

Text Mining with Youtube

유튜브 크리에이터 비교 및 분석

2017052060 강현지
2017052588 백송이
2017053245 정은지

Contents

01 주제 소개 및 전처리

- 분석할 데이터 및 목표
- 전처리 과정 (Crawling, 데이터 통합, 토큰화/명사/빈도분석)

02 데이터 시각화

- 제목/댓글 분석 및 시각화
- 워드클라우드 등 시각화

03 Machine Learning

- 1) NMF(Non-negative Matrix Factorization)
 - 2) Logistic Regression, Linear SVC, Random Forest, KNN, Naïve Bayes, Perceptron, SGDClassifier, DecisionTree, SVM .
- 최종 Score 정렬 및 파일 생성

분석할 데이터 및 목표

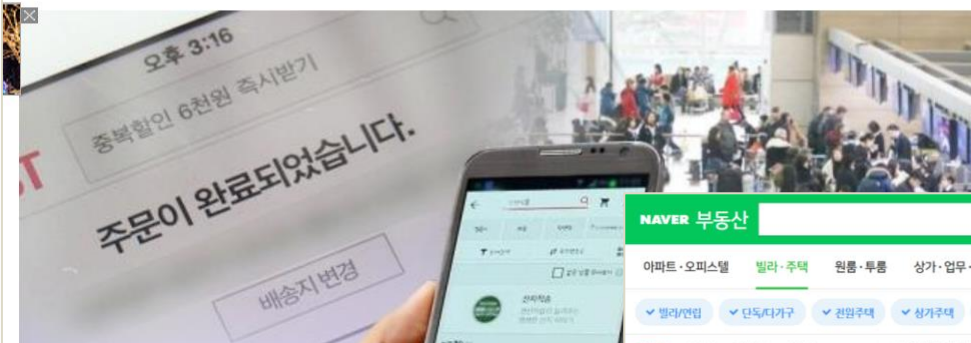
주제에 들어가기 앞서...

꾸준히 늘고있는 인가구를 위해 '네이버 부동산' 페이지를 텍스트 마이닝하려고 계획함
오피스텔 위주 주요 특징 특정 매물이 어느 지역에 많은지 등

나홀로족, 홈코노미 등 1인가구 늘며 소비트렌드도 변화

2019-11-21기사 편집 2019-11-21 17:42:20

대전일보 > 사회 > 종합



The screenshot displays the Naver Real Estate (NAVER 부동산) interface. The top navigation bar includes search and filter options. Below, there are tabs for different property types: 아파트·오피스텔, 빌라·주택, 원룸·투룸, 상가·업무·공장·토지, 분양, and MY관심. A list of properties is shown, including a 2-story villa (연립) and a 2-story apartment (빌라). Each listing includes details like price, area, and location. A map on the right shows the location of the selected property, with labels for nearby landmarks like the Samang Arts Center and Samang Sports Complex.

Property Type	Price	Area	Location
연립 (2층)	월세 7,000/100	빌라 · 120/92㎡, 2/3층	저렴한 리모델링 에어컨 방세게 넓은거실 ...
빌라 (전세 2억 4,000)	전세 2억 4,000	빌라 · 89/71㎡, 1/3층, 남향	상당대경문앞 깨끗해요

분석할 데이터 및 목표

그러나...

웹 페이지 크롤링 불가 크롬 브라우저 불가 태그를 크롤링 해오지 못함
결국 요즘 폭발적 인기인 크리에이터를 분석 주제로 선정

The screenshot shows the PyCharm IDE interface. The main editor window displays a Python script named `test.py` with the following code:

```
1 import requests
2 from bs4 import BeautifulSoup
3 import re
4
5 # mod1.py는 클래스에서 멤버 변수가 동작할 기능을 구현한 곳임. 클래스에서 멤버함수를 생성할 때 import하여 사용할 예정임
6
7 def get_name(pagenumber):
8     # 부동산 사이트에서 매물 이름을 크롤링하여 저장하는 함수임
9     url_start = 'https://new.land.naver.com/rooms?ms=37.5127742,127.0424075,10&a=APT:OPST:ABYG:GM:OR:VL:DDGG:JWJT:SGJT:W'
10    url = url_start + str(pagenumber)
11    result = requests.get(url)
12    soup = BeautifulSoup(result.text, "html.parser")
13    score_result = soup.find('div', class_='detail_contents_inner')
14    print(score_result)
15    # b = score_result.find('strong', class_='info_title_name').text
```

The Python Console at the bottom shows the output of the script:

```
print(score_result)
#b = score_result.find('strong', class_='info_title_name').text
#print(b)
return

get_name(1928874499)

None
```

The console output is highlighted with a red box. The status bar at the bottom indicates the file encoding is UTF-8, the line length is 1:1, and the Python version is 3.7 (finaltask).

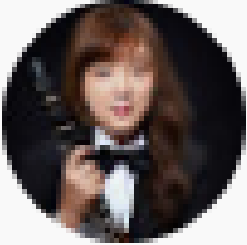
분석할 데이터 및 목표

유튜브의 인기채널인 '펭수'와 '워크맨'을 비교해봄으로써 최근의 트렌드 변화와 인기에 영향을 미치는 요인에 대해 분석

목표 어떤 영상이 가장 인기가 많은지 왜 인기가 많은지 게시글 및 댓글 분석을 통해 알아보기



페 하



2주 전

선 넘는 인간 : 장성규
선 넘는 펭귄 : 펭수

1.6천 답글



워크맨-Workman
구독자 353만명

홈 동영상 재생목록 커뮤니티 채널 정보

분석할 데이터 및 목표

1) DataSet: 펭수 및 워크맨 동영상 및 동영상 별 댓글

```
intro_paw.describe()
```

	Unnamed: 0	영상수
count	2.000000	2.000000
mean	1.500000	84.000000
std	0.707107	69.296465
min	1.000000	35.000000
25%	1.250000	59.500000
50%	1.500000	84.000000
75%	1.750000	108.500000
max	2.000000	133.000000

```
paw_reply.describe() # 통합한 동영상별 댓글 list DF 확인
```

	Unnamed: 0	Like	Title_number
count	3160.000000	3160.000000	3160.0
mean	9.500000	1430.986392	0.0
std	5.767194	2652.356452	0.0
min	0.000000	0.000000	0.0
25%	4.750000	193.000000	0.0
50%	9.500000	546.500000	0.0
75%	14.250000	1500.000000	0.0
max	19.000000	38000.000000	0.0

```
paw.describe()
```

	Unnamed: 0	싫어요
count	158.000000	158.000000
mean	51.253165	467.43038
std	36.693070	685.10947
min	0.000000	8.000000
25%	19.250000	70.000000
50%	43.500000	163.500000
75%	67.750000	404.750000
max	100.000000	1000.000000

개 비교 채널 수 펭수 워크맨
개 영상 수 펭수 워크맨
개 영상에서 추출한 댓글 수

실제 댓글은 영상 당 약 천개가 존재하나
부하로 인해 동영상당 개 한정 동영상 개 으로 수집함
코드상 숫자만 변경하면 전체 댓글 수집 가능함

데이터 전처리 과정 패키지

처리용

등

```
# 펍수 패키지 import
import datetime as dt
import pandas as pd
import requests
import time
import urllib.request #
import re
import konlpy

from bs4 import BeautifulSoup
from pandas import DataFrame
from selenium.webdriver import Chrome
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.keys import Keys
```

```
#워크맨 패키지 import
import datetime as dt
import pandas as pd
import requests
import time
import urllib.request #
import re
import konlpy

from bs4 import BeautifulSoup
from pandas import DataFrame
from selenium.webdriver import Chrome
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.keys import Keys
```

펍수와 워크맨 코드 개별 구현, 전처리 단계에서 병합 함.
(사용한 패키지는 두 파일 모두 동일)

-
- └ Konlpy : 한국어 분석을 위한 패키지
형태소 분석, 명사 분석 등 한국어에 특화된 분석 가능
 - └ selenium : 웹 브라우저를 제어하는 패키지
유튜브 화면 클릭, 스크롤 제어 등에 사용

시각화 및

추출한 데이터를 이용하여 시각화 및 기계 학습을 위한 패키지 시각화 및 사용

데이터 전처리 과정

데이터 입력

공통

```
#크롬 드라이버 연결
delay=0.1
browser = Chrome()
browser.implicitly_wait(delay)

start_url = 'https://www.youtube.com/channel/UCtckgmUcpzqGnzcs7xEqMzQ/videos'
browser.get(start_url) #browser로 위의 url 실행(펍수 videos directory)
browser.maximize_window()

body = browser.find_element_by_tag_name('body') #스크롤 위한 소스 추출
num_of_pagedowns = 20 # 페이지 down 수

#스크롤 다운
while num_of_pagedowns:
    body.send_keys(Keys.PAGE_DOWN)
    time.sleep(0.1)
    num_of_pagedowns -= 1
```

를 통해 별도의 창을 후 해당 창에서 를 진행함
스크롤 다운이 필요한 페이지이므로 스크롤 제어를 사용함

데이터 전처리 과정

데이터 입력

공통

```
# 페이지 소스 받아오기
html0 = browser.page_source
html = BeautifulSoup(html0, 'html.parser')
```

```
# 동영상 directory에 있는 각 영상들의 key값 생성을 위한 리스트 선언
# title(제목), href(링크), viewcount(조회수)
```

```
title_list = [] #제목 리스트 생성
href_list = [] #주소 리스트 생성
viewcountmake_list = [] #조회수 크롤링 위한 리스트 생성
viewcount_list = [] #조회수 리스트 생성
```

```
# 구독자 수 저장
subsc = html.find(id="subscriber-count").text
```

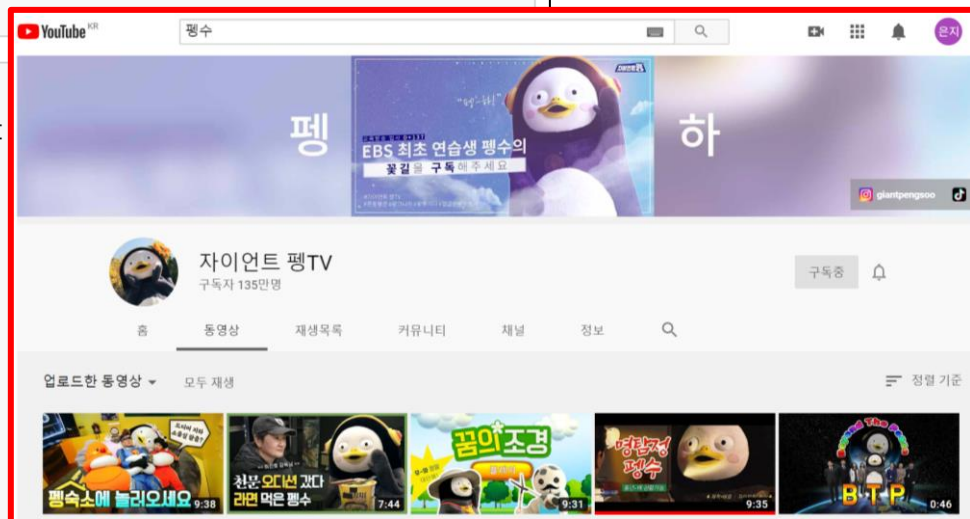
```
# title 저장
for tr in html.find_all(id="video-title"):
    title = tr.get('title')
    title_list.append(title)
```

```
#href 태그 내용 저장
for tr in html.find_all(id="video-title"):
    href = tr.get('href')
    href_list.append(href)
```

```
# 조회수 저장
for tr in html.find_all('span', class_="style-scope ytd-grid-video-renderer"):
    viewcount = tr.get_text('span')
    viewcountmake_list.append(viewcount)
```

```
#구독자수, 동영상 수
print(subsc, ", 영상 개수:", len(title_list))
```

구독자 133만명 , 영상 개수: 133



전체 화면에서 크롤링

데이터 전처리 과정

데이터 입력

공통

```
# 각 array별 개수 확인
print(len(title_list), len(href_list), len(viewcountmake_list))

#title, 주소, 좋아요수 데이터프레임 생성
peng_list = pd.DataFrame({'title':title_list, 'href':href_list, 'viewcount':viewcountmake_list})
```

구독자 133만명 , 영상 개수: 133

133 133 266

```
c:\users\백송0\appdata\local\programs\python\python36\lib\site-packages\pandas\core\internals
#construction.py in extract_index(data)
    315         lengths = list(set(raw_lengths))
    316         if len(lengths) > 1:
--> 317             raise ValueError('arrays must all be same length')
    318
    319         if have_dicts:
```

ValueError: arrays must all be same length

생성 시 조회수
 불일치
에러 발생 따라서 불필요한 데이터인 '등록일' 제거하는 전처리 진행

는 조회수 등록일이

로 포함되어

로 인해

데이터 전처리 과정

데이터 입력 및 처리 데이터 클리닝 작업 공통

```
# 데이터 클리닝 작업(조회수와 등록일에서 등록일 제거)
```

```
for tr in range(0, len(viewcountmake_list), 2):
```

```
    a = viewcountmake_list[tr]
```

```
    viewcount_list.append(a)
```

```
# 조회수 데이터 전처리
```

```
clean_viewcount = []
```

```
for i in viewcount_list:
```

```
    a = i[4:-2]
```

```
    amul = float(a)*10000
```

```
    clean_viewcount.append(int(amul))
```

```
print(clean_viewcount)
```

• 데이터 클리닝 작업

등록일 제거

기존 개월 전 삭제

조회수 데이터 전처리 작업

기존: '조회수 회' → '860000'

```
[860000, 1120000, 610000, 1480000, 660000, 1130000, 1350000, 890000, 1300000, 10
10000, 1370000, 1550000, 600000, 1940000, 640000, 1400000, 1390000, 880000, 7400
00, 1040000, 680000, 320000, 1430000, 580000, 900000, 800000, 710000, 410000, 22
30000, 1170000, 180000, 1530000, 200000, 510000, 320000, 800000, 530000, 810000,
430000, 1730000, 480000, 540000, 560000, 350000, 3150000, 300000, 230000, 66000
0, 340000, 480000, 300000, 540000, 1360000, 100000, 2100000, 1550000, 380000, 84
0000, 1230000, 770000, 630000, 600000, 360000, 280000, 890000, 500000, 430000, 3
20000, 630000, 1380000, 290000, 300000, 330000, 92000, 1070000, 360000, 440000,
220000, 1520000, 170000, 290000, 580000, 180000, 230000, 96000, 790000, 1500000,
380000, 830000, 720000, 660000, 99000, 500000, 820000, 360000, 230000, 440000, 4
70000, 510000, 150000, 130000, 210000, 650000, 210000, 550000, 190000, 100000, 8
20000, 910000, 1020000, 930000, 230000, 490000, 680000, 680000, 480000, 930000,
220000, 390000, 330000, 470000, 550000, 430000, 280000, 300000, 290000, 460000,
480000, 590000, 960000, 380000, 460000, 610000]
```

데이터 전처리 과정

데이터 입력 및 처리

생성 및 확인

공통

#title, 주소, 좋아요수 데이터프레임 생성

```
peng_list = pd.DataFrame({'title':title_list, 'href':href_list, 'viewcount':clean_viewcount})  
peng_list.sort_values(by=['viewcount'], axis=0, ascending=False)
```

	title	href	viewcount
44	EBS 최초 연습생 펑수의 오디션 합격 TIP *최초공개*	/watch?v=K_5lal40ICk	3150000
28	[단독] 펑권 의혹 전격 해부! 독점 인터뷰: 김민교, 양치승 [Ep.57]	/watch?v=UD-WQvgjsng	2230000
54	[Ep.44] 펑수, 드디어 크로스를 만났다	/watch?v=TckHOovKE2I	2100000
13	[Ep.65] 세상에 나쁜 펑권은 없다.	/watch?v=wedLGh2jxkQ	1940000
39	EBS 옥상에서 똑딱이 선배님을 만났다 (feat. 역대급 깜짝손님)	/watch?v=yN7wrzXtlWM	1730000
11	[Ep.66] 화보 모델 펑수	/watch?v=XUM3sH1kBtw	1560000
55	[Ep.43]EBS 복지 클래스 전격 공개! [힐링 선물 3종세트]	/watch?v=80cyxKBcYXA	1550000
31	[Ep.56] '펑TV 야유회'라고 쓰고 '지옥'이라 읽는다	/watch?v=pv84Qcu9DOQ	1530000
78	'자꾸 교육방송 선 넘는' 이말년(침착맨)과 펑수(EBS 연습생)의 짤방 폭격! [...	/watch?v=S9q4BorGgOI	1520000
86	예술천재 펑수, 고양예고 테스트 도전! 음악과 미술과 뿌셔! [EP.27]	/watch?v=NjS52IUjHdu	1500000
3	[Ep.71] 내 헤드셋 누가 가져갔어?	/watch?v=rRZtVrh6kgI	1480000
22	[Ep.60] 펑수, EBS 퇴사??? SBS 정복기(2)	/watch?v=ze1wbV20qAU	1430000
15	1회 남극 유치원 동창회 시작합니다(Feat. 돌리 선배님 등장)	/watch?v=E5f4pRlycsU	1400000
16			
69	(납량특집) EBS 펑권		
10			

- 가장 인기있는 동영상 찾기 위해
동영상별 를 생성 후 조회수

기준 정렬

데이터 전처리 과정

데이터 입력 및 처리

동영상 별 속성값

추출

공통

```
# 영상 본문 key(제목, 조회수, 좋아요, 싫어요, 본문내용, 전체 댓글 수) 추출 - 1
```

```
title_num = 0;
n_title_list = [] # 영상별 제목 리스트 생성
story_sub_list = [] # 영상 본문내용 리스트
n_viewcount_list = [] # 영상별 조회수 리스트 생성
n_like_list = [] # 영상별 좋아요 리스트 생성
n_unlike_list = [] # 영상별 싫어요 리스트 생성
n_href_list = [] # 영상별 주소 리스트 생성
str_n_comments = [] # 영상별 댓글 총 수 리스트 생성
start_url = 'https://www.youtube.com/'
```

```
# 데이터
```

```
for i in href_list[:2]: # :2 빼면 전체 데이터, 시간 단축을 위해 2개 샘플만 시행
```

```
url = start_url + i
```

```
browser.get(url)
```

```
print(i) # 추출 주소값
```

```
n_href_list.append(i)
```

```
source = browser.page_source
```

```
bs = BeautifulSoup(source, 'html.parser')
```

```
bsstory = BeautifulSoup(source, "html.parser")
```

상세 화면에서 크롤링



영상별 '제목 본문 조회수 좋아요 싫어요 댓글수' 가 포함된
펭수 동영상 디렉토리에 있는
모든 동영상을 조회하여 정보 추출
생성 예정

데이터 전처리 과정

데이터 입력 및 처리

동영상 별 속성값

추출

공통

```
# 영상에 대한 기본 정보 수집
info1 = bs.find('div',{'id':'info-contents'})

#n_title = "" #초기화 계속
n_title = info1.find('h1',{'class':'title style-scope ytd-video-primary-info-renderer'}).text
n_title_list.append(n_title)
n_viewcount = info1.find('yt-view-count-renderer',{'class':'style-scope ytd-video-primary-info-renderer'}).find_all('span')[0].text
n_viewcount_list.append(n_viewcount)
n_like = info1.find('div',{'id':'top-level-buttons'}).find_all('yt-formatted-string')[0].text #좋아요수
n_like_list.append(n_like)
n_unlike = info1.find('div',{'id':'top-level-buttons'}).find_all('yt-formatted-string')[1].text
n_unlike_list.append(n_unlike)

# 영상 본문 내용 추출
story_sub = bsstory.find('yt-formatted-string', class_='content style-scope ytd-video-secondary-info-renderer').text
story_sub_list.append(story_sub)
title_num = title_num + 1; # 동영상 번호

# 영상별 댓글수 추출

for i in range(len(n_comments)):
    str_tmp = str(n_comments[i].text)
    str_tmp = str_tmp.replace('댓글 ', '')
    str_tmp = str_tmp.replace('개', '')
    str_n_comments.append(str_tmp)
```

각 영상마다 ‘제목 본문 조회수 좋아요 싫어요 댓글수’를 추출하여
각 항목

의 에 추가

데이터 전처리 과정

데이터 입력 및 처리

동영상 별 속성값

추출

공통

조회수 데이터 전처리

```
clean_n_viewcount = []
for i in n_viewcount_list:
    a = i[4:-1]
    clean_n_viewcount.append(a)
print(clean_n_viewcount)
```

좋아요 데이터 전처리

```
clean_n_like_list = []
for i in n_like_list:
    if i[-1] == '천' or '만':
        a = i[:-1]
        if i[-1] == '천': amul = float(a)*1000
        if i[-1] == '만': amul = float(a)*10000
        clean_n_like_list.append(int(amul))
    else:
        clean_n_like_list.append(i)
print(clean_n_like_list)
```

싫어요 데이터 전처리

```
clean_n_unlike_list = []
for i in n_unlike_list:
    clean_n_unlike_list.append(i)
print(clean_n_unlike_list)
```

```
['1,374,702', '1,562,005', '600,930', '1,944,586', '648,478', '1,409,749',
'1,399,059', '887,330', '745,696', '1,048,624', '685,071', '329,770', '1,440,
539', '585,841', '906,534', '805,053', '719,986', '416,550', '2,239,768', '1,
180,868', '184,352', '1,541,343', '204,786', '517,546', '325,203', '801,839',
'540,834', '816,829', '430,905', '1,732,931', '485,258', '549,376', '568,06
4', '354,701', '3,159,281', '310,126', '232,295', '668,482', '347,716', '482,
736', '305,721', '541,394', '1,366,438', '101,847', '2,109,712', '1,556,128',
'384,418', '845,470', '1,236,459', '780,503', '633,499', '605,853', '364,52
9', '281,998', '900,953', '506,483', '433,951', '325,786', '631,440', '1,392,
969', '300,465', '305,365', '332,093', '92,308', '1,078,602', '366,121', '44
4,650', '229,006', '1,530,687', '171,710', '295,701', '589,027', '188,131',
'230,547', '96,789', '799,954', '1,506,724', '388,573', '832,745', '727,670',
'671,355', '99,810', '510,981', '830,523', '361,678', '230,559', '449,038',
'472,802', '514,906', '151,038', '130,488', '211,608', '655,297', '211,119',
'552,914', '196,212', '107,366', '830,121', '919,272', '1,021,378', '938,69
0', '233,913', '499,892', '681,557', '689,878', '487,521', '934,838', '226,65
1', '393,587', '334,374', '471,266', '555,697', '438,033', '286,852', '304,25
2', '300,175', '461,337', '481,318', '593,577', '964,559', '386,790', '463,77
4', '611,418']
[41000, 45000, 28000, 50000, 20000, 37000, 38000, 24000, 27000, 35000, 21000,
13000, 26000, 21000, 21000, 25000, 17000, 14000, 44000, 30000, 6500, 28000, 6
200, 13000, 12000, 25000, 13000, 19000, 14000, 36000, 12000, 14000, 13000, 90
00, 49000, 7800, 6200, 18000, 10000, 14000, 9600, 12000, 23000, 3300, 35000,
28000, 17000, 14000, 18000, 21000, 18000, 17000, 8300, 8200, 18000, 13000, 94
00, 9600, 12000, 23000, 12000, 24000, 7900, 3300, 15000, 7500, 23000, 5500, 2
2000, 9700, 9100, 11000, 3300, 4400, 2800, 15000, 24000, 10000, 18000, 23000,
15000, 2300, 8500, 13000, 6500, 6700, 12000, 11000, 21000, 3400, 4400, 6700,
21000, 5700, 13000, 5800, 4000, 12000, 18000, 19000, 18000, 7600, 12000, 1600
0, 12000, 7500, 14000, 5400, 11000, 8700, 11000, 12000, 9300, 4900, 6900, 700
0, 10000, 18000, 14000, 22000, 8100, 8700, 12000]
['748', '453', '133', '539', '135', '421', '373', '193', '130', '275', '126',
'57', '347', '125', '242', '157', '159', '84', '779', '261', '23', '387', '2
5', '128', '47', '136', '93', '178', '68', '360', '83', '102', '137', '39',
'808', '45', '23', '94', '62', '119', '55', '227', '294', '27', '442', '374',
'40', '131', '462', '132', '165', '148', '67', '51', '120', '112', '102', '14
```

조회수 좋아요 싫어요의 경우 '조회수 회', 'X.X만 만)' 으로 추출되어
문자 삭제('조회수') 및 정수형 으로 전처리 진행

데이터 전처리 과정

데이터 입력 및 처리

동영상 별 속성값

추출

공통

```
n_peng_list = pd.DataFrame({'제목':n_title_list, '주소':n_href_list, '조회수':clean_n_viewcount,
                             '좋아요':clean_n_like_list, '싫어요':clean_n_unlike_list, '댓글수':str_n_comments,
                             '본문내용':story_sub_list})
```

```
n_peng_list.head()
```

	제목	주소	조회수	좋아 요	싫어 요	댓글 수	본문내용
0	[Ep.73] 10살 펭귄 벌써 집 장만	/watch?v=0jQ6W6BPUFc	1,102,406	37000	403	5,730	드디어 소품실을 탈출한 펭수, 새로운 펭숙소에서 집들이를 하는데... 과연 어떤 모...
1	영화 '천문: 하늘에 묻는다' 오디션 보러 간 펭수(* 쿠기영상 있음!)	/watch?v=8qYmLVOIYXI	1,175,032	38000	437	5,166	첫 영화 오디션을 본 펭수\n라면 먹고 돌아오다?\n\n[MUSIC INFO]...

각 영상에 대한 속성값의

생성 후

를 통해 생성 값 확인

데이터 전처리 과정

데이터 입력 및 처리

각 영상 별 댓글

추출

공통

```
# 영상별 댓글 추출
title_num = 0;
youtube_pd = pd.DataFrame() # 영상 당 댓글 df 생성용
total_youtube_pd = pd.DataFrame() # 최종 df 생성용
str_youtube_comments_len = [] # 영상별 댓글수 리스트 생성

start_url = 'https://www.youtube.com/'
for i in href_list[:2]: # :2 빼면 전체 데이터, 시간 단축을 위해 2개 샘플만 시행
    url = start_url + i
    browser.get(url)
    print(i) # 추출 주소값
    source = browser.page_source
    bs = BeautifulSoup(source, "html.parser")
```

동영상 별 상위 댓글 개씩을 추출하여 생성 예정
댓글 개 개 동영상 개의 행

데이터 전처리 과정

데이터 입력 및 처리

각 영상 별 댓글

추출

공통

```
# 댓글 추출 코드
reply_list = [] #댓글 리스트 생성
body = browser.find_element_by_tag_name('body') #스크롤하기 위해 소스 추출
num_of_pagedowns = 20

#스크롤 다운
while num_of_pagedowns:
    body.send_keys(Keys.PAGE_DOWN)
    num_of_pagedowns -= 1

    html0 = browser.page_source
    html = BeautifulSoup(html0, 'html.parser')

    youtube_userIDs = html.select('div#header-author > a > span')
    youtube_comments = html.select('yt-formatted-string#content-text')
    youtube_likes = html.select('div#toolbar > span')

    str_youtube_userIDs = []
    str_youtube_comments = []
    str_youtube_likes = []
    str_youtube_title_number = [] #영상 타이틀 넘버
```

각 영상의 댓글 추출을 위한
댓글의 속성값

셋팅 및
생성

데이터 전처리 과정

데이터 입력 및 처리

각 영상 별 댓글

추출

공통

```
for i in range(len(youtube_user_IDs)):
    str_tmp = str(youtube_user_IDs[i].text)
    str_tmp = str_tmp.replace('\\\\n', '')
    str_tmp = str_tmp.replace('\\\\t', '')
    str_tmp = str_tmp.replace(' ', ',')
    str_youtube_userIDs.append(str_tmp)

    str_tmp = str(youtube_comments[i].text)
    str_tmp = str_tmp.replace('\\\\n', '')
    str_tmp = str_tmp.replace('\\\\t', '')
    str_tmp = str_tmp.replace(' ', ',')
    str_youtube_comments.append(str_tmp)

    str_tmp = str(youtube_likes[i].text)
    str_tmp = str_tmp.replace('\\\\n', '')
    str_tmp = str_tmp.replace('\\\\t', '')
    str_tmp = str_tmp.replace(' ', ',')
    str_youtube_likes.append(str_tmp)

    # 어떤 영상인지 넘버링으로 확인
    str_youtube_title_number.append(title_num)
```

에 포함된 공백 탭 과도한 띄어쓰기 제거

데이터 전처리 과정

데이터 입력 및 처리

각 영상 별 댓글

추출

공통

```
# 좋아요 수 전처리, 좋아요수가 str+공백으로 들어가 있어 공백 제거 후 계산
clean_str_youtube_likes = []
for i in str_youtube_likes:
    i = i.replace(" ", '')
    if i.find('만') != -1:
        a = i.replace('만', '')
        amul = float(a)*10000
        clean_str_youtube_likes.append(int(amul))
    elif i.find('천') != -1:
        a = i.replace('천', '')
        amul = float(a)*1000
        clean_str_youtube_likes.append(int(amul))
    else:
        clean_str_youtube_likes.append(int(i))

# 각 영상별 댓글 데이터 프레임 생성
youtube_pd = pd.DataFrame({"ID":str_youtube_userIDs, "Comment":str_youtube_comments,
                           "Like":clean_str_youtube_likes, "Title_number":str_youtube_title_number})
total_youtube_pd = pd.concat([total_youtube_pd, youtube_pd])
print(len(youtube_pd))
```

좋아요 수 전처리

좋아요 수에

공백이 포함되어 공백 제거 후 계산

만 →

댓글 데이터 프레임 생성

유저 댓글 본문 좋아요 수 영상 번호 로 구성

데이터 전처리 과정

데이터 입력 및 처리

각 영상 별 댓글

추출

공통

```
# 게시물당 댓글 추출개수를 구하고, 이를 모두 합해 전체 동영상의 댓글 전체 개수를 구함.  
print("* 본문 추출 개수: ", len(story_sub_list)) #추출한 본문 개수 확인  
print("* 댓글 추출 개수: ", len(total_youtube_pd)) #추출한 댓글 개수 확인  
/watch?v=U34Z1Ky11CU  
20  
  
/watch?v=2gg1QKk7fjw  
20  
  
/watch?v=M-52pdhp-ao  
20  
  
/watch?v=Er016Y9sE8Q  
20  
  
/watch?v=4vkzT47_Vdg  
20  
  
/watch?v=XeH_8w4AWS4  
20  
  
* 본문 추출 개수: 123  
* 댓글 추출 개수: 2460
```

본문 및 댓글 추출 개수 확인

개의 동영상을 확인하였으며 각 영상 별 상위 개의 댓글을 추출함

개 개 댓글을 추출하였음

이는 상단의 값을 조절하여 영상 별 댓글 추출 수를 늘릴 수 있으나
컴퓨터 부하 및 엄청난 시간 소요로 인해 금번 과제에서는 으로 설정

데이터 전처리 과정

데이터 입력 및 처리

각 영상 별 댓글

추출

공통

```
print(str_youtube_comments_len)
```

[illegible]

#추출한 본문 예시 1개 확인

```
print("예시 1개", print(story_sub_list[1]))
```

평수가 화보 모델로서 촬영을 하게 되었다!
 화보 촬영을 위한 평수의 눈물(?) 나는 노력!
 그리고 현직 모델에게 직접 받는 모델 포즈까지!
 예시 1개 None

실제 추출된 값 확인

각 본문 당 개의 댓글을 추출하였으며 영상 본문 또한 정상 추출되었음

데이터 전처리 과정


데이터 입력 및 처리 본문내용 데이터 전처리 공통

```
# 본문내용 데이터 전처리
clean_story_sub_list = []
for i in story_sub_list:
    a = i.replace('\n', '')
    # 첨부 음악 리스트가 본문내용에 포함된 경우 찾아서 삭제
    delresult = a.find("[MUSIC INFO]")
    delresult1 = a.find("음원 정보")
    delresult2 = a.find("1.")
    if delresult != -1:
        a = a[:delresult]
        clean_story_sub_list.append(a)
    elif delresult1 != -1:
        a = a[:delresult1]
        clean_story_sub_list.append(a)
    elif delresult2 != -1:
        a = a[:delresult2]
        clean_story_sub_list.append(a)
    else:
        clean_story_sub_list.append(a)
print(clean_story_sub_list)
```

['K-펑크 한다 해외진출', '펑수가 화보 모델로서 촬영을 하게 직접 받는 모델 포즈까지!', '힙합펑수의 커버영상이 드. 디. 이번 신상뮤비 공개챌린지에 참여해준 모든 분들 펑러뷰~♡잇기 두', "어느 날 갑자기 펑수에게 이상한 행동들이 보인다!? 걱정 ===== 칠판연 '꿈을 이루기 위한 7가지 충'을 앞둔여러분들을 위해 이 영상을 바칩니다 펑펑', '펑수가 워창회에서 떠나는 즐거운 시간들 시작합니다', 'SBS에 내레이션 매니저스펑수 사관학교를 통해 진정한 매니저로 거듭나보자펑!' 이 근질근질했는데 이제 대놓고 자랑해도 돼?', '고3들이 기다려라! 펑수가 간다!', '이거 보고 내일 11화 보면 너 꿀잤', '펑수를 5 분위로 초대할 셀럽의 정체가 드러났다..!S 본부의 특급 셀럽 배성재가 직접 펑수의 자질을 확인하고 모종의 거래까지..?', '우리 친구들을 위해 해일요일 밤에 올림니당 펑펑', 'SBS의 셀럽을 자초한 누군가가 초대장을 보내왔다! 펑수의 SBS 정복기 1편 (feat. 재재)', '펑수는 구독자 여러분들이 있어서하나도 외롭지 않습니다!', '펑수가 드디어 해외진출을!???펑수 친구 엘로디의 프랑스 진출 꿀팁 대방출!!펑수는 프랑스 진출에 성공할 수 있을까??', '이 영광을구독자 여러분들께 돌립니다!', '펑수가 펑권이 아니다!?!? 펑수를 둘러싼 정체에 대한 의혹 날날이 파헤쳐 보겠습니다 펑펑!!', '펑수 당근 10살 맞거든용?(당근을 흔든다)', '목요일 20시30분 전주재방금요일 20시 30분 본방 연속 2편일요일 12시30분 전주재방, 본방재방 총 4편"생방송 톡!톡! 보니하니" 인서트 18시 10분경 전주재방 2편자주 만나도 자주 사랑해주세요

영화 '천문: 하늘에 묻는다' 오디션 보러 간 펑수(*쿠키영상 있음!)

조회수 1,136,170회 · 2019. 12. 10. 👍 3.8만 💬 416 🔗 공유 📌 저장 ...

 자이언트 펑TV
구독자 133만명 구독

첫 영화 오디션을 본 펑수!
라면 먹고 돌아오다?

불필요한 음원 정보 삭제

[MUSIC INFO]

1. What's Your Name (If You Want The Part, Earn It)_J.K. Simmons_Whiplash OST
2. The Avengers (From _Avengers Assemble_)_London Music Works
3. ㅋ_장기하와 얼굴들
4. All You Need Is Love (With Orchestra)_The Beatles Revival Band
5. 로망_장미여관_미생 OST
6. under pressure / Rhythm Nation (Feat. Happy Feet Two Chorus) _Pink
7. 명랑_김태성_명랑 OST
8. 광해 왕이 된 남자 Opening_모그
9. 급할수록 서둘러라_Unknown_미생 OST
10. Slow Slow Quick Quick_김종천_경숙이, 경숙아버지 OST

그 외 음원, 모두كم 제공

카테고리 [영화/애니메이션](#)

데이터 전처리 과정

데이터 입력 및 처리 최종

공통

```
# title, 주소, 조회수, 본문내용 추가한 데이터 프레임
peng_list_update = pd.DataFrame({'제목':n_title_list, '주소':n_href_list, '조회수':clean_n_viewcount,
                                '좋아요':clean_n_like_list, '싫어요':clean_n_unlike_list, '댓글수':str_n_comments, '본문내용':clean_s
```

```
peng_list_update.head()
```

	제목	주소	조회수	좋아 요	싫어 요	댓글 수	본문내용
0	[Ep.67] 전 세계 게 샀거라 (feat. 외교부)	/watch?v=yUJAvW2Rykc	1,380,758	42000	749	6,838	K-펑권 한다 해외진출
1	[Ep.66] 화보 모델 펑수	/watch?v=XUM3sH1kBtw	1,580,523	46000	455	7,678	펑수가 화보 모델로서 촬영을 하게 되었다!화보 촬영을 위한 펑수의 눈물(?) 나는 ...
2	펑수와 팬들의最強 콜라베이션 신상유비 (feat. 찰린지♡)	/watch?v=LPmyxMH96S8	603,876	29000	135	3,921	힙합펑수의 커버영상이 드. 디. 어. 찾아왔다!월미도에서 촬영했던 미공개 영상과 여...
3	[Ep.65] 세상에 나쁜 펑권은 없다.	/watch?v=wedLGh2jxkQ	1,971,318	50000	548	6,974	어느 날 갑자기 펑수에게 이상한 행동들이 보인다? 걱정된 제작진들이 긴급 솔루션 ...
4	수험생은 지금 당장 이 영상을 봅니다 (feat. 정승제쌤)	/watch?v=25RhZK3HuYM	650,703	20000	135	2,760	내일 모레 엄청 큰 시험을 앞둔여러분들을 위해 이 영상을 바칩니다!당 펑펑

댓글 수까지 최종 업데이트된 펑수 영상의

데이터 전처리 과정

데이터 입력 및 처리 예시 확인 공통

0번째 영상에서 추출한 댓글 DataFrame
youtube_pd.head()

번째 영상에서 추출한 댓글

	ID	Comment	Like	Title_number
0	Sol Sol	평수전용 빨대 제작해서 물 좀 주세요염 !!!..	1100	0
1	윤하슬	방송국놈들 평수 인기 많다고 스케줄 많이 잡지마 ;	1100	0
2	문성수	각 방송사 아나운서 조차 이름 잘 모르는 지금 이시대....EBS 김명중이 누군지 ...	3000	0
3	내삶 유포리아	제작진이 평수를 너무좋아하는거같음ㅋㅋㅋㅋㅋㅋ	3000	0
4	Music of The Night Korea	평수야 안녕! 뮤지컬 <오페라의 유령> 월드투어 팀인데 우리 공연 보러 오지 않을래...	1200	0

전체 영상에서 추출한 댓글 DataFrame
total_youtube_pd

전체 영상에서 추출한 댓글

	ID	Comment	Like	Title_number
0	대한민국외교부	평수 와줘서 고마워요! 해외진출의 꿈을 응원할게요~ 2019 한·아세안 특별정상회의...	3600	0
1	평랑단1호	악플러들 고소하고 EBS 빛 갓자	3600	0
2	Isabel la	아니 무슨 교육방송이 ㅋㅋㅋ 웬만한 예능프로그램보다 더 잘만들어 자막이고 편집이고 ...	1300	0
3	월드곰탕이	조반 영상부터 다 봐온 사람으로서열심히 한 죄밖에 없는데너무 힘들게 하는듯나쁜 댓글...	1300	0
4	gravity	위험한 물품 가지고 있어여? "제 자신" ㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋ	1000	0
5	Henry Kang	평수 마트료시가 굿즈 나오면 구매하실 분👉	1000	0
6	Zaya Kim	평수야 나는 덴마크 사는 한국사람이야 그런데 여기 친구들한테 너의 동영상을 전도하고...	908	0
7	Bak Shin	외교부편이 진짜 레전드네요. 어른이들과의 합도 좋고. 꿀벌평수, 판다평수, 취권평수...	908	0
8	열공열공	외교부 10살한테 다산의 상징이라뇨 ㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋ	635	0

데이터 전처리 과정

데이터 입력 및 처리

파일 저장

공통

```
# 펭수 intro csv 저장
intro_df.to_csv("peng_intro.csv", mode='w', encoding="utf-8-sig")

# 펭수 동영상 csv 저장
peng_list_update.to_csv("peng.csv", mode='w', encoding="utf-8-sig")

# 펭수 댓글 csv 저장
total_youtube_pd.to_csv("peng_reply.csv", mode='w', encoding="utf-8-sig")
```

최종 추출된 개의

을 워크맨 자료와 비교하기 위해

파일로 변환

이후 워크맨에서도 동일한 작업 진행하여
데이터 시각화 및 머신러닝 진행

파일 변환 후

데이터 전처리 과정

두 데이터 통합 새 파일 생성하여 통합 – 패키지

#펍수/워크맨 csv 불러와서 시각화 및 머신러닝

패키지 import

`import datetime as dt`

`import pandas as pd` *#df 분석용*

`import requests`

`import time`

`import urllib.request #`

`import re`

`import konlpy`

#import konlpy.tag import Okt

`from bs4 import BeautifulSoup`

`from pandas import DataFrame`

`from selenium.webdriver import Chrome`

`from selenium.webdriver.chrome.options import Options`

`from selenium.webdriver.common.keys import Keys`

`from nltk import FreqDist`

새로운 파일 생성하여
펍수의 워크맨의
시각화 및 머신러닝 진행

를 통합 후

시각화

`import matplotlib`

`import matplotlib.pyplot as plt`

`import numpy as np`

`import seaborn as sns`

`from PIL import Image`

#워드클라우드용

`from wordcloud import WordCloud`

#워드클라우드용

`fpath = "NotoSansCJKkr-Bold.otf"`

#워드클라우드 국문지원을 위한 별도 폰트 path 설정

`%matplotlib inline`

기계 학습

`from sklearn.linear_model import LogisticRegression`

`from sklearn.svm import SVC, LinearSVC`

`from sklearn.ensemble import RandomForestClassifier`

`from sklearn.neighbors import KNeighborsClassifier`

`from sklearn.naive_bayes import GaussianNB`

`from sklearn.linear_model import Perceptron`

`from sklearn.linear_model import SGDClassifier`

`from sklearn.tree import DecisionTreeClassifier`

`from sklearn.neural_network import MLPClassifier`

데이터 전처리 과정

두 데이터 통합 새 파일 생성하여 통합 – 파일 및 통합

```
##### Read #####
peng_intro = pd.read_csv('peng_intro.csv') # 펭수의 intro DF
work_intro = pd.read_csv('work_intro.csv') # 워크맨의 intro DF

peng = pd.read_csv('peng.csv')           #펭수의 동영상list DF
peng_reply = pd.read_csv('peng_reply.csv') #펭수의 동영상별 댓글list DF

work = pd.read_csv('work.csv')           #워크맨의 동영상list DF
work_reply = pd.read_csv('work_reply.csv') #워크맨의 동영상별 댓글list DF
```

```
##### DF 통합 #####
# 1) 펭수 + 워크맨 intro DF 통합
intro_paw = pd.DataFrame(peng_intro)
intro_paw = intro_paw.append(work_intro)

# 2) 펭수 + 워크맨 동영상list DF 통합
paw = pd.DataFrame(peng)
paw = paw.append(work)

# 3) 펭수 + 워크맨 동영상별 댓글list DF 통합
paw_reply = pd.DataFrame(peng_reply)
paw_reply = paw_reply.append(work_reply)
```

intro_paw.head()

	Unnamed: 0	구분	구독자	영상수
0	1	펭수	135만명	133
0	2	워크맨	353만명	35

각 에서 추출한 를
하여 새 파일로 통합

데이터 전처리 과정

두 데이터 통합 새 파일 생성하여 통합 - 새 속성 추가

```
##### 구분자 추가 #####
```

```
for i in range(len(peng)): peng['구분'] = 1 # 펭수 1
for i in range(len(work)): work['구분'] = 0 # 워크맨 0
for i in range(len(peng_reply)): peng_reply['구분'] = 1 # 펭수 1
for i in range(len(work_reply)): work_reply['구분'] = 0 # 워크맨 0
```

통합 후 항목 구분을 위해
'구분' 열 추가

```
paw.head()
```

Unnamed: 0		제목	주소	조회수	좋아요	싫어요	댓글수	본문내용	구분
0	0	[Ep.67] 전 세계 게 샀거라 (feat. 외교부)	/watch?v=yUJAvW2Rykc	1,381,698	42000	749	6,838	K-펭귄 한다 해외진출	1
1	1	[Ep.66] 화보 모델 펭수	/watch?v=XUM3sH1kBtw	1,583,971	46000	455	7,678	펭수가 화보 모델로서 촬영을 하게 되었다! 화보 촬영을 위한 펭수의 눈물(?) 나는 ...	1
2	2	펭수와 팬들의 최강 컬래버레이션 신상뮤비 (feat. 챌린지♡)	/watch?v=LPmyxMH96S8	604,376	29000	135	3,921	힙합펭수의 커버영상이 드. 디. 어. 찾아왔다! 월미도에서 촬영했던 미공개 영상과 여...	1
3	3	[Ep.65] 세상에 나쁜 펭귄은 없다.	/watch?v=wedLGh2jxkQ	1,975,593	50000	548	6,974	어느 날 갑자기 펭수에게 이상한 행동들이 보인다? 걱정된 제작진들이 긴급 솔루션 ...	1
4	4	수험생은 지금 당장 이 영상을 봅니다 (feat. 정승제쌤)	/watch?v=25RhZK3HuYM	651,078	20000	135	2,760	내일 모레 엄청 큰 시험을 앞둔 여러분들을 위해 이 영상을 바칩니다! 펭펭	1

데이터 전처리 과정

두 데이터 통합 새 파일 생성하여 통합 - 별

```
intro_paw.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
Int64Index: 2 entries, 0 to 0  
Data columns (total 4 columns):  
Unnamed: 0      2 non-null int64  
구분            2 non-null object  
구독자          2 non-null object  
영상수          2 non-null int64  
dtypes: int64(2), object(2)  
memory usage: 80.0+ bytes
```

```
paw.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
Int64Index: 158 entries, 0 to 34  
Data columns (total 9 columns):  
Unnamed: 0      158 non-null int64  
제목            158 non-null object  
주소            158 non-null object  
조회수          158 non-null object  
좋아요          158 non-null int32  
싫어요          158 non-null int64  
댓글수          158 non-null object  
본문내용        158 non-null object  
구분            158 non-null int64  
dtypes: int32(1), int64(3), object(5)  
memory usage: 11.7+ KB
```

개
개

비교 채널 수
영상 수
영상에서 추출한 댓글 수

```
paw_reply.info() # 통합한 동영상별 댓글list DF 확인
```

```
<class 'pandas.core.frame.DataFrame'>  
Int64Index: 3160 entries, 0 to 699  
Data columns (total 6 columns):  
Unnamed: 0      3160 non-null int64  
ID              3160 non-null object  
Comment         3160 non-null object  
Like            3160 non-null int64  
Title_number    3160 non-null int64  
구분            3160 non-null int64  
dtypes: int64(4), object(2)  
memory usage: 172.8+ KB
```

데이터 전처리 과정

두 데이터 통합 새 파일 생성하여 통합 – 및 통합

paw.head()

Unnamed: 0	제목	주소	조회수	좋아요	싫어요	댓글수	본문내용	구분
0	[Ep.67] 전 세계 세터라 (feat. 외교부)	/watch?v=yUJAvW2Rykc	1,381,698	42000	749	6,838	K-펭귄 한다 해외진출	1
1	[Ep.66] 화보 모델 펭수	/watch?v=XUM3sH1kBtw	1,583,971	46000	455	7,678	펭수가 화보 모델로서 촬영을 하게 되었 다! 화보 촬영을 위한 펭수의 눈물(?) 나는 ...	1
2	펭수와 팬들의 최강 콜라베이션 신상유비 (feat. 찰린지♡)	/watch?v=LPmyxMH96S8	604,376	29000	135	3,921	힙합펭수의 커버영상이 드. 디. 어. 찾아 왔다! 월미도에서 촬영했던 미공개 영상과 여...	1
3	[Ep.65] 세상에 나쁜 펭귄은 없다.	/watch?v=wedLGh2jxkQ	1,975,593	50000	548	6,974	어느 날 갑자기 펭수에게 이상한 행동들 이 보인다? 걱정된 제작진들이 긴급 솔 루션 ...	1
4	수험생은 지금 당장 이 영상을 보 니다 (feat. 정승제쌤)	/watch?v=25RhzK3HuYM	651,078	20000	135	2,760	내일 모레 엄청 큰 시험을 앞둔 여러분들 을 위해 이 영상을 바칩니다! 펭	1

paw_reply.head()

Unnamed: 0	ID	Comment	Like	Title_number	구분
0	EBSDocumentary (EBS 다큐)	펭수...곤 백만이네,,, 짜식 멋지다,,, 날씨가 많이 춥다,,, 감기 조심허구...	2500	0	1
1	펭랑단1호	악플러들 고소하고 EBS 빛 갇자	1300	0	1
2	Isabel la	아니 무슨 교육방송이 ㅋㅋㅋ 웬만한 예능프로그램보다 더 잘만들어 자막이고 편집이고 ...	1000	0	1
3	gravity	위험한 물품 가지고 있어여? "제 자신" ㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋ	652	0	1
4	월드곰탕이	초반 영상부터 다 봐온 사람으로서 열심히 한 죄밖에 없는데너무 힘들게 하는듯나쁜 댓글...	924	0	1

통합한

정보 확인

동영상

개의 영상

동영상 별 댓글

개의 댓글

데이터 전처리 과정

통합된 데이터 전처리 – 토큰화 명사분석 불용어 제거 어간추출

명사 분석 한글 데이터 특성 상 유의미한 가장 작은 단위인 형태소로 변환 후
의미 없는 데이터들 조사 연결어 등 을 확인

불용어 제거 및 의미 있는 단어 추출이 가능한 명사 분석으로 토큰화

```
# 형태소 분석
from konlpy.tag import Okt
okt=Okt()
```

```
# 제목 리스트 명사 분석
title_list=[]
for i in range(len(peng_and_work_title_list)):
    title_list.append(okt.nouns(peng_and_work_title_list[i]))
```

```
# 본문 내용 명사 분석
content_list=[]
for i in range(len(peng_and_work_content_list)):
    content_list.append(okt.nouns(peng_and_work_content_list[i]))
```

```
# 댓글 내용 명사 분석
reply_list=[]
for i in range(len(peng_and_work_reply_list)):
    reply_list.append(okt.nouns(peng_and_work_reply_list[i]))
```

```
# 리스트 안의 리스트 하나의 리스트로 만들기
def flatten (n):
    org =[]
    for i in n :
        if (isinstance(i,list)):
            org += flatten(i)
        else:
            org.append(i)
    return org
```

데이터 전처리 과정

통합된 데이터 전처리 – 토큰화 명사분석 불용어 제거 어간추출

```
flatten(title_list)
flatten(content_list)
flatten(reply_list)
```

```
['핑수',
 '해외진출',
 '꿈',
 '응원',
 '아세안',
 '특별',
 '정상',
 '회의',
 '홍보',
 '핑권',
 '악플러',
 '빛',
 '무스',
 '교육방송',
 '예능',
 '프로그램',
 '더',
 '자막',
 '편집',
 '프로그램']
```

명사 추출결과 출력

```
title_f = flatten(title_list)
content_f = flatten(content_list)
reply_f = flatten(reply_list)
```

데이터 전처리 과정

두 데이터 통합 – 빈도분석 확인

```
title_f = flatten(title_list)
content_f = flatten(content_list)
reply_f = flatten(reply_list)
```

```
# 단어 빈도수 확인을 위한 패키지 import
from nltk.book import *
import operator
from nltk.corpus import brown
from nltk.corpus import stopwords
```

```
# 제목 빈도수 확인
fdist1 = FreqDist(title_f)
dict_w = {}
for w in title_f:
    dict_w[w] = fdist1[w]
resultdict = sorted(dict_w.items(), key=operator.itemgetter(1), reverse=True)
print(resultdict)
```

```
[('펍수', 87), ('워크맨', 38), ('펍권', 30), ('리뷰', 29), ('알바', 28), ('펍', 13), ('채널', 12), ('공개', 11), ('독립', 10), ('매니저', 9), ('연습생', 9), ('세상', 8), ('자이언트', 8), ('장성규', 8), ('라이브', 7), ('독자', 7), ('눈물', 7), ('역대', 7), ('최초', 7), ('선배', 5), ('직캠', 5), ('투표', 5), ('게임', 5), ('인', 5), ('것', 5), ('데뷔', 5), ('민속촌', 5), ('언박싱', 4), ('대결', 4), ('편', 4), ('방송', 4), ('마감', 4), ('선공', 4), ('현장', 4), ('도전', 4), ('거대', 4), ('이말년', 4), ('고양예고', 4), ('기념', 4), ('관종', 4), ('남매', 4), ('폭발', 4), ('세계', 3), ('린지', 3), ('영상', 3), ('남극', 3), ('등장', 3), ('우리', 3), ('남', 3), ('첫', 3), ('학교', 3), ('극장', 3), ('스타', 3), ('진짜', 3), ('펍수쇼', 3), ('안', 3), ('힐링', 3), ('선물', 3), ('본격', 3), ('알', 3), ('의', 3), ('수능', 3), ('이야기', 3), ('말', 3), ('피지', 3), ('컬', 3), ('갤러리', 3), ('빽빽이', 3), ('아저씨', 3), ('굿', 3), ('선', 3), ('대회', 3), ('전지영', 3), ('은밀', 3), ('명', 3), ('올파', 3), ('댓글', 3), ('두', 3), ('주의', 3), ('일', 3), ('홍대', 3), ('뽀로로', 3), ('인생', 3), ('먹방', 3), ('배달', 3), ('시급', 3), ('사장', 3), ('드립', 3), ('끝판', 3), ('꿀', 3), ('극한', 3), ('게', 2), ('팬', 2), ('신상', 2), ('이', 2), ('스', 2), ('인간', 2), ('버튼', 2), ('처음', 2), ('나', 2), ('공', 2), ('비화', 2), ('개', 2), ('정복기', 2), ('추억', 2), ('친구', 2), ('한류', 2), ('축하', 2), ('전격', 2), ('나이', 2), ('살', 2), ('내', 2), ('시간', 2), ('선생님', 2), ('준비', 2), ('쑥쑥', 2), ('부', 2), ('당신', 2), ('눈', 2), ('잔소리', 2), ('오디션', 2), ('참치', 2), ('송', 2), ('소개', 2), ('테스트', 2), ('복지', 2), ('종', 2), ('세트', 2), ('담양', 2), ('누가', 2), ('낙시', 2), ('공
```

데이터 추이 확인을 위해 빈도분석 진행

의

를 활용하여 빈도수 기준 정렬함 제목 본문내용 댓글 각각 정렬

데이터 전처리 과정

두 데이터 통합 – 빈도분석 확인

본문 내용 빈도수 확인

```
fdist1 = FreqDist(content_f)
```

```
dict_w = {}
```

```
for w in content_f:
```

```
    dict_w[w] = fdist1[w]
```

```
content_dict = sorted(dict_w.items(), key=operator.itemgetter(1), reverse=True)
```

```
print(content_dict)
```

```
[('핑수', 130), ('리뷰', 74), ('알바', 69), ('장성규', 60), ('워크맨', 47), ('직업', 37), ('스튜디오', 34), ('핑', 28), ('핑권', 25), ('인력', 23), ('것', 20), ('영상', 19), ('독자', 18), ('자이언트', 18), ('매니저', 16), ('소장', 16), ('잡것', 16), ('공개', 14), ('수', 14), ('친구', 13), ('여러분', 12), ('연습생', 12), ('나', 11), ('우리', 11), ('위해', 10), ('선배', 10), ('덜', 10), ('위', 9), ('이번', 9), ('이', 9), ('남극', 9), ('더', 9), ('편', 9), ('세상', 7), ('거', 7), ('꿀팁', 7), ('앞', 7), ('크리에이터', 7), ('뽀로로', 7), ('시간', 6), ('하나', 6), ('방', 6), ('게', 6), ('내', 6), ('준비', 6), ('말', 6), ('지금', 6), ('눈', 6), ('게임', 6), ('사람', 6), ('방송', 6), ('팬', 6), ('체험', 6), ('스승', 6), ('채널', 6), ('에버랜드', 6), ('눈물', 5), ('핑핑', 5), ('워', 5), ('오', 5), ('과연', 5), ('이유', 5), ('시작', 5), ('플', 5), ('인', 5), ('알', 5), ('라이브', 5), ('도전', 5), ('만', 5), ('명', 5), ('안', 5), ('첫', 5), ('배달', 5), ('데뷔', 5), ('실화', 5), ('남매', 5), ('알바생', 5), ('술집', 5), ('촬영', 4), ('직접', 4), ('힐합', 4), ('린지', 4), ('참여', 4), ('날', 4), ('갑자기', 4), ('진', 4), ('가지', 4), ('기', 4), ('셀럽', 4), ('한국', 4), ('제일', 4), ('안전', 4), ('대결', 4), ('시민', 4), ('잔소리', 4), ('수가', 4), ('콜라보', 4), ('오늘', 4), ('성우', 4), ('고민', 4), ('달성', 4), ('한강', 4), ('꿀', 4), ('덧글', 4), ('민속촌', 4), ('버스킹', 4), ('주의', 4), ('제주도', 4), ('일', 4), ('소', 4), ('화보', 3), ('모델', 3), ('디', 3), ('보고', 3), ('확인', 3), ('의', 3), ('진출', 3), ('방출', 3), ('주재', 3), ('사랑', 3), ('건', 3), ('구독', 3), ('난', 3), ('패션', 3), ('특별', 3), ('일일', 3), ('핑수쇼', 3), ('공연', 3), ('발라드', 3), ('락', 3), ('분', 3), ('투표', 3), ('직캠', 3), ('원', 3), ('공포', 3), ('특집', 3), ('핑하', 3), ('미디어', 3), ('때문', 3), ('두', 3), ('여름', 3), ('끝', 3), ('미용실', 3), ('사실', 3), ('홈', 3), ('맨', 3), ('이말년', 3), ('손흥민', 3), ('고양예고', 3), ('예술', 3), ('주목', 3), ('덕분', 3), ('아이돌', 3), ('현장', 3), ('홍보', 3), ('광', 3), ('손배침', 3), ('동년배', 3), ('소통', 3), ('인생', 3), ('유튜브', 3), ('보기', 3), ('진짜', 3), ('보이', 3), ('계속', 3), ('전지영', 3), ('선생님', 3), ('때', 3), ('어린이집', 3), ('항상', 3), ('요즘', 3), ('울', 3), ('술', 3), ('편의점', 3), ('무엇', 3), ('등장', 3), ('도미노피자', 3), ('독립', 3), ('맥주', 3), ('뜻밖', 3), ('영화관', 3), ('워터파크', 3), ('해외진출', 2), ('노력', 2), ('수의', 2), ('커버', 2), ('드', 2), ('월미도', 2), ('무비', 2), ('모든', 2), ('리크', 2), ('연', 2), ('꽃', 2), ('내외', 2), ('비행', 2), ('동차현', 2), ('장', 2), ('통해', 2)]
```

본문내용 빈도수 기준 정렬

두 데이터 통합 – 빈도분석 확인

```
print(reply_dict)
```

가장 많은 댓글에서 확인됨

	0	1
0	평수	87
1	워크맨	38
2	평권	30
3	리뷰	29
4	알바	28

```
reply_df = DataFrame(reply_dict)
```

댓글 내용 빈도수 정렬까지 완료 후 각 데이터 프레임 형태로 저장

데이터 시각화 및 분석

좋아요 순 정렬 및 시각화

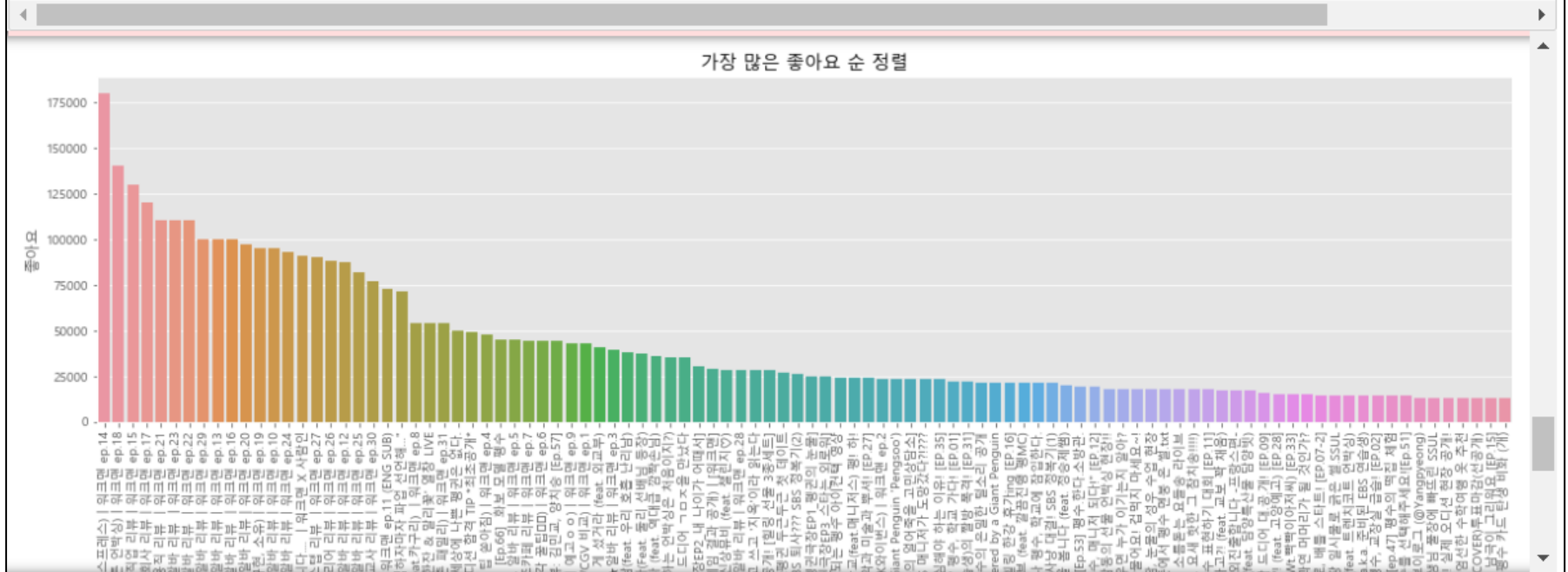
새로운 값 생성 및 시각화

재분리를 통해 두 값의 차이를 분별

워드클라우드 생성 동영상 별 제목 이용

데이터 시각화 – 좋아요 순 정렬 및 시각화

```
#시각화
fig = plt.figure(figsize=(20,5))
plt.title('가장 많은 좋아요 순 정렬')
sns.barplot(x='제목', y='좋아요', data=paw[['제목', '좋아요']].groupby('제목', as_index=False).mean().sort_values(by='좋아요', ascending=False))
plt.xticks(rotation=90); #x축(country) 회전
```



```
# 구분에 따른 요약
paw[['좋아요', '구분']].groupby('구분', as_index=False).mean().sort_values(by='좋아요', ascending=False).head()
```

구분	좋아요	워크맨의 좋아요가 약 배 이상 많음
0 워크맨	79628.571429	이는 워크맨의 구독자가 배 이상 많으며 조회수 또한 동일함
1 팽수	15737.398374	조회수는 많은 경우 이 넘어 에서 가끔 처리를 못해 '좋아요' 기준을 정렬하였음

데이터 시각화 – ‘ 추가 생성 및 시각화 둘 간 기존 속성값들의 수치 차이로 인해 평균값 비교를 위한 생성

```
##### UNLIKE RATIO 구하기 #####
# 워킹맨 구독자 수가(353만명) 펑수(134만명)의 3배이상으로,
# 비율을 구해 서로의 값을 확인 예정

unlike_ratio_list = []
for i in range(len(paw)):
    unlike_ratio = (paw.iloc[i, 5] / paw.iloc[i, 4]) * 100 # 퍼센테이지 표현
    unlike_ratio_ar = round(unlike_ratio, 2) # 소숫점 2자리까지 표현
    unlike_ratio_list.append(unlike_ratio_ar)

paw['unlike_ratio(%)'] = unlike_ratio_list # ratio 열 생성
```

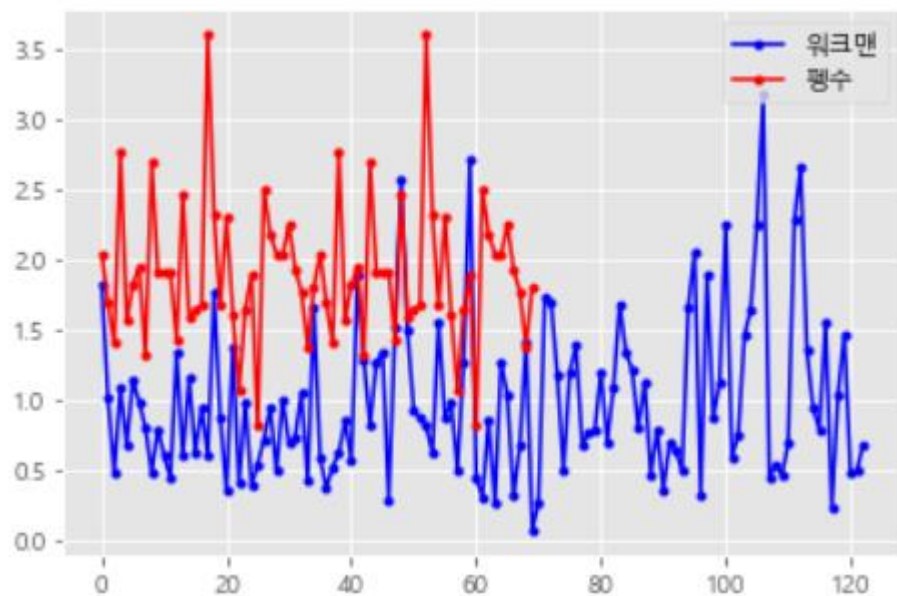
```
paw.head()
```

Unnamed: 0		제목	주소	조회수	좋아요	싫어요	댓글 수	본문내용	구분	unlike_ratio(%)
0	0	[Ep.67] 전 세계 게 샀거라 (feat. 외교부)	/watch?v=yUJAvW2Ryk	1,374,702	41000	748	20	K-펑권 한다 해외진출	펑수	1.82
1	1	[Ep.66] 화보 모델 펑수	/watch?v=XUM3sH1kBtw	1,562,005	45000	453	20	펑수가 화보 모델로서 촬영을 하게 되었다! 화보 촬영을 위한 펑수의 눈물(?) 나는 ...	펑수	1.01
2	2	펑수와 팬들의 최강 콜라베이션 신상유비 (feat. 챌린지♡)	/watch?v=LPmyxMH96S8	600,930	28000	133	20	힙합펑수의 커버영상이 드. 디. 어. 찾아왔다! 웰미도에서 촬영했던 미공개 영상과 여...	펑수	0.48
3	3	[Ep.65] 세상에 나쁜 펑권은 없다.	/watch?v=wedLGh2jxkQ	1,944,586	50000	539	20	어느 날 갑자기 펑수에게 이상한 행동들이 보인다? 걱정된 제작진들이 긴급 솔루션 ...	펑수	1.08
4	4	수험생은 지금 당장 이 영상을 봅시다 (feat. 정승제쌤)	/watch?v=25RhZK3HuYM	648,478	20000	135	20	내일 모레 엄청 큰 시험을 앞둔 여러분들을 위해 이 영상을 바칩니다! 펑펑	펑수	0.68

데이터 시각화 – 시각화 가 통합되어 있어 색상 구별을 위해 를 재분리 후 시각화

```
# unlike에 따른 정렬 및 시각화
peng_list=[] # 현재 df가 통합되어, 색상 분리를 위해 다시 분리
work_list=[]

for i in range(len(paw)):
    if (paw.iloc[i,8]) == '펭수': # '구분'이 펭수일 경우
        peng=paw.iloc[i,9] # 펭수의 ratio를 peng_list에 삽입
        peng_list.append(peng)
    else:
        work=paw.iloc[i,9] # '구분'이 워크맨일 경우
        work_list.append(work) # 워크맨의 ratio를 work_list에 삽입
```



```
plt.plot(peng_list, marker='.', color='b') # 펭수: blue
plt.plot(work_list, marker='.', color='r') # 워크맨: red
plt.legend(['워크맨', '펭수'], loc='upper right')
```

	구분	unlike_ratio(%)
0	워크맨	1.901143
1	펭수	0.995691

상대적으로
워크맨의 비선호도가 높음 약

두 데이터 통합 – 워드클라우드 확인

워드 클라우드 설정

```
title_12s=(''.join(peng_and_work["제목"])) # peng_and_work의 제목df를 string으로
wc = WordCloud(max_font_size=200,font_path=fpath,stopwords=STOPWORDS,background_color='#FFFFFF',
               ,width=1200,height=800).generate(title_12s)
```

#사이즈 등 설정

```
plt.figure(figsize=(10,8))
plt.imshow(wc)
plt.tight_layout(pad=0)
plt.axis('off')
```



데이터 시각화 – 워드클라우드

동영상 를 통해 ‘제목’ 속성으로 워드 클라우드

‘펑수’, ‘워크맨’이 가장 많이 출력되나

해당 채널의 성격을 보여주는 ‘알바 리뷰’, ‘feat’, ‘EBS’ 등도 보임

```
# 워드 클라우드 설정
```

```
title_l2s=(''.join(peng_and_work["제목"])) # peng_and_work의 제목df를 string으로
```

```
#사이즈 등 설정
```

```
plt.figure(figsize=(10,8))
```

```
plt.imshow(wc)
```

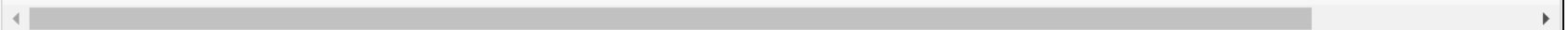
```
plt.tight_layout(pad=0)
```

```
plt.axis('off')
```

```
image_test=np.array(Image.open("ytbb.png")) #mask 이미지 설정
```

```
wc = WordCloud(max_font_size=200,font_path=fpath,stopwords=STOPWORDS,background_color='#FFFFFF',width=1200,height=800,mask=image_test)
```

```
|
```



사용 알고리즘

학습 집합 및 테스트 집합 준비

```
##### Machine Learning #####
```

```
data = paw
data.head()
```

Unnamed: 0		제목	주소	조회수	좋아요	싫어요	댓글 수	본문내용	구분	unlike_ratio(%)
0	0	[Ep.67] 전 세계 게 샀거라 (feat. 외교부)	/watch?v=yUJAvW2Rykc	1,381,698	42000	749	6,838	K-펭귄 한다 해외진출	1	1.78
1	1	[Ep.66] 화보 모델 펭수	/watch?v=XUM3sH1kBTw	1,583,971	46000	455	7,678	펭수가 화보 모델로서 촬영을 하게 되었다! 화보 촬영을 위한 펭수의 눈물(?) 나는 ...	1	0.99
2	2	펭수와 팬들의 최강 콜라보레이션 신상뮤비 (feat. 챌린지♡)	/watch?v=LPmyxMH96S8	604,376	29000	135	3,921	힙합펭수의 커버영상이 드. 디. 어. 찾아왔 다!월미도에서 촬영했던 미공개 영상과 여...	1	0.47
3	3	[Ep.65] 세상에 나쁜 펭귄은 없다.	/watch?v=wedLGh2jxkQ	1,975,593	50000	548	6,974	어느 날 갑자기 펭수에게 이상한 행동들이 보인다? 걱정된 제작진들이 긴급 솔루션 ...	1	1.10
4	4	수험생은 지금 당장 이 영상을 보시다 (feat. 정승제쌤)	/watch?v=25RhZK3HuYM	651,078	20000	135	2,760	내일 모레 엄청 큰 시험을 앞둔여러분들을 위해 이 영상을 바칩니다! 펭펭	1	0.68

머신러닝 사용
동영상의

준비
에 따른 구분 펭수 워크맨 을 예측

학습 집합 및 테스트 집합 준비

```
print ("Data shape:", paw.shape, "\n")
print (data.info())
```

Data shape: (158, 10)

행
열

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 158 entries, 0 to 34
Data columns (total 10 columns):
Unnamed: 0      158 non-null int64
제목            158 non-null object
주소            158 non-null object
조회수          158 non-null object
좋아요          158 non-null int32
싫어요          158 non-null int64
댓글수          158 non-null object
본문내용        158 non-null object
구분            158 non-null int64
unlike_ratio(%) 158 non-null float64
dtypes: float64(1), int32(1), int64(3), object(5)
memory usage: 18.0+ KB
None
```

불필요한 데이터 삭제

```
data.drop('제목', axis=1, inplace=True) # 무의미
data.drop('주소', axis=1, inplace=True) # 무의미
data.drop('조회수', axis=1, inplace=True) # 120만 넘을 경우, python에서 처리 못함
data.drop('댓글수', axis=1, inplace=True) # object
data.drop('좋아요', axis=1, inplace=True) # unlike_ratio 속성과 성격 동일
data.drop('싫어요', axis=1, inplace=True) # unlike_ratio 속성과 성격 동일
data.drop('본문내용', axis=1, inplace=True) # 무의미
```

학습 집합 및 테스트 집합 준비

```
# test / train data 분리
rows_count=len(data) # train : test data를 7:3으로 나누기위한 수
print("Total-data:",rows_count) # 158

train_rows = int(rows_count*0.7); # train data는 70% = 110
test_rows = int(rows_count*0.3); # test data는 30% = 48

data_train = data[:train_rows]
data_test = data[train_rows:rows_count]
print("실제로 나뉜 수:", "(Train)",len(data_train), "(Test)",len(data_test)) # 첫자리 포함하기 때문
```

Total-data: 158
실제로 나뉜 수: (Train) 110 (Test) 48

기존
를
비율로 나눔

```
X_train = data_train.drop("구분", axis=1) # x_train에서는 '구분' 없이 예측
Y_train = data_train["구분"] # y_train은 결과값 뿐이므로

X_test = data_test.drop("구분", axis=1).copy()
print("X_train:", X_train.shape, ", Y_train:", Y_train.shape, ", X_test:", X_test.shape)
```

X_train: (110, 2) , Y_train: (110,) , X_test: (48, 2)

와
를
하여
구분 펄수 워크맨
가 값을 잘 예측하는지 확인

1. Logistic Regression

```
logreg = LogisticRegression(solver='lbfgs')  
logreg.fit(X_train, Y_train)  
  
type(X_train)  
type(Y_train)  
  
Y_pred = logreg.predict(X_test)  
acc_log = round(logreg.score(X_train, Y_train) * 100, 2)  
acc_log
```

93.64

2. Support Vector Machines

```
svc = SVC()  
solver='liblinear'  
svc.fit(X_train, Y_train)  
  
Y_pred = svc.predict(X_test)  
acc_svc = round(svc.score(X_train, Y_train) * 100, 2)  
print(acc_svc)
```

93.64

3. Decision Tree

```
decision_tree = DecisionTreeClassifier()  
decision_tree.fit(X_train, Y_train)  
Y_pred = decision_tree.predict(X_test)  
acc_decision_tree = round(decision_tree.score(X_train, Y_train) * 100, 2)  
acc_decision_tree
```

100.0

4. KNN

```
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, Y_train)
Y_pred = knn.predict(X_test)
acc_knn = round(knn.score(X_train, Y_train) * 100, 2)
acc_knn
```

93.64

5. Gaussian Naive Bayes

```
gaussian = GaussianNB()
gaussian.fit(X_train, Y_train)
Y_pred = gaussian.predict(X_test)
acc_gaussian = round(gaussian.score(X_train, Y_train) * 100, 2)
acc_gaussian
```

94.55

6. Linear SVC

```
linear_svc = LinearSVC()
linear_svc.fit(X_train, Y_train)
Y_pred = linear_svc.predict(X_test)
acc_linear_svc = round(linear_svc.score(X_train, Y_train) * 100, 2)
acc_linear_svc
```

c:\users\백송이\appdata\local\programs\python\python36\lib\site-packages\sk
 iled to converge, increase the number of iterations.
 "the number of iterations.", ConvergenceWarning)

93.64

7. Perceptron

```
perceptron = Perceptron()
perceptron.fit(X_train, Y_train)
Y_pred = perceptron.predict(X_test)
acc_perceptron = round(perceptron.score(X_train, Y_train) * 100, 2)
acc_perceptron
```

90.0

8. Random Forest

```
random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train, Y_train)
Y_pred = random_forest.predict(X_test)
random_forest.score(X_train, Y_train)
acc_random_forest = round(random_forest.score(X_train, Y_train) * 100, 2)
acc_random_forest
```

100.0

9. Stochastic Gradient Descent

```
sgd = SGDClassifier()
sgd.fit(X_train, Y_train)
Y_pred = sgd.predict(X_test)
acc_sgd = round(sgd.score(X_train, Y_train) * 100, 2)
acc_sgd
```

80.91

알고리즘 별

정렬

```
models = pd.DataFrame({  
    'Model': ['Support Vector Machines', 'KNN', 'Logistic Regression',  
              'Random Forest', 'Naive Bayes', 'Perceptron',  
              'Stochastic Gradient Decent', 'Linear SVC',  
              'Decision Tree'],  
    'Score': [acc_svc, acc_knn, acc_log,  
              acc_random_forest, acc_gaussian, acc_perceptron,  
              acc_sgd, acc_linear_svc, acc_decision_tree]})  
models.sort_values(by='Score', ascending=False)
```

	Model	Score
3	Random Forest	100.00
8	Decision Tree	100.00
4	Naive Bayes	94.55
0	Support Vector Machines	93.64
1	KNN	93.64
2	Logistic Regression	93.64
7	Linear SVC	93.64
5	Perceptron	90.00
6	Stochastic Gradient Decent	80.91

의 성공률이
가
가장 높았으며
가장 낮았음
는 지난 학기 과제에서 또한
높은 성공율을 보여주었음

최종 파일 생성

```
submission = pd.DataFrame({
    "Unlike_ratio(%)": data_test["unlike_ratio(%)"],
    "구분(펍수:1/워크맨:0)": Y_pred
})
submission.to_csv('2019_TextMining_Team_num_4.csv', mode='w', encoding="utf-8-sig", index=False)
```

```
data_final = pd.read_csv('2019_TextMining_Team_num_4.csv')
data_final.info()
data_final.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48 entries, 0 to 47
Data columns (total 2 columns):
Unlike_ratio(%)      48 non-null float64
구분(펍수:1/워크맨:0)  48 non-null int64
dtypes: float64(1), int64(1)
memory usage: 848.0 bytes
```

	Unlike_ratio(%)	구분(펍수:1/워크맨:0)
count	48.000000	48.000000
mean	2.057500	0.416667
std	0.450317	0.498224
min	1.560000	0.000000
25%	1.720000	0.000000
50%	1.910000	0.000000
75%	2.302500	1.000000
max	3.600000	1.000000

최종 값을 csv에 저장.

48개의 test data는 높은 성공율을 보였음.
초기 데이터 수는 펍수 121개 / 워크맨 35개로
구성되었으나, Unlike_ratio로 미리 정렬 후
Train/Test Data를 분리하여 해당 Test data는
펍수 / 워크맨 골고루 분석된 것으로 보임.

```
#####  
##### NMF #####  
reply_f
```

```
['펜수',  
'골',  
'백만이',  
'짜척',  
'날씨',  
'출다',  
'감기',  
'조심',  
'허구',  
'국밥',  
'악플러',  
'빔',  
'무슨',  
'교육방송',  
'예능',  
'프로그램',  
'더',  
'자막',  
'편집',  
'공감'
```

```
# sklearn의 CountVectorizer 함수 사용, bow(bag of words)로 변환하는 객체 생성  
bow_transformer = CountVectorizer().fit(reply_f)
```

```
len(bow_transformer.vocabulary_)
```

```
5177
```

명사분석한 리뷰데이터 추가

```
##### NMF #####
reply_f

['펜수',
 '곧',
 '백만이',
 '자식',
 '날씨',
 '출다',
 '감기',
 '조심',
 '허구',
 '국밥',
 '있잖아',
 '빛',
 '무슨',
 '교육방송',
 '예능',
 '프로그램',
 '더',
 '자만',
 '면접',
 '유명인사',

# sklearn의 CountVectorizer 함수 사용, bow(bag of words)로
bow_transformer = CountVectorizer().fit(reply_f)

len(bow_transformer.vocabulary_)

5177
```

```
# reply_f 변형
reply_f = bow_transformer.transform(reply_f)

# reply_f 확인
print('Shape of Sparse Matrix: ', reply_f.shape)
print('Amount of Non-Zero occurrences: '), reply_f.nnz
```

```
Shape of Sparse Matrix: (27879, 5177)
Amount of Non-Zero occurrences:
(None, 20457)
```

```
# reply_f는 현재 document-term matrix로 구성됨.
# NMF를 적용하기 위해서는 term-document matrix가 필요.
# 현재의 Matrix를 transposed matrix(전치행렬)로 바꿔줘야 함.
type(reply_f)
```

```
scipy.sparse.csr.csr_matrix
```

```
# compressed sparse row matrix인 reply_f를 transpose 한다.
reply_f_trans = reply_f.transpose()
type(reply_f_trans)
```

```
scipy.sparse.csc.csc_matrix
```

```
print('Shape of Sparse Matrix: ', reply_f_trans.shape)
print('Amount of Non-Zero occurrences: '), reply_f_trans.nnz
```

```
Shape of Sparse Matrix: (5177, 27879)
Amount of Non-Zero occurrences:
(None, 20457)
```

명사분석한 리뷰데이터 추가
Trasform 후
Transposed matrix로 변환

```
# NMF 모델 객체 생성
from sklearn.decomposition import NMF
model = NMF(n_components=2)
```

```
# NMF 모델 학습
W = model.fit_transform(reply_f_trans)
```

```
H = model.components_
```

```
type(H)
```

```
numpy.ndarray
```

```
H.shape
```

```
(2, 27879)
```

```
# 클러스터링 결과 점수
```

```
print(H)
```

```
[[0.15668896 0.         ... 0.         0.         0.         ]
 [0.         0.         0.         ... 0.         0.         ]]
```

```
# 전체 문서 별 best 점수에 해당하는 클러스터 label
```

```
pred_labels = H.argmax(axis=0)
```

```
len(pred_labels)
```

```
27879
```

```
# 전체 문서 별 best 점수에 해당하는 클러스터 label
```

```
pred_labels = H.argmax(axis=0)
```

```
len(pred_labels)
```

```
27879
```

```
pred_labels
```

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

```
# 실제 labels이...? 없.....?
```

모델 학습 후
클러스터링 결과 점수(H)
Best 점수 추출 후 확인

nmf

전체 동영상 이미지 지도 쇼핑 더보기

설정 도구

검색결과 약 11,800,000개 (0.28초)

음수 미포함 행렬 분해(Non-negative matrix factorization, **NMF**)는 음수를 포함하지 않은 행렬 V 를 음수를 포함하지 않은 행렬 W 와 H 의 곱으로 분해하는 알고리즘이다. 행렬이 음수를 포함하지 않는 성질은 분해 결과 행렬을 찾기 쉽게 만든다.

NMF의 해석

행렬 V 의 행은 특성이며, 행렬 W 의 행은 특성의 강도, 행렬 H 의 행은 특성의 위치를 나타낸다.

www.slideshare.net

음수 미포함 행렬 분해 - 위키백과, 우리 모두의 백과사전

https://ko.wikipedia.org/wiki/음수_미포함_행렬_분해

이 결과에 관한 정보 사용자 의견

음수 미포함 행렬 분해 - 위키백과, 우리 모두의 백과사전

https://ko.wikipedia.org/wiki/음수_미포함_행렬_분해

음수 미포함 행렬 분해(Non-negative matrix factorization, **NMF**)는 음수를 포함하지 않은 행렬 V 를 음수를 포함하지 않은 행렬 W 와 H 의 곱으로 분해하는 알고리즘 ...

종류 · 알고리즘 · 성질 · 응용 사례

구글링 & 구글링...

NMF(non-negative matrix factorization)는 유용한 특성을 뽑아내기 위한 또 다른 비지도 학습 알고리즘

NMF에서는 음수가 아닌 성분과 계수 값을 찾음.

즉 주성분과 계수가 모두 0보다 크거나 같아야 함 ==> 이 방식은 음수가 아닌 특성을 가진 데이터에만 적용

음수가 아닌 **가중치 합으로 데이터를 분해하는 기능은 오디오 트랙이나 음악처럼 독립된 소스를 추가하여 만들어진 데이터에 특히 유용.**

이럴 때 NMF는 섞여 있는 데이터에서 원본 성분을 구분할 수 있음

음수로 된 성분이나 계수가 만드는 상쇄 효과를 이해하기 어려운 PCA보다 대체로 **NMF의 주성분이 해석하기는 쉬움**

인위적 데이터에 NMF 적용하기

NMF로 데이터를 다루려면 주어진 데이터가 **양수인지 확인**해야함

즉 원점 (0, 0)에서 상대적으로 어디에 놓여 있는지(**위치 벡터**)가 **NMF에서는 중요함**

그렇기 때문에 원점 (0, 0)에서 데이터로 가는 방향을 추출한 것으로 음수 미포함 성분을 이해가능

```
# library import
```

```
import mglearn
```

```
import matplotlib
```

```
import matplotlib.pyplot as plt
```

```
# matplotlib 설정
```

```
plt.rcParams['axes.unicode_minus']=False # 줄-설정
```

음성에서 특성을 추출하거나 사진인식을 인식하는 것처럼 특성을 추출하는데 유용한 알고리즘. -무감독 학습 알고리즘임.

- 인수분해라는것은 어떤 원소를 더 기초적이고 간단한 조각으로 분해하는것이 목적이기 때문에 데이터를 조금더 기초적인 특성으로 분해하는데 이용할수 있다.
- 이것을 행렬의 인수분해라는것을 이용해서 유도해 낼수 있다.

뉴스기사에서 단어를 기반으로 특성을 추출하는 예를 이용해서 알고리즘을 설명하면 다음과 같다.

1. 데이터 행렬 = 특성행렬 * 가중치 행렬.

--> 이러한 원리를 이용해서 데이터를 기초로 특성을 추출할수 있다.

	단어1, 단어2, 단어3		특성1, 특성2, 특성3
기사 1.	0 1 1	기사1	10 9 1
특성1	9 10 1		
기사 2.	2 5 3	기사2	9 2 2
특성2	7 2 1		
기사 3	0 4 2	기사 3	0 9 1
특성3	8 2 9		