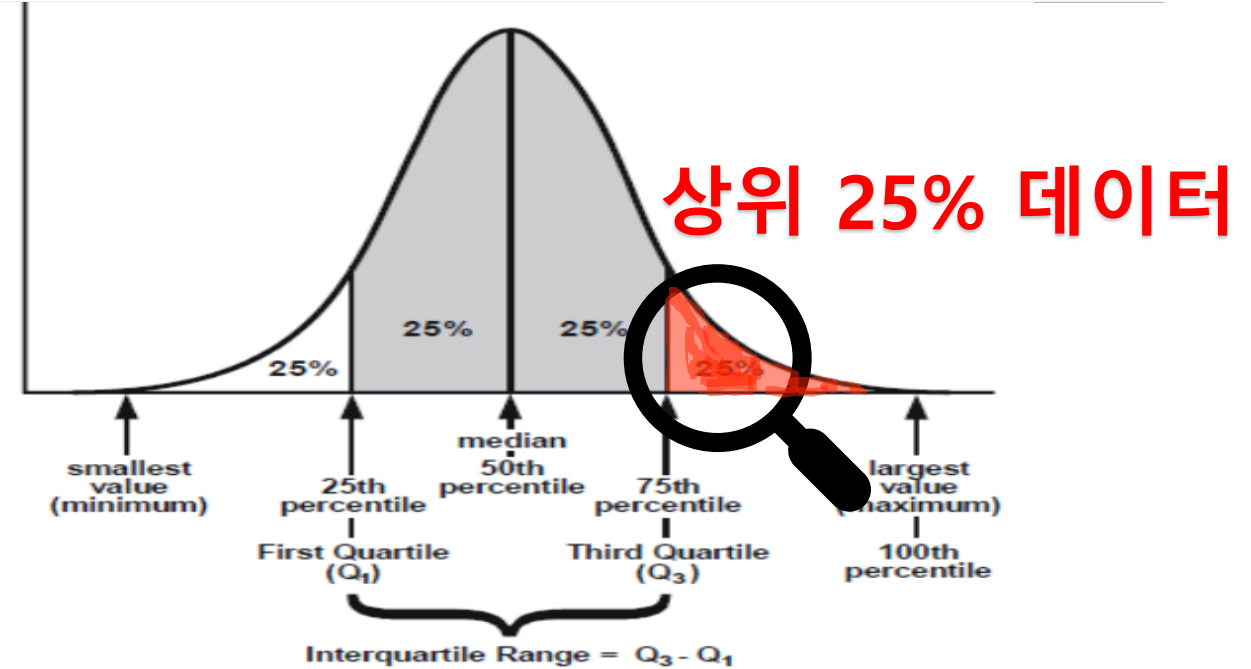
모든 데이터 = 가치?

인간의 해석?

good character
watch unique
seong ki-hun

상위 25%의 데이터

- 최신 자료의 경우 조회수가 적어 공감 수도 적은 형태를 보일 수 있음
- $\frac{\text{공감수}}{\text{조회수}} \times 100$ 으로 공감수가 적더라도 조회수와 비율을 반영
- $\frac{\text{공감수}}{\text{조회수}} \times 100$ 을 기준으로 상위 25%의 데이터를 분석에 반영



공감 수



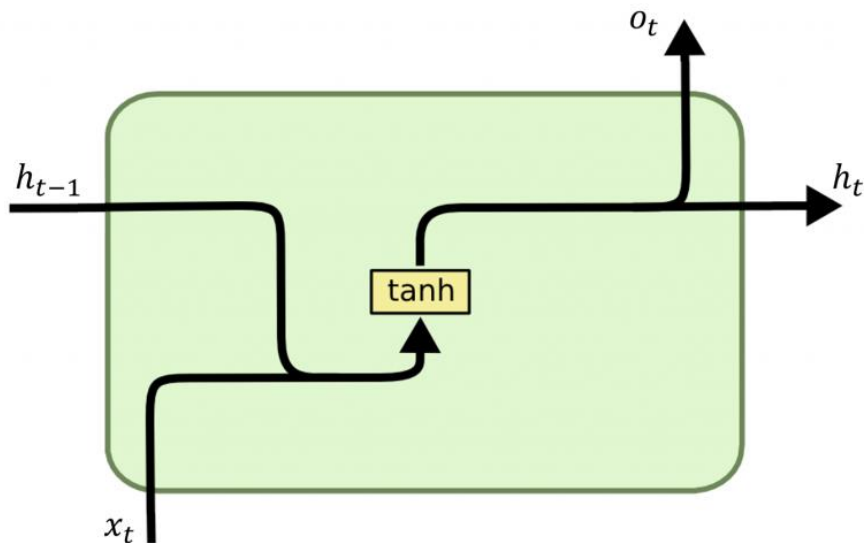
$\frac{\text{공감수}}{\text{조회수}} \times 100$

01 주제 선정

LOGO

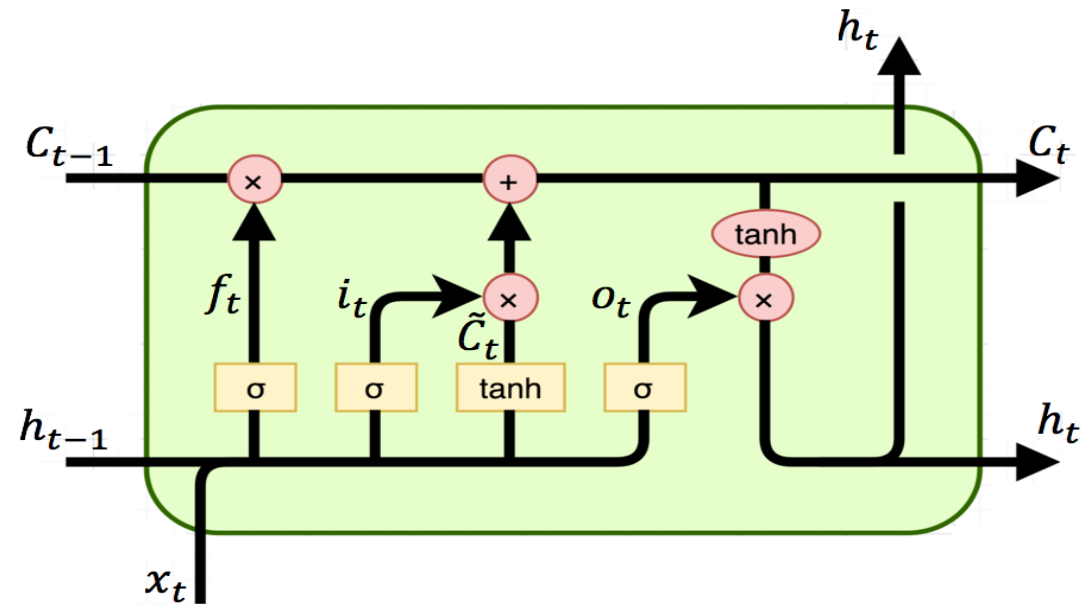
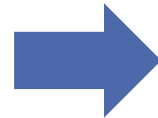
LSTM

RNN 모델에서 발생하는 기울기 소실 문제를 해결하기 위하여 제안된 모델로 어떠한 정보를 버릴지 결정하는데 사용되는 Forget Gate, 어떤 값을 업데이트 할 것인 지를 결정하는 Input Gate, 출력 값을 결정하는 Output Gate 구조를 거쳐서 Cell State에 정보가 저장됨



RNN

- 한 층의 출력이 순환하며 다시 입력되는 재귀 결합을 가진 신경망
- 현재 상태의 정보는 바로 다음 상태로만 전달 이 가능하며, 긴 데이터에 대하여 학습 할 경우 처음 정보가 뒤로 갈 수록 전달되지 않는 **기울기 소실 문제**가 발생할 수 있음



LSTM

- Cell State를 통하여 중요한 정보와 필요 없는 정보를 구분하여 사용하기 때문에 **장기 기억을 보존**할 수 있음

데이터 프레임화

- 폴더에서 데이터 불러오기
- 파일명 순으로 데이터 프레임 정렬

데이터 전처리

- 조회 수, 공감 수 column 생성
- $\frac{\text{공감수}}{\text{조회수}} \times 100$ column 생성

상위 25%데이터

- $\frac{\text{공감수}}{\text{조회수}} \times 100$ column 기준으로 상위 25%의 데이터 추출

K-Means

- 추출한 상위 25%의 데이터들로 군집화
- 군집별 단어 분석
- 해당하는 군집 column생성

LSTM

- 군집별로 LSTM 모델에 넣어서 학습
- 군집에서 가장 유의미한 단어 넣어서 Text생성

LOGO

데이터 프레임화

비주얼 프로그래밍

02

기본 데이터 프레임

filename		review_text
0	0	...
1	1	...
2	2	...
3	3	...
4	4	I honestly don't know wtf the main character...
...
259	288	...
260	301	Not original at all but refreshing to see th...
261	316	...
262	331	It is disturbing in a way that why the hell ...
263	346	DefinitelyThe direct...

264 rows × 2 columns



문서내용 리스트에 저장

지정된 경로에 있는 파일들 데이터 프레임화



파일명 가공 + 리스트에 저장

절대 경로로 주어진 파일명을 가공



최종 데이터 프레임 생성

파일명 리스트와 파일 내용 리스트를 데이터 프레임으로 생성



데이터 프레임 정렬

데이터 프레임을 파일명 순으로 정렬시킴

```
import pandas as pd
import glob, os

path = r'C:\Users\Tony\Desktop\3-2기말\비주얼 기말\프로젝트\Squid game_reviews\Squid game_reviews\reviews\review'
# path로 지정한 디렉토리 밑에 있는 모든 .txt 파일들의 파일명을 리스트로 취합
all_files = glob.glob(os.path.join(path, "*.txt"))
```



```
for file_ in all_files:
    # 개별 파일을 읽어서 df생성
    df = pd.read_table(file_, index_col=None, header=0, encoding='latin1')
```



```
review_text.append(df.to_string())
```

☑ 문서내용 리스트에 저장

- 파일이 있는 경로를 지정하고 .txt파일들의 파일명을 all_files에 넣는다.(경로형태까지 파일명에 들어간다.)
- all_files안의 file들을 read_table로 읽어서 df에 저장한다.
- df를 string형태로 옮겨서 review_text 리스트에 추가한다.

☑ 파일명 가공 + 리스트에 저장

☑ 최종 데이터 프레임 생성

☑ 데이터 프레임 정렬

```
for file_ in all_files:
```



```
# 절대경로로 주어진 file 명을 가공  
filename_ = file_.split('\\\\\\\\')[-1]  
filename = filename_.split('.')[0]
```



```
filename_list.append(filename)
```

① 문서내용 리스트에 저장

② 파일명 가공 + 리스트에 저장

- 절대 경로로 주어진 file명을 가공한다. '\\\\\\\\'단위로 끊으면 마지막 리스트에 ex> 0.txt 이런 형태로 파일이름이 남는다.
- 0.txt에서 '.'단위로 끊고 첫번째 리스트 요소를 파일이름으로 한다.
- 그 파일이름을 filename_list에 저장한다.

③ 최종 데이터 프레임 생성

④ 데이터 프레임 정렬

02 데이터 프레임화

LOGO

```
# 파일명 리스트와 파일내용 리스트를 DataFrame으로 생성
```

```
document_df = pd.DataFrame({'filename':filename_list, 'review_text':review_text})
```



document_df

[:]

	filename	review_text
0	0	...
1	1	...
2	10	...
3	100	As survival series go, "Squid Game" brought...
4	101	...
...
259	95	Terrible performances from t...
260	96	The show has an interesting storyline - you ...
261	97	...
262	98	Netflix K-drama might offer nothing new as ...
263	99	Dont listen to twitter on this show its not ...

264 rows × 2 columns

✓ 문서내용 리스트에 저장

✓ 파일명 가공 + 리스트에 저장

✓ 최종 데이터 프레임 생성

- filename_list와 review_text를 데이터프레임으로 생성한다.

✓ 데이터 프레임 정렬

02 데이터 프레임화

LOGO

```
document_df['filename']=document_df['filename'].astype(int)
```



```
document_df=document_df.sort_values(by='filename')
```



```
document_df.reset_index(inplace=True, drop=True)
```



	filename	review_text
0	0	...
1	1	...
2	2	...
3	3	...
4	4	I honestly don't know wtf the main character...
...
259	288	...
260	301	Not original at all but refreshing to see th...
261	316	...
262	331	It is disturbing in a way that why the hell ...
263	346	DefinitelyThe direct...

264 rows × 2 columns

- ☑ 문서내용 리스트에 저장
- ☑ 파일명 가공 + 리스트에 저장
- ☑ 최종 데이터 프레임 생성
- ☑ 데이터 프레임 정렬

- filename의 type을 int형으로 바꾼다.
- filename이라는 column을 기준으로 오름차순 정렬한다.
- Index값을 초기화하고 기존의 index는 제거한다.

데이터 전처리 전략

비주얼 프로그래밍

03

데이터프레임 살펴보기

₩정수 + 특수문자 & 공백

- 데이터 마다 ₩정수 값이 들어간다. Table로 읽어오면서 생긴 것으로 보인다.
- 또한 텍스트 처리를 하기전에 특수문자나 공백을 어떻게 처리할 것인지 결정해야 한다.



Just finished up SQUID GAME, the Korean TV show that's become a global smash hit on Netflix. It's truly brilliant and even better than the similar Japanese show ALICE IN BORDERLAND! The show is centred around contestants competing in children's games for a cash prize, but the twist is that they get killed if they lose. Shades of SAW and THE HUNGER GAMES here, but this is rivalled only by BATTLE ROYALE in terms of sheer quality. As usual for Korea, the writing, acting, and direction are all superb and the games themselves are hugely suspenseful, unlike anything I've seen before. Definitely horrifying enough both in premise and in terms of violent bloodshed to count as horror too.

조회 수와 공감 수 + 필요 없는 글귀 처리

- 조회 수와 공감 수 만 뽑아서 2개의 column을 생성한다.
- 남은 부분은 전부 제거한다.

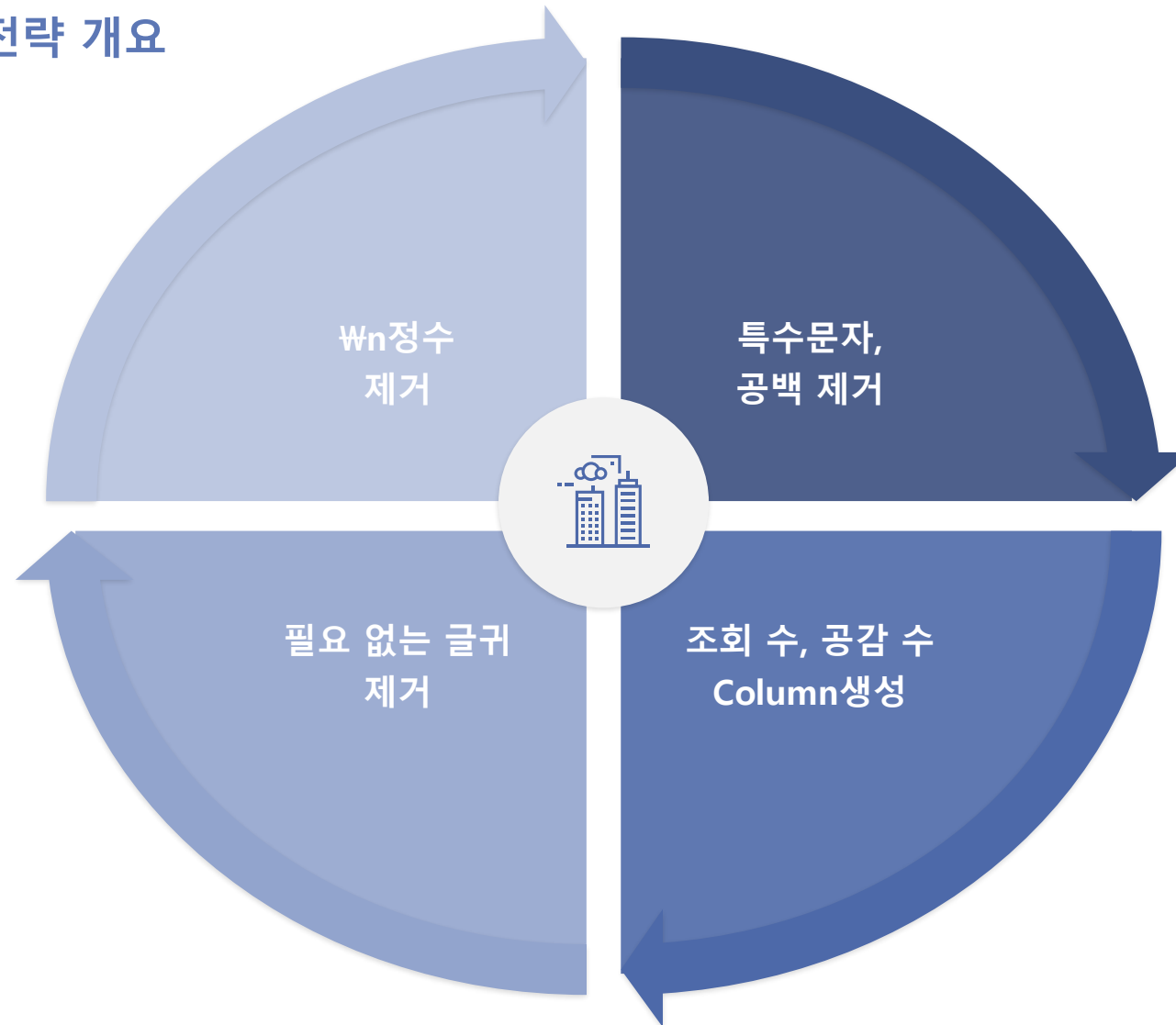


67 out of 144 found this helpful. Was this review helpful? Sign in to vote. Permalink"

03 데이터 전처리 전략

LOGO

데이터 전처리 전략 개요



데이터 전처리

비주얼 프로그래밍

04

```
from nltk.tokenize import word_tokenize
print('단어 토큰화: ', word_tokenize(document_df['review_text'][0]))
```

단어 토큰화1: ['Just', 'finished', 'up', 'SQUID', 'GAME', ',', 'the', 'Korean', 'TV', 'show', 'that', '"', 's', 'become', 'a', 'global', 'smash', 'hit', 'on', 'Netflix', 'O', 'It', '"', 's', 'truly', 'brilliant', 'and', 'even', 'better', 'than', 'the', 'similar', 'Japanese', 'show', 'ALICE', 'IN', 'BORDERLAND', '!', 'The', 'show', 'is', 'centred', 'around', 'contestants', 'competing', 'in', 'children', '"', 's', 'games', 'for', 'a', 'cash', 'prize', ',', 'but', 'the', 'twist', 'is', 'that', 'they', 'get', 'killed', 'if', 'they', 'lose', 'I', 'Shades', 'of', 'SAW', 'and', 'THE', 'HUNGER', 'GAMES', 'here', ',', 'but', 'this', 'is', 'rivalled', 'only', 'by', 'BATTLE', 'ROYALE', 'in', 'terms', 'of', 'sheer', 'quality', '2', 'As', 'usual', 'for', 'Korea', ',', 'the', 'writing', ',', 'acting', ',', 'and', 'direction', 'are', 'all', 'superb', 'and', 'the', 'games', 'themselves', 'are', 'hugely', 'suspenseful', ',', 'unlike', 'anything', 'I', 've', 'seen', 'before', '3', 'Definitely', 'horrifying', 'enough', 'both', 'in', 'premise', 'and', 'in', 'terms', 'of', 'violent', 'bloodshed', 'to', 'count', 'as', 'horror', 'too', '4', '67', 'out', 'of', '144', 'found', 'that', 'is', 'helpful', '5', 'Was', 'this', 'review', 'helpful', '?', 'Sign', 'in', 'to', 'vote', '6', 'Permalink']

✓ 특수문자, 공백 제거

- word_tokenize가 단어를 토큰화 하면서 공백을 고려해서 토큰화 한다.
- 또한 특수문자는 나중에 Lemmatization 함수를 따로 구현하여 string.punctuation을 이용하여 제거할 것이다.

✓ Wn정수 제거

✓ 조회수 column 생성

✓ 공감 수 column 생성 + 필요 없는 글귀 제거

04 데이터 전처리

LOGO

```
def delenter(data):
    loc=data.find('\n')#데이터의 \n으로 바꾸기 위해 저장
    for i in range(99,-1,-1):
        part=data[loc]
        target=part+str(i) # \n정수 이렇게 target값이 만들어짐
        if(data.find(target)!=-1): #target이 있으면 삭제시킴
            data=data.replace(target,'')
    data=data.replace(' ','')#텍스트 100의 문제
    return data
```



```
new=delenter(document_df['review_text'][0])
new
```

"

Just finished up SQUID GAME, the Korean TV show that's become a global smash hit on Netflix. It's truly brilliant and even better than the similar Japanese show ALICE IN BORDERLAND! The show is centred around contestants competing in children's games for a cash prize, but the twist is that they get killed if they lose.

Shades of SAW and THE HUNGER GAMES here, but this is rivalled only by BATTLE ROYALE in terms of sheer quality.

As usual for Korea, the writing, acting, and direction are all superb and the games themselves are hugely suspenseful, unlike anything I've seen before.

Definitely horrifying enough both in premise and in terms of violent bloodshed to count as horror too.

67 out of 144 found this helpful.

Was this review helpful? Sign in to vote.

Permalink"

✓ 특수문자, 공백 제거

✓ \n정수 제거

- delenter함수를 생성
- 파일의 \n이 세 자리 수는 나오지 않으므로 두 자리 수 정수 부터 “\n”문자에 연결시켜서 연결 값이 data에 있으면 제거한다.
- ₩ 특수기호 전부 제거한다.(text 100의 문제)

✓ 조회 수 column 생성

✓ 공감 수 column 생성 + 필요 없는 글귀 제거

```
import numpy as np
views=[]
def makeviews(data):
    global views
    loc=data.rfind("out of")# out of는 문장구조상 나올 수도 있지만 앞에 정수가 붙어서 나오면 거의 100퍼센트 확실하다고 보았다.
    if(loc!=-1):
        temp=[]
        loc+=7# 공백 포함해서 제일 뒷숫자 찾기
        if(48<=ord(data[loc])<=57):# 만약 뒷자리 수가 정수라면 위치 하나씩 늘리면서 뒷칸으로 이동시킴
            while((48<=ord(data[loc])<=57) & (loc<=len(data)-2)): #만약 0~9사이 정수면 temp배열에 추가 & loc가 데이터의 끝 직전까.
                temp.append(data[loc])
                loc+=1
            org=""
            temp.reverse()# reverse해주어야 선입선출구조로 뺄 수 있음
            for i in range(0,len(temp)): # 배열에 저장한거 pop으로 빼서 문자열로 만들어 주면 원래 숫자의 형태로 나온다.
                org+=temp.pop()
            print(org)
            views.append(int(org))
            return data
        else:# 만약 뒷자리수가 정수 아니면 문장 구조상의 out of다 따라서 append np.nan해줌
            views.append(np.nan)
            return data
    else:
        views.append(np.nan)
        return data
```



```
document_df['review_text']=document_df['review_text'].apply(makeviews)
```

✓ 특수문자, 공백 제거

✓ Wn정수 제거

✓ 조회 수 column 생성

- makeviews함수 생성

- “out of”라는 글귀의 위치 저장 하고 +7의 위치 부터의 값이 정수인지 계산

- 정수이고 문자의 위치가 데이터의 끝이 아니라면 리스트에 넣었다가 reverse하고 pop해서 합친 상태로 views 리스트에 넣는다.(reverse해야 선입선출구조)

- 뒷자리수가 정수 아니면 문장 구조상의 out of이므로 np.nan을 append

- 만약 “out of”이라는 글귀가 없으면 np.nan을 저장함(결측치)

✓ 공감 수 column 생성 + 필요 없는 글귀 제거

04 데이터 전처리

LOGO

```
len(views)
```

264

=

```
len(document_df)
```

264



	filename	review_text	views
0	0	...	144.0
1	1	...	61.0
2	2	...	54.0
3	3	...	NaN
4	4	I honestly don't know wtf the main character...	228.0
...
259	288	...	3.0
260	301	Not original at all but refreshing to see th...	9.0
261	316	...	26.0
262	331	It is disturbing in a way that why the hell ...	4.0
263	346	DefinitelyThe direct...	17.0

264 rows × 3 columns

① 특수문자, 다수 공백 제거

② Wn정수 제거

③ 조회 수 column 생성

- views리스트와 document_df(행)의 길이의 수가 같음을 확인
- document_df에 views 열을 추가함

④ 공감 수 column 생성 + 필요 없는 글귀 제거


```
empathy=[]
def makeempathy(data):
    global empathy
    loc=data.rfind("out of")# out of는 문장구조상 나올 수도 있지만 앞에 정수가 붙어서 나오면 거의 100퍼센트 확실하다고
    if(loc!=-1):
        temp=[]
        loc-=2# 공백 포함해서 제일 뒷숫자 찾기
        if(48<=ord(data[loc])<=57):# 만약 앞자리 수가 정수라면
            while(48<=ord(data[loc])<=57): #위치 하나씩 줄이면서 앞칸으로 이동시킴 만약 0~9사이 정수면 temp배열에 추가
                temp.append(data[loc])
                loc-=1
            org=""
            for i in range(0,len(temp)): # 배열에 저장한거 pop으로 빼서 문자열로 만들어 주면 원래 숫자의 형태로 나온다.
                org+=temp.pop()
            print(org)
            empathy.append(int(org))
            data=data[:loc] #뒤에 전부제거
            return data
        else:# 만약 앞자리수가 정수 아니면 문장 구조상의 out of다 따라서 append 0해줌
            empathy.append(np.nan)
            return data
    else:
        empathy.append(np.nan)
        return data
```




```
document_df['review_text']=document_df['review_text'].apply(makeempathy)
```

- ① 특수문자, 공백 제거
 - ② Wn정수 제거
 - ③ 조회 수 column 생성
 - ④ 공감 수 column생성 + 필요 없는 글귀 제거
- makeempathy라는 함수 생성
 - "out of"의 위치를 loc변수에 저장한다.
 - loc-2인 부분이 만약에 정수이면 하나씩 리스트에 넣는다. 그리고 pop해서 연결하여 empathy에 저장한다.
 - 만약 loc-2부분이 정수가 아니라면 ,이는 문장 구조상의 "out of"이므로 np.nan을 empathy에 저장한다.
 - "out of"자체가 없어도 np.nan을 empathy에 저장한다.
 - data=data[:loc]로 마지막 정수위치부터 전부 제거한다.

04 데이터 전처리

LOGO

`len(empathy)` = `len(document_df)`
264 = 264



filename		review_text	views	empathy
0	0	...	144.0	67.0
1	1	...	61.0	34.0
2	2	...	54.0	21.0
3	3	...	NaN	NaN
4	4	I honestly dont know wtf the main character ...	228.0	188.0
...
259	288	...	3.0	0.0
260	301	Not original at all but refreshing to see th...	9.0	2.0
261	316	...	26.0	17.0
262	331	It is disturbing in a way that why the hell ...	4.0	2.0
263	346	DefinitelyThe direct...	17.0	5.0

264 rows × 4 columns

- ✓ 특수문자, 공백 제거
 - ✓ Wn정수 제거
 - ✓ 조회 수 column 생성
 - ✓ 공감 수 column 생성 + 필요 없는 글귀 제거
- empathy리스트와 document_df(행)의 길이의 수가 같음을 확인
 - document_df에 empathy 열을 추가함
 - NaN값은 같은 행에 존재

상위 25% 데이터 추출

비주얼 프로그래밍

05

결측치 확인

- IMDB 사이트에서 확인해보니 모든 글에 공감과 조회수가 있었음
->크롤링 과정에서의 문제로 보임
- 결측치의 비율이 12.5%로 아주 높은 편은 아니지만 삭제해서도 안될 수치임
- 어떤 것을 기준으로 결측치를 예측할 수 있을까?

filename		review_text	views	empathy
3	3	...	NaN	NaN
9	9	...	NaN	NaN
21	21	...	NaN	NaN
23	23	...	NaN	NaN
36	36	...	NaN	NaN
39	39	...	NaN	NaN
49	49	...	NaN	NaN
55	55	...	NaN	NaN
63	63	I guess I got sucked into...	NaN	NaN
64	64	While the series deserves its enthusiastic r...	NaN	NaN
71	71	...	NaN	NaN

$$\text{결측치의 비율} : \frac{33}{264} \times 100 = 12.5\%$$

VADER

주로 소셜 미디어의 텍스트에 대한 감성 분석을 제공하기 위한 패키지로 뛰어난 감성 분석 결과를 제공하며 비교적 빠른 수행 시간을 보장해 대용량 텍스트 데이터에 잘 사용됨

```
def vader_polarity(review, threshold=0.1):
    analyzer = SentimentIntensityAnalyzer()
    scores = analyzer.polarity_scores(review)

    agg_score = scores['compound']
    final_sentiment = 1 if agg_score >= threshold else 0
    return final_sentiment

document_df['Sentiment'] = document_df['review_text'].apply(lambda x: vader_polarity(x, 0.4))
```

neg & pos

- 부정 지수, 긍정 지수

neu

- 중립적인 감성 지수

compound

- Neg, pos, neu 적절히 조합 (-1 ~ 1)
- 보통 0.1이상이면 긍정
->사용자가 임계값 적절히 조절
->0.4로 설정(부정의 비율이 많이 부족)

05 상위 25% 데이터 추출

LOGO

```
test_df['EV']=test_df['empathy']/test_df['views']*100
```



```
test_df['EV'].quantile([.75])
```

```
: 0.75      57.894737  
   Name: EV, dtype: float64
```

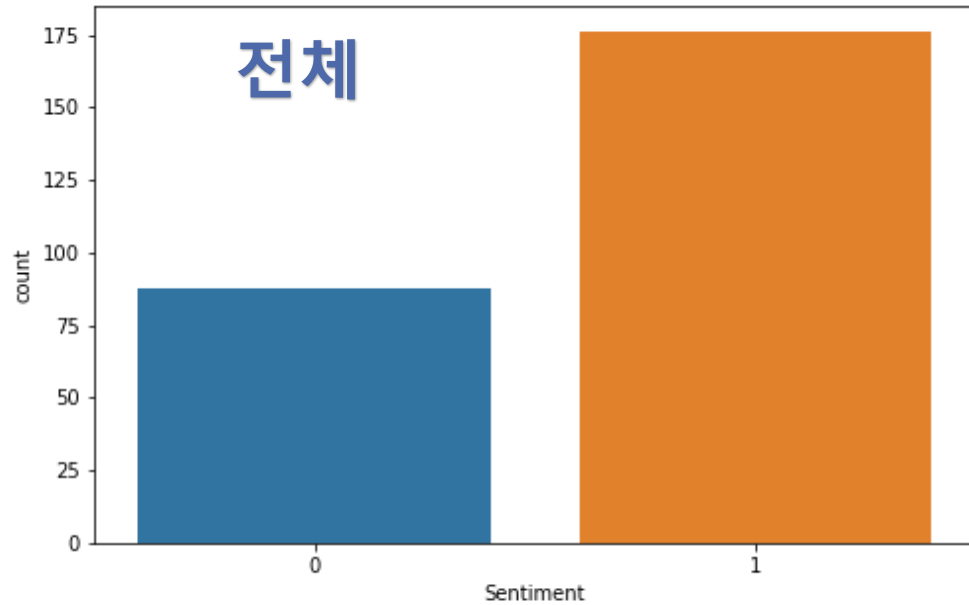
```
high=test_df[test_df['EV']>=57.894737]
```

EV값 상*하위 25% 데이터 추출

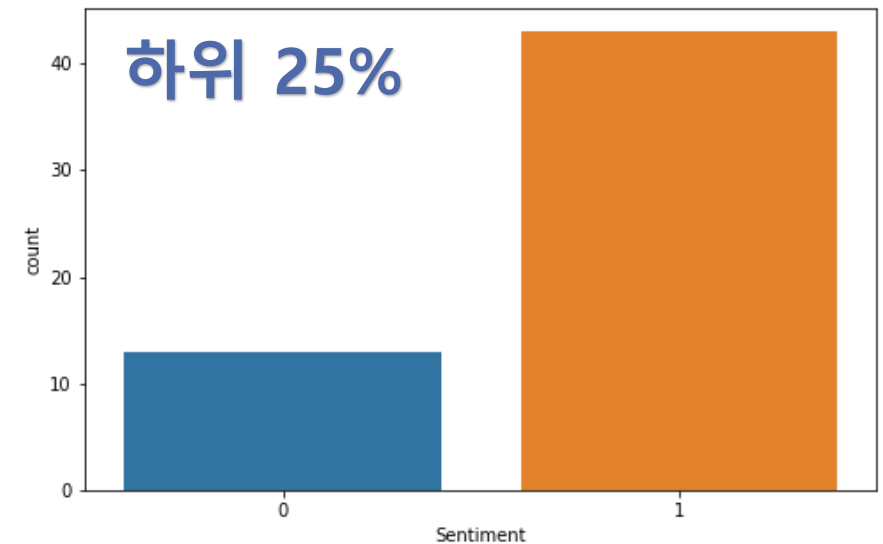
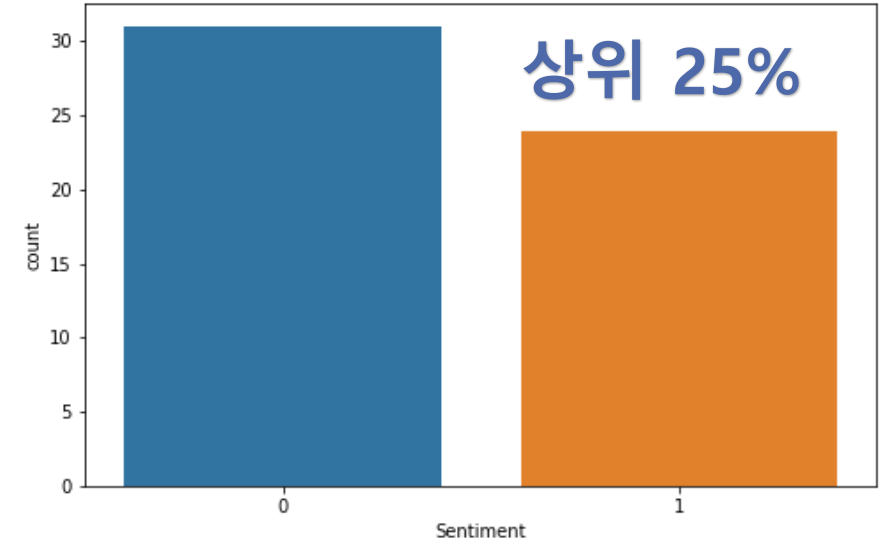
- $\frac{\text{공감수}}{\text{조회수}} \times 100$ 을 **EV column** 으로 생성
- Quantile로 상위 25%의 데이터들만 high 변수로 추출
- 하위 25%도 같은 방법으로 추출

05 상위 25% 데이터 추출

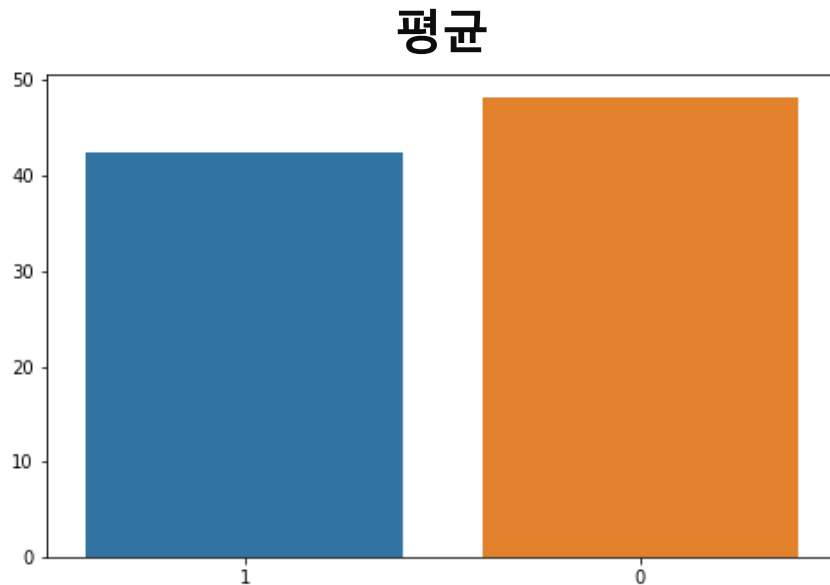
LOGO



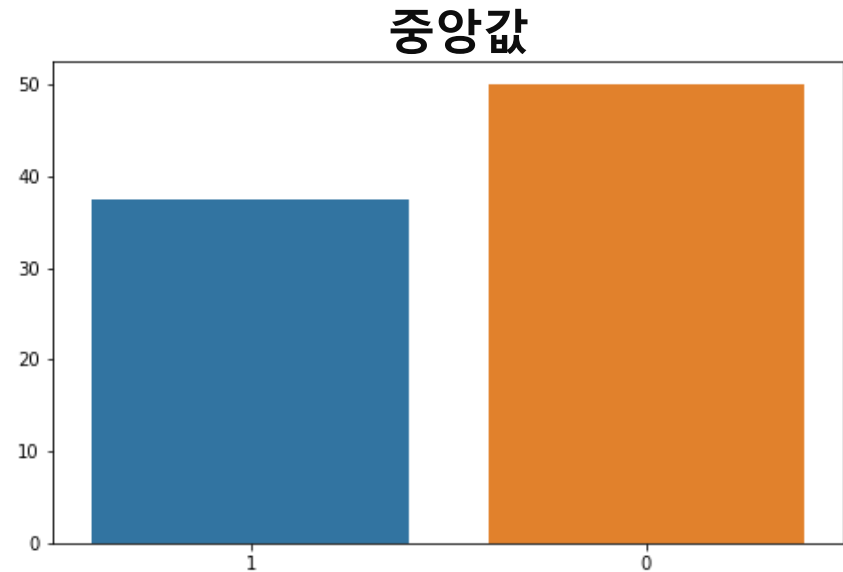
- ① 전체 데이터는 긍정지수가 약 2배정도 많음
- ② EV 상위 25%에서는 부정지수가 더 많음
- ③ EV 하위 25%는 긍정지수가 압도적으로 많음



긍정, 부정 별 EV의 평균과 중앙값 비교



VS



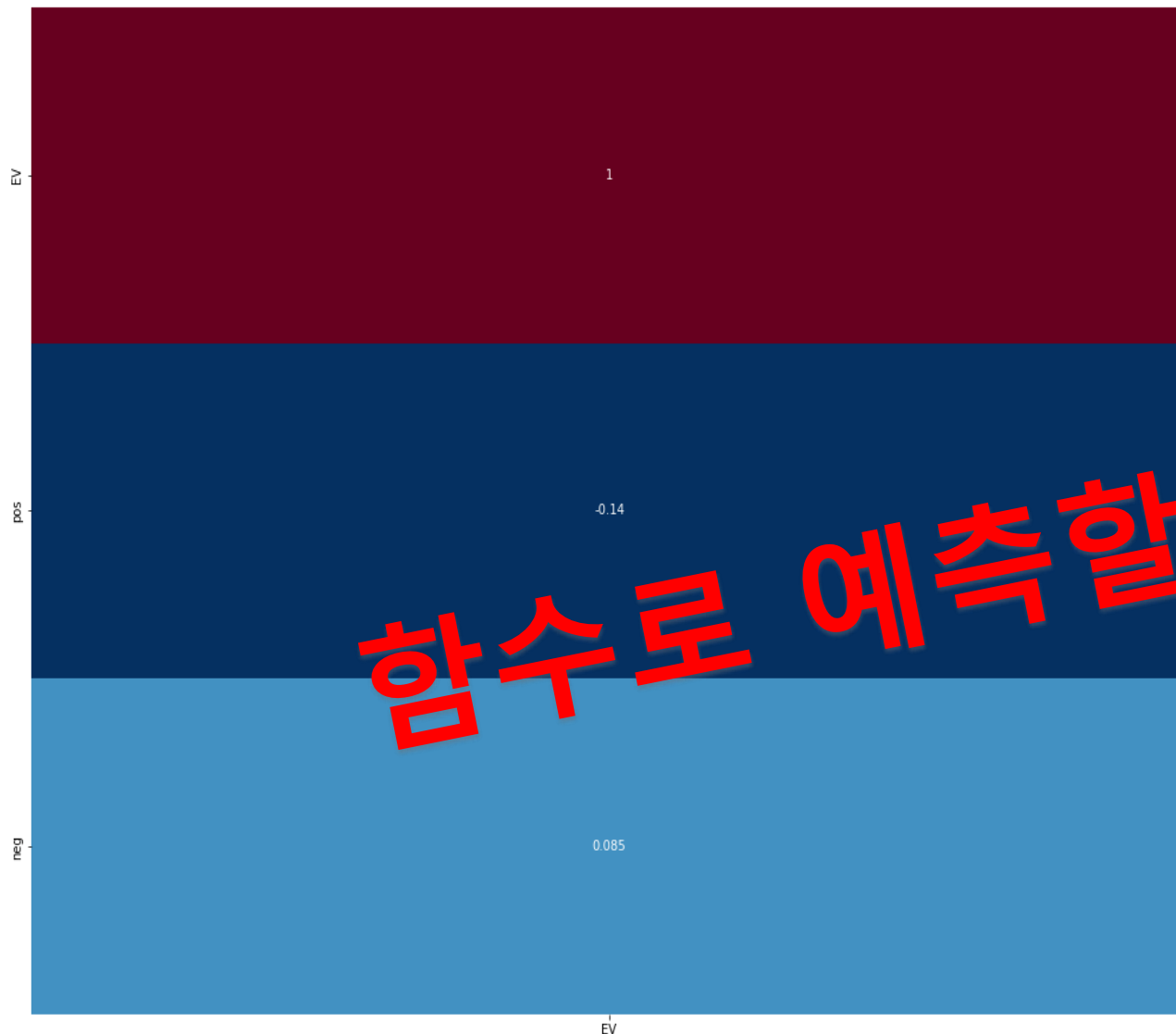
수학적인 수치?

- ① EV의 평균은 부정이 높음
- ② EV의 중앙값은 부정이 높음

③ 앞의 내용 고려하면 EV 값은 **부정이 대체적으로 높을 것임을** 추측할 수 있음

05 상위 25% 데이터 추출

LOGO



EV와 긍정, 부정 지수와의 상관관계

- EV와 긍정, 부정 지수는 상관관계가 없음
- EV와 긍정, 부정의 지수는 선형관계가 없음

LinearRegression()MSE : 460.227, RMSE:21.453
LinearRegression()R2 score : -20.184
Ridge()MSE : 471.847, RMSE:21.722
Ridge()R2 score : -45.872
Lasso()MSE : 510.996, RMSE:22.605
Lasso()R2 score : 0.000

결측치 처리 및 상위 25% 데이터 추출



0/0이 NA로 채워진 부분은 0으로 채움

- 조회수와 공감 수가 모두 0인 경우에 나눗셈 연산을 한 경우 결측치가 EV에 들어감
- 따라서 0으로 초기화



결측치의 sentiment로 결측치 예측

- 결측치의 감성 column이 긍정이면 긍정의 중앙값으로 채움
- 결측치의 감성 column이 부정이면 부정의 중앙값으로 채움



상위 25% 데이터 추출

K-Means 적용

비주얼 프로그래밍

06

TFIDF

특정 단어가 문서 내에 등장(가중치)하는 빈도와 그 단어가 문서 전체 집합에서 등장(패널티)하는 빈도를 고려하여 벡터화 하는 방법

	bed	cat	dog	face
문서 A	0	0.116	0	0.116
문서 B	0.116	0	0.116	0

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vect = TfidfVectorizer(tokenizer=LemNormalize, stop_words=stop_words, ngram_range=(1,2), min_df=0.05, max_df=0.85)
feature_vect = tfidf_vect.fit_transform(document_df['review_text'])
```

ngram_range

문맥상의 의미 반영

Ex> I love cat

->(I, love), (love, cat)

stop_words

빈번히 등장하는 의미 없는 단어

들 ex> is, the a, will 등

max_df & min_df

DF: 특정 단어가 나타나는 문서의 수

max_df: 최대 이 정도 까지 허용

min_df: 최소 이 정도는 있어야 함

```
stop_words.append('')  
def LemTokens(tokens): #입력으로 들어온 token 단어들에 대해 lemmatization 어근 변환  
    return [lemmar.lemmatize(token) for token in tokens]  
def LemNormalize(text): # tokenize한다 -> 1) 소문자로 먼저 만든다 -> 2)  
    return LemTokens(nltk.word_tokenize(text.lower().translate(remove_punct_dict)))
```

Lemmatization(어근 추출)

LemNormalize 함수

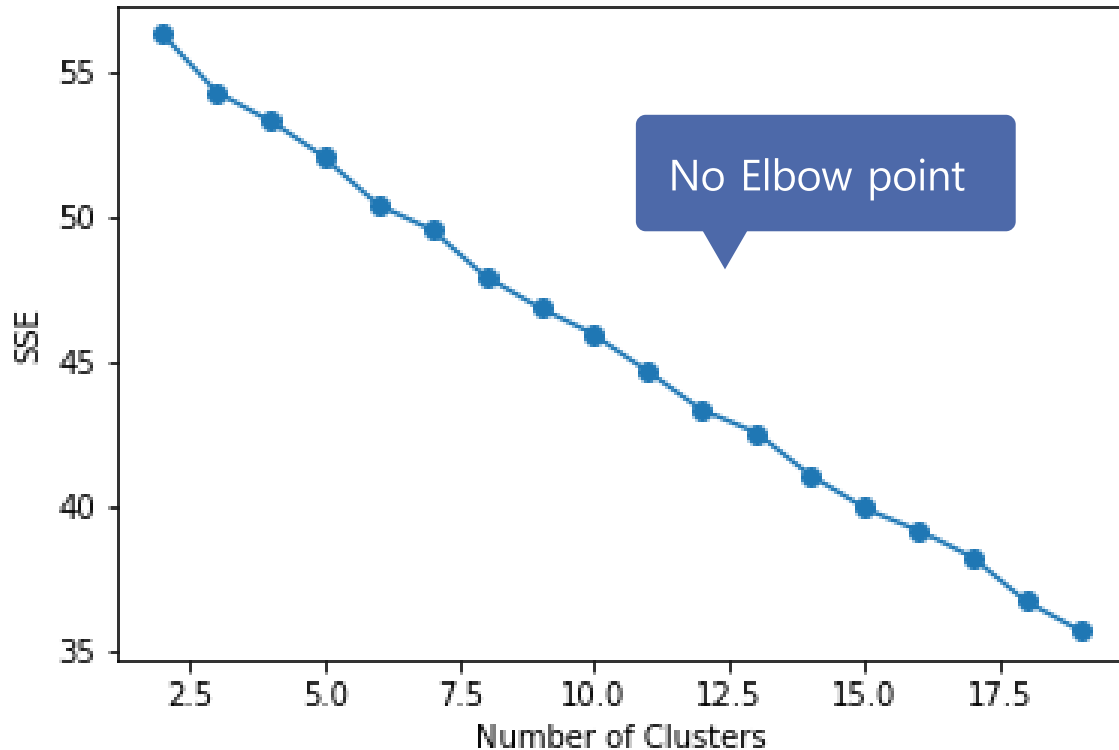
- 1) 소문자로 변환
- 2) 특수기호 제거
- 3) 토큰화

LemTokens 함수

- 1) 어근 추출(lemmatize)
- 2) 배열에 넣고 반환

최적의 군집 수 찾기

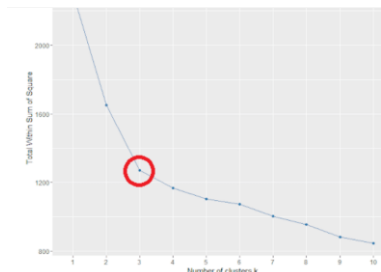
1. Elbow Method



```
import matplotlib.pyplot as plt
def elbow(feature_vect):
    sse=[]
    for i in range(2, 20):
        km_cluster = KMeans(n_clusters=i, max_iter=10000, random_state=42)
        km_cluster.fit(feature_vect)
        sse.append(km_cluster.inertia_)
    plt.plot(range(2,20), sse, marker='o')
    plt.xlabel('Number of Clusters')
    plt.ylabel('SSE')
    plt.show()
    elbow(feature_vect)
```

01. Elbow point

- Inertia: cluster에 속한 샘플들의 거리 제곱 합 -> 낮을수록 좋음
- 어느 한 지점 이후로 성능이 크게 개선되지 않음(팔처럼 굽어짐)



최적의 군집 수 찾기 2. Silhouette Analysis

$$s(i) = \frac{b(i) - a(i)}{\max \{a(i), b(i)\}}$$

02. Silhouette Analysis

- 실루엣 계수: 각각의 데이터가 같은 군집 내의 데이터와는 얼마나 가깝게 군집화 되었고, 다른 군집에 있는 데이터와는 얼마나 멀리 분포 되어 있는지를 나타내는 지표

-> 종합적인 지표(높을수록 좋음)

```
from sklearn.metrics.cluster import silhouette_score
sscore={}
for i in range(2, 15):
    km_cluster = KMeans(n_clusters=i, max_iter=10000, random_state=42)
    km_cluster.fit(feature_vect)
    cluster=km_cluster.labels_
    sscore[i]=silhouette_score(feature_vect,cluster)
print(max(sscore.values()))
print(sscore)
```



0.022930579508343384
{2: 0.018810112504652723, 3: 0.019295016551136652, 4: 0.007407383519793496, 5: 0.013553119482186281, 6: 0.020730163089823186, 7:
0.01665327437392266, 8: 0.022930579508343384, 9: 0.020115063873772668, 10: 0.011528649470397457, 11: 0.01793807566518074, 12: 0.02
176050300433471, 13: 0.019822669339817644, 14: 0.020420264082582632}

최적의 K=8

군집화 수행 및 결과

```
##### Cluster 0
Top features: ['royal', 'battle royal', 'battle', 'better', 'watched']
Top features value: [0.20626824040326244, 0.20626824040326244, 0.1725455333952561, 0.15496565884903477, 0.1359922241769557]
=====

##### Cluster 1
Top features: ['show', 'great', 'good', 'watched', 'acting']
Top features value: [0.20035576791218362, 0.12168183543531486, 0.11378968855407053, 0.11175518847892416, 0.0993466467032157]
=====

##### Cluster 2
Top features: ['episode', 'id', 'hype', 'season', 'someone']
Top features value: [0.36761913130574886, 0.14166360734341393, 0.1310173698965599, 0.11876397535137234, 0.11612212731899337]
=====

##### Cluster 3
Top features: ['old man', 'guy', 'bad', 'character', 'old']
Top features value: [0.0830082005407092, 0.08134908710818439, 0.07991264215190012, 0.07788081265346554, 0.07779330611946884]
=====

##### Cluster 4
Top features: ['end', 'episode', 'first', 'done', 'well']
Top features value: [0.23319590765204465, 0.19927305558251476, 0.13840135276257884, 0.12148541350196639, 0.11164147992319232]
=====

##### Cluster 5
Top features: ['life', 'style', 'play', 'acting', 'squid']
Top features value: [0.6102510598768176, 0.3735549246003652, 0.29379198443517274, 0.18947450229518026, 0.15005471873232723]
=====

##### Cluster 6
Top features: ['people', 'character', 'show', 'make', 'way']
Top features value: [0.2105428646369402, 0.10245174621606072, 0.10207013488988444, 0.0979623681317986, 0.0929350149819306]
=====

##### Cluster 7
Top features: ['last', 'last episode', 'episode', 'worth', 'three']
Top features value: [0.3186341979545478, 0.26718411233899875, 0.23045480758050285, 0.11687622700703572, 0.09962927810953376]
=====
```

```
cluster_centers = km_cluster.cluster_centers_
print('cluster_centers shape: ', cluster_centers.shape)
print(cluster_centers) #군집화에 대해서 어떤 것들이 가깝게 있는가(그 군집화를 대변함)
```

```
cluster_centers shape: (8, 208)
[[0.0334694  0.          0.09820616 ... 0.          0.          0.          ]
 [0.09934665 0.0945256  0.01846937 ... 0.          0.          0.          ]
 [0.          0.          0.          ... 0.          0.05694876 0.          ]
 ...
 [0.1894745  0.          0.          ... 0.          0.          0.          ]
 [0.02645724 0.0287576  0.          ... 0.01870972 0.02736264 0.02927581]
 [0.          0.02309751 0.          ... 0.          0.11687623 0.          ]]
```

군집별 top 5 핵심단어

- 값이 클 수록 해당 군집에 **중심**에 있는 단어
- 값이 클 수록 군집을 대표할 수 있는 단어

군집화 수행 및 결과



Cluster 0

- Battle royal 관련 단어들 묶임



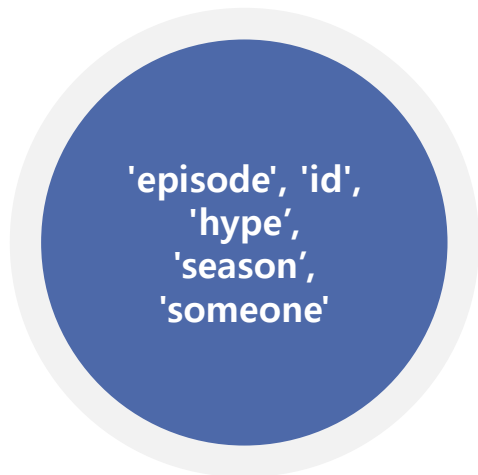
Cluster 1

- 쇼, 최고, 연기 라는 단어들이 묶여 있음
- 연기력이 훌륭했다고(great) 추측해볼 수 있음



Cluster 3

- old man 캐릭터와 관련 단어들이 묶임



Cluster 2

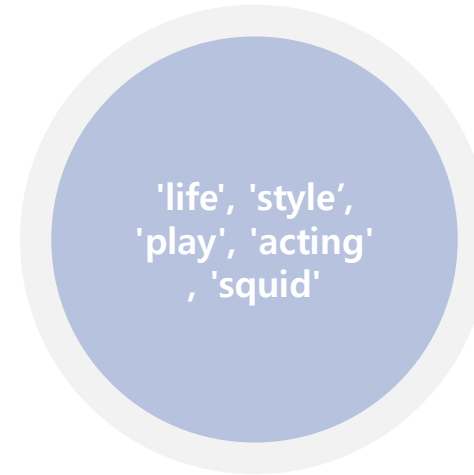
- 에피소드, id, 시즌 단어들이 묶여 있음

군집화 수행 및 결과



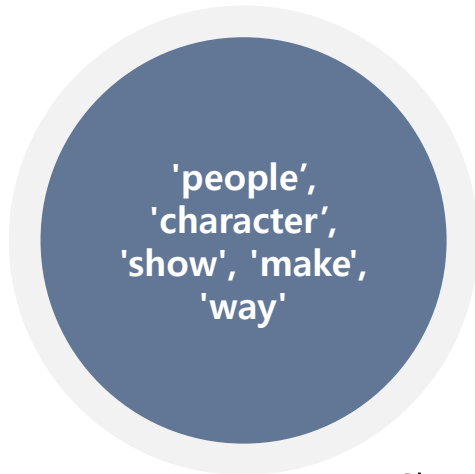
Cluster 4

- Episode와 순서에 관련한 단어 묶임



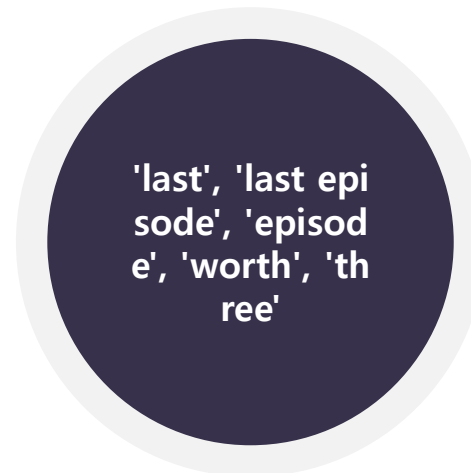
Cluster 5

- 삶, 연기, 오징어 게임 등의 단어가 묶임



Cluster 6

- 사람, 캐릭터, 쇼 등의 단어가 묶임



Cluster 7

- 마지막 에피소드에 관한 단어들이 묶임

LSTM 모델로 Text 생성 및 시나리오 방향성 제시

비주얼 프로그래밍

LOGO

07

07 LSTM 모델로 Text 생성 및 시나리오 방향성 제시

LOGO

```
sentences = []
for sentence in preprocessed_text:
    encoded = tokenizer0.texts_to_sequences([sentence])[0]

    for i in range(1, len(encoded)):
        sentences.append(encoded[:i+1])
```



30	66			
30	66	67		
30	66	67	14	
30	66	67	14	15

학습 데이터 구성

문장 리스트 구성

- 모델이 단어를 예측하기 위해 이전에 등장한 단어를 모두 활용하기 위함

07 LSTM 모델로 Text 생성 및 시나리오 방향성 제시

LOGO

학습 데이터 구성

```
max_len0 = max(len(i) for i in sentences)
sequences = pad_sequences(sentences, maxlen = max_len0, padding='pre')
```



Y

0	0	0	30	66
0	0	30	66	67
0	30	66	67	14
30	66	67	14	15

Padding 작업

- Sentence안의 문장의 길이가 모두 다르게 때문에 0으로 padding

07 LSTM 모델로 Text 생성 및 시나리오 방향성 제시

LOGO

Model: "sequential_1"

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, None, 10)	2200
lstm_1 (LSTM)	(None, 128)	71168
dense_1 (Dense)	(None, 220)	28380

=====
Total params: 101,748
Trainable params: 101,748
Non-trainable params: 0
=====

LSTM 모델 구성

- 128개의 유닛
- Lstm & Dense Layer

07 LSTM 모델로 Text 생성 및 시나리오 방향성 제시

LOGO

LSTM TEXT 생성

입력: 'Royal watched'

01

Royal watched the whole lot and its just plain silly the cast bring overacting to a new level all the characters are spectacularly annoying i wanted them all to die by episode



- Royal은 전체를 보았고 캐스트가 과잉 행동을 새로운 수준으로 가져 오는 것은 어리석은 일입니다. 모든 캐릭터는 엄청나게 짜증이 납니다. 에피소드별로 모두 죽기를 원했습니다.

입력: 'Show'

02

Show of the best shows ive watched everyone of the actors was so so good and the second like which it really slow i often have the feeling that many scenes are there just to make the show longer adding very little to the plot



- 내가 본 최고의 쇼의 쇼는 모든 배우들이 너무 좋았고 두 번째는 정말 느려서 쇼를 더 길게 만들기 위해 많은 장면이 거기에 있다는 느낌이 듭니다.

입력: 'Episode'

03

Episode 1 was amazing but things went downhill a little bit each episode episode 4 i was totally over it i did watch all 9 episodes because id already invested so much time into it



- 1화는 굉장했지만 각 에피소드 에피소드 4화는 완전히 끝났습니다. 이미 너무 많은 시간을 투자했기 때문에 9화를 모두 시청했습니다.

입력: 'Old man'

04

Old man the korean version of hunger game with some cheap metaphors to criticize capitalism democracy ethics etc terrible acting cliché dialogues and predictable scenes that drag for minutes and minutes



- 올드맨 한국판 헝거게임 자본주의 민주주의 윤리 등을 비판하는 싸구려 은유 등 진부한 연기의 진부한 대사 와 예측 가능한 장면들이 몇 분씩 질질 끌린다.

07 LSTM 모델로 Text 생성 및 시나리오 방향성 제시

LOGO

LSTM TEXT 생성

입력: 'First' 05 First started squid game i didnt expect much just something to pass time at work after the first two episodes i was hooked the show is thrilling and makes your heart beat increase i highly recommend watching if you are looking to have some entertainment for a couple hours hours



- 처음 시작한 오징어 게임 나는 첫 두 에피소드 후 직장에서 시간이 지나갈 것을 기대하지 않았습니다. 나는 쇼가 스릴 있고 심장 박동을 증가시킵니다. 몇 시간 동안 엔터테인먼트를 찾고 있다면 시청하는 것이 좋습니다.

입력: 'Life' 06 Life overrated it keeps the same drum about what you do in a game if it was your life versus another's life it's got a certain style but the acting was amateurish and melodramatic sorta like anime



- 과대평가된 인생은 당신의 삶과 다른 사람의 삶이라면 게임에서 하는 일에 대해 같은 복을 유지합니다. 특정 스타일이 있지만 연기는 애니메이션처럼 아마추어적이고 멜로드라마적이었습니다.

입력: 'Character' 07 Character mean am an ardent follower of the Korean movies even though most of the movies are laden with violence gore and bloodbath ... i risked my time on the rave reviews the show is getting and boy it has paid back i cannot recall any single show more edge of the seat than this yes there may be better shows in terms of intrinsic production values on the rave reviews the show is getting and boy it has paid back i cannot recall any single show more edge of the seat than this yes there may be better shows in terms of intrinsic production values



- 캐릭터 의미는 대부분의 영화가 폭력적이지만 나는 약간 불안한 한국 영화의 열렬한 추종자입니다. 한국 웹시리즈가 엄청나게 인기 있고 오징어 게임이라는 이름 넷플릭스의 최신 제품이 그다지 매력적이지 않은 것 같지만 쇼가 받고 있는 격찬에 내 시간을 걸었습니다. 그리고 보답을 받았습니다.

입력: 'Last Episode' 08 Last Episode very very very minor i really didn't like the last three episodes and the wrapup was dissatisfying gganbu was one of the strongest episodes in an emotional perspective



- 마지막화 아주아주아주사소함 마지막 3화가 정말 마음에 안들었고 마무리가 불만족스러웠음 깐부는 감성적으로 가장 강한 에피소드중 하나였다

LSTM TEXT 해석

01 입력: Royal watched

- 모든 캐릭터는 엄청나게 짜증이 납니다. 에피소드별로 모두 죽기를 원했습니다.



- 캐릭터들에 대한 부정적인 의견
- 너무 몰입을 해서 부정적인 것인지 알 수 없음

02 입력: Show

- 내가 본 최고의 쇼의 쇼는 모든 배우들이 너무 좋았고 두 번째는 정말 느려서 쇼를 더 길게 만들기 위해 많은 장면이 거기에 있다는 느낌이 듭니다.



- 배우들에 대한 긍정적인 의견
- 쇼를 의도적으로 길게 만든 것이 눈에 보였음

03 입력: Episode

- 1화는 굉장했지만 각 에피소드 에피소드 4화는 완전히 끝났습니다. 이미 너무 많은 시간을 투자했기 때문에 9화를 모두 시청했습니다.



- 에피소드 1화는 좋았음 그러나 중반부터 부정적
- 앞에 거를 다 봐서 그냥 끝까지 보았음

04 입력: Old man

- 올드맨 한국판 헝거게임 자본주의 민주주의 윤리 등을 비판하는 싸구려 은유 등 진부한 연기의 진부한 대사와 예측 가능한 장면들이 몇 분씩 질질 끌린다.



- 전체적으로 비판적인 의견
- 대사, 연기, 장면들이 예측이 가능하며 질질끄는 경향이 보임

07 LSTM 모델로 Text 생성 및 시나리오 방향성 제시

LOGO

LSTM TEXT 해석

05 입력: First

- 처음 시작한 오징어 게임 나는 첫 두 에피소드 후 직장에서 시간이 지나갈 것을 기대하지 않았습니다. 나는 쇼가 스릴 있고 심장 박동을 증가시킵니다. 몇 시간 동안 엔터테인먼트를 찾고 있다면 시청하는 것이 좋습니다.

- 긍정적인 의견
- 킬링타임으로 좋다는 것이 살짝 녹아 있음

06 입력: Life

- 과대평가된 인생은 당신의 삶과 다른 사람의 삶이라면 게임에서 하는 일에 대해 같은 복을 유지합니다. 특정 스타일이 있지만 연기는 애니메이션처럼 아마추어적이고 멜로드라마적이었습니다.

- 연기가 애니메이션처럼 아마추어적이고 멜로드라마
- 연기의 깊이가 깊지 않다는 의견으로 추측

07 입력: Character

- 캐릭터 의미는 대부분의 영화가 폭력적이지만 나는 약간 불안한 한국 영화의 열렬한 추종자입니다. 한국 웹시리즈가 엄청나게 인기 있고 오징어 게임이라는 이름 넷플릭스의 최신 제품이 그다지 매력적이지 않은 것 같지만 쇼가 받고 있는 격찬에 내 시간을 걸었습니다. 그리고 보답을 받았습니다.

- 제목이 매력적이지 않았음
- 그러나 재미있었음

08 입력: Last Episode

- 마지막화 아주아주아주사소함 마지막 3화가 정말 마음에 안들었고 마무리가 불만족스러웠음 깐부는 감성적으로 가장 강한 에피소드중 하나였다.

- 마지막 3화 그리고 결말에 부정적인 의견
- 깐부라는 에피소드는 임팩트가 있었음



시나리오
방향성

제시

- 1) 에피소드를 늘리고자 **억지요소의 스토리 라인** 구성X
- 2) **배우들이** 매우 중요->**단순한 연기지만 개성 있게** 할 수 있는 배우가 필요
- 3) **임팩트 있는 에피소드** 하나는 있어야 함 -> '**깐부**'
- 4) 에피소드 초반에 비해 **중반부터 결말까지의 부정적인 의견** 고려해야함

Thank you!