

# 중간 보고서

미디어 통합 리뷰 어플

Vol.5



제출일	2020. 04. 03	전공	컴퓨터공학과
과목	졸업작품 프로젝트	학번	2015722084
			2015722083
담당교수	이기훈	이름	한승주
			김성종

## 목 차

### I 개요

#### 1. 배경 및 필요성

#### 2. 목적

ㄱ. 정보의 신뢰성

ㄴ. 접근의 용이성

#### 3. 설계 내용

ㄱ. Flow Chart

ㄴ. 개념 설계

### II 과제 수행

#### 1. 수행 일정

#### 2. 웹크롤링

#### 3. DB 구축

#### 4. 단어 사용 빈도 추출

#### 5. UI 제작

### III 과제 평가

#### 1. 개선방안

#### 2. 기대효과

ㄱ. 기업적 측면

ㄴ. 사용자 측면

### < 참고문헌 >

# I. 개요

## 1. 배경 및 필요성

- 최근에는 스트리밍 플랫폼이라는 새로운 서비스가 음악이나 단순 동영상이 아니라 영화, TV, 서적까지 분야에 있어 확장되면서 미디어계의 새로운 패러다임으로 떠오르기 시작했다. 영화와 TV라는 미디어는 “넷플릭스”를 선두로 현재까지 많은 스트리밍 플랫폼이 생겨났는데 단순히 미디어 스트리밍뿐만이 아닌 자체 오리지널 시리즈를 제작하기도 하고 기존 영화 개봉은 영화관이라는 틀을 바꾸어 플랫폼 최초 공개까지 영화 산업의 전반적인 흐름을 바꾸고 있다고 해도 과언이 아니다. 이런 스트리밍 서비스는 출판업에까지 그 분야를 더욱 확장하여 기존 서점이나 도서관에서 실제 종이로 된 책을 인터넷으로 더욱 쉽게 가볍게 접할 수 있는 E-북 시장이 활성화 되어 “네이버 시리즈”, “카카오 페이지”, “리디북스” 여러 플랫폼이 만들어졌다.
- 이렇듯 과거의 문화 소비와는 달리, 많은 변화를 거친 현대에 미디어를 접하는 환경뿐 아니라 이에 대한 리뷰를 남기는 방식이나 플랫폼도 사용자에게 따라 저마다의 다른 의견을 가지고 서로 다른 특성을 보이기 때문에 특정 미디어에 대한 리뷰를 단 한 곳에서만 보고서 이 미디어가 어떤 미디어인지 판단하는 것은 불가능해졌다.
- 다시 말해, 미디어는 연령, 성별, 자라온 환경 등 많은 요인에 따라 각기 다른 후기를 남기며 개개인마다 보는 관점이 다르다. 따라서, 일반적인 제품 리뷰와는 다르게 미디어의 리뷰는 미디어를 접하고 난 후에 가지고 있는 신념, 자라온 문화 환경, 평소 자주 사용하는 어투 등에 따라 자신의 생각이 여과 없이 리뷰에 드러난다. 최근에는 이러한 리뷰들이 기업의 매출에 직결되고 있음을 종종 확인할 수 있다.
- 미디어를 접하는 플랫폼 뿐만 아니라, SNS에다가도 리뷰를 많이 남기는데 SNS 또한 나이대 혹은 성별에 따라서 선호하는 소셜미디어가 다르다. 실제 국내 페이스북 유저 비율은 남성이 여성 대비 14% 많고, 인스타그램은 여성이 남성 대비 4% 많은 비율을 가지고 있다. 또한 페이스북은 연령대가 고른 반면, 인스타그램은 20~30대 비율이 상대적으로 높다.
- 또, SNS의 특성에 따라 따라서도 리뷰를 남기는 방식의 차이가 있다. 인스타그램이 키워드 중심의 간결한 표현이 많다면 트위터는 일상생활 속 자신이 사용하는 어투 그대로를 적는 경우가 많다. 특히, 블로그는 상대적으로 육하원칙을 갖춘 완전한 말로 작성한다.

- 이러한 자신의 정치성향이나 가치관을 직접 반영할 수 있는 리뷰의 특성에 따라 일어나는 별점 테러와 같은 행위, 광고성 포스트들로 인해 진정한 리뷰를 찾기 힘들어진 지금, 미디어 플랫폼의 리뷰와, SNS를 통한 리뷰를 분석하여 신뢰성 있는 리뷰를 사용자에게 보이고, 그 외에도 트렌드나 추천 여부를 판단해보고자 한다.
- 특히, 미디어 중 웹소설을 중점으로 분석하려고 하는데, 2018년 한국 콘텐츠 진흥원의 조사에 따르면 국내 웹소설 시장은 2013년 100억원 규모에서 2018년 4000억원까지 40배 성장했다. 이러한 배경에는 웹소설을 바탕으로 제작된 드라마로 익히 알려져 있는 ‘성균관 스캔들’, ‘구르미 그린 달빛’ 등의 성공이 있다. 또한, 주변국인 일본은 남녀노소 상관없이 서브컬처 문화를 수용하고 이를 즐기고 있어 이러한 콘텐츠 시장이 이미 발달한 상태이고, 중국도 연간 2조원의 시장 규모를 가지고 있을 만큼 웹툰 및 웹소설 시장이 큰 편이다. 최근에는 중국, 일본 외에도 세계 각국의 작품들을 수입 및 수출하고 있고 one source multi use라는 개념에 따라, 새로이 제작되는 웹툰이나 드라마, 영화 등이 웹소설을 기반으로 한 것이 굉장히 많아 웹소설 시장의 가능성이 크다고 판단한다.
- 하지만 시장이 커진 만큼 너무나 많은 웹소설이 등장하고 있고, 이를 선택하기에는 하나씩 읽어봐야 하는 어려움이 있기 때문에 위의 기술들을 이용하여 의사결정을 보조하는 수단을 제작하고자 함이 본 작품의 배경이다.

## 2. 목적

### ㄱ. 정보의 신뢰성

기존의 리뷰 사이트뿐 아니라 여러 소셜미디어에 있는 관련 리뷰를 분석합니다. 여기서 관련 중요 키워드를 중심으로 데이터베이스를 구축하여 이렇게 만들어진 데이터베이스를 통해 현재 미디어 트렌드를 파악해보고 이와 관련되어 긍정적인 반응을 얻고 있는 작품을 추천할 수 있습니다. 전반적인 작품에 대한 반응을 살펴보고 이에 대해 긍정적 리뷰와 부정적 리뷰를 모두 보여줌으로써 이러한 리뷰를 통해 사용자는 해당 미디어에 대한 정보를 사전 습득할 수 있고 이와 비슷한 평가를 받은 미디어가 어떤 것이 있는지 확인하며 미디어의 추천으로 이어질 수 있고 또는 반대되는 평가를 받은 미디어 확인을 통해 미디어 선택에 도움을 줄 수 있다.

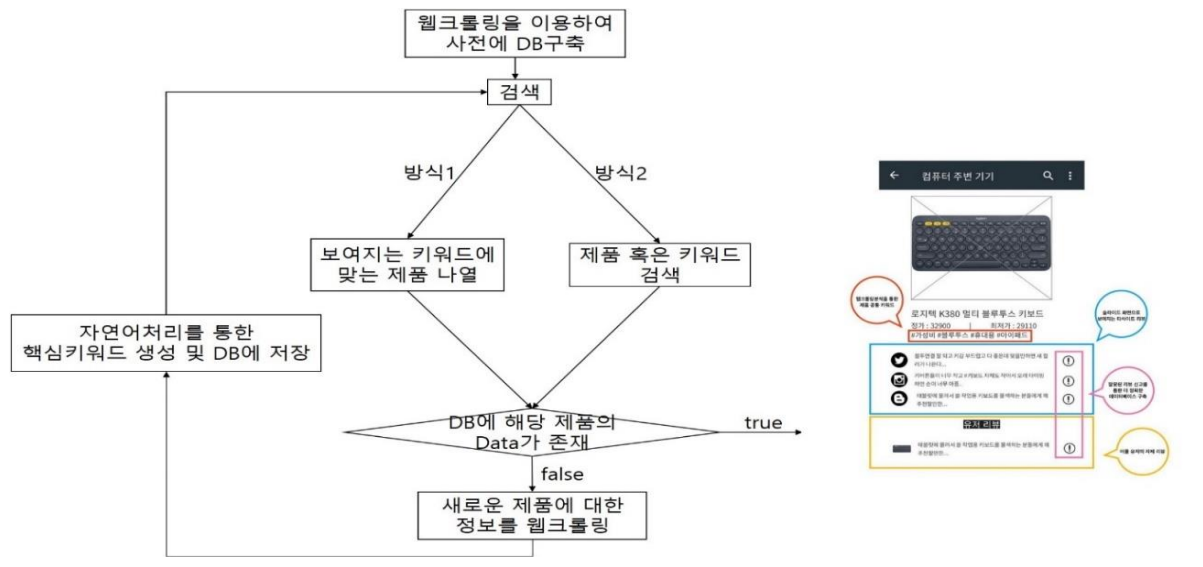
### ㄴ. 접근의 용이성

이를 어플리케이션(혹은 웹사이트)로 구현하여 사용자가 언제 어디서든 쉽게 접근하고 사용할 수 있도록 합니다.



### 3. 설계내용

#### ㄱ. Flow Chart



#### ㄴ. 개념 설계

##### 1) 플랫폼의 웹사이트 코드 분석 및 크롤링(Beautiful soup)

- DB 를 구축하기 위해 해당 웹사이트 접속 (\*리뷰 사이트, 블로그, 트위터, 인스타그램)
- 작품 기본 정보와 사용자 및 구매자의 리뷰가 담긴 웹사이트 코드 분석
- BS4를 이용해 페이지 데이터 호출
- 작품의 기본 정보 tag 를 찾아 추출
- 각 사이트와 페이지별로 링크를 재귀적으로 검색하여 데이터를 추출

##### 2) 크롤링된 정보를 이용한 DB 구축(Mysql or sqlite)

- MySQL 서버에 접속하여 데이터베이스 생성
- cursor 를 추출하여 execute 메서드로 SQL 을 실행, 테이블 생성
- Execute 메서드에 데이터를 계속 확장

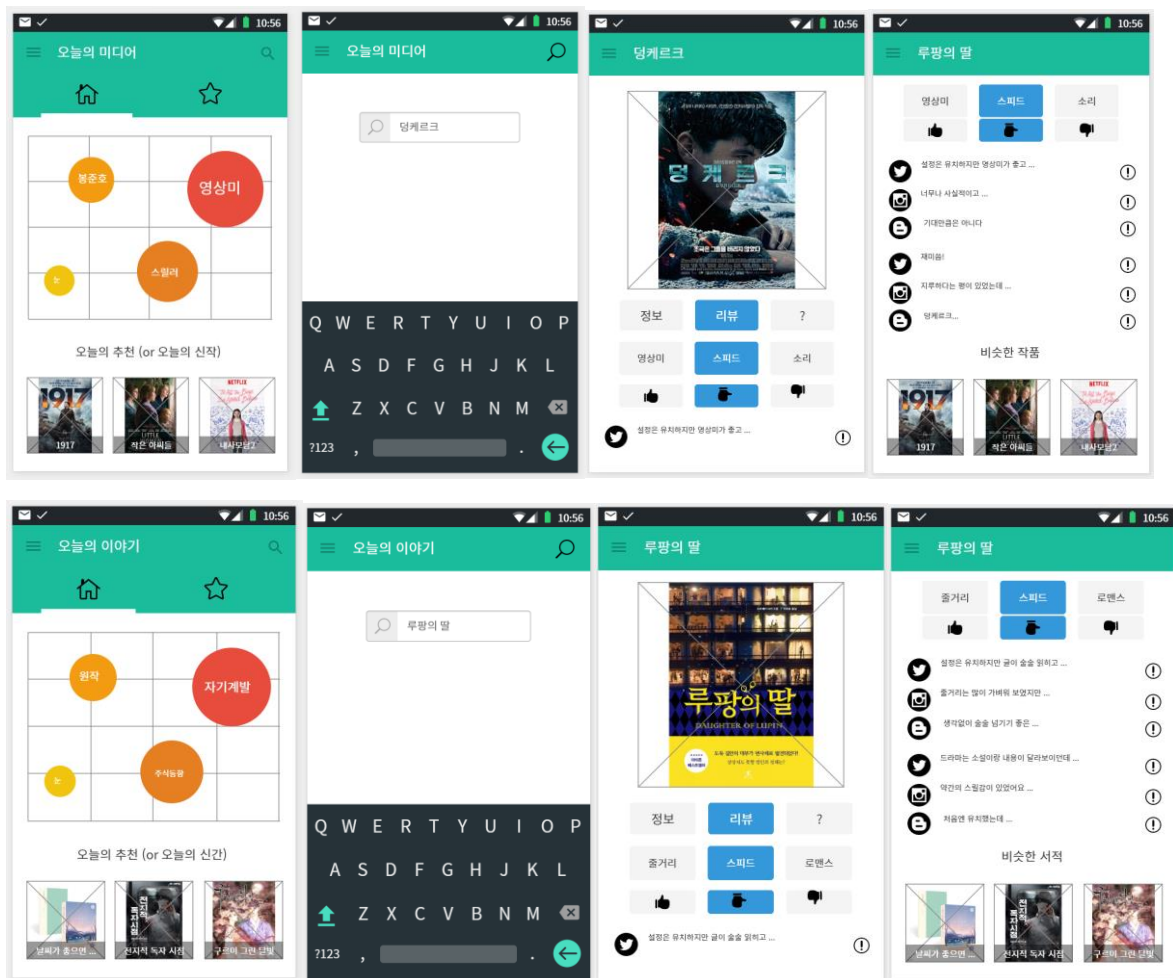
### 3) Kkma, Hannanum 을 이용한 KoNLP(키워드 생성)

- Kkma 나 Hannanum 모듈을 이용하여, 해당 모듈에 맞추어 입력된 문자열에서 키워드로 표현할 품사 추출
- 가장 빈도수가 높은 단어(키워드로 설정할 단어)를 DB 에 저장
- 오피니언 마이닝(감성 분석)을 이용하여 리뷰의 긍부정을 평가하고 어떤 평가가 많은지 분석한다. 최근 1-2달간의 리뷰를 토대로 현재의 트렌드를 파악한다.
- 긍부정 리뷰 평가 과정에서 각 단어의 속성 값을 추출하여 판별을 하는데 이 속성 값이 작품을 선정함에 있어 사용자의 기준이 되므로 이 키워드를 중요 키워드로 설정하고 관련 데이터 현황을 조사, 분석한다.
- 이 과정에서 마이닝의 정확도를 높이는 작업이 필요하다. 이 부분은 현재 나와있는 다양한 모델 기반 접근법을 이용한 감성사전을 이용하여 데이터의 정확도를 끌어올릴 계획이다. 각 감성 사전마다 어떤 단어나 조사를 제거하고 비속어, 신조어, 오타의 처리를 어떻게 하냐에 따라 정확도가 달라진다. 현 소셜미디어의 특성상 표준어를 사용하지 않고 발음을 있는 그대로 적어 두거나 비속어, 신조어, 약어 혹은 “이 장면 정말 사이다였다.”와 같이 원래 가지고 있는 역할과 다르게 빗대어 많이 사용되고 있는 언어들이 있으므로 이 부분에서 더 효과적인 판별법과 사전이 어떤 것인지 정확도를 높이는 데 중점을 두고자 한다. 표준어만을 이용한 감성 분석 API 및 툴들은 많이 존재하지만 비표준어는 경우의 수가 굉장히 다양하기 때문에 이를 처리하는 API 나 사전 구축에 대한 조사는 더 필요하다.
- 참고 분석 예정 : word2vec, sentiwordnet, [www.openhanquel.com](http://www.openhanquel.com), KTS
- 음소 단위 분할 조합을 통해 뜻을 파악하는 Trigram-Signature 를 사용하면 오타가 있는 “조ㅎ아”, “실ㅎ어”와 같은 문맥을 파악하는 데 유용할 것이라고 생각한다. 이를 이용한 API 가 있는지 혹은 이와 관련된 사전을 제작하여 사용하는지 더 조사할 계획이다.

#### 4) Android UI 제작

- 자신이 보고 싶은 작품의 리뷰를 보기 위한 제품의 검색창을 사용자가 보기 편하도록 UI로 구현
- 검색 결과로 작품의 기본 정보와 함께 리뷰를 보여준다. 리뷰의 경우 공통적으로 많이 언급이 되는 속성(엔터티, 특징)을 보여주고 이에 대한 리뷰를 선택 시 해당 리뷰를 사이트 별로 보여준다.

#### 5) 예시 결과





## II. 과제 수행

### 1. 수행 일정

	1月	2月	3月	4月	5月	6月	7月	8月	9月	10月
제안서 작성	■									
PYTHON,Android 기초및심화 학습	■	■								
Beautiful soup 등 API를 사용한 웹크롤링 및 DB구축		■	■	■						
중간보고서 작성				■						
자연어 처리를 이용한 데이터 가공					■	■	■			
가공된 데이터의 정확도 파악						■	■			
중간보고서 작성								■		
UI 제작									■	■
최종보고서 작성										■

## 2. 웹 크롤링

### ㄱ. 트위터

Vol1. 트위터의 정식 API 인 tweepy 가 존재는 하나 단점이 최근 7일간의 트윗만을 가져올 수 있기 때문에, 리뷰 분석을 하는데에 있어서 한계가 존재한다. 그래서 사용한 API 는 getolddtweet3(<https://github.com/Jefferson-Henrique/GetOldTweets-python>)이며, 해당 API 는 기간을 설정하여 트윗을 수집할 수 있다. 기간은 2019년 1년 동안 '로지텍'이라는 검색어가 들어간 트윗을 수집해보았다.

```
In [6]: try:
import GetOldTweets3 as got
except:
!pip install GetOldTweets3
import GetOldTweets3 as got

In [7]: import datetime

days_range = []

start = datetime.datetime.strptime("2019-01-01", "%Y-%m-%d")
end = datetime.datetime.strptime("2019-12-31", "%Y-%m-%d")
date_generated = [start + datetime.timedelta(days=x) for x in range(0, (end-start).days)]

for date in date_generated:
    days_range.append(date.strftime("%Y-%m-%d"))

print("=== 설정된 트윗 수집 기간은 {} 에서 {} 까지 입니다 ===".format(days_range[0], days_range[-1]))
print("=== 총 {}일 간의 데이터 수집 ===".format(len(days_range)))

=== 설정된 트윗 수집 기간은 2019-01-01 에서 2019-12-30 까지 입니다 ===
=== 총 364일 간의 데이터 수집 ===

In [8]: import time

# 수집 기간 맞추기
start_date = days_range[0]
end_date = (datetime.datetime.strptime(days_range[-1], "%Y-%m-%d")
            + datetime.timedelta(days=1)).strftime("%Y-%m-%d") # setUntil() 끝을 포함하지 않으므로, day + 1

# 트윗 수집 기준 정의
tweetCriteria = got.manager.TweetCriteria().setQuerySearch('로지텍')\
                .setSince(start_date)\
                .setUntil(end_date)\
                .setMaxTweets(-1)

# 수집 with GetOldTweets3
print("Collecting data start.. from {} to {}".format(days_range[0], days_range[-1]))
start_time = time.time()

tweet = got.manager.TweetManager.getTweets(tweetCriteria)

print("Collecting data end.. {0:0.2f} Minutes".format((time.time() - start_time)/60))
print("=== Total num of tweets is {} ===".format(len(tweet)))

Collecting data start.. from 2019-01-01 to 2019-12-30
Collecting data end.. 15.82 Minutes
=== Total num of tweets is 7521 ===
```

가져온 트윗에서 사용할 정보는 유저, 트윗 개시 날짜, 트윗 링크, 내용이며 해당 내용을 csv 파일로 저장한다.

```
In [10]: from random import uniform
from tqdm import tqdm_notebook

# initialize
tweet_list = []

for index in tqdm_notebook(tweet):

    # 메타데이터 목록
    username = index.username
    link = index.permalink
    content = index.text
    tweet_date = index.date.strftime("%Y-%m-%d")
    tweet_time = index.date.strftime("%H:%M:%S")

    # 결과 합치기
    info_list = [tweet_date, tweet_time, username, content, link]
    tweet_list.append(info_list)

    # 휴식
    time.sleep(uniform(1,2))

HBox(children=(IntProgress(value=0, max=7521), HTML(value='')))
```

```
In [12]: # 파일 저장하기

import pandas as pd

twitter_df = pd.DataFrame(tweet_list,
                           columns = ["date", "time", "user_name", "text", "link"])

# csv 파일 만들기
twitter_df.to_csv("sample_twitter_data_{}_to_{}.csv".format(days_range[0], days_range[-1]), index=False)
print("=== {} tweets are successfully saved ===".format(len(tweet_list)))

=== 7521 tweets are successfully saved ===
```

## 저장한 파일을 확인하면

```
In [13]: # 파일 확인하기

df_tweet = pd.read_csv('sample_twitter_data_{}_to_{}.csv'.format(days_range[0], days_range[-1]))
df_tweet.head(10) # 위에서 10개만 출력
```

	date	time	user_name	text	link
0	2019-12-30	18:37:18	s_NEGEV_s	와 근데 로지텍 마우스 G903이 HERO 센서 달고 오니까 배터리가 거의 닳지...	https://twitter.com/s_NEGEV_s/status/121171791...
1	2019-12-30	16:29:14	mookbini	로지텍 뻘손이라 키보드도 마우스도 헤드셋도 로지텍으로 하고싶었지만 만만...	https://twitter.com/mookbini/status/1211685685...
2	2019-12-30	16:09:43	KeepGoingMasiro	지금 게임용으로 쓰고 있는 키보드.. 사실 엄청 오래 되기는 했는데.. 아직도 잘...	https://twitter.com/KeepGoingMasiro/status/121...
3	2019-12-30	14:03:32	RyuZU_Seed	음 그리고 키보드는 COX CK450 마우스는 로지텍 G102 정도면.. 80 살짜...	https://twitter.com/RyuZU_Seed/status/12116490...
4	2019-12-30	11:34:39	nabislife	마우스 로지텍 mx 버티컬 쓰고있음	https://twitter.com/nabislife/status/121161155...
5	2019-12-30	10:37:24	lulul_jd	안녕하세요.. 멍청하게 자기 아이디도 모르는 놈,, 팔로 해주셔서 감사해요...	https://twitter.com/lulul_jd/status/1211597147...
6	2019-12-30	09:18:35	sinrisoung	킹킥히 유로트럭 최고세팅은 이거아님? 트리플모니터 하는 비용보다 vr저렴하...	https://twitter.com/sinrisoung/status/12115773...
7	2019-12-30	09:12:54	wannabecoolman	로지텍 블루키보드 고장났어.. 미쳤나... 내손에 남아나는게 없다는 뜻이니.....	https://twitter.com/wannabecoolman/status/1211...
8	2019-12-30	09:01:08	kkibaek	로지텍 키즈투고, 이거 사지 마세요. 블루투스 키보드의 본질을 놓친 키보드	https://twitter.com/kkibaek/status/12115729177...
9	2019-12-30	09:00:22	binu_4_lviz	로지텍 keys to go 키보드 타자 느낌이 좋아서 계속 치고 있음 ㅋㅋㅋㅋ	https://twitter.com/binu_4_lviz/status/1211572...

이러한 결과를 얻을 수 있다.

## ㄴ. 네이버

1차시: 실제로 많은 사용자들이 많이 보는 리뷰는 네이버일 것이다. 따라서 네이버 블로그의 검색어에 따른 결과 크롤링을 해보면 제목과 블로그 링크를 가져올 수 있다. 무한정으로 많은 양의 검색 결과를 가져올 수 없는데 이는 좋은 검색 결과를 위해 네이버가 1000건의 검색 결과만을 보여주고 있기 때문이다. 하지만 이것만으로도 충분히 키워드를 추출한다던지, 제품 한 개를 분석함에 있어서 부족함이 보이지는 않는다.

```
In [30]: import requests
import pandas as pd
from bs4 import BeautifulSoup
from collections import OrderedDict
from itertools import count

def mycrawler(input_search):
    url='https://search.naver.com/search.naver'
    post_dict=OrderedDict()

    for page in count(1):
        params={
            'query': input_search,
            'where': 'post',
            'start': (page-1)*10+1,
            'date_from': 20191231,
            'date_to': 20200220,
        }
        print(params)
        response = requests.get(url, params=params)
        html=response.text

        soup=BeautifulSoup(html, 'html.parser')

        title_list=soup.select('.sh_blog_title')

        for tag in title_list:
            if tag['href'] in post_dict:
                return post_dict

            print(tag.text, tag['href'])
            post_dict[tag['href']] = tag.text

        return post_dict
```

```
In [31]: result=mycrawler('로지텍')
print(len(result))
```

```
게이밍 키보드 로지텍 G910 엔터티의 추천 https://blog.naver.com/mhmkh1001?Redirect=Log&logNo=2216632200
로지텍 MX KEYS 무선키보드 : 매력적이고 편리한 팬타그래프... https://blog.naver.com/purplecrom?Redirect=Log&logNo=221678278985
로지텍 G900 마우스 스위치 교체 도전 https://blog.naver.com/soundbross?Redirect=Log&logNo=221731982821
맥북 마우스 추천! 로지텍m350 (디자인중심) https://blog.naver.com/sevenlove_?Redirect=Log&logNo=221709979670
무선 게이밍 키보드 로지텍 G613 블루투스까지 품다 https://neces2.blog.me/221465140806
로지텍 G410 기계식 키보드 수리 - 스위치 불량 https://blog.naver.com/azrama?Redirect=Log&logNo=221599052278
무선 게이밍 마우스 로지텍 G304으로 옮긴 로스타크 http://isaac.pe.kr/221444525789
{'query': '로지텍', 'where': 'post', 'start': 981, 'date_from': 20191231, 'date_to': 20200220}
로지텍G PRO GAMING MOUSE... 증상으로 로지텍마우스수리... http://cardin.co.kr/221531572598
게이밍 스피커 로지텍 G560 후기 http://mnteye.com/221665338654
블루투스 마우스! 로지텍 MX 버티컬 후기 http://blingyue.com/221460503384
게이밍 헤드셋 추천 로지텍 G933S http://gondora.com/221608392732
로지텍 K380 블루투스 키보드 오프라인 파는곳 / 연결방법 https://blog.naver.com/sini0222?Redirect=Log&logNo=221811762513
[IT주변기기구입] 무선마우스 로지텍 M171 (Wireless... https://blog.naver.com/dazzling_jun?Redirect=Log&logNo=221722749598
태블릿 무선 키보드 '로지텍 K380' https://blog.naver.com/xhd1alr45?Redirect=Log&logNo=221687643672
게이밍 키보드 즉각적인 반응을 보여준 로지텍 G512 액타일... https://blog.naver.com/dogs1ife78?Redirect=Log&logNo=221584762334
로지텍 마우스 렉키박스 개편기 / 소소하고 확실한 도박 https://blog.naver.com/pinkbomi?Redirect=Log&logNo=221701602578
로지텍 MX MASTER 더블클릭이 잘 안되는 증상으로 마우스수리... http://cardin.co.kr/221559555784
{'query': '로지텍', 'where': 'post', 'start': 991, 'date_from': 20191231, 'date_to': 20200220}
990
```

위의 글에서 글의 제목과 링크 주소를 가져온 스크래핑의 결과인데, 해당 링크를 타서 본문을 가져올 수 있지만, 해당 기능은 다음인 인스타그램에서 구현해보고자 한다.

## ㄷ. 인스타그램

Vol1. 네이버 블로그가 긴 리뷰, 트위터가 짧은 리뷰라고 하면 인스타그램은 해쉬태그라는 자신만의 키워드를 표현하는 기능이 있지만, 실제로 분석함에 있어서는 해쉬태그를 따로 볼 것이 아니라 전체적인 리뷰로 보고 트위터와 유사한 짧은 리뷰라 판단하여 전체적인 분석을 하고자 한다.

```
def InstagramUrlFromKeyword(browser, keyword, numofpage):
    keyword_url_encode = quote(keyword) # 한글인식

    url = 'https://www.instagram.com/explore/tags/' + keyword_url_encode + '/?hl=ko'

    browser.get(url)

    arr_href = []

    body = browser.find_element_by_tag_name('body')

    for i in range(numofpage):
        body.send_keys(Keys.PAGE_DOWN)

        time.sleep(1)

    time.sleep(3)

    post = browser.find_elements_by_class_name('v1Nh3')

    for j in post:
        href_str = j.find_element_by_css_selector('a').get_attribute('href')

        arr_href.append(href_str) # append 추가시키는거

    return arr_href
```

인스타그램에서 어떤 키워드로 검색어를 하면 해당 키워드인 한글이나 영어는 컴퓨터가 해석하기 난해하므로, 인코딩을 통해 해당 검색어에 맞추어 주소변환이 가능하다. 또한,

```
▼<div class="v1Nh3 kIKUG _bz0w">
  ▼<a href="/p/B1Fh3MvhY1J/"> == $0
    ▼<div class="eLAPa"> https://www.instagram.com/p/B1Fh3MvhY1J/
```

여러 글의 본문을 찾기 위해서 해당 함수를 사용하게 되는데, 인스타그램 웹의 소스코드에는 v1Nh3 class 밑에 href 를 통해 모든 글의 주소를 가져올 수 있어서 해당 글의 ref 를 가져오기 위한 정의이다.

```
def IdHashTagFromInstagram(browser, url):
    browser.get(url)

    insta_id = ""

    hash_data = ""

    wait = WebDriverWait(browser, 20)

    wait.until(EC.presence_of_element_located((By.CLASS_NAME, "e1e1d"))) # e1e1d는 아이디가 적혀있는 소스코드

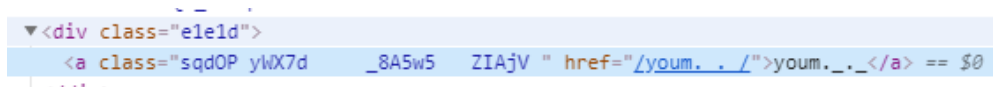
    id_href = browser.find_elements_by_class_name('e1e1d')

    insta_id = id_href[0].find_element_by_css_selector('a').text # id_href[0] 첫번째 있는 a 찾기

    wait.until(EC.presence_of_element_located((By.CLASS_NAME, "C4VMK"))) # 댓글들

    href = browser.find_elements_by_class_name('C4VMK')

    total_hash_text = []
```



아이디를 찾는 부분으로, e1e1d 클래스 영역에 href 로 text 만 뽑아내면 해당 부분이 인스타그램 id 이므로 이를 추출한다.

```
for i in range(0, len(href)): # 댓글 가져와서 하나씩 끝까지 보는 거 len 몇개 개수

    hash_text = href[i].find_element_by_css_selector('span').text

    total_hash_text.append(hash_text)

    image_src = ''

    try:

        image_temp = browser.find_element_by_class_name('KL4Bh').find_element_by_css_selector('img') # 이미지 찾기

        image_list = image_temp.get_attribute('srcset') # srcset이란 속성을 가지고 있는 예를 가져와라

        temp = image_list.split(',') # ,로 구분해서 temp로 가져와라

        for i in temp:

            if '1080w' in i: # 사진의 많은 url 중에서 1080w 있는 문자열 찾기

                image_src = i.split(' ')[0] # url 1080w이 있는 링크에서 1080w를 떼고 공백 앞의 정보를 가져오기

    except:

        image_src = '' # 동영상이면(이미지가 아니면) 빈칸으로 뒤라

    pass

return insta_id, image_src, total_hash_text
```

해당 코드는 for 문을 이용하여 참조할 reference 가 있는 동안 해당 ref 의 본문을 축적하는 부분과 본문에 덧붙인 이미지의 src 를 찾는 부분으로 구성이 되어있다.

```

▼<span class title="수정됨"> == $0
  "저한테 로지텍 핑크 k380이 있지만"
  <br>
  "늘 화이트색상의 키보드를 가지고 싶다는 생각이 마음 저 구석에 있는데, 드디어 저도 화이트 키보드와 마우스를 하나 더 가지게 되었어요!"
  <br>
  <br>
  "제 책상과도 아주 찰떡인 k580과 m350"
  <br>

```

본문의 부분으로 댓글 또한 위와 같이 span 의 영역에 글이 작성되어 있다.

```

▼<div class="KL48h" style="padding-bottom: 100%;">
  
  </div>

```

Img\_src 또한 KL48h 의 영역에 img 를 find 하여 찾을 수 있다.

```

browser = webdriver.Chrome('C:\chromedriver.exe')

keyword = input("검색어를 입력하세요 : ")

num_of_pages = 2

arr = InstagramUrlFromKeyword(browser, keyword, num_of_pages)

insta_df = pd.DataFrame(columns=['Insta ID', 'Image Src', 'Content'])

for url in arr:

    try:

        insta_id, image_src, hash_data = IdHashTagFromInstagram(browser, url)

        char = re.compile('[^0-9a-z- | -가-힣!#?]*') # 재 정비
        """
        정규식을 두 번 이상 사용한다면, 모듈의 match, search 함수는 효율적이지 않다.
        매번 match 혹은 search를 수행할 때마다, 정규식을 분석해서 처리하기 때문이다.
        효과적인 처리 방법은 정규식을 내부 표현식으로 일단 변환하고, 그것을 계속 활용하는 것이다.
        compile 함수가 정규식을 내부 표현식으로 변환하여 정규식 객체를 리턴한다.
        """
        hash_data_str = ""

```

```

for data in hash_data:
    hash_data_str = hash_data_str + data

hash_data_str = char.sub("", hash_data_str) # ""를 hash_data_str으로 바꿔주기

dic_insta = {"Insta ID": insta_id, "Image Src": image_src, "Content": hash_data_str}

temp_df = pd.DataFrame(dic_insta, index=[0]) # index=0은 dic을 temp로 바꾸는데 예러가 나지 않도록 하는 것

insta_df = insta_df.append(temp_df, ignore_index=True)

except:

    print(sys.exc_info()[0])

    pass

nsta_df.to_csv('insta_temp.csv', mode='w', encoding='euc-kr')

```

실제로 동작하는 부분은 검색어를 입력받아서 위의 두 가지 함수를 활용하여 인스타그램의 주소를 가져와서 해당 주소의 해시태그를 데이터 프레임에 쌓고 이를 csv로 저장하는 부분이다.

Insta ID	Image Src	Content
0 youm_	https://scc	저한테로지택핑크380이있지만늘화이트색상의키보드를가지고싶다는생각이마주곡석에있는데디어저도화이트키보드와마우스를하나더가지게되었어요!제작상과도아주잘먹인k580과m350제가키
1 picn2k	https://scc	마우스없이는노트북쓰는일인항상저렴이마우스쓰다가새로장만하마우스!그립감도환상인데다크필드기능으로패드없이도유리대석에서사용가능하니친짜가장편해변잡지깔고쓰는거너무불편했는데
2 kkomtokki	https://scc	라이언오채워본책상#라이언데스크매트#라이언마우스패드#하브그리고요즘쓰고있는#로지텍#무선키보드와#무선마우스디자인이알끔해서예쁜데사용감도좋아서계속쓰는중#kakao#akaofriends#카
3 jiuji	https://scc	2년만에우리오빠이렇게진심답아웃는거정본다ㅋㅋ
4 so_yeOn_0	https://scc	새로산키보드로일기쓰고오늘하루마무리#오늘하루도수고했어#애플#아이패드#아이패드프로3세대#아이패드다꾸#애플펜슬#애플펜슬2세대#애플이#애플이의김#로지텍#로지텍k380#
5 sweetp_nu	https://scc	기계육심많은살아이라마우스와키보드만해도몇개씩갖고있음요즘은새로나온#제닉스#버티컬마우스쓰고있는데기존에스던#로지텍#mxmaster2s보다확실히더가볍다!로지텍쓰다가제닉스들면손간나
6 lotx_	https://scc	#로지텍과#프라이탁ㅋㅋ
7 dndbajwk	https://scc	#나를위한스피우로컴퓨터로일하는나를위한손목보마우스새로구매!#버티컬마우스#관절염#거갈아서로지택옆로추가구매!손목보호패드도귀여운형구로했키보드는사는김에#지름신강림함으로도열심하
8 mazect	https://scc	요즘내가사진작업할때주로사용하는키보드#광공한디자인에부드러운키감이좋다-중그릴게생긴너석은크라운이라고하는데저다이얼을돌려자주사용하는기능들을간편하게사용할수있다!생생한사용장
9 habi_snap	https://scc	아이패드미니를위한키보드두종!정실먹고옆에이패드매트를삼가서냉큼사왔다국민키보드크크크
10 best_t_soc	https://scc	아침제일이어나서고시문쓰고유제국갔다가이제서야제대로#작성일단내해설서문제부터!#작성인증#스티디를레너#플레너#모트모트를레너#study#공스타그램#공부스타그램#공부인증#유아인
11 davidthegi	https://scc	와내가이마우스들은지5년만에처음으로빠터리없다고불들어옴빠터리도정실때들어있던거게속은건대여곡시#로지텍#510빠터리라이프죽이네이번에빠터리갈까면도아날5년갈까나??ㅎㅎ
12 aanong	https://scc	새세팅완료!새로운작업실새로운사람들새로운환경과새로운일상안수어린이집입학-7#브라보택디파이b40#라이젠2700x#로지텍#lg4k
13 best_t_soc	https://scc	계획이들어지는바람에울을못했다오늘갑자기동기부여가막막내일더열심히달려보자#내일은해설서문제부터!#작성인증#스티디를레너#플레너#모트모트를레너#study#공스타그램#공부스타그램#
14 koo_9_che	https://scc	컴퓨터도착이제피시방안가요#게이밍컴퓨터#로지텍#키보드#마우스#헤드셋#오버워치#맥들그라운드#ram16g#geforce10606gb#ssd240g#56600#영상속노래는#여자친구#교차로형키보드는다시사
15 flowerin-	https://scc	바지락술집면거울시즌한정이레오진짜바지락이든라면이러니!#gs25원의점#gs25추천#바지락술집면#진짜바지락#신기#신기방기#겨울#시즌한정#리미티드#에디션#상암라면#라면의원조#라면꿀만왕!
16 jstagram1	https://scc	#꿀자료가수북하다고좋은거아니지?알맹이가알맹이중요하지!!!저도처음엔자료가수두름그게나만알기싫겠다는생각이들면서매번자료정리하고필수요만핵심만주려서이해하기힘겨운자료만드는중-
17 no.1_assac	https://scc	http://www.wassacom.com주문정보20021456141김형고객님제품가격1787110원제작담당홍석오동일사양바로구매하기http://www.wassacom.comshopproductautomakesystemmn2abog33제품정보#인텔코0
18 no.1_assac	https://scc	http://www.wassacom.com주문정보20021635691이병고객님제품가격981700원제작담당김정훈동일사양바로구매하기http://www.wassacom.comshopproductautomakesystemmymg33제품정보#라이젠53
19 computerj	https://scc	매장방문구매여주신최석고객님로지텍#ireless게이밍마우스#htcpanavercombusancomputerplaza부산컴퓨터프라자는게이밍기어제점형매장입니다#로지텍#무선마우스#지프로무선#마우스#부산!
20 computerj	https://scc	매장방문구매여주신유익고객님로지텍#ireless게이밍마우스#htcpanavercombusancomputerplaza부산컴퓨터프라자는게이밍기어제점형매장입니다#로지텍#무선마우스#지프로무선#마우스#부산!

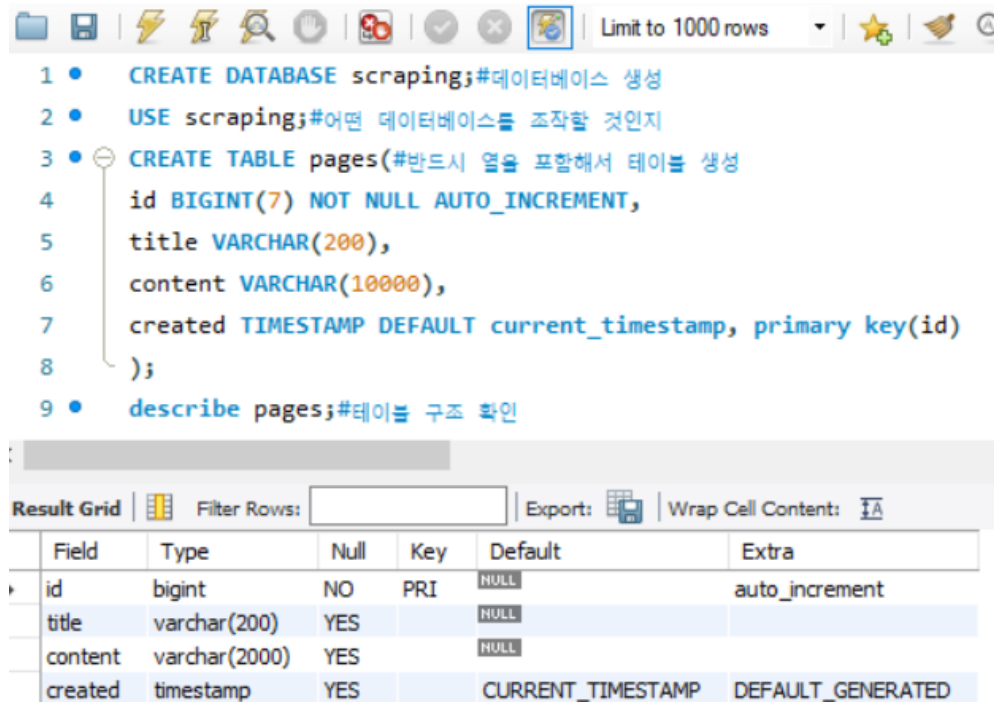
라는 결과를 얻을 수 있다. 여기서, 활용한 것은 저번 2주 차 동안의 목표였던 링크를 타서 본문을 가져오는 것과, 검색어를 입력하여 해당 검색어에 맞는 리뷰를 가져오는 것을 사용해보았으며, 추가적으로 csv 파일로 저장하는 것까지 구현해보았다.

해당 주차를 크롤링하면서 생각한 부분은 페이스북은 리뷰를 분석하기에는 자료가 너무 적어서 의미가 없다는 점과 크롤링된 결과에서 필요한 키워드를 뽑아내는 것을 연구해야 할 것 같다. 따라서 자연어 처리를 실습하는 것을 계획하고 있으며, 어떠한 DB를 사용할지까지 고민해보고자 한다. 또한, 최근 1년 사이의 리뷰를 확인 기간을 정하고자 하는데 이는 리뷰얼된 제품 즉, 최신의 정보도 업데이트가 가능할 것이라고 판단하여 해당 기간으로 리뷰를 분석하고자 한다. 해당 기간 설정이 필요한 이유는 리뷰얼된 제품의 경우, 과거의 제품과는 다른 제품력을 보여주기 때문이다.



### 3. DB 구축

#### ㄱ. MySQL 환경설정 및 DDL 명령어 사용, pyMySQL을 이용한 DB연동



The screenshot shows a MySQL IDE interface. The top toolbar includes icons for file operations, execution, and search. Below the toolbar, a list of SQL commands is displayed with line numbers 1 through 9. The commands are: 1. CREATE DATABASE scraping; #데이터베이스 생성, 2. USE scraping; #어떤 데이터베이스를 조작할 것인지, 3. CREATE TABLE pages(#반드시 열을 포함해서 테이블 생성), 4. id BIGINT(7) NOT NULL AUTO\_INCREMENT, 5. title VARCHAR(200), 6. content VARCHAR(10000), 7. created TIMESTAMP DEFAULT current\_timestamp, primary key(id), 8. );, 9. describe pages; #테이블 구조 확인. Below the commands, a 'Result Grid' shows the table structure for 'pages'. The grid has columns: Field, Type, Null, Key, Default, and Extra. The rows are: id (bigint, NO, PRI, NULL, auto\_increment), title (varchar(200), YES, NULL), content (varchar(20000), YES, NULL), and created (timestamp, YES, CURRENT\_TIMESTAMP, DEFAULT\_GENERATED).

```
1 • CREATE DATABASE scraping;#데이터베이스 생성
2 • USE scraping;#어떤 데이터베이스를 조작할 것인지
3 • CREATE TABLE pages(#반드시 열을 포함해서 테이블 생성
4   id BIGINT(7) NOT NULL AUTO_INCREMENT,
5   title VARCHAR(200),
6   content VARCHAR(10000),
7   created TIMESTAMP DEFAULT current_timestamp, primary key(id)
8 );
9 • describe pages;#테이블 구조 확인
```

Field	Type	Null	Key	Default	Extra
id	bigint	NO	PRI	NULL	auto_increment
title	varchar(200)	YES		NULL	
content	varchar(20000)	YES		NULL	
created	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

여러 개의 테이블이 아닌 한 개의 테이블을 이용하여, 제목과 내용을 저장해보기 위한 테이블을 구성하였습니다.

```
22 #####문자셋을 utf8mb4에서 unicode_ci로 바꿔주는 코드#####
23 • alter database scraping character set=utf8mb4 collate=utf8mb4_unicode_ci;
24 • use scraping;
25
26 • alter table pages convert to character set utf8mb4 collate utf8mb4_unicode_ci;
27 • alter table pages change title title varchar(200) character set utf8mb4 collate utf8mb4_unicode_ci;
28 • alter table pages change content content varchar(2000) character set utf8mb4 collate utf8mb4_unicode_ci;
29 #####
30 • alter table pages modify column content varchar(2000);
```

이후, 데이터를 저장하는 과정에서 하나의 열에 내용을 포함시키지 못하는 경우가 많아 alter 기능을 이용하여 데이터 열의 정보를 수정하였습니다.

```

In [1]: pip install PyMySQL

Requirement already satisfied: PyMySQL in c:\users\anaruk\anaconda3\lib\site-packages (0.9.3)
Note: you may need to restart the kernel to use updated packages.

In [3]: import pymysql
        conn=pymysql.connect(host='127.0.0.1',user='root',passwd='3721',db='mysql')

In [6]: cur=conn.cursor()
        cur.execute("USE scraping")
        cur.execute("SELECT * from pages WHERE id=2")

Out[6]: 1

In [7]: print(cur.fetchone()) #마지막에 실행한 쿼리 결과 출력

(2, 'A new title', 'Some new content', datetime.datetime(2020, 4, 2, 7, 28, 17))

In [8]: cur.close()
        conn.close()

```

pyMySQL 을 이용하여 cur 과 conn 이라는 변수를 활용하여 execute 문을 통한 MySQL 명령어를 실행해 보았습니다.

## L. 크롤링된 결과를 DML 명령어를 사용하여 저장

```

insert into pages(title,content)#id는 자동증가, timestamp는 현재시간 자동 저장
values(
    "Test page title",
    "This is some test page content. It can be up to 10,000 characters long."
);
select * from pages where id=2;#id가 2인 행이없으므로 none return
select * from pages where title like "%test%";
select id,title from pages where content like "%page content%";
#delete를 실행하기전에는 select를 먼저 실행하는 것이 좋다.;
select * from pages where id=1;
delete from pages where id=1;
update pages set title="A new title", content="Some new content" where id=2;

```

크롤링한 결과를 DB 에 저장해보기 앞서서 DML 명령어를 기본적으로 활용해보았습니다.

```

In [5]: from urllib.request import urlopen
        from bs4 import BeautifulSoup
        import datetime
        import random
        import pymysql
        import re

        conn = pymysql.connect(host='127.0.0.1', user='root', passwd='3721', db='mysql', charset='utf8')
        cur = conn.cursor()
        cur.execute('USE scraping')

        random.seed(datetime.datetime.now())

        def store(title, content):
            cur.execute('INSERT INTO pages (title, content) VALUES ("%s", "%s")', (title, content))
            cur.connection.commit()

        def getLinks(articleUrl):
            html = urlopen('http://en.wikipedia.org'+articleUrl)
            bs = BeautifulSoup(html, 'html.parser')
            title = bs.find('h1').get_text()
            content = bs.find('div', {'id': 'mw-content-text'}).find('p').get_text()
            store(title, content)
            return bs.find('div', {'id': 'bodyContent'}).findAll('a', href=re.compile('^(/wiki/)((?!:).)*$'))

        links = getLinks('/wiki/KevinBacon')
        try:
            while len(links) > 0:
                newArticle = links[random.randint(0, len(links)-1)].attrs['href']
                print(newArticle)
                links = getLinks(newArticle)
        finally:
            cur.close()
            conn.close()

/wiki/List_of_gothic_festivals
/wiki/Moldova
/wiki/Taracia_District
/wiki/Romani_people
/wiki/Turkey
/wiki/Karachays
/wiki/Soyot
/wiki/Chechens
/wiki/Berbers_in_Belgium
/wiki/Aghul_people
/wiki/Archil_people
/wiki/Rutul_people
/wiki/Poles_in_Azerbaijan
/wiki/Azerbaijan
/wiki/United_Nations_Development_Program
/wiki/UNICEF
/wiki/Chapter_XIV_of_the_United_Nations_Charter

```

위와 동일하게, conn 과 cur 변수를 활용하여 pyMySQL 의 데이터에 접근합니다. 기본적으로 위키백과의 데이터를 하나 참조하되, 데이터 내부의 href 를 모두 찾아서 ref 의 제목과 내용을 모두 가져와서 DB 에 저장하는 코드를 작성해 보았습니다. 이는, 추후에 네이버 블로그나 티스토리등에서 작품을 검색하고, 그 내용을 수집하는 과정과 매우 유사할 것이라 판단합니다. 특히, 네이버 블로그 위키백과와 동일하게 request 를 활용합니다.

이때, DML 명령어를 사용하기 위해서 execute 문을 사용하여 insert 문을 실행하고 있음을 확인할 수 있습니다.

```

31 • select * from pages where id=1800;

```

id	title	content	created
1800	'Checkbox'	'A checkbox (check box, tickbox, tick box) is a GUI widget that permit...	2020-04-03 02:50:32

Edit Data for content (VARCHAR)

Binary Text

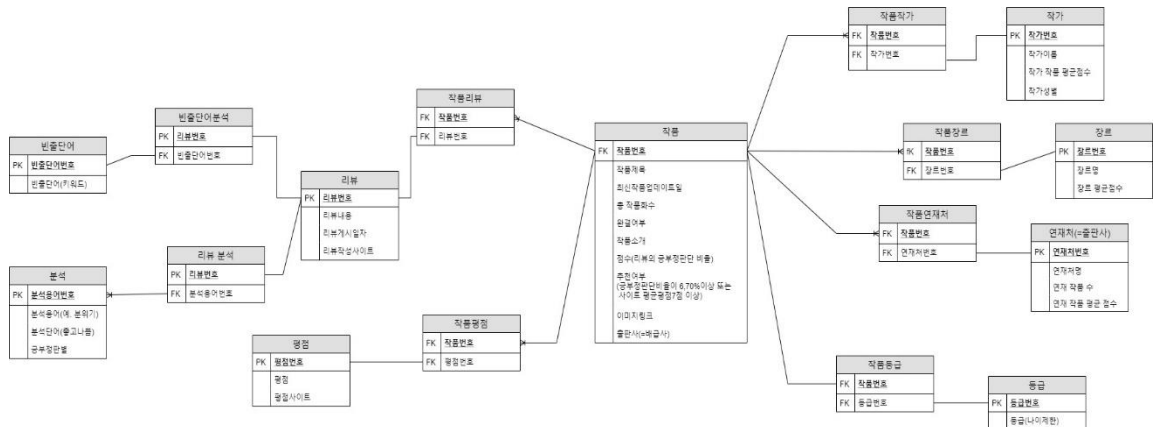
```

1 'A checkbox (check box, tickbox, tick box) is a GUI widget that permits the user to make a binary choice, i.e. a
  choice between one of two possible mutually exclusive options. For example, the user may have to answer 'yes'
  (checked) or 'no' (not checked) on a simple yes/no question.
2 '

```

컴퓨터의 속도에 비해 데이터의 길이가 크다고는 생각을 안했는데, 생각보다 긴 시간이 소요되는 것을 확인하였습니다. 웹페이지당 1초정도 소요되긴 하지만 추후에는 더 많은 데이터를 저장할 것이라 판단하므로 빠른 작업이 필요할 것 같습니다.

## ㄷ. DB Logical-ERD 구성



기본적인 작품 정보로 작품 이미지, 제목, 작가, 출판사, 연재사, 완결 여부, 현재 총 작품 회수, 최신화 업데이트일, 작품 소개, 장르가 들어가게 됩니다. 이 중 작가, 장르와 연재처는 단일 테이블로 만들어 간단한 작가 정보와 연재처 정보를 가지게됩니다. 이 테이블에는 장르, 작가와 연재처의 다른 작품을 포함한 통합적 평균 점수가 개재되게 됩니다. 가장 중요한 리뷰의 경우 해당 작품에서 화제가 되며 많은 언급이 일어나고 있는 키워드와 리뷰의 긍정/부정 판단을 위해 두가지 테이블을 만들었습니다. 또한 각 리뷰 사이트마다 가지고 있는 평점에 대한 평균을 측정하여 종합적인 리뷰점수를 판단하기 위해 평점에 관련된 테이블도 만들었습니다.

\*이는 추후 제작과정에서 변경될 수 있으며 계속 진행되는 회의를 통해 상세 내역을 수정해 나가고 있습니다.

## 4. 단어 사용 빈도 추출

### ㄱ. 전처리 과정(불용어 제거, 어근 동일화, N-gram)

```
In [3]: import nltk
nltk.download('stopwords')

[nltk_data] Downloading package stopwords to
[nltk_data] C:\Users\naru\AppData\Local\nltk_data...
[nltk_data] Unzipping corpora\stopwords.zip.

Out[3]: True

In [5]: words_Korean=["추석","연휴","민족","대이동","시작","늦어","교통량","교통사고","특히","자동차"]
stopwords=["가다","늦어","나타","것","특히"]
[if for i in words_Korean if i not in stopwords]

Out[5]: ['추석', '연휴', '민족', '대이동', '시작', '교통량', '교통사고', '자동차']

In [7]: from nltk.corpus import stopwords
words_English=["apple","banana","chief","roberts","','president','you','']
print([w for w in words_English if not w in stopwords.words('english')])

['apple', 'banana', 'chief', 'roberts', ',', 'president', '']
```

Nltk api 의 stopwords 를 이용하여 불용어를 제거한 결과입니다. 이 외에도 re.compile 과 정규식 의 조합을 이용하여 특수문자나 특정 조합(메일)등을 지우는 처리를 해보았습니다.

```
In [1]: #####
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

In [4]: import nltk

In [7]: nltk.download('punkt')

[nltk_data] Downloading package punkt to
[nltk_data] C:\Users\maruk\AppData\Roaming\nltk_data...
[nltk_data] Unzipping tokenizers\punkt.zip.

Out[7]: True

In [9]: ps_stemmer=PorterStemmer()
new_text="It is important to be immersed while you are pythoning with python. All pythoners have pythoned poorly at least once."
words=word_tokenize(new_text)
for w in words:
    print(ps_stemmer.stem(w),end=' ')

It is import to be immers while you are python with python . all python have python poorli at least onc .

In [10]: from nltk.stem.lancaster import LancasterStemmer
LS_stemmer=LancasterStemmer()

In [11]: for w in words:
    print(LS_stemmer.stem(w),end=' ')

it is import to be immers whil you ar python with python . al python hav python poor at least ont .

In [15]: from nltk.stem.regex import RegexpStemmer
RS_stemmer=RegexpStemmer('python')#python이라는 불자 제거
for w in words:
    print(RS_stemmer.stem(w),end=' ')

It is important to be immersed while you are ing with . All ers have ed poorly at least once .
```

해당 실행문은 영어의 경우 시제에 따라 같은 동사임에도 형태가 달라지는 경우가 있는데, 이때 달라지게 하는 원인을 제거하는 것을 실행해 보았습니다. 한국어에도 이러한 형태가 있는데 형태소 분석뒤에 가능할 것이라고 판단합니다.

```
In [2]: ### ## K-gram
n-gram이란 n번 연이어 등장하는 단어들의 연쇄를 의미한다.
ex) Republic of korea 트라이그램, 보통 바이(2)그램까지만 사용한다.

from nltk import ngrams
sentence = "Chief Justice Roberts, President Clinton, President Obama, President Carter, the citizen of Americans and people of the world."
grams=ngrams(sentence.split(),2)
for gram in grams:
    print(gram,end="- ")
#Chief, President와 같이 두개의 단어가 연이어 생성되는 단어를 바이그램을 통해 추출할 수 있다.
```

```
('Chief,' ('Justice' ('Justice', ('Roberts,' ('Roberts,', ('President' ('President', ('Clinton,' ('Clinton.', ('President' ('Preside', ('Obama,' ('Obama.', ('President' ('President', ('Carter,' ('Carter.', ('the' ('the', ('citizen' ('citizen', ('of' ('of', ('Americans' ('Americans', ('and' ('and', ('people' ('people', ('of' ('of', ('of', ('the' ('the', ('world.' ('world.', ('thank' ('thank', ('you.'
```

n-gram 을 이용하여 '대통령 트럼프'와 같이 한 단어로 취급해야할 필요가 있는 단어를 표기해주는 전처리 과정을 실행해 보았습니다.

## └. KoNLPy 한국어 분석 中 Hannanum

```
In [24]: import pandas as pd
import nltk
from nltk.stem.porter import PorterStemmer
from nltk.tokenize import RegexpTokenizer
from nltk.corpus import stopwords
```

```
In [30]: from konlpy.tag import Hannanum
hannanum=Hannanum()
```

```
In [31]: temp = []
for i in range(len(lines)):
    temp.append(hannanum.nouns(lines[i]))#명사만추출
```

```
In [32]: # 중첩 리스트(개념을 알 것) 하나의 리스트로 변환하는 함수
def flatten(l):
    flatList = []
    for elem in l:
        if type(elem) == list:
            for e in elem:
                flatList.append(e)
        else:
            flatList.append(elem)
    return flatList

word_list=flatten(temp)
# 두글자 이상인 단어만 추출
word_list=pd.Series([x for x in word_list if len(x)>1])
word_list.value_counts().head(10)
```

```
Out [32]: 대통령    29
국민            19
대한민국        9
우리             8
여러분           7
역사             6
국민들           6
나라             6
대통령의         5
세상             5
dtype: int64
```

대통령 담화문 중, 명사만을 추출하여 빈도를 추출하고자 한 실행문입니다. 담화문의 경우 문단별 구성에 따라서 리스트가 중첩이 되므로 이를 하나로 만들어줄 필요가 있습니다. 따라서 def 문을 통해 함수를 정의하여 실행하였습니다.

```
from wordcloud import WordCloud
from collections import Counter
```

```
font_path = 'C:\Users\wnaruk\Desktop\잡아라! 텍스트마이닝\NanumBarunGothic.ttf'
```

```
wordcloud = WordCloud(
    font_path = font_path,
    width = 800,
    height = 800,
    background_color="white"
)
```

```
count = Counter(stopped_tokens2)
wordcloud = wordcloud.generate_from_frequencies(count)
```

```
def __array__(self):
    """Convert to numpy array.
    Returns
    =====
    image : nd-array size (width, height, 3)
           Word cloud image as numpy matrix.
    """
    return self.to_array()

def to_array(self):
    """Convert to numpy array.
    Returns
    =====
    image : nd-array size (width, height, 3)
           Word cloud image as numpy matrix.
    """
    return np.array(self.to_image())

array = wordcloud.to_array()
```

```
count = Counter(word_list)
wordcloud = wordcloud.generate_from_frequencies(count)
array = wordcloud.to_array()
```

```
get_ipython().run_line_magic('matplotlib', 'inline')
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(10, 10))
plt.imshow(array, interpolation="bilinear")
plt.show()
fig.savefig('wordcloud.png')
```



자연어처리를 한 결과만을 이용하여 UI 를 구성하고자 했는데, 시각적인 처리가 있다면 UI 가 좀 더 다양화할 수 있지 않을까 하여 수행해보았습니다. 이 외에도 matplotlib 를 이용하여 그래프 등의 시각적인 자료를 제작해보았는데, 이러한 시각적인 자료가 트렌드를 파악할 때 사용될 수 있으리라 판단합니다.

## ㄴ. 감성분석(감성사전이용)

```
In [1]: #사전기반감성분석
import pandas as pd
import glob
from afinn import Affinn
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.tokenize import RegexpTokenizer
import numpy as np
import matplotlib.pyplot as plt

In [2]: pos_review=(glob.glob("C:\Users\***\Desktop\***\아라! 텍스트 마이닝 데이터\***\train\***pos***.txt"))[20]
#1008 리뷰의 긍정(pos) 데이터셋 중 20번째 데이터의 경로를 받아옴
f = open(pos_review, 'r')
lines1 = f.readlines()[0]
#해당 문자열을 받아옴
f.close()

In [3]: affinn = Affinn()
affinn.score(lines1)#감성점수 산출(긍정이나까 하이스코어)

Out [3]: 8.0

In [4]: #부정
neg_review=(glob.glob("C:\Users\***\Desktop\***\아라! 텍스트 마이닝 데이터\***\train\***neg***.txt"))[20]
f = open(neg_review, 'r')
lines2 = f.readlines()[0]
f.close()
affinn.score(lines2)

Out [4]: -4.0
```

영화사이트 IDMB 의 리뷰자료를 이용하였고, 2500개의 data set 을 가지고 있는 Affinn 을 이용하여 불러온 리뷰의 긍부정 척도를 계산하는 것을 실행해 보았습니다. 저희는 한국어 감성사전을 이용하여 이러한 척도를 계산해보려고 했었는데, 한국어에서 영어로 변환하여 해당 사전을 활용해도 괜찮겠다라는 생각을 해보았습니다.

```
In [7]: #EmoLex
NRC=pd.read_csv('C:\Users\***\Desktop\***\아라! 텍스트 마이닝 데이터\***\nrc.txt', engine="python", header=None, sep="\t")
#감성사전 오픈

NRC=NRC[(NRC != 0).all(1)]
NRC=NRC.reset_index(drop=True)
#감성어와 감성표현이 유의미한 라벨만 추출

tokenizer = RegexpTokenizer('[\w]+')
stop_words = stopwords.words('english')
#불용어 처리

p_stemmer = PorterStemmer()

raw = lines1.lower()
tokens = tokenizer.tokenize(raw)
stopped_tokens = [i for i in tokens if not i in stop_words]

#긍정 텍스트 전처리

match_words = [x for x in stopped_tokens if x in list(NRC[0])]
emotion=[]
for i in match_words:
    temp=list(NRC.iloc[np.where(NRC[0] == i)[0],1])
    for j in temp:
        emotion.append(j)
#감성사전과 텍스트의 감성어들의 맵핑

sentiment_result1=pd.Series(emotion).value_counts()
#감성표현 출력횟수

sentiment_result1

Out [7]: positive      8
trust              7
negative          5
joy               4
anticipation      4
sadness           3
fear              3
anger             2
surprise          2
```



자연어처리를 하기전에 불용어 처리를 하여 단어별 감정을 매핑하여 감정이 몇번 매핑되었나를 확인하는 실행문입니다. 실습자료로서 구성해보았지만 프로젝트에 따로 활용할 것 같진 않습니다.

## ㄷ. 감성분석(지도 기계학습기반 감성 분석)

```
In [2]: import pandas as pd
import glob
from afinn import Affin
from nltk.corpus import stopwords
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer

In [3]: pos_review=(glob.glob("C:\Users\naruk\Desktop\잡아라! 텍스트 마이닝 데이터\imdb\train\pos*.txt"))
# 긍정, 부정 텍스트 읽어오기
lines_pos=[]
for i in pos_review:
    try:
        f = open(i, 'r')
        temp = f.readlines()[0]
        lines_pos.append(temp)
        f.close()
    except Exception as e:
        continue

len(lines_pos)

Out[3]: 12490

In [4]: neg_review=(glob.glob("C:\Users\naruk\Desktop\잡아라! 텍스트 마이닝 데이터\imdb\train\neg*.txt"))
lines_neg=[]
for i in neg_review:
    try:
        f = open(i, 'r')
        temp = f.readlines()[0]
        lines_neg.append(temp)
        f.close()
    except Exception as e:
        continue

len(lines_neg)

Out[4]: 12489

In [5]: total_text=lines_pos+lines_neg
len(total_text)

Out[5]: 24979
```

위의 IDMB 데이터셋을 불러오는 과정입니다.

```

In [7]: x = np.array(["pos", "neg"])
class_index=np.repeat(x, [len(lines_pos), len(lines_neg)], axis=0)
#금. 부정 클래스 라벨링
stop_words = stopwords.words('english')

vect = TfidfVectorizer(stop_words=stop_words).fit(total_text)
X_train_vectorized = vect.transform(total_text)
#TF-IDF가중치를 준 후에 문서-단어 매트릭스로 바꾸어줌

In [8]: from sklearn.linear_model import LogisticRegression,SGDClassifier
model = LogisticRegression()
model.fit(X_train_vectorized, class_index)
#로지스틱 회귀모형을 세움

C:\Users\wnaruk\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be changed to
'lbfgs' in 0.22. Specify a solver to silence this warning.
FutureWarning)

Out[8]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, l1_ratio=None, max_iter=100,
multi_class='warn', n_jobs=None, penalty='l2',
random_state=None, solver='warn', tol=0.0001, verbose=0,
warm_start=False)

In [17]: pos_review_test=(glob.glob("C:\Users\wnaruk\Desktop\잡아라! 텍스트 마이닝 데이터\aclImdb\test\pos\*.txt"))[10]

test=[]
f = open(pos_review_test, 'r')
test.append(f.readlines()[0])
f.close()

predictions = model.predict(vect.transform(test))
predictions

Out[17]: array(['pos'], dtype='<U3')

In [18]: neg_review_test=(glob.glob("C:\Users\wnaruk\Desktop\잡아라! 텍스트 마이닝 데이터\aclImdb\test\neg\*.txt"))[20]

test2=[]
f = open(neg_review_test, 'r')
test2.append(f.readlines()[0])
f.close()

predictions = model.predict(vect.transform(test2))
predictions

Out[18]: array(['neg'], dtype='<U3')

```

여러 모델 중 로지스틱 회귀분석 모델을 사용한 결과로 불러온 리뷰의 감정을 나타낸 결과입니다.

위와 동일하게 서포트벡터머신, 의사결정나무 모형 등 지도 기계학습기반의 여러 모델을 동일한 데이터에 대해 수행해보았습니다.

```

In [19]: #의사결정나무모형으로 위와 동일함 실행
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
clf.fit(X_train_vectorized, class_index)
predictions = clf.predict(vect.transform(test))
predictions

Out[19]: array(['neg'], dtype='<U3')

In [20]: predictions = clf.predict(vect.transform(test2))
predictions

Out[20]: array(['neg'], dtype='<U3')

In [15]: #서포트벡터머신-감나오래결함
from sklearn.svm import SVC
clf = SVC() #SVC(gamma='scale') 이런식으로 변경가능
clf.fit(X_train_vectorized, class_index)
predictions = clf.predict(vect.transform(test))
predictions

C:\Users\wnaruk\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The default value of gamma will change from 'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
FutureWarning)

Out[15]: array(['pos'], dtype='<U3')

In [16]: predictions = clf.predict(vect.transform(test2))
predictions

Out[16]: array(['pos'], dtype='<U3')

```

그 결과로 로지스틱 회귀분석의 경우 원하는 대로 긍정, 부정의 결과를 가져왔고 수행속도도 해당 데이터에 한해서는 다른 두 모델보다 속도도 빨랐습니다. 하지만, 나머지 두개의 모델은

수행속도도 굉장히 느리고 원하는 결과를 도출해내지도 않았습니다. 이를 통해서 모델별로 적용해서 실제 데이터의 예측률을 파악하는 게 중요할 것이라고 생각하였으며, 저희가 수행하고자할 프로젝트에서 사용할 수 있는 api 나 모델들이 여러가지 있으므로 이를 적용하여 최선의 값을 도출해낼 필요가 있다고 판단하여 추후에 작업해보고자 합니다.

## ㄹ. Word2vec을 이용한 단어 임베딩 中 단어 유사도 판단

추후에 Aspect Analysis 를 할 때 단어 유사도 판단의 과정이 포함될 것이라 생각하여 Word2vec 을 이용하여 단어 임베딩을 실시해보았습니다. 그 중에서 이번에는 단어 유사도를 판단하여, 특정 단어와 유사한 단어가 무엇이 있는지 확인해보았으며, 좌표로서 나타낼 수 있음을 확인하였습니다. 이 때, 네이버 영화 말뭉치 training 이 완료된 Set 를 이용하여 Word2vec 과정을 수행하였으며, Konlpy 를 이용하여 한글에 대한 분석을 실시해 보았습니다. 특히, konlpy 사용중에 norm 이나 stem 을 이용하여 오타나 형태소의 원형을 이용하였습니다.

```
In [5]: import codecs
        #konlpy 0.5.0 버전 이후부터 이름이 Twitter에서 Okt로 바뀌었다.
        from konlpy.tag import Okt
        from gensim.models import word2vec
        from konlpy.utils import pprint

In [6]: def read_data(filename):
        with codecs.open(filename, encoding='utf-8', mode='r') as f:
            data = [line.split('ㄷ') for line in f.read().splitlines()]
            data = data[1:] # header 제외
        return data

In [7]: #파일 위치, 본인의 파일경로로 변경필요
        ratings_train = read_data('ratings_train.txt')
        #konlpy 중에서 트위터 형태소분석기 사용 (1)
        tw_tagger = Okt()

In [8]: # 토큰나이즈(의미단어검출) 함수, 트위터 형태소 분석기 사용 (2)
        # 형태소 / 품사 형태로 리스트화
        def tokens(doc):
            return ['/'.join(t) for t in tw_tagger.pos(doc, norm=True, stem=True)]
        #norm 기능을 이용해 오타를 정정(ex. 사랑해를 사랑해로), stem을 이용해 원형으로 반환(ex. 입니다를 이다로)

In [9]: # 파일중에서 영화 리뷰 데이터만 달기
        docs = []
        for row in ratings_train:
            docs.append(row[1])

        data = [tokens(d) for d in docs]

In [11]: # [TRAIN] word2vec 으로 모델 생성 (3)
        w2v_model = word2vec.Word2Vec(data)

        # init_sims 명령어로 필요없는 메모리 반환
        w2v_model.init_sims(replace=True)

        # [TEST] 가장 유사한 단어 출력 (4)
        pprint(w2v_model.wv.most_similar(positive=tokens(u'남자 여배우'),
            negative=tokens(u'배우'), topn=1))

[('여자/Noun', 0.8150866031646729)]
```

```

In [14]: pprint(w2v_model.wv.most_similar(positive=tokens(u'주인공'), topn=10))
[('기선/Noun', 0.7544881105422974),
 ('얼굴/Noun', 0.6906263828277588),
 ('행동/Noun', 0.6644679307937622),
 ('남자/Noun', 0.6600486636161804),
 ('여자/Noun', 0.6532753705978394),
 ('악역/Noun', 0.6524346470632825),
 ('등장인물/Noun', 0.6311010122299194),
 ('성격/Noun', 0.630095899105072),
 ('캐릭터/Noun', 0.6252544522285461),
 ('악당/Noun', 0.6231570839881897)]

In [16]: from sklearn.manifold import TSNE
from matplotlib import font_manager as fm
from matplotlib import rc
movie_tsne = TSNE(n_components=2)
movie_tsne

Out[16]: TSNE(angle=0.5, early_exaggeration=12.0, init='random', learning_rate=200.0,
method='barnes_hut', metric='euclidean', min_grad_norm=1e-07,
n_components=2, n_iter=1000, n_iter_without_progress=300, perplexity=30.0,
random_state=None, verbose=0)

In [18]: movie_vocab = w2v_model.wv.vocab
movie_similarity = w2v_model[movie_vocab]
movie_similarity

C:\Users\maruk\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: DeprecationWarning: Call to deprecated `__getitem__` (Method w
ill be removed in 4.0.0, use self.wv.__getitem__() instead).

Out[18]: array([[ 0.01583514, -0.01967263,  0.01102952, ..., -0.06493731,
-0.11728221, -0.10047553],
[ 0.02801962, -0.07917922, -0.02239081, ..., -0.11629011,
-0.03628932,  0.07144445],
[ 0.2009029 , -0.05290857,  0.02421443, ..., -0.1569652 ,
 0.07382585, -0.17336367],
...,
[ 0.10578212,  0.0587996 , -0.027095 , ...,  0.02845743,
-0.18845902, -0.03497209],
[ 0.00156862,  0.11541089, -0.0601042 , ...,  0.02754588,
-0.17417371,  0.06905237],
[ 0.00297765, -0.00140684,  0.05425932, ..., -0.0171057 ,
-0.16980827,  0.06580625]], dtype=float32)

```

좌표를 나타내주는 코드까지 수행하였습니다. 단어의 유사도 이므로 특정 단어와 유사한 단어를 보여주는 것인데, Aspect Analysis 의 경우 문법적인 연관성을 판단해야 하므로 임베딩에서 다른 과정을 추가적으로 요구함을 확인하였습니다.

```

In [20]: import pandas as pd
movie_transform_similarity = movie_tsne.fit_transform(movie_similarity)
movie_df = pd.DataFrame(movie_transform_similarity, index=movie_vocab, columns=['x', 'y'])
movie_df[0:10]

```

Out [20]:

	x	y
아/Exclamation	22.172894	-57.955566
더빙/Noun	6.981616	45.611614
../Punctuation	18.876213	-60.003292
친자/Noun	17.515112	-63.976002
짜증나다/Adjective	36.130859	53.686295
목소리/Noun	-56.858418	-29.822529
홀/Noun	23.910069	-57.069225
.../Punctuation	18.831528	-59.998028
포스터/Noun	46.868126	26.451733
보고/Noun	50.604397	-23.869593

작업하고자 하는 타겟이 리뷰이다보니, 'ㅋㅋㅋ'나 '꿀잼' 과 같은 은어들에 대해서 처리하는 과정이 필요하리라 판단하였는데 Okt 의 경우 이러한 은어들도 한국어조사 등으로 처리가 되고 있음을 따로 확인하였습니다. 그 외의 분석기에 대해서도 확인이 필요할 듯 싶습니다.

분석기별로 수행 속도가 다를 것을 확인하였고, 분석을 하여 보여주는 결과의 종류도 각기 다르기 때문입니다.

#### A. 트위터에서 마이닝한 트윗 정보로 유사도 판단하기.

```
In [1]: #https://jeongwookie.github.io/2019/06/10/190610-twitter-data-crawling/
# GetOldTweet3 사용 준비
try:
    import GetOldTweets3 as got
except:
    !pip install GetOldTweets3
    import GetOldTweets3 as got

In [2]: # datetime을 사용해 가져올 범위를 정의
# 예제 : 2020-01-01 ~ 2020-04-01

import datetime

days_range = []

start = datetime.datetime.strptime("2020-01-01", "%Y-%m-%d")
end = datetime.datetime.strptime("2020-04-01", "%Y-%m-%d")
date_generated = [start + datetime.timedelta(days=x) for x in range(0, (end-start).days)]

for date in date_generated:
    days_range.append(date.strftime("%Y-%m-%d"))

print("=== 설정된 트윗 수집 기간은 {} 에서 {} 까지 입니다 ===".format(days_range[0], days_range[-1]))
print("=== 총 {}일 간의 데이터 수집 ===".format(len(days_range)))

=== 설정된 트윗 수집 기간은 2020-01-01 에서 2020-03-31 까지 입니다 ===
=== 총 91일 간의 데이터 수집 ===

In [4]: # 특정 검색어가 포함된 트윗 검색하기 (query search)
# 검색어 : 기묘한이야기

import time

# 수집 기간 맞추기
start_date = days_range[0]
end_date = (datetime.datetime.strptime(days_range[-1], "%Y-%m-%d")
            + datetime.timedelta(days=1)).strftime("%Y-%m-%d") # setUntil()이 끝을 포함하지 않으므로, day + 1

# 트윗 수집 기준 정의
tweetCriteria = got.manager.TweetCriteria().setQuerySearch('기묘한이야기')\
    .setSince(start_date)\
    .setUntil(end_date)\
    .setMaxTweets(-1)

# 수집 with GetOldTweet3
print("Collecting data start.. from {} to {}".format(days_range[0], days_range[-1]))
start_time = time.time()

tweet = got.manager.TweetManager.getTweets(tweetCriteria)

print("Collecting data end.. {0:0.2f} Minutes".format((time.time() - start_time)/60))
print("=== Total num of tweets is {} ===".format(len(tweet)))

Collecting data start.. from 2020-01-01 to 2020-03-31
Collecting data end.. 21.67 Minutes
=== Total num of tweets is 7758 ===

In [5]: # GetOldTweet3 에서 제공하는 기본 변수
# 유저 아이디, 트윗 링크, 트윗 내용, 날짜, 리트윗 수, 관심글 수
# 원하는 변수 골라서 저장하기

from random import uniform
from tqdm import tqdm_notebook

# initialize
tweet_list = []

for index in tqdm_notebook(tweet):

    # 메타데이터 목록
    username = index.username
    link = index.permalink
    content = index.text
    tweet_date = index.date.strftime("%Y-%m-%d")
    tweet_time = index.date.strftime("%H:%M:%S")

    info_list = [tweet_date, tweet_time, username, content, link]
    tweet_list.append(info_list)

    # 휴식
    time.sleep(uniform(1,2))

HBox(children=(IntProgress(value=0, max=7758), HTML(value='')))
```

예시로 2020년 1월 1일부터 3월 31까지 "기묘한이야기"를 검색한 결과를 마이닝하였습니다.

```
In [6]: # csv 파일로 결과 저장 - pandas
# 파일 저장하기

import pandas as pd

#twitter_df = pd.DataFrame(tweet_list,
#                           # columns = ["date", "time", "user_name", "text", "link", "retweet_counts", "favorite_counts",
#                           # "user_created", "user_tweets", "user_followings", "user_followers"])

twitter_df = pd.DataFrame(tweet_list,
                          columns = ["date", "time", "user_name", "text", "link"])

# csv 파일 만들기
twitter_df.to_csv("sample_twitter_data_{}_to_{}.csv".format(days_range[0], days_range[-1]), index=False)
print("=== {} tweets are successfully saved ===".format(len(tweet_list)))

=== 7758 tweets are successfully saved ===

In [7]: #생성 파일 확인
# 파일 확인하기

df_tweet = pd.read_csv('sample_twitter_data_{}_to_{}.csv'.format(days_range[0], days_range[-1]))
df_tweet.head(10) # 위에서 10개만 출력

Out[7]:
```

	date	time	user_name	text	link
0	2020-03-31	23:52:03	H	요즘 킹덤시리즈가 유행하던데 난 넷플 잘, 많이 안봐서 킹덤, 기묘한이야기, 내가	https://twitter.com/status/124...
1	2020-03-31	21:07:31		혁 기묘한 이야기 보러구 했는데!!! 모던 패밀리 재밌어요! 애들이 너무 귀엽습디	https://twitter.com/B... 2450954049...
2	2020-03-31	20:58:52		기묘한 이야기 재밌고 다 좋는데 시즌3는 특히 엄청 그로테스크하고 고여하니 주	https://twitter.com/r... as/12450932...
3	2020-03-31	20:57:28		꿈 속 인물한테 꿈에서 꿈 쓴 거 꿈했다고 말했는데 자기도 똑같은 꿈 썼다고 하더	https://twitter.com/hy... s/124509287...
4	2020-03-31	20:08:00		분야 기묘한이야기 심형어를 보고자옴	https://twitter.com/9... 2450804244...
5	2020-03-31	19:13:12		일요야 기묘한이야기 시즌3 촬영예행연출이었는데 진작에 출시됐네	https://twitter.com/h... s/124506663...
6	2020-03-31	19:03:14		힐 넷플 보세요?!! 산타클라리타다이어트 / 엘리트를 / 나를 자바빈 스팅이 / ...	https://twitter.com/m... 1245064127...
7	2020-03-31	18:22:17		넷플릭스 추천해주세용 무서운거 빼고 ... 저 겁 완전 많아서 기묘한이야기도 못보	https://twitter.com/... 2450538196...

크롤링한 내용을 csv 파일로 저장하여 위와 같이 저장하고 저장한 파일에서 내용과 관련된 부분만 추출하여 konlpy 와 word2vec 을 이용한 분석을 진행해보았습니다.

```
In [24]: from konlpy.tag import Twitter
twitter = Okt()

word_dic={}

for i in df["text"]:
    mallist=twitter.pos(i)
    for word in mallist:
        if word[1]=="Noun" or word[1]=="Verb" or word[1]=="Adjective":
            if not(word[0] in word_dic):
                word_dic[word[0]]=0
            word_dic[word[0]]+=1
print(word_dic)
```

'같은데': 48, '있지': 9, '나': 519, '어제': 68, '학교': 28, '귀신': 30, '뽕다': 71, '년': 9, '또': 119, '헛소리': 4, '어떻다고': 1, '그래': 21, '일정': 2, '팔로워': 2, '도달': 2, '사': 24, '개장': 4, '확정': 3, '읽니다': 13, '너': 102, '그': 339, '소문': 7, '들은': 5, '적': 23, '있어': 36, '아가': 5, '발증': 4, '플귀': 2, '길': 15, '모어': 7, '걸스': 26, '임': 118, '낫': 73, '오케이': 80, '부루클린': 68, '나인': 141, '글리': 10, '엘리스': 1, '활뉴블': 1, '있는데': 63, '다스틴': 21, '엄마': 64, '둘': 51, '나와서': 20, '괴리감': 1, '들어오': 2, '본다': 41, '나탈': 1, '리아': 1, '커피': 7, '나오는': 85, '뒤': 33, '실제': 14, '사귀게': 1, '맹': 33, '취향': 76, '비슷하면': 1, '아메리칸': 14, '반달': 1, '리츠': 6, '봐줘': 21, '좋아하면': 31, '반': 16, '봐': 222, '다': 207, '폴리': 12, '티션': 11, '중이': 220, '꿈': 265, '오류': 80, '불': 81, '오줌': 3, '지리': 3, '재밌음': 51, '보고싶는데': 25, '전도': 6, '나갈': 1, '글루': 15, '리스': 17, '뭇': 209, '레이스': 170, '내일': 33, '만나': 4, '보실수': 3, '보신': 28, '문': 100, '있나올': 1, '다음': 61, '불파': 78, '알파': 3, '고민': 33, '중': 297, '봤다니': 12, '가발': 2, '바느질': 1, '글꼴': 1, '용': 6, '와': 93, '왜': 196, '레오': 3, '색': 5, '국내': 3, '팔자': 1, '알아서': 8, '이런': 66, '불것': 1, '하계': 29, '만드': 1, '켓링': 1, '해야하는데': 5, '자야': 5, '해서': 113, '일단': 81, '글러다니면': 1, '인형': 24, '마려': 175, '씩워': 1, '불': 149, '꼭같다': 1, '게이': 10, '툰이다': 1, '보시나요': 10, '꿈': 1, '이제': 156, '이정트': 1, '피라미드': 1, '세울': 1, '있게': 6, '되었습니다': 4, '비장한': 1, '표정': 11, '진심': 42, '열': 25, '동네': 4, '부': 31, '다봤다': 1, '와': 16, '보고싶다': 32, '보려고': 31, '달': 49, '무료': 17, '하는건데': 2, '제': 176, '끝내고': 8, '온다': 4, '아': 211, '복': 6, '불': 1, '아시': 10, '전': 131, '수박': 2, '결': 1, '할기로': 1, '파는': 7, '편이': 16, '스킨스': 11, '같은것도': 4, '그냥': 83, '할': 2, '분위': 45, '종당': 2, '이러고': 5, '좋아함': 31, '성격': 9, '대사': 39, '관계도': 1, '이런건': 4, '알바': 3, '무지개': 1, '전구': 4, '배송': 5, '와라': 1, '노경': 108, '내면': 2, '방': 32, '처': 14, '박혀있을거라고': 1, '데이': 8, '결제': 49, '했다': 1, '보라': 16, '안보': 5, '28, '아남': 48, '기묘한이야기': 1, '존나': 111, '좋아한다고': 2, '오빠': 22, '나가고': 4, '나가자': 1, '배우': 58, '주하연': 8, '마지막': 7, '4, '한국': 54, '오괴': 27, '대백': 10, '프로젝트': 9, '추월자': 8, '남': 33, '공유': 19, '님': 182, '쟁': 89, '쓰요': 1, '재밌음': 1, '바': 51, '음': 16, '다방송': 1, '게': 84, '위영균요': 1, '말이': 1, '아니저': 3, '같이': 33, '달와지만': 1, '머가': 10, '다

위의 사진과 같이 명사, 동사, 형용사 같은 리뷰와 작품과 연관지어 중요한 품사의 단어들만 다시 추출하였고 그 단어가 마이닝한 결과 내에서 얼마만큼의 횟수만큼 사용되었는지 파악하였습니다.

```
In [26]: for word, count in keys[:20]:
        print("{} {} ".format(word, count))
```

```
이야기 7966
기묘한 7753
시즌 942
천재 607
나 519
거 511
넷플릭스 465
보고 462
추천 459
저 418
넷플 369
것 352
그 339
드라마 338
내 313
중 297
때 265
집 265
안 259
곳 238
```

위와 같이 리뷰에서 가장 많은 쓰인 단어를 추출할 수 있었고 이를 토대로 다음 과정에서는 유용한 단어에 대한 추출방법이 과제가 될 것 같습니다.

```
In [31]: results = []
        for i in dff["text"]:
            mallist = twitter.pos(i, norm=True, stem=True)
            # stem=True -> 어근으로 출력하라는 의미
            # ex) '그려요' -> '그렸다'
            #print(mallist)
            r = []
            for word in mallist:
                if not word[1] in ["Josa", "Punctuation", "Foreign", "Suffix", "Eomi"]:
                    r.append(word[0])
            # 결과에서 제외 할 품사 입력하기
            print(r)
            rl = (" ".join(r)).strip() #공백제거
            results.append(rl)
        print(results)
```

```
['기묘하다', '이야기', '저는', '1', '2', '3', '순서대로', '재미', '출어한다', '출어', '시즌', '4', '된다', '거', '대답', '보기']
['하이큐', '회지', '하산', '기묘하다', '이야기', '하산', 'In', 'our', 'time', '하산', '한자봉', '세남자', '양도', '판매', '구함',
'나', '열', '연선', '부탁드리다']
['그냥', '보다', '거', '계속', '보구', '요즘', '기묘하다', '이야기', '불', '까맣다', '생각', '중', '이다', '닷', '썩어남', '뭉',
'보다']
['기묘하다', '이야기']
['발레', '영상', '유튜브', '보다', '저', '찾다', '보다', '바', '졌다', '기묘하다', '이야기', '저', '준다', '거의', '보다', '같다',
'에', 'ㅋㅋ', 'ㅠㅠ', '새롭다', '나오다', 'ㅠㅠ', '노래', '저', '방탄', '소년단', '런', '들다', 'ㅋㅋ', '모', 'ㅋㅋ']
['배즈', '오브', '프레이', '할리퀸', '예쁘다', '그냥', '할리퀸', '다', '항', '클로켓', '한국판', '기묘하다', '이야기', '일드', '감
투룩', '더', '이상', '노닐', 'ㅠ']
['기묘하다', '이야기', '하나', '영기다', '있다', '영역', '오다']
['앗', 'ㅋㅋ', '저', '그때', '그때', '다르다', '요즘', '짧다', '발레', '영상', '클립', '몇개', '주', '구', '장장', '들리다', '보
구', '드라마', '기묘하다', '이야기', '시즌', '4', '나오다', '같다', '재탕', '중이', 'gt', 'It', '영화', '요즘', '재탕하다', '없
다', 'ㅋㅋ', '유튜브', '요즘', '뭉', '들다', 'gt', 'It']
['기묘하다', '이야기', '미치다', '마감', '하다', '되다', '더', '과', '이', '십', 'ㄱㅇ', 'ㅇ', '모모']
['기묘하다', '이야기', '한번', '보다', '보다']
['마블', '제시카', '존스', 'DC', '타이탄', '기묘하다', '이야기', '한니발']
['기묘하다', '이야기', '4', '연체', '나오다', 'ㅠㅠ', '내', '기', '달리', '있다']
['하이큐', '회지', '양도', '반다', '레스큐일', '원더풀', '데이즈', '기묘하다', '이야기', '마법사', '기록', 'oh', 'kids', '디엠',
'준다', 'ㅠ']
```

다음은 형용사와 동사를 어근으로 바꿔주는 작업을 하였습니다.

```
In [33]: from gensim.models import word2vec
        data = word2vec.LineSentence(data_file)
        print(data)
        model = word2vec.Word2Vec(data, size=100, window=10, hs=1, min_count=2, sg=1)
        # CBOW, Skip-gram(0)
        model.init_sims(replace=True) #필요없는 메모리는 unload
        model.save("news.model")
        print("ok")
```

```
<gensim.models.word2vec.LineSentence object at 0x00000178B8BC6488>
ok
```

```
In [36]: model = word2vec.Word2Vec.load("news.model")
        print(model.similarity("기묘하다", "넷플릭스"))
        print(model.similarity("기묘하다", "일리"))
        #두 단어의 유사도 -> 10개 근접할 수록 서로 상관관계가 있다.
        print(model.most_similar("기묘하다"))

        print(model.most_similar(positive=["기묘하다"]))
        print(model.most_similar(positive=["기묘하다", "이야기"], negative=["정그럽다"], topn=5))
```

```
0.5386249
0.32553965
[('개굴', 0.6190359592437744), ('역시', 0.6100665926933289), ('웁', 0.6065816283226013), ('기정', 0.6044149398803711), ('이따', 0.604
0357947349548), ('알다', 0.6020412445068359), ('이러다가', 0.6007715463638306), ('거왕', 0.6000221967697144), ('도리', 0.598800897598
2666), ('혹', 0.5978043079376221)]
[('개굴', 0.6190359592437744), ('역시', 0.6100665926933289), ('웁', 0.6065816283226013), ('기정', 0.6044149398803711), ('이따', 0.604
0357947349548), ('알다', 0.6020412445068359), ('이러다가', 0.6007715463638306), ('거왕', 0.6000221967697144), ('도리', 0.598800897598
2666), ('혹', 0.5978043079376221)]
[('ㅠㅠ', 0.4821982681751251), ('열', 0.46736279129981995), ('좋아하다', 0.45916640758514404), ('추강', 0.44668009877204895), ('렌
즈', 0.4345983564853668)]
```

이전의 작업을 토대로 단어 간의 유사성을 판단해보았습니다.

이번 과정에서는 추출한 데이터를 konlpy 및 word2vec 과 같은 분석에 유용한 톨과 연결시키는 작업을 해보았고 이 결과 유용한 데이터의 선별 및 축약어와 같은 비속어에 대한 판별 또한 다음 과제가 될 것으로 보입니다.

## 5. UI 제작

(진행도가 있을 때 추가 작성예정)



### III. 과제 평가

#### 1. 개선방안

- 이번 주차는 진행하면서, 이전과 같이 단순히 API를 사용해보는 것이 아니라, 주제를 확장하다보니 개념적인 부분의 이해가 부족한 것 같아 개념적인 이해를 코딩하면서 이해하고 정리하였으며 실제로 어떤 식으로 적용할 수 있을까 생각을 많이 한 주차입니다.

 bs4, selenium, lxml 크롤링.ipynb	12시간 전
 practice.ipynb	Running 한 시간 전
 Twitter 품사분석(phrases 문장을 구단위로 쪼개주는 메서드가 있는게 특징).ipynb	Running 18시간 전
 감성분석(감성사전이용).ipynb	Running 15시간 전
 감성분석(지도 기계학습기반).ipynb	Running 14시간 전
 군집분석.ipynb	Running 13시간 전
 꼬꼬마 연습.ipynb	Running 18시간 전
 불용어제거, n-gram(n개단어), 품사분석, 어근동일화(의미는 같은데 생긴게 달라. s나 es같은게 붙어서).ipynb	Running 한 시간 전
 빈출단어 추출.ipynb	Running 15시간 전
 소셜커머스후기분석.ipynb	11시간 전
 연관어분석.ipynb	Running 13시간 전
 정규표현식 연습.ipynb	Running 18시간 전
 트위터 크롤링.ipynb	12시간 전

- 이번의 경우 konlpy보다는 nltk, 영화리뷰(영어) data set과 api를 이용한 긍부정 점수 산출, 감성 사전 data set의 인덱스화 등 영어를 사용한 분석을 다뤄서 자연어처리와 감성 분석의 개념을 이해하는데 시간을 활용했기에, 다음에는 konlpy와 한국어 감성사전을 이용해보고자 합니다.
- DB구축에 있어서는 교수님께서 조언해주신대로 다른 기능을 확장시키고 있습니다. 또한, data별로 어떻게 활용할지를 생각하는 과정에 있어서 기초적인 ERD모델은 구축되었고 소소한 수정 이후에 확정되는 대로 데이터베이스를 구축하고자 합니다.
- 어떤 데이터를 추출할지(웹소셜 리뷰)를 정했기 때문에 사이트별 크롤러를 제작하고자 합니다.

#### 2. 기대효과

##### ㄱ. 기업적 측면

- 즉각적인 피드백이 필요한 문화 산업의 경우 통합적으로 리뷰를 확인 가능함으로서 앞으로의 홍보, 제작, 투자 방향 선택에 도움이 되는 지표가 될 것이다.

## **L. 사용자 측면**

- 별점 테러와 같이 실제 작품에 대한 후기가 아닌 평가 반영으로 실제 작품의 후기를 원하는 사용자에게 더욱 사실적인 후기를 각기 다른 플랫폼에서 검색해 볼 필요 없이 한 곳에서 확인이 가능할 것이다.
- 리뷰에서 자주 언급된 단어를 통해 중요 키워드를 산출해내기 때문에 선호하는 양상의 작품을 기호에 맞춰 선택하기 쉽다.
- 비슷한 성격의 작품을 추천받을 수 있다.

## **참고문헌**

- 파이썬을 활용한 클로러 개발과 스크레이핑 입문, 2019, 카토 카츠야, 요코야마 유우키, 위키북스
- 파이썬 데이터 수집 자동화 한방에 끝내기 한입에 웹크롤링, 2018, 김경록, 서영덕, 비제이퍼블릭
- 파이썬을 이용한 웹크롤링과 스크레이핑, 2018, 카토 코타, 위키북스
- 파이썬을 이용한 머신러닝, 딥러닝 실전 개발 입문, 2019, 쿠지라 히코우즈쿠에, 위키북스
- Web Scraping with Python, 2019, 라이언미첼, 한빛미디어
- 잡아라! 텍스트 마이닝 with 파이썬, 2019, 서대호, 비제이퍼블릭
- <https://www.crummy.com/software/BeautifulSoup/bs4/doc.ko/>
- 2010, 오피니언 마이닝 기술을 이용한 효율적 상품평 검색 기법, 윤홍준, 김한준, 장재영
- 2011, 한글 텍스트의 오피니언 분류 자동화 기법, 김진옥, 이선숙, 용환승
- 2013, 상품평가 텍스트에 암시된 사용자 관점추출, 장경록, 이강욱, 맹성현
- 2013, 텍스트 마이닝을 이용한 2012년 한국대선 관련 트위치 분석, 배정환, 손지은, 송민

- 2014, 한글 감성어 사전 api구축 및 자연어 처리의 활용, 안정국, 김희웅
- 2015, 한글 음소단위 trigram-signature 기반의 오피니언 마이닝, 장두수, 김도연, 최용석
- 2016, 소셜네트워크서비스에 활용할 비표준어 한글처리 방법연구, 이종화, 레환수, 이현규
- 2017, [https://www.samsungsds.com/global/ko/support/insights/1195888\\_2284.html](https://www.samsungsds.com/global/ko/support/insights/1195888_2284.html), 윤병운 교수
- 2017, 한국어 비정형 데이터 처리를 위한 효율적인 오피니언 마이닝 기법, 남기훈
- 2020, A study on Sentiment Analysis with Multivariate ratings in Online Reviews, 임소현