중간보고서

Vol. 11

프로젝트 명 : 데이터 마이닝과 분석을 통한 웹소설 종합 인포 웹 어플리케이션

지도 교수 : 컴퓨터정보공학부 이기훈

팀 장 : 컴퓨터정보공학부 한승주

팀 원 : 컴퓨터정보공학부 김성종

컴퓨터정보공학부 조예슬

2020. 7. 10

광운대학교
KwangWoon University

목 차

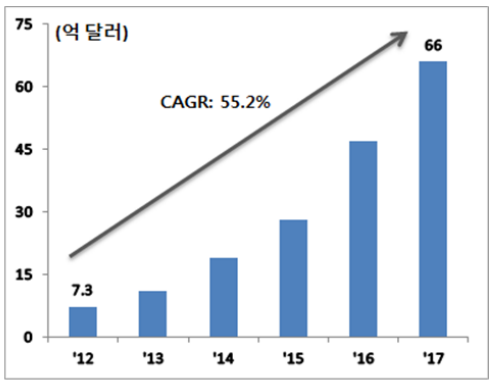
I. 프로젝트의 개요

1. 배경 및 필요성

가. 시장 성장

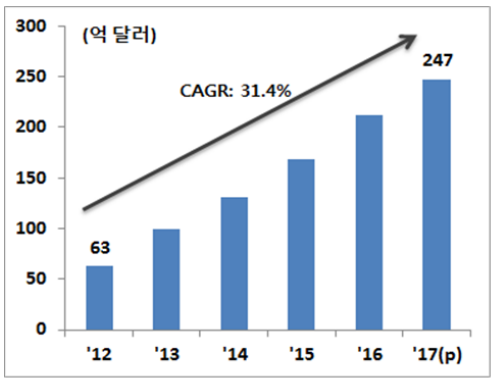
현대에 이르며 미디어 시장은 많은 변화가 이루어졌다. 미디어는 물론 미디어 플랫폼도 다양해지며 현대인들은 더 쉽고 간편하게 빠르고 편리하게 미디어를 접할 수 있게 되었다. CD나 카세트 같은 아날로그식의 접근은 음원 다운로드를 넘어 스트리밍으로 디지털화되었고 영화와 TV 미디어와 같은 영상물도 마찬가지로 영화관, TV를 넘어 다운로드와 VOD 시스템의 활성화를 지나 '넷플릭스', '왓챠플레이', '훌루' 등 스트리밍 플랫폼으로 넘어오며 가히 스트리밍의 시대가 도래했다고 할 수 있다. 이런 변화는 출판 업계도 마찬가지였다. 만화와 소설 등 책들이 손 안의 휴대전화로 들어오게 되며 책을 접하는 방식도 접할 수 있는 곳도 다양해졌다.

< 전 세계 음악 스트리밍시장 규모 >



자료 : 국제음반산업협회(IFPI).

< 전 세계 OTT 서비스시장 규모 >



자료 : PwC(2017), ITU(2017), 정보통신진흥원(2018) 재인용.

그림 1. OTT 서비스 시장 규모 (출처 : 현대경제연구원)[1]

이러한 미디어 시장 속 오늘 우리가 주목할 것은 웹소설이다. 웹소설이 현대에 이르러 새롭게 나타난 장르는 아니다. 웹소설의 전신은 과거 인터넷 소설과 그 전 PC 통신에서 퍼지던 소설부터 시작되었다고 할 수 있다. 그러나 인터넷과 스마트폰의 보급으로 중구난방으로 퍼지던 소설을 모아 웹소설 연재처가 생기고 이 시장에 다음, 네이버 등 한국의 인터넷 시장을 주름잡는 대표 포털 사이트가 뛰어들며 그 시장은 날로 커지고 있다.

웹소설 시장이 커지게 된 또 다른 이유는 OSMU, 원 소스 멀티 유즈가 큰 역할을 하였다. 2018년 한국 콘텐츠 진흥원의 조사에 따르면 국내 웹소설 시장은 2013년

[1] 콘텐츠 스트리밍 산업의 성장동력화가 시급하다. (현대경제연구원, 2019.02)

100억 원 규모에서 2018년 4000억 원까지 40배 성장했다. 이러한 배경에는 웹소설을 바탕으로 제작된 드라마의 성공이 있다. 또한, 주변국인 일본은 남녀노소 상관없이 서브컬처 문화를 수용하고 이를 즐기고 있어 이러한 콘텐츠 시장이 이미 발달한 상태이고, 중국도 연간 2조 원의 시장 규모를 가지고 있을 만큼 웹툰 및 웹소설 시장이 큰 편이다. 최근에는 중국, 일본 외에도 세계 각국의 작품들을 수입 및 수출하고 있고 새로이 제작되는 웹툰이나 드라마, 영화 심지어는 게임까지도 많은 분야에서 웹소설을 원작으로 하여 새로운 콘텐츠를 양산해내고 있다.

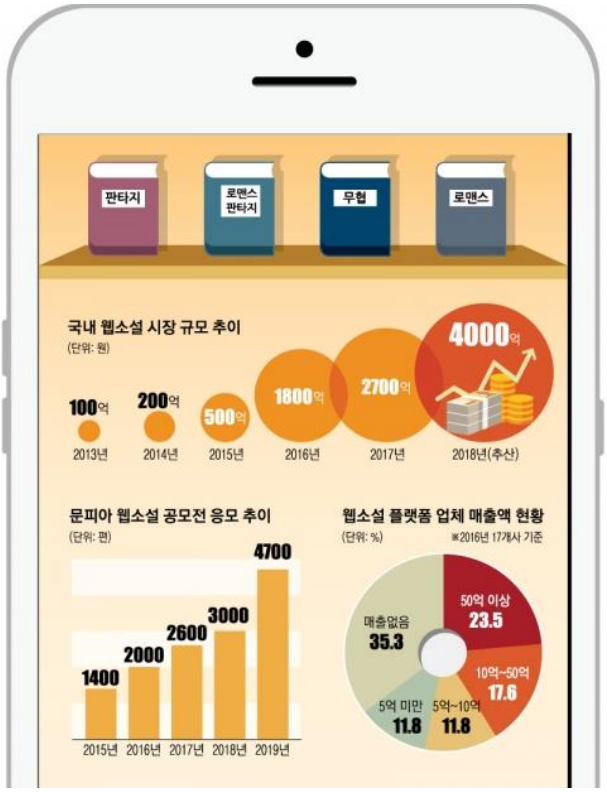


그림 2. 국내 웹소설 시장 규모 추이 (출처 : 서울신문, 2019.05)[2]

이렇게 커진 웹소설 시장을 방증하듯 독점 연재도 추진하며 인지도 높은 플랫폼만 약 6곳, 그 외에도 여러 작품의 연재 서비스를 제공하는 곳까지 합하면 그 수는 10곳을 훌쩍 넘는다. 이렇게 커진 플랫폼들은 저마다 신인 작가를 육성하기 위한 공모전을 열기도 하고 독점 연재작 계약도 진행하며 독자 유입을 위해 총력을 기울이고 있다.

[2] '하루 5분' SNS 하듯 쓰윽~ 4000억 시장 펼친 웹소설 (서울신문, 2019.05)

나. 문제 정의

[1] 양산화

웹소설의 인기와 트렌드에 따른 양산화에 따라 수많은 작품이 탄생을 했지만 작품의 질이 그에 못 미치는 경우가 많아졌다. 독자 또한 이를 직접 다양한 형태로 그 의견을 내비치고 있다. 플랫폼 내부의 댓글과 별점, SNS 나 각종 커뮤니티에 자신의 리뷰를 남김으로써, 직접 칭찬이나 불만을 터트리며 작품에 대한 평가를 하고 때론 그런 행동이 작품에 직접적인 영향을 주고 있다.

[2] 다양화

웹소설을 제공하는 플랫폼뿐 아니라 SNS 나 커뮤니티 등 다양한 플랫폼에 독자는 리뷰를 남기는데 이용자들의 나이, 성별 등 다양한 특성에 따라 사이트에 남겨지는 리뷰의 방식도 저마다 다른 특징을 가진다. 따라서, 이러한 리뷰들을 모아 분석하여 독자에게 제공함으로써 작품 판별에 대한 지표를 제공하고자 한다.

2. 목표

댓글과 소셜미디어, 블로그, 커뮤니티 등 다양한 곳에 퍼져있는 웹소설 리뷰를 다양한 시각에서 분석해보고 현재 웹소설 트렌드와 독자의 작품 평가 변화 양상이 어떤지 살펴본다.

3. 설계 내용

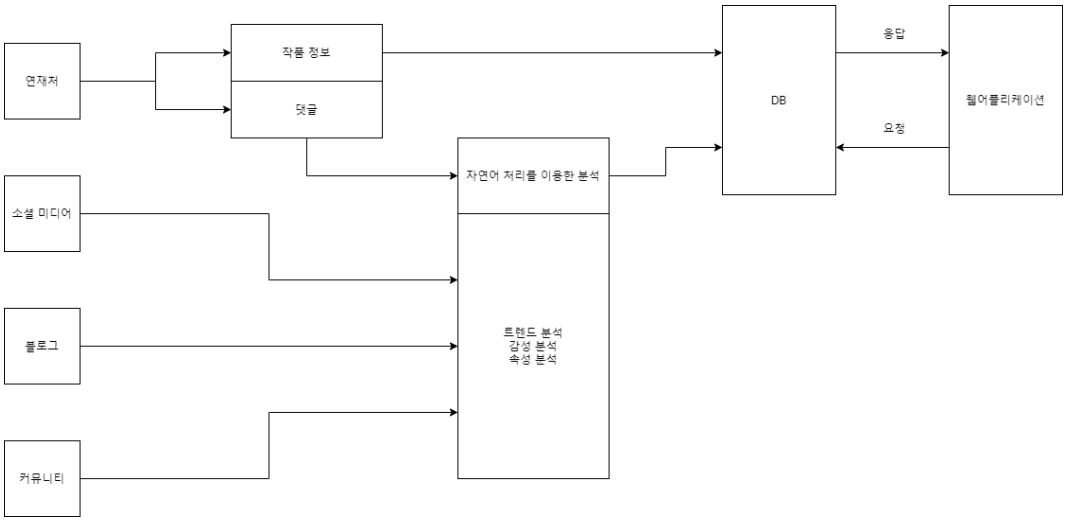


그림 3. 시스템 구조

- 웹소설 플랫폼에서 HTML 코드를 분석하여 작품의 기본 정보와 댓글을 파이썬을 이용하여 추출한다.

- 소셜미디어와 블로그, 커뮤니티에서 작품 제목 검색 시 나오는 글의 내용과 작성 날짜를 추출한다.
- 연재처 속 댓글과 소셜미디어, 블로그, 커뮤니티 글을 자연어 처리를 이용해 분석한다.
- 분석한 결과와 품 정보를 MySQL을 이용해 구축한 DB에 저장한다.
- 웹페이지를 이용하여 DB에 저장된 정보를 사용자에게 제공한다.

II. 프로젝트의 내용

1. 설계 및 개발의 내용

가. 개념 설계 (구조 설계)

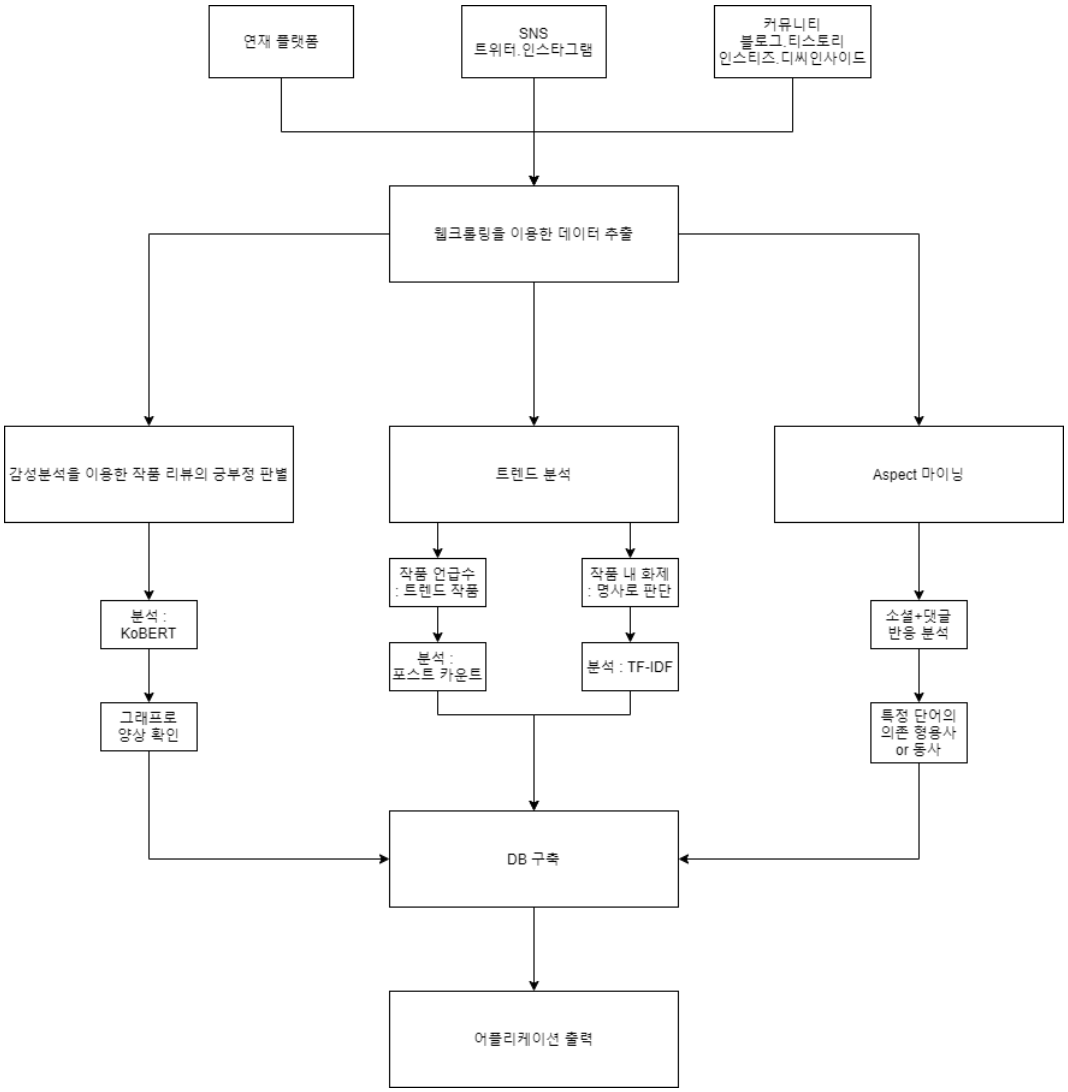


그림 4. Flow Chart

- **[데이터 수집]** 연재 플랫폼에서 작품 정보와 함께 댓글을 소셜 미디어와 커뮤니티, 블로그에서 작품과 관련된 글을 수집하고 수집한 정보의 분석 과정에 들어간다.

- **[데이터 분석 1]** SNS와 커뮤니티에서 해당 작품이 얼마나 언급되었는지 포스트 수를 통해 작품의 화제성을 판단한다. 독자들이 자신의 생각을 펼치는 많은 플랫폼에서 작품 자체의 언급은 호평, 혹평, 논란 등 여러 가지 이유로 이루어지고 이를 통해 순순히 작품의 화제성만을 판단할 것이다.

- **[데이터 분석 2]** 작품과 관련된 글에서 명사만을 추출하여 해당 작품에 대해 어떤 주제가 화제가 되고 있는지 TF-IDF를 이용한 분석을 통해 알아볼 것이다.

- **[데이터 분석 3]** 댓글과 리뷰에 가까운 블로그 글을 KoBERT를 이용한 분석을 하여 현재 작품에 대한 독자들의 반응을 긍부정도 변화 양상을 알아볼 예정이다.

- **[데이터 분석 4]** 소셜미디어와 댓글의 반응을 의존 형용사 혹은 동사를 분석하여 주인공, 줄거리와 같은 속성에서 어떤 평가를 받고 있는지 알아볼 것이다.

- **[데이터 저장]** 연재 플랫폼에서 수집한 작품의 기본 정보와 분석 결과를 MySQL을 이용해 구축한 DB에 저장한다.

- **[결과물 구현]** 웹 애플리케이션을 구현하여 DB에 저장된 결과를 사용자에게 제공한다.

나. 상세 설계 (기능 설계)

1) 플랫폼의 웹사이트 코드 분석 및 파이썬 크롤링(BeautifulSoup, Selenium)

- 정보를 추출할 연재 사이트와 분석할 리뷰가 적힌 플랫폼 선정

플랫폼	비고
조아라 (프리미엄)	자유로운 연재 가능 정식 연재작 선정기준 필요
문피아 (유료 웹소설)	자유로운 연재 가능 정식 연재작 선정기준 필요
카카오페이지	리뷰 크롤링 불가
리디북스	
네이버 시리즈	화수별 리뷰가 전체 리뷰에 포함
네이버 웹소설	네이버 단독 연재작

표 1

SNS 및 커뮤니티	비고
네이버 블로그	리뷰 중심
티스토리	리뷰 중심
트위터	독백형 추천작으로 언급이 多
인스타그램	해시태그 이용한 검색만 가능 리뷰 중심
디씨인사이드	독백, 리뷰, 추천 多
인스티즈	추천, 독백, 리뷰 多

표 2

- **트렌드 분석을 위한 플랫폼**의 경우 실시간 독자 반응을 살피기 좋은 곳으로 선정하였다. 웹소설 독자의 연령층을 고려하여 현대인들이 자주 사용하는 **소셜미디어**와 실시간과 익명이라는 특성을 이용하여 많은 이야기가 오가는 **커뮤니티**를 선택했다. 소셜미디어의 경우, 커뮤니티와 특성은 비슷하지만 일기장처럼 유저 개인의 생각을 가감 없이 들어내는 '트위터'와 해시태그라는 특징을 이용해 다양하게 사용되는 '인스타그램'을 선정하였고 커뮤니티는 오래전부터 많은 사용자를 보유하고 있는 커뮤니티이자 많은 장르의 이야기가 오가는 '디씨인사이드'와 '인스티즈'를 선택했다. 또한 네이버 블로그나 티스토리는 리뷰 중심의 포스트가 많이 게시되는 것으로 미루어 보아 작품의 내용 측면에서 어떤 것이 어떻게 화제가 되고 있는지 알 수 있기 때문에 선택하였다.
- 데이터 수집 방법은 다음과 같다. 먼저, 정보를 추출할 웹사이트의 HTML 코드를 분석한다.
- HTML 코드를 가져오기 위해서 웹사이트에 요청을 해야 하는데 이는 **requests 라이브러리**를 사용한다. 다음은 정보 수집을 위한 크롤링 예시이다.

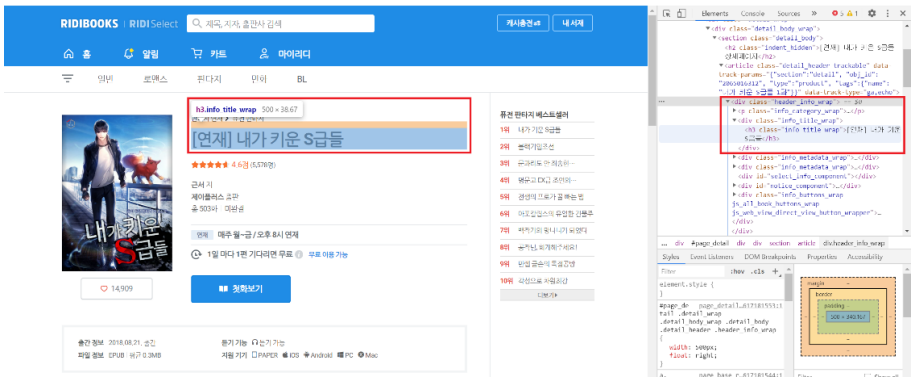


그림 5. 웹사이트 속 HTML 코드 예시

```
import requests
from bs4 import BeautifulSoup

req=requests.get('https://ridibooks.com/books/875103701')
source=req.text
soup=BeautifulSoup(source, 'html.parser')

# 제목
title_list=soup.select("#page_detail > div.detail_wrap > div.detail_body_wrap > section > article.detail_header.trackable > #
div.header_info_wrap > div.info_title_wrap > h3")
title=title_list[0].text
title=title[5:] #'[연재] '가 제목 앞에 붙음
```

그림 6. 분석한 HTML 코드를 바탕으로 한 크롤링 코드 예시 [CSS Selector 이용]

<그림6> 코드 내용

[req = request.get(**URL**)]

- 가져온 코드에서 원하는 정보를 추출하기 위해 BS4의 **BeautifulSoup 라이브러리**를 이용한다.

[soup = BeautifulSoup(**웹페이지 HTML 코드**, 'html.parser')]

- 정보를 추출할 때는 **CSS Selectors, XPATH** 등을 이용하여 작품의 기본 정보 tag를 찾아 추출한다.

[title_list = soup.select(**정보가 있는 태그 경로 selector**)]

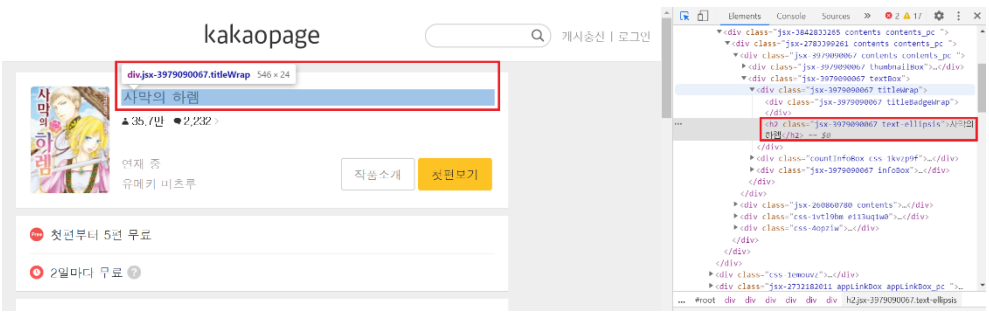


그림 7. 자바스크리트로 렌더링된 웹페이지

```
#작품의 링크
url="https://page.kakao.com" + newPage
driver = webdriver.Chrome("C:/chromedriver.exe")
driver.get(url)
bs_obj = bs4.BeautifulSoup(driver.page_source, "html.parser")

driver.find_element_by_xpath('//*[@id="root"]/div[3]/div/div/div[1]/div[2]/div[2]/div[2]/button[1]').click()

time.sleep(0.2)

if latest.text!="완결":
    update_days = driver.find_element_by_xpath('//*[@id="root"]/div[3]/div/div/div[1]/div[2]/div[2]/div[1]/p[1]')
    print("연재 요일 : " + update_days.text.replace(" 연재",""))
```

그림 8. Selenium을 이용한 크롤링 예시 [XPATH 이용]

<그림8> 코드 내용

[driver = webdriver.Chrome(**Chrome 드라이버 경로**)]

- Java-script로 렌더링되는 동적 웹사이트인 경우 BeautifulSoup으로 크롤링하는데 한계가 있기 때문에 **Selenium**을 이용한다.

[driver.get(**URL**)]

- Selenium을 이용한 크롤링을 위해서 크롬 웹드라이버를 설치하고 request를 통한 요청이 아닌 driver를 통해 직접 창을 열어 수집한다.

[driver.find_element_by_xpath(**정보가 있는 태그 경로 xpath**)]

2) Data Processing (데이터 분석)

추출 데이터의 분석은 연재 플랫폼의 리뷰와 댓글 커뮤니티 및 소셜 데이터를 통해 이뤄진다.

가) 트렌드 분석 : 커뮤니티와 소셜미디어에서 작품이 얼마나 언급되었는지를 통해 알아본다.

[1] 작품이 얼마나 언급되고 있는가

작품 자체가 얼마나 언급되었는지 카운트를 통해 작품의 화제성을 판단한다. 주기는 **1달 간격**으로 설정하고 추이를 살펴본다. **트위터, 인스타그램, 디씨인사이드, 인스티즈의 게시물 수를 카운트**한다.

[가] 실시간으로 소통이 이루어지는 소셜미디어와 커뮤니티 공간에서 작품의 언급수는 작품의 화제성과 인지도와 직결된다고 생각한다. 그래서 소셜미디어와 커뮤니티에서 작품을 검색하여 나온 게시물 수를 카운트하고자 한다.

[나] 월 단위로 **가장 많이 언급이 된 10개 작품을 선정**하고 결과물의 메인 화

면에 보여준다.

[2] 작품 내에서 어떤 것이 화제가 되고 있는가

작품에 대해 독자들이 어떤 얘기를 나누는지 1달을 주기로 텍스트 분석을 통해 알아본다. **커뮤니티 및 소셜 미디어**의 글을 가지고 분석한다.

[가] 사용자는 작품과 관련하여 어떤 것이 화제가 되고 있는지 알 수 있을 것이다.

[나] 독자의 표현이 자유로운 소셜미디어와 커뮤니티 공간 속에서 독자들이 작품에 대해 어떤 얘기를 나누고 있는지 알아보기 위해 포스트 속 내용에서 **형태소 분석**을 통해 **명사**를 구분하고 이 명사를 **TF-IDF 방식**을 통해 분석을 할 것이다.

[다] 단순히 KoNLPy[3]를 이용하여 형태소 분석을 하고 여기서 순수히 빈도를 계산하는 BOW[4]방법도 있지만 단어의 맥락을 고려한 정확도를 위해 TF-IDF[5] 방식을 이용하여 벡터화하는 방식을 이용할 것이다.

[라] TF-IDF를 통해 나온 결과에서 **상위 5개의 데이터를 시각화하여** 유저에게 제공할 것이다.

나) 감성 분석 : 독자들의 작품에 대한 반응이 어떻게 변화되고 있는가

1달 간격으로 작품 댓글과 리뷰를 통해 **독자들의 반응** 변화를 살펴본다. **연재처의 댓글과 블로그** 글을 가지고 분석한다. 평가에 대한 점수를 기간에 대한 **그래프**로 표현한다. 이번 분석을 통해 작품의 즉각적인 반응과 이에 대한 변화를 시각적으로 확인할 수 있다.

---

[3] 별첨 [1] 항목 참조

[4] 별첨 [2] 항목 참조

[5] 별첨 [2] 항목 참조

[1] **KoBERT를 활용한 감성 분석**[6]으로 연재 플랫폼의 댓글과 리뷰와 같이 작품의 질에 대해 살펴볼 수 있는 글에서 긍부정도를 살펴보고 이에 대한 비율 변화를 그래프화하여 유저로 하여금 시각적으로 작품이 지속적으로 좋은 평가를 받는지, 언제 반대의 반응을 받았는지 쉽게 알 수 있도록 제공할 것이다.

다) Aspect 분석 : 작품의 속성에 대한 독자들의 느낌은 어떤 것인가

작품 사이트 속 **리뷰와 네이버, 티스토리** 등 리뷰에 대한 게시글이 많은 플랫폼을 가지고 **작품 내 주인공, 분위기, 스토리 등과 연결되어** 자주 나오는 **반응**을 분석한다.

[1] 작품에 대해 구체적으로 작성한 글이 필요하기 때문에 **네이버, 티스토리, 다음 블로그**의 리뷰를 수집하여 분석한다.

[2] 먼저 **군집분석**을 진행한다. 웹소설 중 '닥터최태수'를 예로 들면, '주인공'은 말 그대로 주인공일 수 있지만 '최태수', '닥터최태수', '태수'와 같이 표현방식이 다양하기 때문에 군집분석을 통해 주인공과 유사한 단어에 대해 분석할 것이다.

[3] 각 키워드 별로 키워드를 설명하는 **키워드 주변의 1-3개 단어를 벡터화**하는 것이 효율적이다. 이 중 형태소 분석을 통해 명사를 표현하는 동사나 형용사에 대해서 벡터를 구성하고 **용언 분석기(lemmatize)**를 활용하여 **'원형'**에 대해 **count**를 진행한다.

3) DB 구축

- 파이썬에서는 PyMySQL 패키지를 통해 Python 으로도 외부에서 DB 내 데이터 조작이 가능하여 MySQL 을 활용하고자 한다.

- 그 외에도 SQLite 등의 가벼운 데이터베이스 등도 활용할 수 있지만, 작품의 수가 많아 데이터 수집 결과와 분석 결과를 데이터베이스에 모두 담기에는 저장공간이나 속도면에서 데이터베이스 활용이 어려울 수 있다는 판단 하에 최종적으로 MySQL 을 선택하였다.

---

[6] 별첨 [3] 항목 참조

```
In [1]: pip install PyMySQL

        Requirement already satisfied: PyMySQL in c:#users#naruk#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()
```

그림 9. 파이썬과 DB 연동 라이브러리 사용 예시

```
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/Kevin_Bacon')
        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/Taraclia_District
        /wiki/Romani_people
        /wiki/Turkey
        /wiki/Karachays
        /wiki/Soyot
        /wiki/Chechens
        /wiki/Berbers_in_Belgium
        /wiki/Aghul_people
        /wiki/Archi_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
```

그림 10. 파이썬 코드로 DML 사용

4) 웹사이트 UI 제작

- 밀러의 법칙, 즉 7±2 법칙이란 사람이 한 번에 수용할 수 있는 정보는 7개 전후에 불과하다는 것이다. 이 논문은 웹 사용성에서 실제로 많이 쓰이기도 했지만 가설일 뿐이라는 논란도 받았다. 하지만 분명 사람들은 보통 3-5개 이상의 단계를 거쳐 결론에 도달하게 되면 그 인터페이스는 복잡하다고 느낀다. 이러한

점을 적용하여 저희 팀은 이번 프로젝트에서 보여주고자 하는 결과물만을 최대한 간결하고 한눈에 정보를 전달받을 수 있도록 구성하려 한다.

- 자신이 보고 싶은 작품의 리뷰를 보기 위한 제품의 검색창을 사용자가 보기 편하도록 UI 로 구현한다.

- 결과물을 보여줄 플랫폼으로 안드로이드 어플과 웹사이트를 고민하였으나 사용자의 접근성 편리를 위해 웹사이트 구축으로 결정하였다.

- 메인 화면에서는 커뮤니티와 소셜미디어, 블로그 등 기타 여러 이유를 통해 작품을 언급할 독자가 작품을 언급한 횟수를 분석한 결과 보여줄 것이다. 해당 분석 결과는 곧 화제성을 의미하고 여기서 가장 많이 언급된 10 작품을 선정하여 '이달의 화제작 TOP10'으로 보여줄 예정이다.

- 왼쪽 상단의 메뉴 탭을 누르면 연재처와 장르별로 화제작을 확인해볼 수 있도록 구성하였다.

- 검색을 통해 유저가 검색하고 싶은 작품을 검색해 볼 수 있도록 구성하였다.

- 작품 페이지에서는 수집한 작품의 기본 정보와 함께 TF-IDF 분석 결과를 키워드로 5가지 정도 나열한다. 그 밑에 주요 속성 (주인공, 스토리 등)에 대한 반응을 나열하고 그 하단에 감성 분석의 변화 양상을 시각화한 그래프를 보여준다.

*** 파이썬 플라스크(Python Fask)** : 마이크로 웹 프레임워크로 데이터 수집부터 분석, DB 조작에 사용하던 언어인 파이썬을 이용하여 결과물을 구현이 수월할 것이라 생각하여 선정하였다.

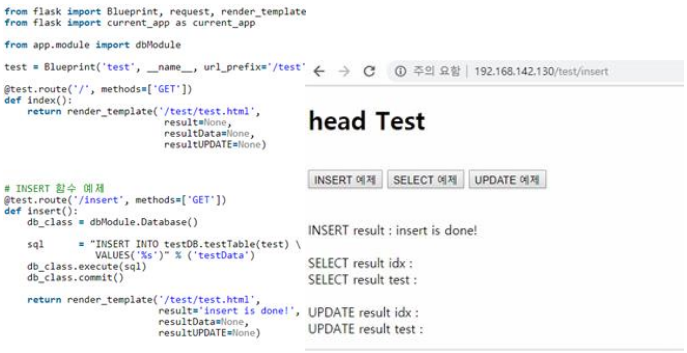


그림 11. 플라스크를 이용한 웹으로 DB 구동하기 예시[7]

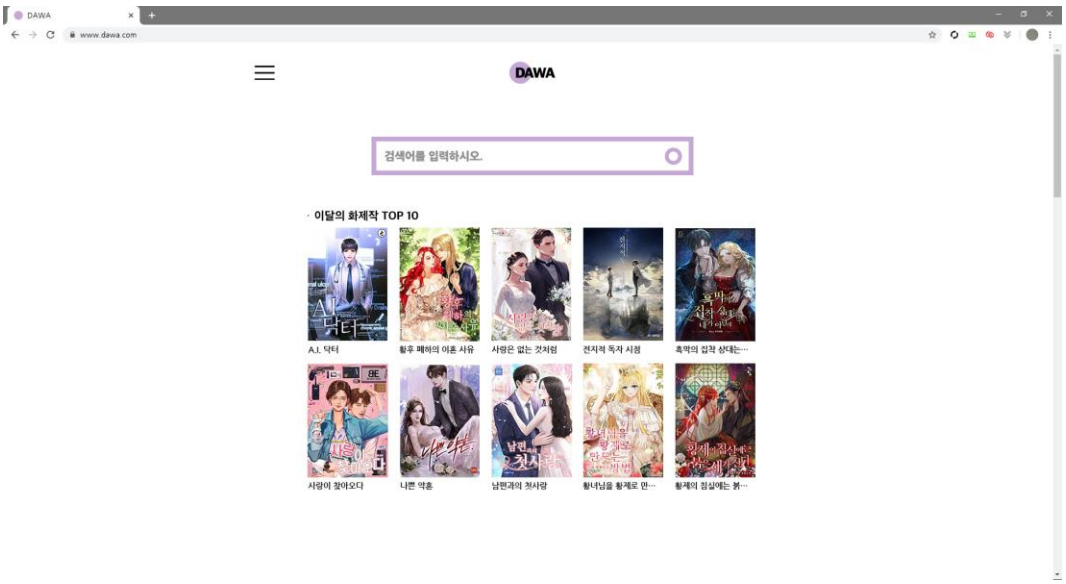


그림 12. 메인화면 예시

[7] 예시 출처 : https://kkamikoon.tistory.com/162

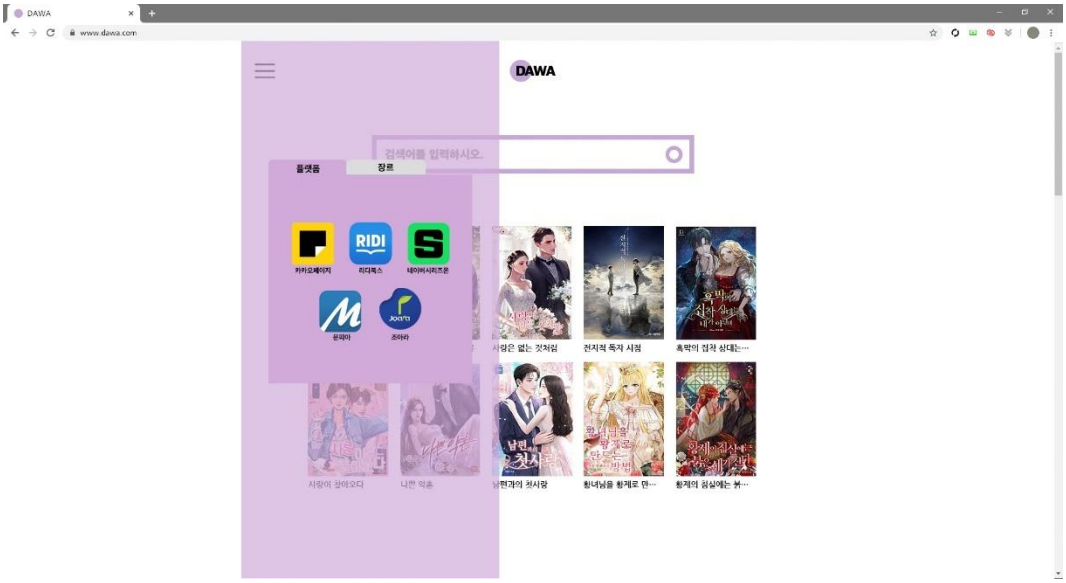


그림 13. 메뉴 화면 디자인 예시

그림 14. 작품 정보 화면 디자인 예시

2. 역할 분담

- **크롤러 제작 [공통]** : 연재처에서는 제목, 저자, 장르, 출판사, 연령 등급, 총 연재 화수, 출간일, 완결 여부, 표지 사진 URL, 작품 소개, 작품 연재처 링크, 댓글까지 12가지 항목에 대해 정보 수집한다. 소셜미디어, 커뮤니티, 블로그에서는 게시글 내용과 작성 날짜, 커뮤니티는 댓글까지 수집한다. 6곳의 연재처와 6곳의 수집처를 조원 모두 나눠 기본 크롤러 제작을 진행한다.

- 데이터 분석 [김성종, 한승주] : 팀 회의를 통해 결정한 6곳의 수집처의 게시글과 댓글의 분석 방법을 바탕으로 구체적인 자연어 처리를 통한 분석 방식을 조사하고 실시한다.

- DB [한승주, 조예슬] : 수집한 작품 정보 데이터와 분석 결과를 웹사이트에 구현할 수 있도록 팀회의를 통해 설계한 Logical-ERD를 수정하며 테이블을 구성하고 DB를 최종 구축한다.

- 웹 애플리케이션 [조예슬, 김성종] : DB에 저장된 데이터를 사용자의 편의성을 고려하여 팀 회의를 통해 구성한 기본 디자인을 가지고 UI를 제작하며 최종 결과물을 구현한다.

성명	역할
한승주 [팀장]	크롤러 제작[공통] : 2곳의 연재처 조아라, 리디북스 및 소셜 미디어(인스타그램, 트위터) 데이터를 수집하는 코드를 작성한다. DB[조장] : 설계한 Logical-ERD를 기반으로 MySQL을 이용한 DB를 구축하고 분석팀과 회의를 통해 지속적으로 ERD 수정을 한다. 데이터 분석[조원] : 데이터 분석을 위한 자료수집과 코드 구축을 지원한다.
김성종 [팀원]	크롤러 제작[공통] : 2곳의 연재처 카카오페이지, 문피아 플랫폼 및 블로그(네이버, 다음, 티스토리) 데이터를 수집하는 코드를 작성한다. 데이터 분석[조장] : Python을 이용한 데이터 전처리 후 분석(Trend in Web-Novel, Trend in each work, Sentimental Analysis, Aspect Extraction)을 위한 코드를 작성한다. 웹페이지 구현[조원] : 웹페이지 제작을 위한 자료수집과 구현을 지원한다.
조예슬 [팀원]	크롤러 제작[공통] : 2곳의 연재처 네이버 시리즈 플랫폼 및 커뮤니티(디시인사이트, 인스티즈) 데이터를 수집하는 코드를 작성한다. 웹페이지 구현[조장] : 회의를 통해 구상한 UI의 구체화와 Java-Script, CSS, HTML을 이용한 웹페이지(PC Web 및 Mobile Web 병행)를 제작한다. DB[조원] : DB 구축을 위한 자료 수집과 ERD 수정 및 구축을 지원한다.

3. 최종 결과물

가. 데이터 수집을 위한 기본 크롤러 제작

1) 플랫폼 크롤링

6곳의 연재처에서 수집할 정보는 총 12가지이다.

[제목, 저자, 장르, 출판사, 연령 등급, 총 연재 화수, 출간일, 완결 여부, 표지 사진 URL, 작품 소개, 연재 링크, 댓글]

가) 카카오페이지 : Selenium과 XPATH를 이용한 마이닝

```
#기다무소설or 소설
#[단행본] x
#[닥터최태수 or 닥터 최태수 포함]

#작품의 링크
url="https://page.kakao.com" + newPage
driver2 = webdriver.Chrome("C:/chromedriver.exe")
driver2.get(url)

driver2.find_element_by_xpath('//*[@id="root"]/div[3]/div/div/div[1]/div[2]/div[2]/div[2]/button[1]').click()

title=driver2.find_element_by_xpath('//*[@id="root"]/div[3]/div/div/div[1]/div[2]/div[1]/h2')
print("제목 : " + title.text)

update_days = driver2.find_element_by_xpath('//*[@id="root"]/div[3]/div/div/div[1]/div[2]/div[2]/div[1]/p[1]')
print("연재일 : " + update_days.text)

writer=driver2.find_element_by_xpath('//*[@id="root"]/div[3]/div/div/div[1]/div[2]/div[2]/div[1]/p[2]')
print("작가 : " + writer.text)



genre=driver2.find_element_by_xpath('/html/body/div[2]/div/div/div/div[2]/div/div/table/tbody/tr[1]/td[2]/div[2]/div[2]')
print("장르 : " + genre.text)

age=driver2.find_element_by_xpath('/html/body/div[2]/div/div/div/div[2]/div/div/table/tbody/tr[1]/td[2]/div[4]/div[2]')
print("연령등급 : " + age.text)

publisher=driver2.find_element_by_xpath('/html/body/div[2]/div/div/div/div[2]/div/div/table/tbody/tr[1]/td[2]/div[3]/div[2]')
print("출판사 : " + publisher.text)

time.sleep(1)
driver2.quit()
```

그림 14. 카카오페이지 작품 정보 크롤링 中

```
작품이름 ->
닥터 최태수
작품이름 : 닥터 최태수
독점여부 : 미독점
감상인원 : 190.5만명
제목 : 닥터 최태수
연재일 : 연재 중
작가 : 조석호
장르 : 기다무소설현판
연령등급 : 전체이용가
출판사 : 시나브로
```

그림 15. 카카오 페이지 작품 정보 크롤링 결과

나) 네이버 시리즈 (+ 네이버 웹소설) : Request 사용

```
req = requests.get(novel_url)
source = req.text
soup = BeautifulSoup(source, 'html.parser')

################## 제목 ###################
container = soup.find('h2')
con = container.text
print("제목: " + con)

################## 글, 그림 ###################
f_wri_ill = soup.find('p', {'class', 'writer'})
author = f_wri_ill.find('a', {'class', 'NPI=a:writer'})
aut = author.text
illustrator = f_wri_ill.find('a', {'class', 'NPI=a:illustrator'})
ill = illustrator.text
print("글: " + aut)
print("그림: " + ill)

################## 별점 ###################
f_stargrade = soup.find('p', {'class', 'grade_area'})
ff_stargrade = f_stargrade.find('em')
stargrade = ff_stargrade.text
print("별점: " + stargrade)

############### 관심, 연재일, 장르 #################
info = soup.find('p', {'class', 'info_book'})
f_like = info.find('span', {'id': 'concernCount'})
like = f_like.text
```

그림 16. 네이버 웹소설 작품 정보 마이닝 中

```
검색: 검은 늑대가 나를 부르면

--------------------

제목: 검은 늑대가 나를 부르면
글: 임혜
그림: 홍복
별점: 9.96
관심: 23,025
연재일: 화, 금
장르: 로판
연재 화수: 8
소개: 첫 번째 삶은 남편의 손에 죽임을 당했고, 두 번째 삶은 가족을 몰살한 남편 앞에서 자살했다. 하지만 그것은 마지막이 아닌 또 다른
  시작이었다. 과거로 돌아가 세 번째 삶을 살게 된 연우. 그녀 앞에 나타난 검은 늑대 휘타. 가족과 제 목숨을 지키고 싶었던 연우는 휘타
 에게 자신을 맡기게 되는데……. 사랑하는 사람을 위해 영혼마저 팔아버린 그들의 가슴 시리도록 아픈 사랑 이야기가 펼쳐집니다.

Process finished with exit code 0
```

그림 17. 네이버 웹소설 작품 정보 마이닝 결과

```
req = requests.get('https://series.naver.com/novel/detail.nhn?productNo=3200031')
source = req.text
soup = BeautifulSoup(source, 'html.parser')

f_container = soup.find('div', {'class', 'end_head'})
container = f_container.find('h2')
con = container.text.split(" [")[0]
print("제목: " + con)
count = container.find('em')
count1 = count.text.replace("- 총","")
count2 = count1.replace("화/","")
if '미완결' in count2:
    count3 = count2.replace("미완결","")
else:
    count3 = count2.replace("완결","")
print("화수: " + count3)

box = soup.find(id='container')
info_list = box.find('li', {'class', 'info_lst'})
li = info_list.findAll('li')
print("저자: " + li[0].find('a').text)
if "그림" in li[1].find('span'):
    n = 1
else:
    n = 0
if n == 1:
    print("그림: " + li[1].find('a').text)
print("장르: " + li[1+n].find('a').text)
print("출판사: " + li[2+n].find('a').text)
print("등급: " + li[3+n].text.replace("등급 ",""))
update = li[4+n].text.replace("업데이트 ","")
if n == 1:
    print("최근 업데이트:", update.replace("(완결)",""))
else:
    print("최근 업데이트:", update.replace("(미완결)", ""))
if n == 1:
    print("완결여부: 완결")
else:
    print("완결여부: 미완결")


f_scorearea = soup.find('div', {'class', 'score_area'})
ff_scorearea = f_scorearea.find('em')
scorearea = ff_scorearea.text
print("별점: " + scorearea)

f_dsc = soup.find('div', {'class', '_synopsis'})
print("소개: " + f_dsc.text)
```

그림 18. 네이버 시리즈온 작품 정보 크롤링 中

```
제목: 입술이 너무해
화수: 106
저자: 갓녀
그림: RM
장르: 로맨스
출판사: 와이엠북스
등급: 전체 이용가
최근 업데이트: 2018.09.21.
완결여부: 완결
별점: 9.8
소개: "키스하면 변해?" 남자가 되는 병에 걸린 지 7년, 그 남자와의 하룻밤은 서연을 다시 여자로 만들었다. 머리카락은 길어지고, 가슴은 봉긋해지고, 입술은 더욱더 새빨갛게 피어났다. "예쁜데요." "네?" "입술 예쁘네." 설렘 대폭발, 심쿵 주의, 치명적으로 섹시한 직진남의 꿀 떨어지는 막강 들이댐이 시작됐다. 운명에 얽힌 세계 최고 달달한 썸씽과 더 달달한 꼰적꼰적 연애! 예측불허 입술 로맨스.
```

그림 19. 네이버 시리즈온 작품 정보 크롤링 결과

다) 조아라 : Request, BeautifulSoup 이용

```
req=requests.get('http://www.joara.com/premium_new/book_intro.html?book_code=1360670')
source=req.text
soup=BeautifulSoup(source, 'html.parser')

# 제목
title_list=soup.select("#premium_content > div.latest > div.best_list_list_wrap > div.box_sty01 > div > div > div.txt_c_sty01 > str
title=title_list[0].text

# 저자
author_list=soup.select("#premium_content > div.latest > div.best_list_list_wrap > div.box_sty01 > div > div > div.txt_c_sty01 > st
author=author_list[0].text

# 총 연재화수 - 정수
book_count_list=soup.select("#premium_content > div.latest > div.best_list_list_wrap > div.box_sty01 > div > div > div.txt_c_sty01
book_count=book_count_list[0].text
book_count=book_count.replace("편","")
book_count=int(book_count)
```

그림 20. 조아라 작품 정보 크롤링 中

```
제목: 무한서고의 계약자!
장르: 판타지
저자: 준솔
표지 url: http://cf.joara.com/literature_file/20190418_121420.jpg_thumb.png
총 연재화수:  220
최근 업데이트일: 2019.06.04.
출간일: 19/04/03
완결 여부: 완결
조회수:  613218
추천수:  3965
관심도:  724
소개:
스킬북을 포함한 모든 서적들이 기록되어 있는 무한서고.
나는 그런 무한서고를 열람할 수 있는 권능을 얻게 되는데 · · · .
```

그림 21. 조아라 작품 정보 크롤링 결과

라) 문피아 : Request, BeautifulSoup 이용

```
    if(all(value==True for value in days.values())==True):
        period="매일"

    print("연재 주기 : " + period[:-1] + " 연재")



else:
    print("연재 주기 : 비주기적 연재")


age=box.find({'class','xui-icon xui-adult'})
if(age==None):
    print("연령등급 : 전체이용가")
else:
    print("연령등급 : " + age.text)

writer_dl=box.find('dl',{'class','meta-author meta'})
writer=dl.find('strong').text
print("작가 : " + writer)

etc_dl=box.findAll('dl',{'class','meta-etc meta'})
days_dl=etc_dl[0]
days_dd=days_dl.findAll('dd')
first_update=days_dd[0].text
print("작품등록일 : " + first_update)
recently_update=days_dd[1].text
print("작품등록일 : " + recently_update)

number_dl=etc_dl[1]
number_dd=number_dl.findAll('dd')

recommend=number_dd[1].text
print("추천수 : " + recommend)
enjoyer=number_dd[2].text
print("조회수 : " + enjoyer)
```

그림 22. 문피아 작품 정보 크롤링 中

```
독점여부 : 선독점
제목 : 영웅 - 삼국지
대체역사, 판타지
장르 : 대체역사,  판타지
연재 주기 : 월,화,수,목 연재
연령등급 : 전체이용가
작가 : 와이키키진
작품등록일 : 2016.10.13 11:00
작품등록일 : 2020.04.28 23:43
추천수 : 3,826,385
조회수 : 108,882
```

그림 23. 문피아 작품 정보 크롤링 결과

마) 리디북스 : Request, BeautifulSoup 이용

```
ridi_base = "https://ridibooks.com/search/?q="
ridi_work = input("작품 검색 : ")
print("\n-------------------------\n")
ridi_search = ridi_base + ridi_work.replace(" ", "+")

req = requests.get(ridi_search)
source = req.text
soup = BeautifulSoup(source, 'html.parser')

url_list = soup.select(
    "#page_search_result > div.result_list_wrapper > article > div.book_macro_wrapper.js_book_macro_wrapper > div")
check = 0
for i in url_list:
    if (i.text.find("연재") > 0):
        ridi_bookID = i.find(class_="book_thumbnail_wrapper")
        ridi_bookID = ridi_bookID.get("data-book_id_for_tracking")
        ridi_url = "https://ridibooks.com/books/" + ridi_bookID
        get_info(ridi_url, ridi_bookID)
        check = 1
        # print(book_id)
        break;

if check == 0:
    print("검색 결과가 없습니다.")
else:
    print("\n-------------------------\n검색 결과 출력 완료")
```

그림 24. 리디북스 작품 정보 크롤링 中

```
제목: 백작가의 망나니가 되었다
저자: 유려한
출판사: 도서출판 청어람
표지 url: //img.ridicdn.net/cover/875125819/xxlarge
총 연재화수:  568
완결 여부: 미완결
출간일: 2018.10.02.
최근 업데이트일: 2020.04.24
장르: 퓨전 판타지
별점: 4.8점
별점 참여자: 3,900명
관심도: 0
소개:
눈을 떠보니 소설 속이었다.
그것도 망나니로 유명한 백작가 도련님 몸으로.

하지만,
그렇다고 망나니가 될 순 없잖아?
```

그림 25. 리디북스 작품 정보 크롤링 결과

2) 소셜미디어 크롤링

가) 소셜미디어 [트위터] : API 사용

- 트위터의 특징은 불특정 다수가 독백을 하는 경향이 있다. 불특정 다수의 독백을 통해 작품의 언급수는 자연스레 현재 작품의 화제성을 대변해줄 수 있다. 이를 통해 트렌드 작품을 파악하기 좋을 것이라고 판단했다.

- 트위터 크롤링을 위한 API는 여러 가지가 존재한다. 트위터의 정식 API Tweepy는 키 신청을 하면 이용이 가능하지만 최근 7일간의 트윗만 확인이 가능하다. 이외에도 GetOldTweets3[8] , twitterscraper[9]와 같은 사설 API가 많이 존재하고 있다. 지난 약 1년 간의 데이터를 분석할 것이기 때문에 위의 API를 이용할 것이다.

- 크롤링에 사용한 API getoldtweet3는 기간 설정이 가능하여 설정한 기간에 설정한 키워드를 작성한 트윗을 수집했다.

```
import GetOldTweets3 as got
import datetime
import time
from random import uniform
from tqdm import tqdm_notebook
import pandas as pd

# 트윗 수집 기간
days_range = []

starting_date = input("트윗 수집 시작일 [ex. 양식 : 2010-07-23] : ")
ending_date = input("트윗 수집 마지막일 [ex. 양식 : 2015-07-23] : ")

start = datetime.datetime.strptime(starting_date, "%Y-%m-%d")
end = datetime.datetime.strptime(ending_date, "%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 ===\n".format(len(days_range)))

# 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")

# 작품 검색
search_key = input("작품 검색 #")
print()

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

---

[8] https://github.com/Jefferson-Henrique/GetOldTweets-python

[9] https://github.com/taspinar/twitterscraper

```
# 트윗 수집
print("트윗 수집 시작")
start_time = time.time()

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

print("트윗 수집 완료 [{0:0.2f} Minutes]".format((time.time() - start_time) / 60))
print("=== 수집 트윗 총 {}개 ===".format(len(tweet)))

# initialize
tweet_list = []

for index in tqdm_notebook(tweet):
    # 데이터 목록
    username = index.username
    link = index.permalink
    content = index.text
    tweet_date = index.date.strftime("%Y-%m-%d")

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

    time.sleep(uniform(1, 2))

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

# csv 파일 만들기
tw_df.to_csv("{}_tw.csv".format(search_key), index=False, encoding='utf-8')
print("저장 완료.\n")

# In[59]:


# 파일 확인하기
df_tweet = pd.read_csv("전지적독자시점_tw.csv".format(search_key))
df_tweet
```

그림 26. 기간에 따른 트위터 검색 정보 추출

<그림24> 코드 내용

트윗 수집 기준을 먼저 정의하고 트윗을 수집한다.

- got.manager.TweetCriteria().setQuerySearch(검색어).setSince(시작일).setUntil(마지막일).setMaxTweets(-1)
- got.manager.TweetManager.getTweets(트윗 수집 기준)

각 사이트와 페이지별로 링크를 재귀적으로 검색하여 필요한 정보를 수집한다.

```
=== 총 4일 간의 데이터 수집 ===

작품 검색 #전지적독자시점

트윗 수집 시작
트윗 수집 완료 [0.18 Minutes]
=== 수집 트윗 총 120개 ===

HBox(children=(IntProgress(value=0, max=120), HTML(value='')))

저장 완료.
```

```
In [59]:
# 파일 확인하기
df_tweet = pd.read_csv("전지적독자시점_tw.csv".format(search_key))
df_tweet
```

Out[59]:

	date	user_name	text	link
0	2020-05-04	autumn_teatime	洗清秋(徐浪 事吧? #전지적_독자_시점 #백작가의_망나니가_되었다 #人渣反派自救系統	https://twitter.com/autumn_teatime/status/1257...
1	2020-05-04	R_LA_	[전지적 독자 시점] 한수영_리아님 @kangela1117 어린 독자_R_LA_ 갓...	https://twitter.com/R_LA_/status/1257326622727...
2	2020-05-04	ripi32478	오소마즈상 사이퍼즈 럽프듀 러브앤프로듀서 방드림 을양사 페이트 패그오 페스나 fat...	https://twitter.com/ripi32478/status/125732364...
3	2020-05-04	kangela1117	전지적 독자 시점 - 5부 한수영 - 리아 (@kangela1117) 어린 독자 -...	https://twitter.com/kangela1117/status/1257322...
4	2020-05-04	cooing_o91o	유중혁씨 찾아요!! 중혁이 오면 알티 이벤? 깁티 뭐라도 올릴테니 상위 트윗 알티부...	https://twitter.com/cooing_o91o/status/1257321...
...	...	...	...	...
115	2020-05-01	silversphere_	전지적 독자 시점 108화 이 부분이 왼쪽 회차들 중에선 제일 슬펐던 기억ㅠ #도결쓰다	https://twitter.com/silversphere_/status/12560...
116	2020-05-01	silversphere_	전지적 독자 시점 107화 #도결쓰다	https://twitter.com/silversphere_/status/12560...
117	2020-05-01	silversphere_	전지적 독자 시점 97화 제일 좋아하는 탑3 안에 드는 대사들이에요 특히 서울! 유...	https://twitter.com/silversphere_/status/12560...
118	2020-05-01	silversphere_	전지적 독자 시점 76, 88화 #도결쓰다 #전독시	https://twitter.com/silversphere_/status/12560...
119	2020-05-01	wtf5cx	전지적 독자 시점 단행본 혐툰 외전 양장본 설정집 비나이다 비나이다 공식 덕밥 5합...	https://twitter.com/wtf5cx/status/125602080460...

120 rows × 4 columns

그림 27. 트위터 크롤링 결과

나) 소셜미디어 [인스타그램] : Selenium, XPATH 사용

- 인스타그램은 트위터와 같이 간단한 소감을 함께 적어 놓는 경우가 많다. 그렇기에 트렌드 지표를 알아보기 위한 언급수에 유용할 것이다.
- Selenium과 XPATH를 이용하여 마이닝을 진행하며 일정 이상 스크롤을 하면 로그인하라는 창이 나오도록 구성되어 있어 먼저 로그인을 하고 마지막까지 스크롤을 내린 후에 포스트를 클릭하여 각 포스트의 내용을 가져올 수 있도록 코드를 구성하였다.
- 로그인을 하고 난 뒤, 작품을 검색하면 해당 페이지 마지막까지 스크롤하고 포스트 링크를 추출하여 해당 포스트 링크에서 글의 내용, 작성 유저, 작성 날짜를 찾아 수집한다.

```
# 크롬드라이버 호출
driver = webdriver.Chrome("C:/project/chromedriver.exe")

# 로그인 [ID :          / PW :          ]
driver.get('https://www.instagram.com/accounts/login/?source=auth_switcher')

time.sleep(3)

id_input = driver.find_elements_by_css_selector('#react-root > section > main > div > article > div > div > div > form > div > div > l
id_input.send_keys(ig_id)
password_input = driver.find_elements_by_css_selector('#react-root > section > main > div > article > div > div > div > form > div > d
password_input.send_keys(ig_pwd)
password_input.submit()
time.sleep(3)
```

그림 28. 인스타그램 크롤링 中 Selenium을 통한 로그인

```
# 포스트 링크 추출
while True:
    html = driver.page_source
    soup = BeautifulSoup(html, "lxml")

    for link1 in soup.find_all(name="div",attrs={"class":"Nnq7C weEfm"}):
        for num in range(3):
            try:
                title = link1.select('a')[num]
                real = title.attrs['href']
                post_list.append(real)
            except:
                break

    last_height = driver.execute_script("return document.body.scrollHeight")
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
    time.sleep(SCROLL_PAUSE_TIME)
    new_height = driver.execute_script("return document.body.scrollHeight")
    if new_height == last_height:
        driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
        time.sleep(SCROLL_PAUSE_TIME)
        new_height = driver.execute_script("return document.body.scrollHeight")

        if new_height == last_height:
            break

        else:
            last_height = new_height
            continue

post_list = duplicate(post_list) # 중복포스트 제거
post_num = len(post_list) # 검색 결과 개수
print("총 "+str(post_num)+"개의 데이터.₩n")
```

그림 29. 인스타그램 크롤링 中 마지막 포스트까지 스크롤

	포스트 URL	유저ID	내용	작성 날짜
0	https://www.instagram.com/p/B-00H9PHaS-/	diana.pontin	리셋팅 레이디- 이셀라 에반스 낙서사실 읽은지는 좀 돼서 가물가물하기 때문에 외모묘...	2020-04-11
1	https://www.instagram.com/p/B7ivmF3Jr6a/	choinuri17	리셋팅 레이디- 이셀라 에반스 낙서사실 읽은지는 좀 돼서 가물가물하기 때문에 외모묘...	2020-01-20
2	https://www.instagram.com/p/ByxfvloFIYk/	gillllip	."이번에도 저와 결혼해 주시겠습니까?"."...이번이 두번째로 최악인 청혼이예요. ...	2019-06-16
3	https://www.instagram.com/p/B8BazVend_w/	raaaheen	2020.01. [완독] 리셋팅레이디회귀물 로판. 회귀도 로판도 새롭게 해석한 소설...	2020-02-01
4	https://www.instagram.com/p/B8ekVDkHyMx/	ilikehouse1	#리셋팅레이디 #리디북스 #로맨스판타지소설 #차서진 #회귀물 #피폐물 #완결 #완결...	2020-02-13
5	https://www.instagram.com/p/Bj8_8GIHh_y/	happy_hjj	#리셋팅레이디등장인물들마저도 깨알같이 취저.시온경. 밑바닥 인생에서 돈 되는 일이면...	2018-06-13
6	https://www.instagram.com/p/CAIvqHlgSO6/	u_u9oo	한줄 감상 : 불닭볶음면보다 매운 로맨스릴러....🔪. .『리셋팅 레이디』, 차서진...	2020-05-14
7	https://www.instagram.com/p/B73jRjhnQwX/	rooomr_0202	캐럿은 117세 생일을 맞이하여 살인마가 되기로 결심했다.#리셋팅레이디	2020-01-29

그림 30. 인스타그램 크롤링 결과 - 포스트 URL, 작성 유저, 내용, 날짜 추출

3) 블로그 크롤링

가) 블로그 [네이버 블로그] : Request, BeautifulSoup 이용

- 네이버 블로그와 티스토리는 기타 소셜미디어(트위터, 인스타그램)와 다르게 세세한 리뷰 중심의 글이 특징이다. 상대적으로 더 정확한 연관어를 찾을 수 있을 것 같다고 예상되기에 Aspect 마이닝에 사용할 예정이다.
- 네이버 블로그의 검색어에 따른 결과 크롤링을 해보면 제목과 블로그 링크를 가져올 수 있다. 무한정으로 많은 양의 검색 결과를 가져올 수는 없는데 이는 좋은 검색 결과를 위해 네이버가 1000건의 검색 결과만을 보여주고 있기 때문

이다.

- 네이버 검색창에 검색 결과를 통해 나온 글의 URL을 수집하고 검색을 통해 글의 제목과 작성자, 작성 일시, 게시물의 내용을 추출한다.

```
else:
    title = html.find('span', {'class', 'se-fs- se-ff-'})
    if title == None:
        title = html.find('div', {'class', 'se_textView'})
    print("제목 : " + title.text)

    info_area = html.find('div', {'class', 'blog_authorArea'})

    author = info_area.find('strong', {'class', 'ell'})
    print("작성자 : " + author.text)

    date = info_area.find('p', {'class', 'blog_date'})
    print("작성일시 : " + date.text)

    print("\n----------------------------------------\n" + "게시글\n" + "----------------------------------------\n")

    post_area = html.find('div', {'class', 'se-main-container'})
    if post_area != None:
        posts = post_area.findAll('p')
        for post in posts:
            print(post.text)

    else:
        post_area = html.findAll('div', {'class', 'se_component_wrap'})[1]
        posts = post_area.findAll('span')
        for post in posts:
            print(post.text)

    print("\n----------------------------------------\n")
```

그림 31. 네이버 블로그 크롤링 中 - 제목, 작성자, 작성 날짜, 게시글

```
작품이름 ->
닥터최태수
140
제목 : [리뷰] 닥터 최태수
작성자 : RM MC
작성일시 : 2020. 3. 4. 8:19

----------------------------------------
게시글
----------------------------------------

[닥터 최태수 리뷰] 닥터 최태수 읽으며 문득문득 드는 생각이극한 상황과 제한된 인력으로 살리기 위해어떻게 해야하나 하는 게임을 보는 것 같다.게임 보고 있지만 게이머가 펼치는 완벽하고 절제된 전개에 넋을 놓게 되어 그냥 빨려들어 몰입의 경지가 되고갑자기 정전이 되거나 광고가 나오는 것 같이한 편이 끝난다. ^^

----------------------------------------
```

그림 32. 네이버 블로그 크롤링 – 추출 결과 예시

나) 블로그 [티스토리, 다음 블로그] : Request, BeautifulSoup 이용

- 네이버 블로그와 마찬가지로 검색어 결과로 나온 글들의 링크를 통해 작성글 제목, 작성자, 작성 일시를 출력한다.
- 각 게시물마다 구성 방식이 달라 이에 따라 여러 버전의 크롤링 코드를 구성하였다.

```
main = html.find('div', {'class', 'coll_cont'})

total = html.find('span', {'class', 'txt_info'}).text.split("/ ")[1].replace("건", "").replace(",", "")
print(int(total))

ul = main.find('ul')
li_s = ul.findAll('li')
for li in li_s:
    url = li.find('a').attrs['href'].replace("://", "://m.")
    r = session.get(url, headers=header)
    html = bs4.BeautifulSoup(r.content, "html.parser")

    title_area = html.find('div', {'class', 'view_head'})

    title = title_area.find('h3', {'class', 'tit_view'})
    print("제목 : " + title.text.strip())

    info_area = html.find('div', {'class', 'info_writer'})

    author = info_area.find('span', {'class', 'txt_writer'})
    print("작성자 : " + author.text.strip())

    date = title_area.find('time', {'class', 'txt_time'})
    print("작성일시 : " + date.text)

    post_area = html.find('div', {'class', 'small'})

    posts = post_area.findAll('p')
    print("\n----------------------------------------\n" + "게시글\n" + "----------------------------------------\n")
    for post in posts:
        print(post.text)
    print("\n----------------------------------------\n")
```

그림 33. 티스토리 크롤링 中

```
total = html.find('span', {'class', 'txt_info'}).text.split("/ ")[1].replace("건", "").replace(",", "")
print(int(total))
ul = main.find('ul')
li_s = ul.findAll('li')
for li in li_s:
    url = li.find('a').attrs['href'].replace(".com/", ".com/m/")
    r = session.get(url, headers=header)
    html = bs4.BeautifulSoup(r.content, "html.parser")

    title_area = html.find('div', {'class', 'blogview_tit'})

    title = title_area.find('h2', {'class', 'tit_blogview'})
    print("제목 : " + title.text)

    author = title_area.find('span', {'class', 'txt_by'})
    print("작성자 : " + author.text)

    info_area = html.find('div', {'class', 'blogview_info'})
    date = info_area.find('time', {'class', 'txt_date'})
    print("작성일시 : " + date.text)

    post_area = html.find('div', {'class', 'blogview_content'})

    posts = post_area.findAll('p')
    print("\n----------------------------------------\n" + "게시글\n" + "----------------------------------------\n")
    for post in posts:
        print(post.text)
    print("\n----------------------------------------\n")
```

그림 34. 티스토리 크롤링 中

```
작품이름 ->
재혼 황후
301
제목 : 【역사/로맨스】 연록흔 재련 _ 한수영
작성자 : 흰고래
작성일시 : 2008.01.18 01:58

----------------------------------------
게시글
----------------------------------------

▶원    제 - 연록흔 재련
▶시 리 즈 - 1_5[완]
▶작    가 - 한수영
▶발 행 일 - 2007년 08월
▶발 행 처 - 마야
```

그림 35. 다음 블로그 크롤링 결과 예시

4) 커뮤니티 크롤링 : [디씨인사이드]

가) 커뮤니티 [디씨인사이드] : Request 사용

- 디씨인사이드는 갤러리 전체에서의 검색과 장르별 갤러리 모두 검색이 가능하다. 갤러리 전체를 통틀어 검색을 할 경우 최대 120페이지(한 페이지 당 25개)의 검색 결과를 보여준다.
- 디씨인사이드의 경우 리뷰보다는 언급 횟수에 따른 관심도 및 인지도 지표를 파악하기 좋은 특징을 가지고 있어 언급된 횟수를 카운트해 반영할 것이다.
- 다음은 디씨인사이드 중 장르소설 갤러리의 게시글 크롤링 코드이다. Base_url을 디씨인사이드로 설정하고, 각 갤러리가 가지고 있는 아이디를 받아 params 값으로 입력을 해 연결하는 방식이다. 장르소설 갤러리의 첫 페이지에 작성된 게시글의 제목, 작성자, 날짜, 조회수, 추천수, 내용을 추출한다.

```
for i in contents:
    print('-' * 15)

    title_tag = i.find('a')
    title = title_tag.text
    print("제목: ", title)

    writer_tag = i.find('td', class_='gall_writer ub-writer').find('span', class_='nickname')
    if writer_tag is not None:
        writer = writer_tag.text
        print("글쓴이: ", writer)

    else:
        print("글쓴이: ", "없음")

    ip_tag = i.find('td', class_='gall_writer ub-writer').find('span', class_='ip')
    if ip_tag is not None:
        ip = ip_tag.text
        print("ip: ", ip)

    date_tag = i.find('td', class_='gall_date')
    date_dict = date_tag.attrs
```

```
if len(date_dict) == 2:
    print("날짜: ", date_dict['title'])

else:
    print("날짜: ", date_tag.text)
    pass

views_tag = i.find('td', class_='gall_count')
views = views_tag.text
print("조회수: ", views)

recommend_tag = i.find('td', class_='gall_recommend')
recommend = recommend_tag.text
print("추천수: ", recommend)

link = i.find('td', class_='gall_tit ub-word')
href = link.find('a').attrs['href']
if href=="javascript:;":
    continue
content_url = "https://gall.dcinside.com/" + href
```

그림 36. 디씨인사이드 크롤링 中

```
---------------
제목:  예의를 좀 아는 그랜절 마스터 연예인은?
글쓴이:  없음
날짜:  20.05.12
조회수:  -
추천수:  -
---------------
제목:  장갤 통합 공지 0.8
글쓴이:  ㅇㅇ
날짜:  2020-02-01 21:20:41
조회수:  16590
추천수:  40
https://gall.dcinside.com//mgallery/board/view/?id=genrenovel&no=457640&page=1
[일반] 장갤 통합 공지 0.8


차단 및 삭제 대상글먹 분충소설 제목 말 안 하고 튀기소설과 전혀 무관한 떡밥이나 잡담 ( 막줄에 소설이야기 달아두는 것도 포함 )참독 어그로타 갤 옹 퍼오기특히 특정 소설 얘기 없는 처1,2년 , TS 빨글은 댓글로 호응해 주는 것도 차단함 ( 글쓰놈 7일 차단, 댓글 단 놈 3일 차단)특정 소설 얘기 없는 백합,보빔충 글절치색 섞인 글(차단)낚시 뇌절닉언급이런 소설 없냐? 등등의 제목으로 소설과 관련없는 짤 올리려고 쓰는 글 (차단)소설이랑 관련 없는 씹덕 만화(3일 차단)--- 당분간 소설 관련 없는 빨글들은 최소 1일 차단 하겠음 ------ 당분간 연독률,구매수 언급하는 글들 최소 1일 차단 하겠음 ---작가 홍보는 15화 이상 쓴 글만해당 글 제외 작가 티 내지 마셈차단 단어,아이피http s://gall.dcinside.com/mgallery/board/view/?id=genrenovel&no;=698098

---------------
제목:  장마갤 신문고
글쓴이:  Qwer1234m
날짜:  2020-04-25 20:24:45
조회수:  4111
추천수:  6
https://gall.dcinside.com//mgallery/board/view/?id=genrenovel&no=708312&page=1
[일반] 장마갤 신문고앱에서 작성


완장 호출벨입니다  댓글달 때 링크 뒤에 글자 붙이지 말아주십쇼  기본적으로 새벽반이긴 한데 새벽 아닐 때 사용해도 상관 없습니다  좋은 예)  https://m.dcinside.com/board/genrenovel/708312 삭제 좀  틀린 예) https://m.dcinside.com/board/genrenovel/708312삭제 좀
```

그림 37. 디씨인사이드 크롤링 결과

나) 커뮤니티 [인스티즈] : Selenium과 XPATH를 이용한 마이닝

- 커뮤니티 인스티즈는 게시글에 작품의 내용이 포함되어 있다면 스포 방지 팝업창이 나오도록 설정되어있다.
- 어떤 게시물의 경우, 검색 결과로는 나오지만 로그인을 한 회원만 게시물을 확인할 수 있도록 되어 있어 이럴 경우는 제외했다.
- 댓글 작성자의 이름이나 어떤 게시글의 연관 게시글 제목에 검색 키워드가 있다면 검색 결과로 나오기도 하여 이런 케이스들에 대한 예외 케이스문을 작성하였다.

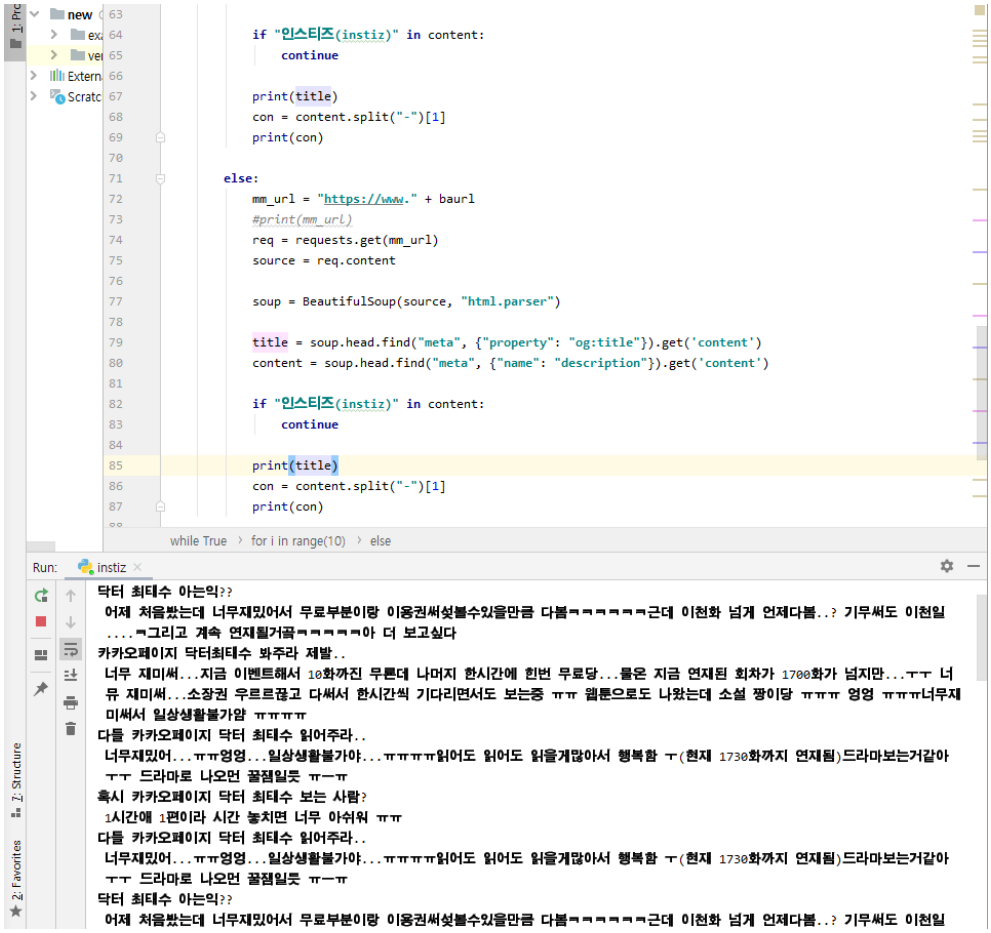
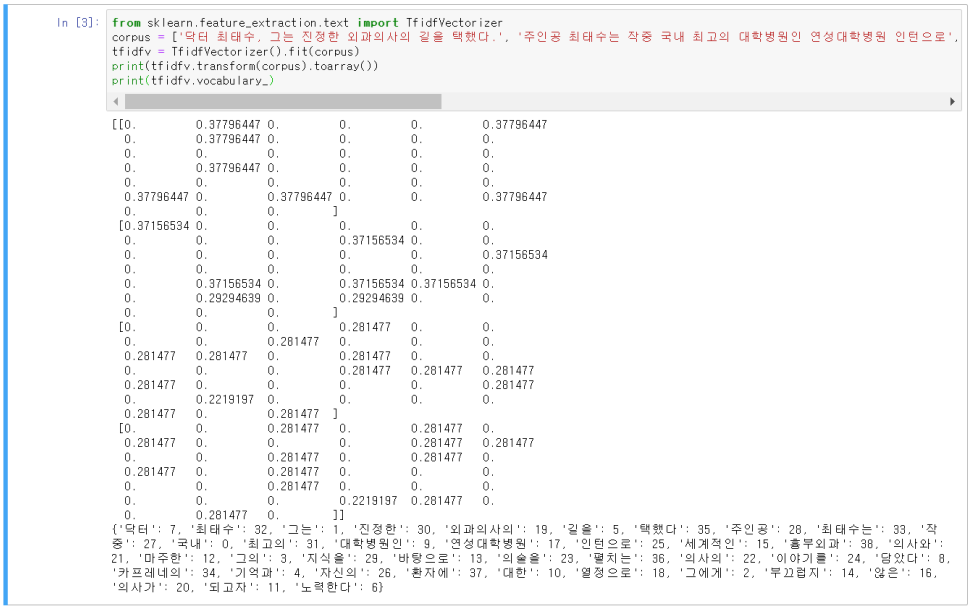


그림 38. 인스티즈 크롤링 코드 中 / 크롤링 결과

나. 데이터 분석을 위한 전처리과정과 분석 코드 제작

1) 트렌드 분석 : TF-IDF를 활용한 문장 내 단어 별 가중치 벡터화



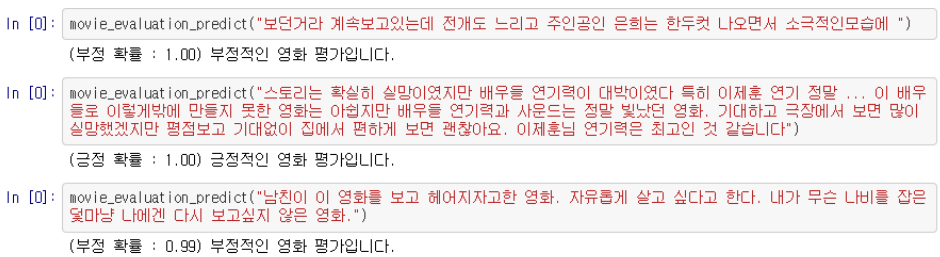
```
In [3]: from sklearn.feature_extraction.text import TfidfVectorizer
corpus = ['닥터 최태수, 그는 진정한 외과의사의 길을 택했다.', '주인공 최태수는 작중 국내 최고의 대학병원인 연성대학병원 인턴으로',
tfidfv = TfidfVectorizer().fit(corpus)
print(tfidfv.transform(corpus).toarray())
print(tfidfv.vocabulary_)

[[0.         0.37796447 0.         0.         0.         0.37796447
  0.         0.37796447 0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.37796447 0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.37796447 0.         0.37796447 0.         0.         0.37796447
  0.         0.         0.        ]
 [0.37156534 0.         0.         0.         0.         0.
  0.         0.         0.         0.37156534 0.         0.
  0.         0.         0.         0.         0.         0.37156534
  0.         0.         0.         0.         0.         0.
  0.         0.37156534 0.         0.37156534 0.37156534 0.
  0.         0.29294639 0.         0.29294639 0.         0.
  0.         0.         0.        ]
 [0.         0.         0.         0.281477   0.         0.
  0.         0.         0.281477   0.         0.         0.
  0.281477   0.281477   0.         0.281477   0.         0.
  0.         0.         0.         0.281477   0.281477   0.281477
  0.281477   0.         0.         0.         0.         0.281477
  0.         0.2219197  0.         0.         0.         0.
  0.281477   0.         0.281477  ]
 [0.         0.         0.281477   0.         0.281477   0.
  0.281477   0.         0.         0.         0.281477   0.281477
  0.         0.         0.281477   0.         0.281477   0.
  0.281477   0.         0.281477   0.         0.         0.
  0.         0.         0.281477   0.         0.         0.
  0.         0.         0.         0.2219197  0.281477   0.
  0.         0.281477   0.         0.        ]]
{'닥터': 7, '최태수': 32, '그는': 1, '진정한': 30, '외과의사의': 19, '길을': 5, '택했다': 35, '주인공': 28, '최태수는': 33, '작
중': 27, '국내': 0, '최고의': 31, '대학병원인': 9, '연성대학병원': 17, '인턴으로': 25, '세계적인': 15, '흉부외과': 38, '의사와':
21, '마주한': 12, '그의': 3, '지식을': 29, '바탕으로': 13, '의술을': 23, '펼치는': 36, '의사의': 22, '이야기를': 24, '담았다': 8,
'카프레네의': 34, '기억과': 4, '자신의': 26, '환자에': 37, '대한': 10, '열정으로': 18, '그에게': 2, '부끄럽지': 14, '않은': 16,
'의사가': 20, '되고자': 11, '노력한다': 6}
```

그림 39. TF-IDF 코드 예시

<그림 39>의 결과 해석 : 위의 리스트는 각 문서에서 단어의 출현을 나타낸 것으로 0은 출현하지 않은 것을 의미한다. 마지막 줄의 딕셔너리는 각 단어들이 몇 번 등장했는지를 나타낸다.

2) 감성 분석 – KoBERT를 이용한 문장별 긍부정 판단

오피니언 마이닝(감성 분석)을 이용하여 리뷰의 긍부정을 판단하고 평가에 대한 점수를 기간에 대한 그래프로 표현하여 시간이 지남에 따라 작품의 평가 변화 양상을 확인한다.



```
In [0]: movie_evaluation_predict("보던거라 계속보고있는데 전개도 느리고 주인공인 은희는 한두컷 나오면서 소극적인모습에 ")
        (부정 확률 : 1.00) 부정적인 영화 평가입니다.

In [0]: movie_evaluation_predict("스토리는 확실히 실망이였지만 배우들 연기력이 대박이였다 특히 이제훈 연기 정말 ... 이 배우
        들로 이렇게밖에 만들지 못한 영화는 아쉽지만 배우들 연기력과 사운드는 정말 빛났던 영화. 기대하고 극장에서 보면 많이
        실망했겠지만 평점보고 기대없이 집에서 편하게 보면 괜찮아요. 이제훈님 연기력은 최고인 것 같습니다")
        (긍정 확률 : 1.00) 긍정적인 영화 평가입니다.

In [0]: movie_evaluation_predict("남친이 이 영화를 보고 헤어지자고한 영화. 자유롭게 살고 싶다고 한다. 내가 무슨 나비를 잡은
        덫마냥 나에겐 다시 보고싶지 않은 영화.")
        (부정 확률 : 0.99) 부정적인 영화 평가입니다.
```

그림 40. KoBERT 기반 감성 분석 결과

3) Aspect Mining – 군집분석, 구문분석 및 용언 분석을 활용한 속성 추출

Aspect Mining 이란 속성(Aspect)을 나타내는 특정 키워드와 관련된 원소(Element), 즉 속성 값을 추출하는 기법이다. 웹소설에서 주요 요소인 '주인공', '스토리', '분위기'를 키워드라고 한다면 속성 값은 주인공은 '밝다', 스토리는 '진부하다', 분위기는 '칙칙하다'의 형용사와 동사들이다.

가) 군집분석

주어진 데이터들의 특성을 분석하여 이와 유사한 것을 묶는 것이다. word2vec을 이용하여 작품의 다양한 요소에 대해 의존적인 연결 형용사나 동사를 추출한다.

```
In [8]: import csv
from konlpy.tag import Okt
from gensim.models import word2vec

#네이버 영화 코퍼스를 읽는다.
f = open('ratings_train.txt', 'r', encoding='utf-8')
rdr = csv.reader(f, delimiter='₩t')
rdw = list(rdr)
f.close()

#트위터 형태소 분석기를 로드한다. Twiter가 KoNLPy v0.4.5 부터 Okt로 변경 되었다.
twitter = Okt()

#텍스트를 한줄씩 처리합니다.
result = []
for line in rdw:
#형태소 분석하기, 단어 기본형 사용
    malist = twitter.pos( line[1], norm=True, stem=True)
    r = []
    for word in malist:
     #Josa", "Eomi", "Punctuation" 는 제외하고 처리
        if not word[1] in ["Josa","Eomi","Punctuation"]:
            r.append(word[0])
 #형태소 사이에 공백 ' ' 을 넣습니다. 그리고 양쪽 공백을 지웁니다.
    rl = (" ".join(r)).strip()
    result.append(rl)
 #print(rl)

#형태소들을 별도의 파일로 저장 합니다.
with open("NaverMovie.nlp",'w', encoding='utf-8') as fp:
    fp.write("₩n".join(result))

#Word2Vec 모델 만들기
wData = word2vec.LineSentence("NaverMovie.nlp")
wModel =word2vec.Word2Vec(wData, size=200, window=10, hs=1, min_count=2, sg=1)
wModel.save("NaverMovie.model")
```

```
In [10]: from gensim.models import word2vec

model = word2vec.Word2Vec.load("NaverMovie.model")

print(model.most_similar(positive=["주인공"]))

[('한도', 0.7152603864669), ('안문숙', 0.7000996470451355), ('shiT', 0.6950535774230957), ('화서', 0.6600076556205), ('땡잡다', 0.6486603021621704), ('기선', 0.6467784643173218), ('성취향', 0.6442456245422363), ('히스테릭', 0.6414129734039307), ('오락가락', 0.6256510019302368), ('비합리적', 0.6216648817062378)]
```

그림 41. Word2vec 을 이용한 유사도 결과 예시

Vol.12에서의 변경점은 word2vec분석에서 명사 뿐 아니라, 조사와 어미, 구두점을 제외하고 모든 단어를 사용하여 분석을 실시하였을 때, 명사를 중심으로 형용사나 동사를 추출할 수 있음을 확인하고, word2vec만으로 속성에 대한 값을 추출할 수 있겠다는 판단이 들어 다)항목과 함께 aspect 마이닝의 한 방향성으로 생각하고 있다.

나) 동시 출현 기반 연관어 분석

문장 내 의존 관계를 추출하는 분석 방식인 구문분석을 활용할 수도 있지만, 해당 프로젝트에서는 전체 문장 '구조'를 분석할 필요성이 없으므로 단어를 중심으로 어떤 단어가 있는지 파악하는 동시 출현 기반 연관어 분석 기법을 활용한다.

```
pos_review=(glob.glob("C:₩₩Users₩₩shung₩₩Desktop₩₩pos₩₩*.txt"))[0:100]

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

tokenizer = RegexpTokenizer('[₩w]+')

stop_words = stopwords.words('english')

count = {}  #동시출현 빈도가 저장될 dict
for line in lines_pos:
    words =  line.lower()
    tokens = tokenizer.tokenize(words)
    stopped_tokens = [i for i in list(set(tokens)) if not i in stop_words+["br"]]
    stopped_tokens2 = [i for i in stopped_tokens if len(i)>1]
    for i, a in enumerate(stopped_tokens2):
        for b in stopped_tokens2[i+1:]:
            if a>b:
                count[b, a] = count.get((b, a),0) + 1
            else :
                count[a, b] = count.get((a, b),0) + 1

df=pd.DataFrame.from_dict(count, orient='index')
```

```
In [7]:
list1=[]
for i in range(len(df)):
    list1.append([df.index[i][0],df.index[i][1],df[0][i]])
```

```
In [8]:
df2=pd.DataFrame(list1, columns=["term1","term2","freq"])
```

```
In [9]:
df3=df2.sort_values(by=['freq'],ascending=False)
```

```
In [10]:
df3_pos=df3.reset_index(drop=True)
df3_pos.head(20)
```

Out [10]:

	term1	term2	freq
0	film	one	41
1	film	like	34
2	like	one	34
3	film	movie	29
4	film	story	27
5	like	movie	26

그림 42. 동시 출현 연관어 결과 예시

```
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", "밀리"))
#두 단어의 유사도 -> 1에 근접할 수록 서로 상관관계가 있다.
print(model.most_similar("기묘하다"))

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

```
0.5386249
0.32553965
[('개꿀', 0.6190359592437744), ('역시', 0.6100665926933289), ('뭥', 0.6065816283226013), ('기점', 0.6044149398803711), ('이따', 0.6040357947349548), ('앜다', 0.6020412445068359), ('이러다가', 0.6007715463638306), ('거양', 0.6000221967697144), ('도리', 0.5988008975982666), ('훅', 0.5978043079376221)]
[('개꿀', 0.6190359592437744), ('역시', 0.6100665926933289), ('뭥', 0.6065816283226013), ('기점', 0.6044149398803711), ('이따', 0.6040357947349548), ('앜다', 0.6020412445068359), ('이러다가', 0.6007715463638306), ('거양', 0.6000221967697144), ('도리', 0.5988008975982666), ('훅', 0.5978043079376221)]
[('ㅠㅠㅠ', 0.4821982681751251), ('엽', 0.46736279129981995), ('좋아하다', 0.45916640758514404), ('추강', 0.44668009877204895), ('렌즈', 0.4345983564853668)]
```

그림 43. 어근 변경 후 유사성 판단 예시

다) 의존 구문 분석 : koalanlp의 parser를 활용

'나)항목'의 동시 출현 연관어 분석을 수행한 이유는, 특성 '속성'에 대한 값(원소)가 어떤 형용사나 동사를 가지고 있는지 알아보기 위함이다. 이를 수행하는 다른 방식으로 의존 구문 분석이 있다. 이것은, 해당 문장의 구조를 분석하는 방식으로

속성을 꾸미는 형용사나 동사등을 문장 구조 속에서 찾을 수 있다.

```
In [1]: from koalanlp.Util import initialize, finalize
        from koalanlp.proc import Parser
        from koalanlp import API

        initialize(KKMA='LATEST')  #: HNN=2.0.4, ETRI=2.0.4

        parser = Parser(API.KKMA)

        while True:
            text = input("분석할 문장을 입력하세요>> ").strip()

            if len(text) == 0:
                break

            sentences = parser(text)

            for sent in sentences:
                print("===== Sentence =====")
                print(sent.singleLineString())
                print("# Dependency Parse result")

                dependencies = sent.getDependencies()
                if len(dependencies) > 0:
                    for edge in dependencies:
                        print("[%s]는 [%s]의 %s-%s" % (edge.getDependent().getSurface(),
                                                      edge.getGovernor().getSurface() if edge.getGovernor() is n
                                                      str(edge.getType()),
                                                      str(edge.getDepType())))
                else:
                    print("(Unexpected) NULL!")

        finalize()
```

```
분석할 문장을 입력하세요>> 그 웨이터는 너무나도 불친절해서 기분이 나빴습니다.
===== Sentence =====
그/MM 웨이터/NNG+는/JX 너무나/MAG+도/JX 불친절/NNG+하/XSV+어서/EC 기분/NNG+이/JKS 나쁘/VA+았/EP+습니다/
EF+./SF
# Dependency Parse result
[나빴습니다.]는 [ROOT]의 X-UNDEF
[불친절해서]는 [나빴습니다.]의 VP-CNJ
[웨이터는]는 [불친절해서]의 NP-AJT
[그]는 [웨이터는]의 DP-MOD
[너무나도]는 [불친절해서]의 NP-AJT
[기분이]는 [나빴습니다.]의 NP-SBJ
분석할 문장을 입력하세요>> 주인공은 밝고 괘활한 성격입니다.
===== Sentence =====
주인공/NNG+은/JX 밝/VA+고/EC 괘활/XR+하/XSA+ㄴ/ETM 성격/NNG+이/VCP+ㅂ니다/EF+./SF
# Dependency Parse result
[성격입니다.]는 [ROOT]의 X-UNDEF
[괘활한]는 [성격입니다.]의 DP-MOD
[밝고]는 [괘활한]의 VP-CNJ
[주인공은]는 [밝고]의 NP-AJT
```

그림 44. 의존 구문 분석 결과 예시

이 결과를 통해서 알 수 있는 부분은, 동시 출현 연관어 분석의 경우 n-gram을 미리 고려하여, window size를 조절하여 해당 단어를 중심으로 모든 단어를 count하기 때문에 해당 속성을 꾸며주는 모든 값을 대부분 추출할 수 있는 반면, 의존 구문의 경우 의존 관계에 의하여 해당 속성을 꾸며주는 단어를 하나만 추출할 수 있다는 단점이 있음을 확인하였다. 따라서, 이내영 신라대학교 교수의 '집단따돌림' 관련 온라인 기사 빅데이터 의미연결망 분석 논문을 참조하여 동시출현 연관어 분석을 이용하는 방법을 고려하고 있다.

라) 용언 분석기(Lemmatizer) : 말뭉치를 이용한 한국어 용언 분석기 Lemmatizer

한국어는 어미의 변용으로 같은 동사나 형용사가 '먹다', '먹고', '먹니', '아름다우니', '아름다워' 등과 같이 형태가 변형되어 문장에서 사용. 여기서 변형되지 않는 것을 어간, 변하는 부분을 어미라 칭한다. 문장에서 쓰인 단어들을 원형으로 바꿔주기 위해 용언 분석기를 사용한다.

Aspect Extraction에서 특정 속성과 연관된 동사 혹은 형용사를 찾기 위해 문장 속 단어의 형태를 원형으로 복원시켜야 하는데 이때 용언 분석기를 사용한다.

```
lemmatizer.lemmatize('차가우니까')
```

```
[('차갑다', 'Adjective')]
```

그림 45. 용언 분석기를 이용한 품사의 원형 복원

다. 수집한 데이터와 분석 결과를 저장할 DB 제작

1) Logical-ERD

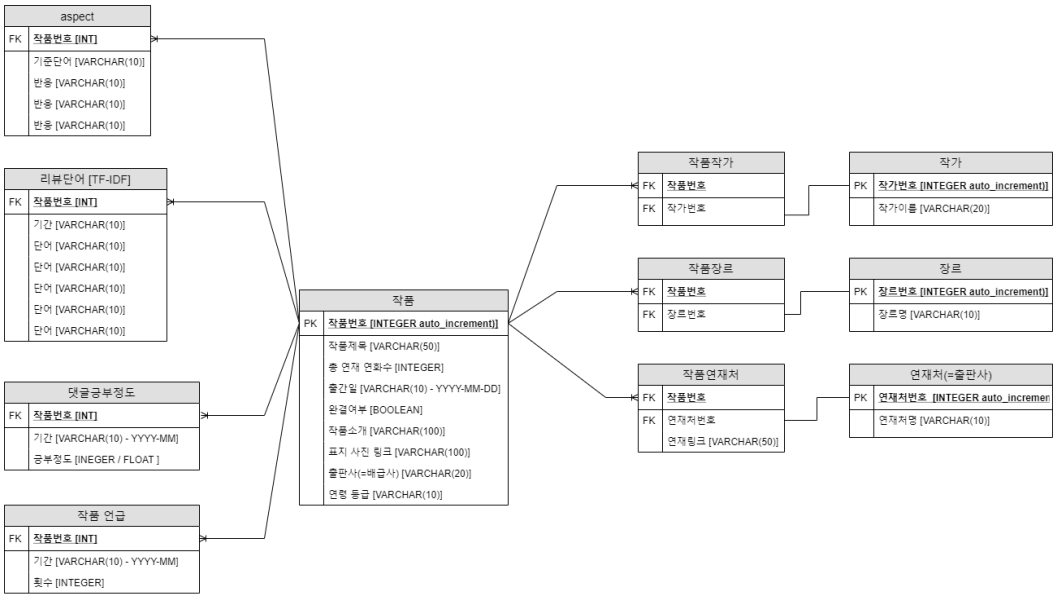


그림 46. Logical-ERD

[작품 테이블] 작품의 기본 정보가 저장되는 테이블로 작품이 삽입될 때마다 작품 번호를 자동으로 등록시켜주고 작품제목, 총 연재 화수, 출간일, 완결이며, 작품 소개, 표지 사진 링크, 연령 등급, 출판사를 속성으로 가진다.

[저자 테이블] 기본키 지정을 위한 작가 번호와 작가 이름을 속성으로 가진다.

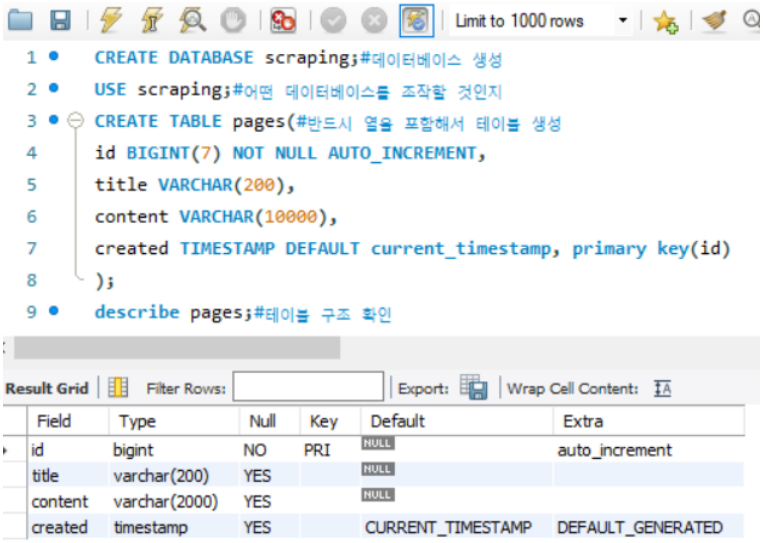
[작품작가 테이블] 작품과 작가를 연결하기 위한 테이블로 [작품 테이블]의 작품 번호와 [작가 테이블]의 작가 번호를 외래키로 가지는 테이블이다. [작가 테이블]의 경우 다수의 작가가 한 작품의 집필하는 경우를 고려하여 따로 테이블을 만들었다. 이와 같은 이유로 [장르 테이블], [연재처 테이블]도 별도로 테이블을 구성하였다. [연재처 테이블]의 경우 작품과 연재처럼 연결하는 [작품 연재처 테이블]에 추가로 해당 연재처럼 작품 링크 속성을 가진다.

분석 결과의 경우 총 4가지의 분석을 하기 때문에 4개의 테이블로 구성되어 있다. [작품 언급 테이블]은 소셜 미디어에서 작품이 얼마나 언급되는지 그 결과를 저장하는 릴레이션으로 기간과 횟수를 속성으로 가진다. [댓글긍부정도 테이블]은 댓글과 리뷰의 긍부정 양상을 저장하는 테이블로 기간과 긍부정도 비율을 속성으로 가진다. [리뷰 단어 테이블]은 리뷰 속 빈출 단어를 저장하는 테이블로

단어와 TF-IDF 결과를 속성으로 가진다. [aspect 테이블]은 기준 속성 단어와 그에 대한 반응 키워드는 속성으로 가진다.

2) DB 구축

MySQL 서버에 접속하여 데이터베이스를 생성한다.



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

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
id	bigint	NO	PRI	NULL	auto_increment
title	varchar(200)	YES		NULL	
content	varchar(2000)	YES		NULL	
created	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

그림 47. 테이블 구축 예시

```
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;
```

그림 48. DML 명령어 활용

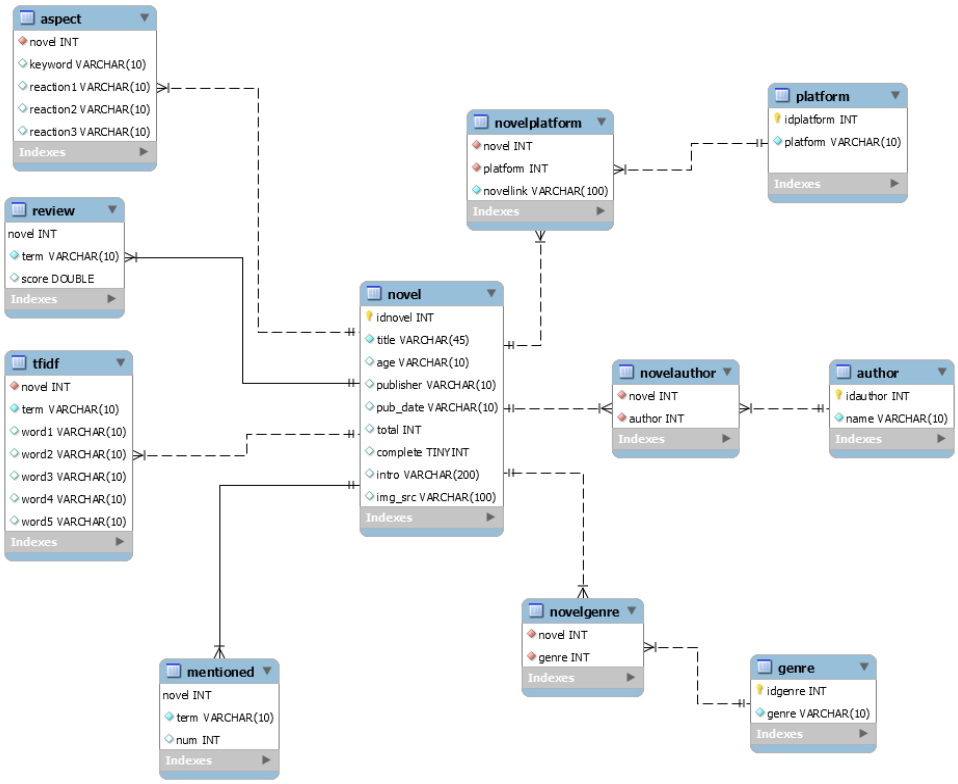


그림 49. Logical-ERD 기반 실제 테이블 구성(객체의 속성 표현)

라. 사용자 편의성을 고려한 UI를 구성한 웹어플리케이션 제작

1) Web에 대한 이해

웹 서비스는 현재 우리가 사용하는 모든 서비스의 기본이 된다. 이 웹 서비스를 구축할 때 주로 insert, select, update, delete 의 단계를 고려하여 제작하게 된다. 이 각 단계는 MVC 패턴에 의해 구축이 되는데 이는 Model View Controller 의 줄임말로, Model 은 웹에 구축되는 DB(Data Base)를 처리하는 곳, View 는 HTML 을 화면에 형성하는 것, Controller 는 클라이언트와 서버의 상황을 제어 관리하는 것을 의미한다. 이는 프로그램 디자인 메커니즘 중 하나이며 가장 기본적이다.

최근에는 클라이언트를 Front_end, 서버를 Back_end 라고도 부른다. 서버와 클라이언트는 기본적으로 위와 같은 주고받는 식의 메커니즘을 따른다. 클라이언트가 서버에 웹을 요청하면, 서버에서는 HTML 로 응답한다. 클라이언트에서 요청한 내용은 서버의 자원 (DB)를 바탕으로 controller 가 처리하게 된다. 클라이언트는 HTML 을 다운로드 받아 사용자의 화면에 응답 받은 내용을 출력해 준다. 웹 브라우저는 HTML 을 화면에 띄워주는 역할을 하게 된다.

2) Flask에 대한 이해

Flask 는 웹 어플리케이션을 만드는 프레임워크로, python 에서 대표적으로 사용되는 웹 프레임 워크 중 하나인 'django'와 같은 라이브러리다. Flask 는 기본적인 구조가 명확해 짧은 코드로도 웹 어플리케이션을 만들 수 있는 장점이 있다.

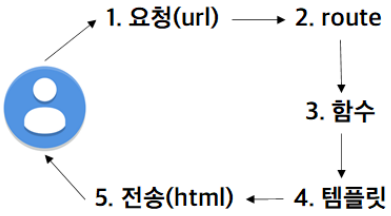


그림 50. Flask의 실행 구조

Flask 의 구조는 다음과 같다. 사용자가 url 로 접속하면 route 를 통해 접속한 url 에 맞는 함수를 호출하고, 그 함수의 결과가 템플릿을 통해 html 로써 사용자에게 보이게 한다. 홈페이지를 구성하는 코드로는 html 도, CSS 도 사용 가능하다.

III. 프로젝트의 활용 및 기여

1. 프로젝트 결과물의 활용

- 즉각적인 피드백이 필요한 문화 산업에서 소셜미디어와 커뮤니티 같은 독자층의 실시간 반응을 통합적으로 확인 가능함으로써 앞으로의 홍보, 제작, 투자 방향 선택에 도움이 되는 지표가 될 것이다.

- 구매자로 하여금 다양한 웹사이트에서의 반응을 한 번에 확인하게 됨으로써 사이트의 성향과 무관하게 리뷰 자체의 신뢰성이 높아져 구매력이 증가할 것이다. 이로 인해 웹소설을 제공하는 플랫폼과, 출판사, 작가에게 방향성에 대한 도움이 될 것이다.

- 제작하고자 하는 웹사이트의 실제 이용고객이 분석 결과를 보고 작품을 감상했을 때 실제로 독자들에게 호평을 받는 작품이라는 것이 입증된다면 웹사이트를 통해 출판사와 작가에 대한 신뢰가 높아짐으로써 차기 출시될 출판사 또는 작가의 신작의 구매력도 증가할 것이다.

2. 프로젝트 결과물의 기여

가. 기업적 측면

- 즉각적인 피드백이 필요한 문화 산업에서 소셜미디어와 커뮤니티 같은 독자층의 실시간 반응이 보이는 곳의 리뷰를 통합적으로 확인 가능함으로써 앞으로의 홍보, 제작, 투자 방향 선택에 도움이 되는 지표가 될 것이다.

- 긍정적인 리뷰를 많이 보이는 작품의 경우, 구매자로 하여금 다양한 웹사이트의 공통적인 반응을 가진 리뷰를 확인 가능함으로써 리뷰의 신뢰성이 높아져 구매력이 증가할 것이다.

- 제작하고자 하는 웹사이트의 실제 이용고객이 분석 결과를 보고 작품을 감상했을 때 실제로 좋은 작품이라면 웹사이트의 신뢰성이 증가하여 차기 출시될 출판사 또는 작가의 신작의 구매력도 증가할 것이다.

나. 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 분석하여 사용자에게 제공하기 때문에 사용자가 선호하는 양상의 작품을 기호에 맞춰 선택하기 쉽다.

- 현재 작품에 대한 주요 평가가 어떻게 되는지 시각적으로 확인 가능하여 작품 평가에 대한 신뢰성을 사용자가 직접 판단할 수 있다.

- 비슷한 성격 또는 조건의 작품을 추천받을 수 있다.

IV. 프로젝트의 향후 계획

1. 수행 일정

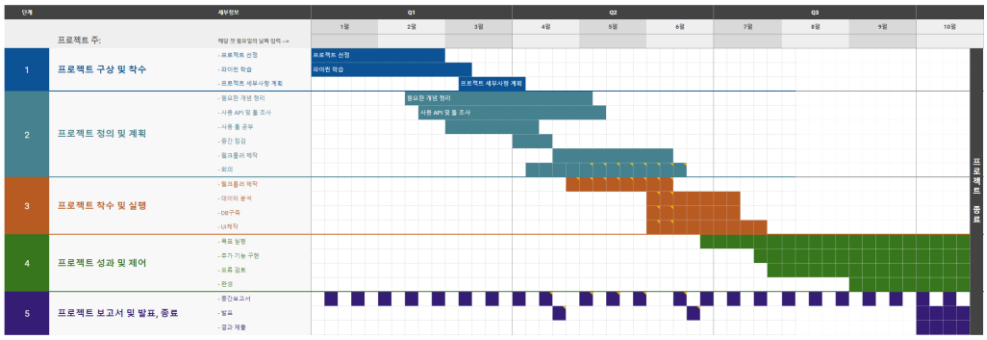


그림 51. 프로젝트 진행 계획

- 크롤러 제작 과정은 마무리되었고, 크롤링을 실시하면서 특정 웹사이트에서 동작하지 않는지, HTML 코드 구성이 바뀌어서 크롤링이 되지 않는지 등을 파악하여 지속적으로 수정 보완하고 있습니다.

- 다양한 유형의 데이터 분석을 어떤 방식을 활용하여 코드를 구성할지와 어떤 DB를 활용할지 등의 필요한 조사와 구조적 설계를 끝마치고 모듈별로 코드 구성중에 있습니다.

- Front-End의 반응형 웹사이트 구현을 위한 조사와 디자인 구성 회의를 마치고 현재 UI 제작 진행 중입니다.

2. 개선 방안

- <별첨 2, 3 항목>과 같이 분석에 있어서 정확도를 올리는 다양한 방법을 지속적으로 수행해봐야 한다.

- DB구축 시 불필요한 정보나, 과도한 정보가 입력되지 않도록 Logical-ERD를 수정해가며 DB구축을 완성해야 한다.

- 데이터 베이스의 데이터 최신화 간격, 방법 등에 대해서 논의해볼 필요성이 있다.
웹페이지 구축 시 접속 시간 또는 데이터 로딩 시작의 지연이 과도하게 발생하지 않도록 구축해야 할 필요성이 있다.

-> 이 모든 방안은 웹 애플리케이션을 사용하는 사용자들로 하여금 좀 더 정확하고 신속하게 정보를 확인할 수 있도록 하는 작업이다.

V. 별첨

1. KoNLPy- Kkma, Hannanum

- KoNLPy는 자바 기반의 형태소 분석기를 파이썬에서 사용할 수 있도록 해주는 라이브러리이다.

- 형태소 분석기 : Hannanum (한나눔), Kkma (꼬꼬마), Komoran(코모란), Twitter(트위터, Okt), Mecab(메캅)

- Kkma나 Hannanum 모듈을 이용하여, 해당 모듈에 맞추어 입력된 문자열에서 단어별 품사를 분별할 수 있고 이를 토대로 각 단어의 빈도수나 명사에 따른 의존 형용사 및 동사를 추출하거나 작품과 함께 가장 많이 언급되고 있는 단어들을 찾을 수 있다.

- 앞으로 사용할 자연어 처리를 이용한 분석을 하기 위해 형태소를 구분하는 것과 같은 한국어 처리를 위해 해당 라이브러리가 사용된다.

```
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
```

그림 52. KoNLPy – Hannanum 예시

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

word_dic={}

for i in dff["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, '봤다': 71, '년': 9, '또': 119, '헛소리': 4, '어딨다고': 1, '그래': 21, '일정': 2, '팔로워': 2, '도달': 2, '시': 24, '개장': 4, '확정': 3, '됩니다': 13, '너': 102, '그': 339, '소문': 7, '들은': 5, '적': 23, '있어': 36, '아가': 5, '발음': 4, '졸귀': 2, '길': 15, '모어': 7, '걸스': 26, '임': 118, '낫': 73, '오케이': 80, '브루클린': 68, '나인': 141, '글리': 10, '엘리스': 1, '칠뉴블': 1, '있는데': 63, '더스틴': 21, '엄마': 64, '돌': 51, '나와서': 20, '괴리감': 1, '들어요': 2, '본다': 41, '나탈': 1, '리아': 1, '커플': 7, '나오는': 85, '뒤': 33, '실제': 14, '사귀게': 1, '뎀': 33, '취향': 76, '비슷하면': 1, '아메리칸': 14, '반달': 1, '리즈': 6, '봐줘': 21, '좋아하면': 31, '반': 16, '봐': 222, '더': 207, '풀리': 12, '티션': 11, '종이': 220, '집': 265, '오늬': 80, '불': 81, '오줌': 3, '지리': 3, '재밌음': 51, '보고싶은데': 25, '진도': 6, '나감': 1, '클루': 15, '리스': 17, '뭐': 209, '레이스': 170, '내일': 33, '만나': 4, '보실수': 3, '보신': 28, '분': 100, '있나용': 1, '다음': 61, '불까': 78, '말까': 3, '고민': 33, '중': 297, '봤더니': 12, '가발': 2, '바느질': 1, '끝났서': 1, '용': 65, '와': 93, '왜': 196, '레오': 3, '색': 5, '국내': 3, '팔지': 1, '않아서': 8, '이런': 66, '별짓': 1, '하게': 29, '만드': 1, '컷팅': 1, '해야하는데': 5, '자야': 5, '해서': 113, '일단': 81, '굴러다니면': 1, '인형': 24, '머리': 175, '씩워': 1, '봄': 149, '똑같다': 1, '게이': 10, '튼이다': 1, '보시나요': 10, '큼': 1, '이제': 156, '이짐트': 1, '피라미드': 1, '세울': 1, '있게': 6, '되었습니다': 4, '비장한': 1, '표정': 11, '진심': 42, '옆': 25, '동네': 4, '부': 31, '다봤다': 1, '앜': 16, '보고싶다': 32, '보려고': 31, '달': 49, '무료': 17, '하는건데': 2, '제': 176, '끌내고': 8, '온다': 4, '이': 211, '복': 6, '불': 1, '아시': 10, '전': 131, '수박': 2, '겉': 1, '찾기로': 1, '파는': 7, '편이': 16, '스킨스': 11, '같은것도': 4, '그냥': 83, '철': 2, '분위기': 45, '좋당': 2, '이러고': 5, '좋아함': 31, '성격': 9, '대사': 39, '관계도': 1, '이런건': 4, '알바': 3, '무지개': 1, '전구': 4, '배송': 5, '와라': 1, '느낌': 108, '내면': 2, '방': 32, '처': 14, '박혀있을거라고': 1, '데이': 8, '결제': 49, '햇다': 1, '봐라': 16, '안보': 28, '아님': 48, '기묘한아기': 1, '존나': 111, '좋아한다고': 2, '오빠': 22, '나가고': 4, '나가자': 1, '배우': 58, '주하연': 8, '마지막': 74, '한국': 54, '요괴': 27, '대백': 10, '프로젝트': 9, '후원자': 8, '님': 33, '공유': 19, '넘': 182, '잼': 89, '쓰요': 1, '재밋엉': 1, '바': 51, '쓱': 16, '다봤음': 1, '게': 84, '있었군요': 1, '많이는': 1, '아너겅': 3, '같': 33, '닼았지만': 1, '먼가': 10, '다
```

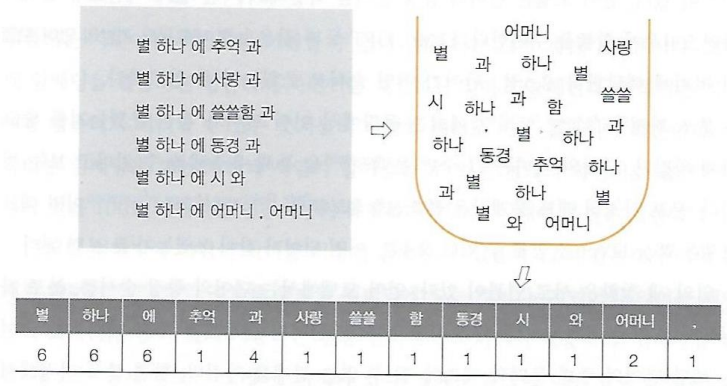
그림 44. KoNLPy – Okt 예시

2. Bag of words Embedding vs. TF-IDF

KoNLPy를 이용한 형태소 분석으로 형태소별 빈도를 계산하는 방식으로 실제 코드에서는 '명사'만을 추출하여 작품에서 독자들의 많은 관심을 받고 있는 주제가 어떤 것인지 알아보고자 한다.

가. Bag of words Embedding

단어의 나열 순서에 상관없이 등장 빈도를 임베딩으로 쓰는 기법이다. 단순히 BOW 방식으로 단어 빈출도를 계산한다면 맥락과 관계없는 단어가 단지 빈도수가 높다는 이유로 높은 점수를 받을 가능성이 있다.



별	하나	에	추억	과	사랑	쓸쓸	함	동경	시	와	어머니	.
6	6	6	1	4	1	1	1	1	1	1	2	1

그림 45. Bag of Words Embedding 예시[10]

[10] 한국어 임베딩, 이기창, 에이콘, 2019

나. TF-IDF

<수식 1>에서 D는 문서, T는 단어, N은 문서의 총개수를 의미한다. TF는 어떤 단어가 특정 문서에 얼마나 많이 쓰였는지 그 빈도를 나타내고 DF는 특정 단어가 나타난 '문서의 수'를 의미한다. DF가 클수록 많은 문서에서 쓰인 단어를 뜻하고 <수식 1>의 IDF는 전체 문서수를 해당 단어의 DF로 나눈 뒤 로그를 취한 값이다. IDF 값이 클수록 해당 단어가 문서의 연관성이 크다는 것을 의미한다.

$$\mathbf{TF-IDF(w)} = \mathbf{TF(w)} - \mathbf{log}\frac{\boldsymbol{N}}{\boldsymbol{DF(w)}})$$

수식 1. TF-IDF 가중치 계산

구분	메밀꽃 필 무렵	운수 좋은 날	사랑 손님과 어머니	삼포 가는 길
담배	0.2603	0.2875	0.0364	0.2932
를	0.0	0.0034	0.0	0.0

그림 55. TF-IDF 결과 예시[11]

<그림 52>에서 '를'이라는 조사가 '담배'에 비해 다빈출 단어이지만 '담배'라는 단어는 '를'이라는 단어보다 TF-IDF가 큰 것으로 미루어 보아 실제론 '를'보다는 '담배'가 문서와 더 연관 있는 단어라는 결론이 도출된다. 정보성이 없는 조사 '를'과 같은 단어의 가중치가 0에 근사하여 불필요한 정보를 없애기 때문에 순서를 생각하지 않고 단순 빈도를 계산하는 BOW방식보다 더 좋은 임베딩 결과를 가져온다.

3. 감성사전 vs 기계학습 vs 딥러닝 방식

가. 감성사전

형태소 분석을 통해 형태소 판별을 한 뒤, 형태소 별로 사전에 점수화된 점수를 가지고 긍부정을 판단하는 방식이다. 단점은 단어별로 판단하기 때문에 전체 맥락의 긍부정을 잘못 판단하는 경우가 많다. 각 감성 사전마다 어떤 단어나 조사를 제거하고 비속어, 신조어, 오타 처리를 어떻게 하느냐에 따라 정확도가 달라진다.

---

[11] 한국어 임베딩, 이기창, 에이콘, 2019

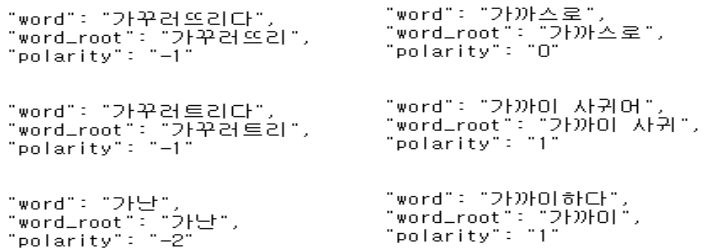


그림 56. KNU 감성 사전 中

나. 지도 기계학습(Supervised Learning) 기반 감성 분석

특정한 입력에 대해 올바른 정답이 무엇인지 알려주는 것이다. 지도 학습에는 연속적인 값을 찾는 회귀 분석과 입력에 대응하는 출력을 분석하여 이산적인 값을 찾는 분류 등이 있다.

1) 로지스틱 회귀모형

기존의 회귀 분석에서 반응변수가 0, 1의 이진형 변수에서 쓰이는 회귀분석 방법이다.

2) 의사결정나무모형

- 지도 기계학습의 일종으로 의사 결정을 위한 규칙을 트리구조로 만들어 분류와 예측을 하는 분석이다.

- <그림 54>와 같이 긍정적인 문장을 부정적으로 보거나 부정적인 문장을 긍정적으로 보는 등 정확도가 떨어진다.

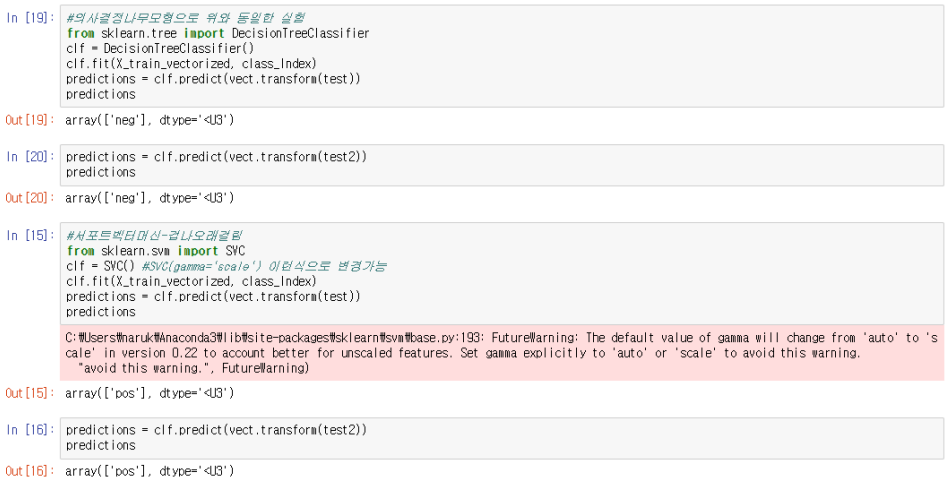


그림 46. 지도 기계 학습 기반 감성 분석 결과

다. 딥러닝 기반 감성 분석

KoBERT : 2019년 9월 SKT[12]에서 오픈소스로 제공된 딥러닝 모델이다. <그림 55>를 보면 기존 감성 분석에서 활용된 Fast-Text나 구글에서 제공하고 있는 BERT에 비해 한국어에 최적화되어 있다는 것을 알 수 있다. 우리 프로젝트에서 다루는 웹소설의 댓글과 리뷰와 유사한 '네이버 영화 리뷰 말뭉치'를 이용해 pre-train을 하고 예측해 보면 2-5% 더 높은 90%의 정확도를 보여준다.

Naver Sentiment Analysis

- Dataset : https://github.com/e9t/nsmc

Model	Accuracy
BERT base multilingual cased	0.875
KoBERT	**0.901**
KoGPT2	0.899

그림 47. Korean BERT pre-trained cased (KoBERT)와 타 Model의 정확도 비교[13]

- BERT[14]는 구글에서 개발한 자연언어를 양방향으로 학습하는 모델이다. BERT의 모델링 방식은 대량의 말뭉치로 미리 학습을 시킨 뒤 분류를 원하는 데이터를 넣으면 여기에 추가적인 머신러닝 모델을 쌓아 원하는 답을 도출시키는 것이다. 이 딥러닝 알고리즘은 실제 구글이 인공지능을 활용한 검색 능력을 향상시키기 위해 2019년 구글 검색에도 도입한 알고리즘이다. 미리 학습시킨 데이터를 통해 문장 속 단어의 뉘앙스나 문맥 파악을 더 효과적으로 하는 것으로 알려져있다.
- KoBERT는 BERT에서 한국어 성능의 한계를 극복하고자 T-Brain에서 개발했다고 한다. 수백만 개의 한국어 문장으로 이루어진 말뭉치를 학습시켰다고 한다.

---

[12] https://www.skt.ai/kr/press/detail.do?seq=27

[13] Korean BERT pre-trained cased (KoBERT) https://github.com/SKTBrain/KoBERT

[14] https://github.com/google-research/bert

VI. 참고문헌

[1] 파이썬을 활용한 클로러 개발과 스크레이핑 입문 (카토 카츠야, 요코야마 유우키, 위키북스, 2019)

[2] 파이썬 데이터 수집 자동화 한방에 끝내기 한입에 웹크롤링 (김경록, 서영덕, 비제이퍼블릭, 2018)

[3] 파이썬을 이용한 웹크롤링과 스크레이핑 (카토 코타, 위키북스, 2018)

[4] 파이썬을 이용한 머신러닝, 딥러닝 실전 개발 입문 (쿠지라 히코우즈쿠에, 위키북스, 2019)

[5] Web Scraping with Python (라이언미첼, 한빛미디어, 2019)

[6] 잡아라! 텍스트 마이닝 with 파이썬 (서대호, 비제이퍼블릭, 2019)

[7] https://www.crummy.com/software/BeautifulSoup/bs4/doc.ko/

[8] 오피니언 마이닝 기술을 이용한 효율적 상품평 검색 기법 (윤홍준, 김한준, 장재영, 2010)

[9] 한글 텍스트의 오피니언 분류 자동화 기법 (김진옥, 이선숙, 용환승, 2011)

[10] 상품평가 텍스트에 암시된 사용자 관점추출 (장경록, 이강욱, 맹성현, 2013)

[11] 텍스트 마이닝을 이용한 2012년 한국대선 관련 트위치 분석 (배정환, 손지은, 송민, 2013)

[12] 한글 감성어 사전 api구축 및 자연어 처리의 활용 (안정국, 김희웅, 2014)

[13] 한글 음소단위 trigram-signature 기반의 오피니언 마이닝 (장두수, 김도연, 최용석, 2015)

[14] 소셜네트워크서비스에 활용할 비표준어 한글처리 방법연구 (이종화, 레환수, 이현규, 2016)

[15] 인공지능을 활용한 오피니언 마이닝 - 소셜 오피니언 마이닝은 무엇인가? (윤병운, 2017)

[16] 한국어 비정형 데이터 처리를 위한 효율적인 오피니언 마이닝 기법 (남기훈, 2017)

[17] A study on Sentiment Analysis with Multivariate ratings in Online Reviews (임소현, 2020)

[18] 한국어 임베딩 (이기창, 에이콘, 2019)