최종보고서

프로젝트 명 : 데이터 마이닝을 이용한 웹소설 종합 인포 웹 어플리케이션

지도 교수 : 컴퓨터정보공학부 이기훈

팀 장 : 컴퓨터정보공학부 한승주

팀 원 : 컴퓨터정보공학부 김성종

컴퓨터정보공학부 조예슬

2020. 10. 31

광운대학교
KwangWoon University

목 차

I. 프로젝트의 개요

1. 배경 및 필요성

가. 시장 성장

최근 몇 년 동안 미디어 시장에는 많은 변화가 이루어졌다. 미디어물 배급 방식에 많은 변화가 생기며 현대인들은 더 쉽고 편하게, 빠르고 다양하게 미디어물을 즐기고 있다.

이번 프로젝트에서는 이 미디어물 중 하나인 웹소설에 주목해 보았다. 웹소설은 PC 통신 속 소설과 인터넷 소설에서 시작되어 현재는 이를 연재하고 배급하는 많은 플랫폼도 생겨나고 신인 작가와 새로운 연재 작품을 찾기 위한 공모전, 플랫폼 내 자유로이 작가가 되어 연재할 수 있는 곳을 만들며 더욱더 다양한 장르, 소재의 웹소설들을 양산해 나가고 있다. 거기다 웹소설이 드라마, 영화화되는 것은 물론 게임 등 OSMU1로 쓰이면서 그 시장은 매 해 점점 더 커지고 있는 추세이다.

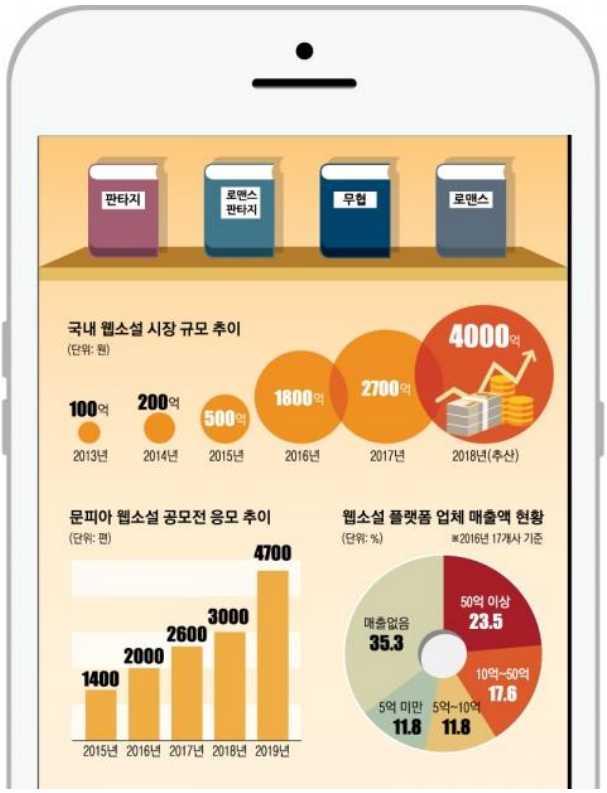

그림 1. 국내 웹소설 시장 규모 추이 (출처 : 서울신문, 2019.05)[2]

[1] OSMU : 원 소스 멀티 유즈

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

나. 문제 정의

(1) 양산화

시대 변화에 따른 트렌드 변화는 양산화로 이어진다. 그러나 그 양에 비해 독자의 기대감을 충족시켜주지 못하는 작품 또한 많아졌다.

(2) 다양화

독자들이 작품에 대한 의견을 표출하는 방식과 장소가 다양해졌다.

2. 목표

작품 댓글과 소셜미디어, 블로그, 커뮤니티 등 여러 웹 사이트를 통해 작품을 다양한 시각에서 분석해보고 현재 웹소설의 트렌드와 작품에 대한 독자들의 생각을 살펴본다.

3. 설계 내용

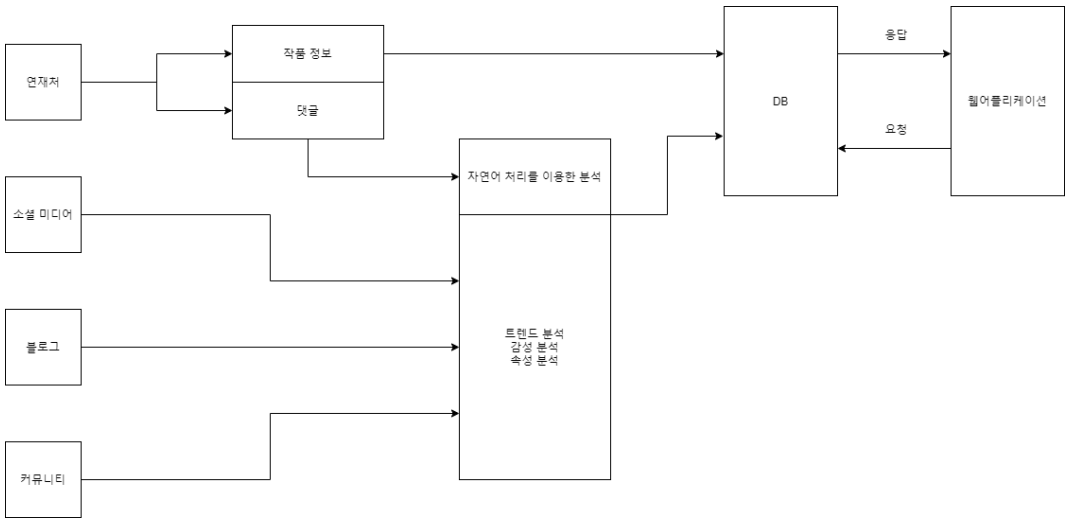

그림 2. 시스템 구조

- 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 작품 정보를 MySQL을 이용해 구축한 DB에 저장한다.
- Flask를 이용해 DB에 저장된 정보를 사용자에게 제공하기 위한 웹 어플리케이션으로 구현한다.

II. 프로젝트의 내용

1. 설계 및 개발의 내용

가. 개념 설계 (구조 설계)

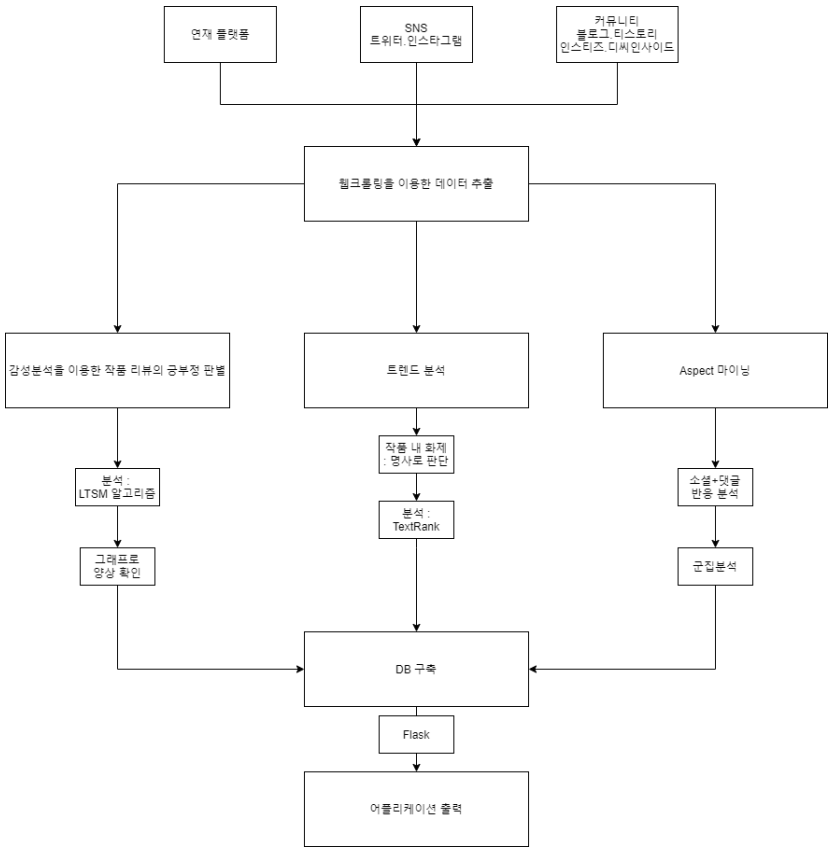

그림 3. Flow Chart

- **[데이터 수집]** 연재 플랫폼에서 작품 정보와 함께 댓글을, 소셜 미디어와 커뮤니티, 블로그에서 작품과 관련된 글을 수집하여 작품 개별 정보는 DB에 저장하고 댓글과 관련 글은 분석 과정에 들어간다.

- **[데이터 분석 1]** 작품과 관련된 글에서 TextRank를 이용한 분석을 통해 개별 작품에 대해 어떤 주제가 화제가 되고 있는지 알아볼 것이다.

- **[데이터 분석 2]** 작품에 대한 독자들의 긍부정 반응 변화 양상을 보기 위해 댓글과 리뷰글을 텐서플로우와 케라스, NSMC, LSTM 알고리즘을 이용하여 분석할 예정이다.

- **[데이터 분석 3]** 리뷰에서 군집분석을 통해 주인공, 줄거리, 분위기, 이 3가지 속성에 대한 독자들의 생각을 알아볼 것이다.

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

- **[결과물 구현]** Flask를 이용한 웹 애플리케이션으로 구현하여 DB에 저장된 결과를 사용자에게 제공한다.

나. 상세 설계 (기능 설계)

1) 플랫폼의 웹사이트 코드 분석 및 파이썬을 이용한 크롤링

- 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, 독자의 반응을 실시간으로 보고 분석하기에 용이한 플랫폼을 골라 선정하였다. 요즘 현대인들이 자주 사용하는 **소셜미디어**, 익명이라는 특성을 이용해 특별한 제약 조건에 구애받지 않는다는 특성의 **커뮤니티**를 선택했다. 소셜미디어의 경우, 커뮤니티와 비슷한 듯한 특성을 가지지만 유저 개인의 공간으로서 각 유저마다 고유의 특징이 보다 더 두드러진다. 반면 커뮤니티는 오래 시간 운영되어 많은 유저를 보유하고 있고 다양한 장르의 이야기가 오가며 유저 개인의 특징을 숨기며 관련 장르 얘기를 주로 게시된다. 또한 블로그와 티스토리는 주로 리뷰 중심의 포스트가 게시되는 특징이 있다. 분석된 각 웹사이트의 특성을 고려하여 여러 분석 방법을 생각하게 되었다.
- 데이터 수집 방법은 먼저 공통적으로 정보를 추출할 웹사이트의 HTML 코드를 분석한다.
- HTML 코드를 가져오기 위해 웹사이트에 요청을 보내는 방법으로 **requests 라이브러리와 Selenium** 두 가지 방법을 사용한다. 다음은 정보 수집을 위한 크롤링 예시이다.

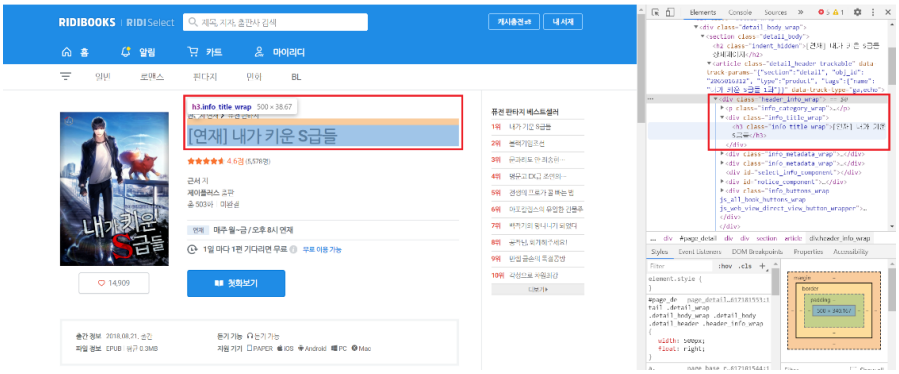

그림 4. 웹사이트 속 HTML 코드 예시

```
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 5. 분석한 HTML 코드를 바탕으로 한 크롤링 코드 예시 [CSS Selector 이용]

<그림 5> 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**)]

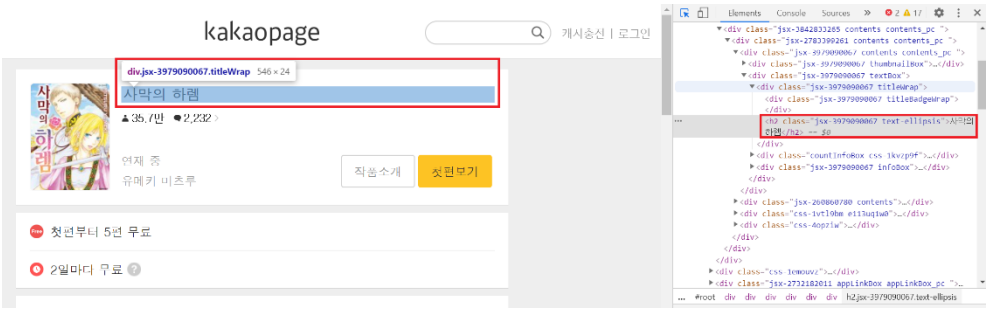

그림 6. 자바스크립트로 랜더링된 웹페이지

```
#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 7. Selenium을 이용한 크롤링 예시 [XPATH 이용]

<그림 7> 코드 내용

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

- Java-script로 랜더링되는 동적 웹사이트의 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**)]

- 각 사이트와 페이지별로 링크를 재귀적으로 검색하여 필요한 정보를 수집한다.

2) Data Processing (데이터 분석)

수집 데이터 분석은 연재 플랫폼의 댓글 리뷰, 커뮤니티 및 소셜 데이터를 통해 이뤄진다.

가) 트렌드 분석 : 커뮤니티와 소셜미디어 속 작품 내 트렌드.

작품에 대해 독자들이 어떤 키워드에 집중하고 있는지 텍스트 분석을 통해 알아본다. 커뮤니티 및 소셜 미디어, 블로그 글을 이용해 분석한다.

(1) 사용자는 작품과 관련하여 어떤 키워드가 독자들에게 화제가 되고 있는지 알 수 있을 것이다.

(2) 소셜미디어와 커뮤니티는 독자의 표현이 자유롭다는 특징을 가지고 있어 이를 **TextRank 기법**을 이용해 핵심 어구 추출을 통한 분석을 진행한다.

(3) 분석을 통해 나온 결과에서 **상위 5개의 데이터를** 유저에게 제공할 것이다.

나) 감성 분석 : 작품에 대한 독자들의 반응 변화

1달 간격으로 작품 댓글과 리뷰를 통해 **독자들의 반응 변화**를 살펴본다.

(1) **연재 플랫폼의 댓글 리뷰, 블로그 포스트**와 같이 독자들이 작품에 대한 감정적 변화를 직접적으로 드러내는 글을 통해 긍부정도를 살펴보고 이에 대한 변화를 그래프화하여 유저로 하여금 독자들의 변화 양상을 시각적으로 확인할 수 있도록 제공할 것이다.

(2) **텐서플로우와 케라스**를 가지고 **NSMC**[3]를 이용하여 학습을 시킨 뒤, **LSTM 알고리즘**[4]을 이용한 분석을 진행한다.

(3) 평가에 대한 점수를 기간에 대한 **그래프**로 표현한다.

---

[3] Naver Sentiment Movie Corpus : https://github.com/e9t/nsmc

[4] 별첨 [5] 항목 참조

다) Aspect 분석 : 캐릭터, 줄거리, 분위기에 대한 독자들의 반응

작품 리뷰와 네이버, 티스토리 **등 리뷰 글이 많은 플랫폼을 이용해** 작품 내 주인공, 분위기, 스토리 등과 연결되어 **자주 나오는** 반응**을 분석한다.**

(1) 작품에 대해 구체적으로 작성한 글이 필요하기에 **네이버, 티스토리, 다음 블로그**의 리뷰를 수집하여 분석한다.

(2) 한 키워드를 표현하는 방식이 단순한 이름, 별명, 줄임말 등으로 다양하기 때문에 이를 종합하기 위해 **군집분석**을 이용한다.

(3) 각 키워드 별로 키워드를 설명하는 **주변 1-3개 단어**를 **벡터화**한다.

3) DB 구축

파이썬에서는 PyMySQL 패키지를 통해 Python으로도 외부에서 DB 내 데이터에 접근 및 조작이 가능하여 MySQL을 활용하고자 한다.

4) 웹사이트 UI 제작

- 유저들이 제공되는 정보를 간결하고 빠르게 전달받을 수 있도록 만드는 것이 웹사이트 제작 시, 가장 중요하게 생각한 부분이다.
- 사용자의 접근성 편리를 위해 웹사이트로 최종 결과물을 구축한다.
- 메인 화면에서는 감성분석을 통해 나온 이달의 점수가 높은 10작품을 순서에 관계없이 나열하여 '이달의 작품 TOP10'이라는 제목 하에 보여준다.
- 왼쪽 상단의 메뉴 탭에서 연재처와 장르별 아이콘을 통해 해당 플랫폼과 장르별 작품을 확인할 수 있도록 구성하였다.
- 검색을 통해 유저가 검색하고 싶은 작품을 검색해 볼 수 있도록 구성하였다.
- 작품 페이지에서는 수집한 작품의 기본 정보와 함께 TextRank 분석 결과를 키워드로 최대 5개 나열한다. 그 밑에 주인공, 스토리, 분위기에 대한 반응을 나열하고 그 하단에 감성 분석의 변화 양상을 시각화한 그래프를 보여준다.

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

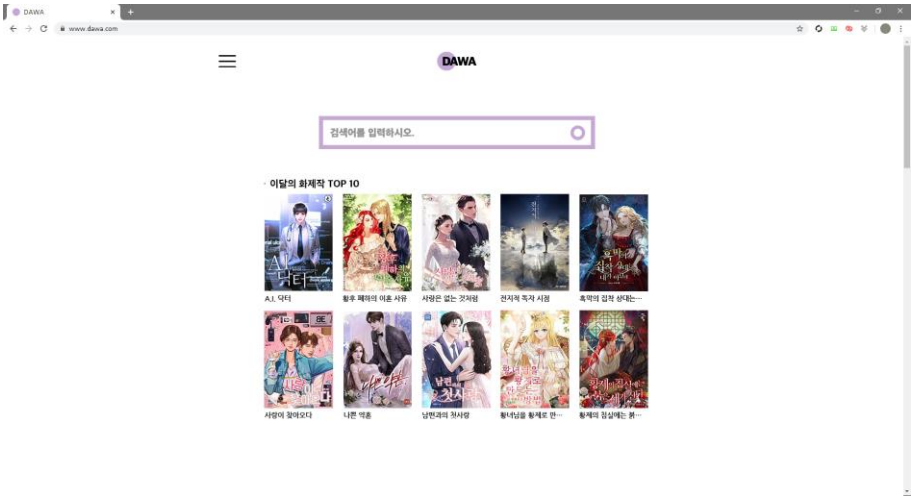

그림 8. 메인화면 예시

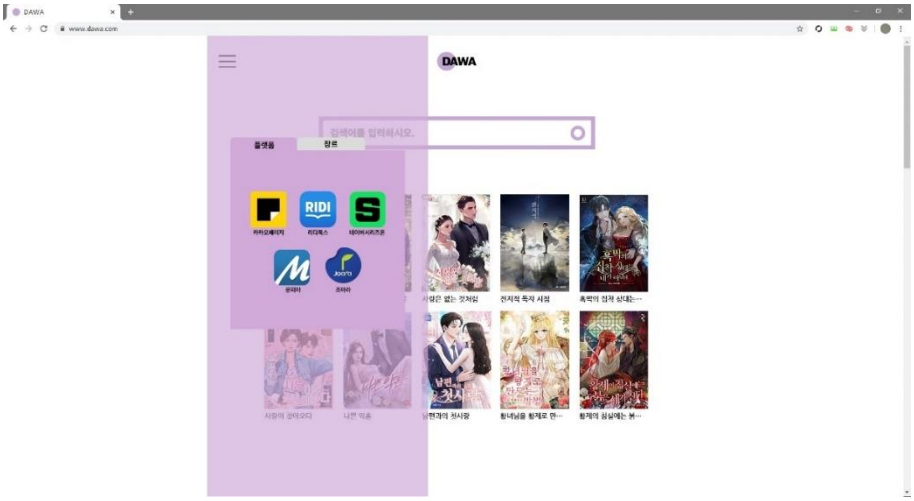

그림 9. 메뉴 화면 디자인 예시

Figure 1. 작품 정보 화면 디자인 예시

2. 역할 분담

가. 계획

- **크롤러 제작 [공통]** : 연재처에서 기본 정보와 댓글 수집, 소셜미디어, 커뮤니티, 블로그에서 포스트 내용과 작성 날짜, 커뮤니티는 댓글까지 수집한다. 6곳의 연재처와 6곳의 수집처를 조원 모두 나눠 기본 크롤러 제작을 진행한다.

- **데이터 분석 [김성종, 한승주]** : 기본 크롤러를 이용한 분석 코드를 작성한다.

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

- **웹 애플리케이션 [조예슬, 김성종]** : DB에 저장된 데이터를 사용자의 편의성을 고려하여 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] : Logical-ERD 설계와 이를 기반으로 MySQL을 이용한 DB를 구축한다. 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 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] : 웹페이지 디자인 UI의 구체화와 Java-Script, CSS, HTML을 이용한 웹페이지를 제작한다. DB[조원] : DB 구축을 위한 자료 수집과 ERD 수정 및 구축을 지원한다.

나. 실제 진행

성명	역할
한승주 [팀장]	크롤러 제작[공통] : 조아라, 리디북스, 트위터, 인스타그램 1차 크롤러 제작 및 이후 수정 작업. 인스티즈, 디씨인사이드 1차 크롤러 제작 서포트. 시리즈온, 디씨인사이드, 네이버 웹소설 에러 수정. DB[조장] : Logical-ERD 설계와 MySQL을 이용한 DB 구축. 데이터 분석[조원] : 한승주/조예슬 담당 크롤러 분석 코드 형식에 맞도록 모듈화 작업.
김성종 [팀원]	크롤러 제작[공통] : 카카오페이지, 문피아, 블로그(네이버, 다음, 티스토리) 1차 크롤러 제작 및 수정, 모듈화 작업. 디씨인사이드, 인스티즈, 시리즈온, 네이버 웹소설 1차 크롤러 제작 서포트. 인스타그램, 인스티즈, 디씨인사이드 수정 작업. 데이터 분석[조장] : 텐서플로우, 케라스, LSTM 알고리즘을 이용한 분석 코드 제작. 웹페이지 구현[조원] : 웹페이지 제작을 위한 자료수집.
조예슬 [팀원]	크롤러 제작[공통] : 시리즈온, 네이버 웹소설, 디씨인사이드, 인스티즈 1차 크롤러 제작. 웹페이지 구현[조장] : 웹페이지 UI 디자인, Flask, JS, CSS, HTML을 이용한 웹페이지를 제작.

3. 최종 결과물 https://github.com/hazelnutlemon/DAWAproject

가. 데이터 수집을 위한 기본 크롤러 제작

1) 플랫폼 크롤링 : 6곳의 연재처에서 수집할 정보는 총 12가지

[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[5]

```
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 10. 카카오 페이지 작품 정보 크롤링 결과

나) 네이버 시리즈 (+ 네이버 웹소설) : Request 사용[6]

```
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 11. 네이버 웹소설 작품 정보 크롤링 결과

```
제목: 입술이 너무해
화수: 105
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, 가슴
은 봉긋해지고, 입술은 더욱더 새빨갛게 피어났다. "예쁜데요." "네?" "입술 예쁘네." 설렘 대폭발, 심쿵 주의, 치명적으로 씩시
한 직진남의 꽃 떨어지는 막강 들이댐이 시작됐다. 운명에 얽힌 세계 최고 달달한 썸씽과 더 달달한 끈적끈적 연애! 예측불허 입술 로맨스.
```

그림 12. 네이버 시리즈온 작품 정보 크롤링 결과

[5] 8 페이지 <그림 7>과 양상 동일. 참조.

[6] 8 페이지 <그림 5>와 양상 동일. 참조.

다) 조아라 : Request, BeautifulSoup 이용[7]

```
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 13. 조아라 작품 정보 크롤링 결과

라) 문피아 : Request, BeautifulSoup 이용[8]

```
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 14. 문피아 작품 정보 크롤링 결과

마) 리디북스 : Request, BeautifulSoup 이용[9]

```
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 15. 리디북스 작품 정보 크롤링 결과

---

[7] 8 페이지 <그림 5> <그림 7>과 양상 동일. 참조.

[8] 8 페이지 <그림 5> <그림 7>과 양상 동일. 참조.

[9] 8 페이지 <그림 5> <그림 7>과 양상 동일. 참조.

2) 소셜미디어 크롤링

가) 소셜미디어 [트위터] : API 사용

- 트위터는 불특정 다수가 독백을 하는 특징이 있다.
- 여러 API 중 속도면을 비교하여 Twint[10] 패키지를 사용했다.
- 2020년 1월 1일부터 올 한 해의 데이터를 수집하여 분석할 것이다.

```
# Twint를 이용한 트위터 검색 결과
c = twint.Config()

# 파라미터 설정
#c.Limit = limit # 트윗 개수 제한 X
c.Search = searchterm
c.Since = since
c.Until = until
c.Popular_tweets = True
# c.Custom= ["date", "tweet"]
c.Store_object = True
c.Store_object_tweets_list = tweets

twint.run.Search(c)

tweet_data = [] # 수집한 트위터를 list 타입으로 저장

for tweet in tweets:
    date = tweet.datestamp # YYYY-MM-DD 형식으로 작성 날짜 저장
    tweet_text = tweet_clean(tweet.tweet) # 트윗 내용 저장
    tweet_data.append([date,tweet_text])
```

그림 16. 기간에 따른 트위터 검색 정보 추출

〈그림 16〉 코드 내용

트윗 수집을 위한 파라미터를 설정하고 트윗을 수집한다.

- Import twint로 패키지를 불러온다.
- twint.Config()로 수집 기준을 정의한다.
- 분석에 이용하기 쉽도록 리스트 타입으로 결과를 추출한다.

나) 소셜미디어 [인스타그램] : Selenium, XPATH 사용

- 인스타그램은 트위터와 같이 간단한 소감을 함께 적어 놓는 경우가 많다.
- Selenium과 XPATH를 이용하며 일정 이상 스크롤시. 로그인을 요구하기 때문에 먼저 로그인을 한 후 마지막까지 스크롤을 내린 후에 포스트를 클릭하여 각 포스트의 내용을 가져올 수 있도록 코드를 구성하였다.
- 포스트 링크에서 글의 내용, 작성 날짜를 찾아 리스트로 결과를 보여준다.

[10] https://github.com/twintproject/twint

```
# 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 17. 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 18. 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/ByxfvIoFIYk/	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. 밀바닥 인생에서 돈 되는 일이면...	2018-06-13
6	https://www.instagram.com/p/CAIvqHIgSO6/	u_u9oo	한줄 감상 : 불닭볶음면보다 매운 로맨스릴러....🔪..「리셋팅 레이디」, 차서진...	2020-05-14
7	https://www.instagram.com/p/B73jRjhnQwX/	rooomr_0202	캐런은 117세 생일을 맞이하여 살인마가 되기로 결심했다.#리셋팅레이디	2020-01-29

그림 19. 인스타그램 크롤링 결과 - 포스트 URL, 작성 유저, 내용, 날짜 추출

3) 블로그 크롤링

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

- 네이버 블로그와 티스토리는 리뷰 중심의 글이 특징이기에 연관어 찾기에 적합하다고 판단되어 Aspect 마이닝에 사용한다.
- 네이버 블로그의 검색 시, 네이버의 정책으로 인해 1000건의 검색 결과만을 보여준다.
- 검색 결과를 통해 나온 글의 URL을 수집하고 검색을 통해 글의 제목과 작성 일시, 게시물의 내용을 추출한다.

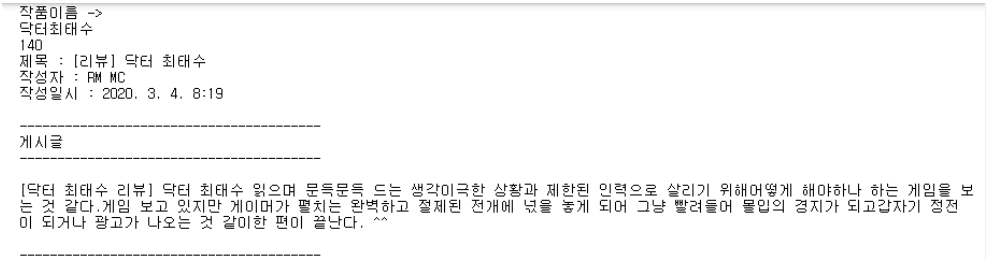

그림 20. 네이버 블로그 크롤링 – 추출 결과 예시

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

- 작성글 제목, 작성 일시를 수집한다.
- 게시물마다 구성 방식이 달라 이에 따른 여러 버전의 크롤링 코드를 구성하였다.

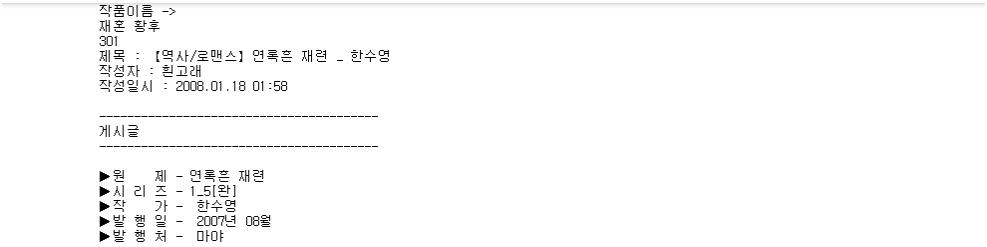

그림 21. 다음 블로그 크롤링 결과 예시

4) 커뮤니티 크롤링 : [디씨인사이드]

가) 커뮤니티 [디씨인사이드] : Request 사용

- 갤러리 전체 검색과 장르별 갤러리 모두 검색이 가능하다. 갤러리 전체 검색의 경우, 최대 120페이지(한 페이지 당 25개)의 검색 결과를 보여준다.
- 각 갤러리가 가지고 있는 아이디를 받아 params 값으로 입력을 해 연결하는 방식이다. 게시글의 제목, 날짜, 내용, 댓글을 추출한다.

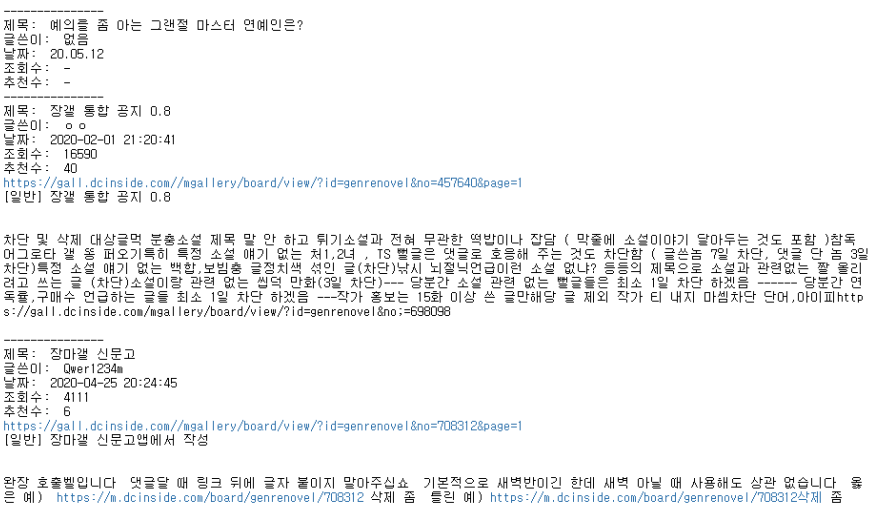

그림 22. 디씨인사이드 크롤링 결과

나) 커뮤니티 [인스티즈] : Selenium과 XPATH를 이용한 마이닝

- 게시글에 작품의 내용이 포함되어 있다면 스포 방지 팝업창의 출현한다.
- 검색 결과에는 존재하지만 로그인 회원만 게시물을 확인할 수 있도록 되어 있는 게시물, 댓글 작성자의 이름이나 연관 게시글 제목에 검색어가 있는 경우는 수집에서 제외했다.

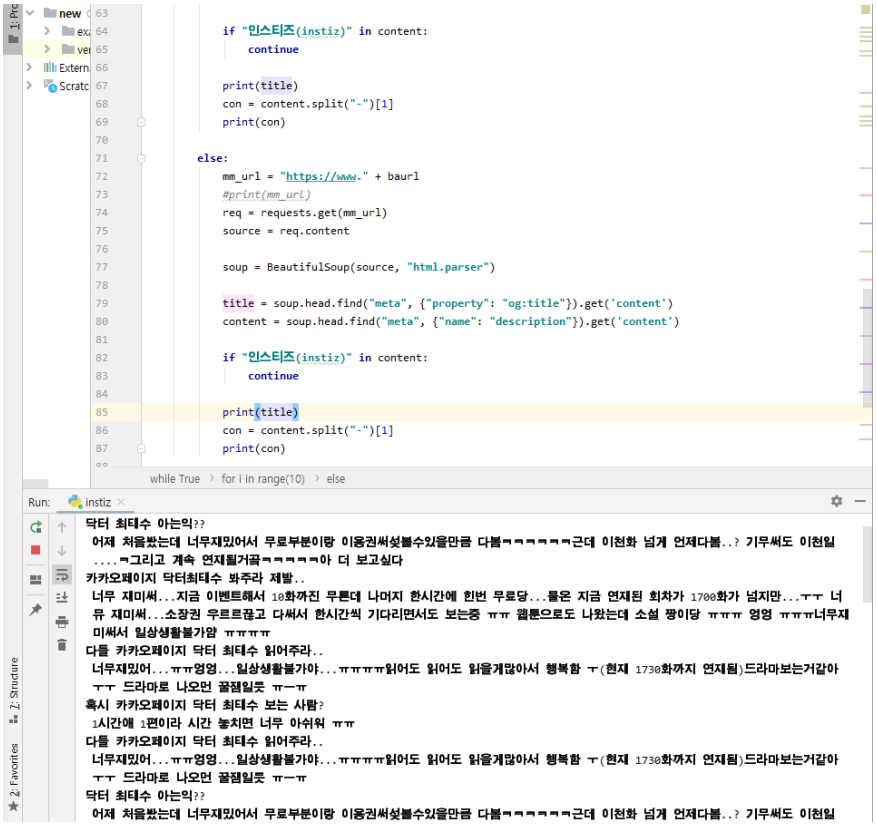

그림 23. 인스티즈 크롤링 코드 中 / 크롤링 결과

나. 데이터 분석을 위한 전처리과정과 분석 코드 제작

1) 트렌드 분석 : TextRank 알고리즘을 활용하여 핵심 키워드를 추출

- 이 분석을 통해 핵심 문장 또한 추출 가능하여 독자들의 핵심 반응을 추출하는데 활용할 수 있다.

가) 핵심 키워드 추출

- 작품 이름 검색 시, 두 단어 간 유사도에 따라 n-gram을 만족하도록 하였다.
- 데이터베이스에 저장할 5개의 단어를 얻을 수 있도록 코드를 수정하였다.
- [11]생성자의 인수로 특정 단어의 좌우로 몇 개의 단어를 활용할 것인지를 나타내는 'window'와 동시 출현 빈도를 가중치에 반영하기 위한 'coef', 연관여부를 판단하기 위한 최소의 유사도를 지정하는 'threshold'값을 설정하고 TextRank 클래스를 생성한다.
- [12]핵심 키워드 추출을 위해 tr.load로 문장을 읽고 tr.extract로 키워드를 추출[13]한다.

```
class TextRank:
    def __init__(self, **kargs):
        self.graph = None
        self.window = kargs.get('window', 5)
        self.coef = kargs.get('coef', 1.0)
        self.threshold = kargs.get('threshold', 0.005)
        self.dictCount = {}
        self.dictBiCount = {}
        self.dictNear = {}
        self.nTotal = 0


    def load(self, sentenceIter, wordFilter = None):
        def insertPair(a, b):
            if a > b: a, b = b, a
            elif a == b: return
            self.dictBiCount[a, b] = self.dictBiCount.get((a, b), 0) + 1

        def insertNearPair(a, b):
            self.dictNear[a, b] = self.dictNear.get((a, b), 0) + 1

        for sent in sentenceIter:
            for i, word in enumerate(sent):
                if wordFilter and not wordFilter(word): continue
                self.dictCount[word] = self.dictCount.get(word, 0) + 1
                self.nTotal += 1
                if i - 1 >= 0 and (not wordFilter or wordFilter(sent[i-1])): insertNearPair(sent[i-1], word)
                if i + 1 < len(sent) and (not wordFilter or wordFilter(sent[i+1])): insertNearPair(word, sent[i+1])
                for j in range(i+1, min(i+self.window+1, len(sent))):
                    if wordFilter and not wordFilter(sent[j]): continue
                    if sent[j] != word: insertPair(word, sent[j])
```

그림 24. 키워드 추출을 위한 Text Rank Class 생성

```
201 def hot_topic_analyzer(work_name):
202     tr = TextRank(window=5, coef=1)
203     #print('Load...')
204     stopword = set([('있', 'VV'), ('하', 'VV'), ('되', 'VV'), ('없', 'VV')])
205     file_name= 'Crawling\\' + work_name + '.txt'
206     tr.load(RawTaggerReader(file_name), lambda w: w not in stopword and (w[1] in ('NNG', 'NNP', 'VV', 'VA')))
207     #print('Build...')
208     tr.build()
209     kw = tr.extract(0.1)
210     title = work_name
211     title_nonspace = title.replace(' ', '')
212     count = 0
213     word_list=[]
214     for k in sorted(kw, key=kw.get, reverse=True):
215         temp = ("%s%g" % (k, kw[k])).split('0')[0]
216         if 'VV' in temp:
217             continue
218         if 'VA' in temp:
219             continue
```

(좌)그림 25. TextRank 분석 코드 中

```
from Hot_Topic_Analyzer import hot_topic_analyzer

hot_topic_analyzer('구르미 그린 달빛')
```

```
Load...
Build...
라온
윤 이수
드라마
별 연
화초 저하
```

(우)그림 26. <그림 25> 분석 코드를 이용한 결과 예시

---

[11] 20 페이지 <그림 24> 참조.

[12] 21 페이지 <그림 25> 참조.

[13] 결과 : 21 페이지 <그림 26> 참조.

2) 감성 분석 – TenserFlow와 Keras의 말뭉치 이용한 학습 후, LSTM 알고리즘 이용한 분석

- 리뷰의 긍부정을 판단하고 평가에 대한 점수를 기간에 대한 그래프로 표현하여 시간이 지남에 따라 작품의 평가 변화 양상을 확인한다.
- 이 분석은 85.5% 정도의 정확도를 가지긴 하지만 긍정의 확률을 보여주기 때문에 변칙적인 확률을 보여줄 수 있다.
- 머신러닝에 이용할 코드를 모듈화하여 별도의 패키지로 만들고 이를 이용해 자동 모델링을 구성하고 predict 함수를 사용하여 결과를 볼 수 있도록 했다.
- 감성분석 코드 중, ratings_test를 이용해 학습시킨다.

```
In [42]: es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=4)
         mc = ModelCheckpoint('best_model.h5', monitor='val_acc', mode='max', verbose=1, save_best_only=True)

In [43]: model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
         history = model.fit(X_train, y_train, epochs=15, callbacks=[es, mc], batch_size=60, validation_split=0.2)

Epoch 1/15
1939/1939 [==============================] - ETA: 0s - loss: 0.3903 - acc: 0.8221
Epoch 00001: val_acc improved from -inf to 0.84468, saving model to best_model.h5
1939/1939 [==============================] - 47s 24ms/step - loss: 0.3903 - acc: 0.8221 - val_loss: 0.3532 - val_acc: 0.8447
Epoch 2/15
1938/1939 [==============================>.] - ETA: 0s - loss: 0.3274 - acc: 0.8572
Epoch 00002: val_acc improved from 0.84468 to 0.85521, saving model to best_model.h5
1939/1939 [==============================] - 47s 24ms/step - loss: 0.3274 - acc: 0.8573 - val_loss: 0.3347 - val_acc: 0.8552
Epoch 3/15
1937/1939 [==============================>.] - ETA: 0s - loss: 0.3024 - acc: 0.8721
Epoch 00003: val_acc improved from 0.85521 to 0.85741, saving model to best_model.h5
1939/1939 [==============================] - 46s 24ms/step - loss: 0.3023 - acc: 0.8721 - val_loss: 0.3308 - val_acc: 0.8574
Epoch 4/15
1937/1939 [==============================>.] - ETA: 0s - loss: 0.2844 - acc: 0.8806
Epoch 00004: val_acc improved from 0.85741 to 0.85875, saving model to best_model.h5
1939/1939 [==============================] - 48s 25ms/step - loss: 0.2844 - acc: 0.8806 - val_loss: 0.3266 - val_acc: 0.8587
Epoch 5/15
1938/1939 [==============================>.] - ETA: 0s - loss: 0.2688 - acc: 0.8890
Epoch 00005: val_acc improved from 0.85875 to 0.85978, saving model to best_model.h5
1939/1939 [==============================] - 48s 25ms/step - loss: 0.2687 - acc: 0.8890 - val_loss: 0.3288 - val_acc: 0.8598
Epoch 6/15
1938/1939 [==============================>.] - ETA: 0s - loss: 0.2543 - acc: 0.8963
Epoch 00006: val_acc did not improve from 0.85978
1939/1939 [==============================] - 47s 24ms/step - loss: 0.2543 - acc: 0.8963 - val_loss: 0.3380 - val_acc: 0.8583
Epoch 7/15
1938/1939 [==============================>.] - ETA: 0s - loss: 0.2397 - acc: 0.9034
Epoch 00007: val_acc did not improve from 0.85978
1939/1939 [==============================] - 49s 25ms/step - loss: 0.2397 - acc: 0.9034 - val_loss: 0.3465 - val_acc: 0.8561
Epoch 8/15
1939/1939 [==============================] - ETA: 0s - loss: 0.2242 - acc: 0.9108
Epoch 00008: val_acc did not improve from 0.85978
1939/1939 [==============================] - 48s 25ms/step - loss: 0.2242 - acc: 0.9108 - val_loss: 0.3591 - val_acc: 0.8534
Epoch 00008: early stopping
```

그림 27. 감성분석 코드 中

```
테스트 정확도: 0.8543

def sentiment_predict(new_sentence):
  new_sentence = okt.morphs(new_sentence, stem=True) # 토큰화
  new_sentence = [word for word in new_sentence if not word in stopwords] # 불용어 제거
  encoded = tokenizer.texts_to_sequences([new_sentence]) # 정수 인코딩
  pad_new = pad_sequences(encoded, maxlen = max_len) # 패딩
  score = float(loaded_model.predict(pad_new)) # 예측
  if(score > 0.5):
    print("{:.2f}% 확률로 긍정 리뷰입니다.\n".format(score * 100))
  else:
    print("{:.2f}% 확률로 부정 리뷰입니다.\n".format((1 - score) * 100))

sentiment_predict('이 영화 개꿀잼 ㅋㅋㅋ')

93.41% 확률로 긍정 리뷰입니다.

sentiment_predict('이 영화 핵노잼 ㅠㅠㅠ')

97.95% 확률로 부정 리뷰입니다.
```

그림 28. <그림 27> 코드를 이용한 리뷰 분석 결과 예시

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

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 데이터의 특성을 분석하여 이와 유사한 것을 묶는 것이다. word2vec을 이용하여 작품의 다양한 요소에 대해 의존적 연결 형용사나 동사를 추출한다.
- word2vec을 수행하기 위해 사전에 처리해야 할 코드를 모듈로 작성하면 작품의 글이 담긴 txt 파일로 word2vec을 수행하고 해당 작품의 aspect를 추출하도록 코드를 수정하였다.

```
#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")
print("Word2Vec Modeling finished")
```

```
Word2Vec Modeling finished
```

```
from gensim.models import word2vec
from konlpy.tag import Okt

twitter = Okt()

model = word2vec.Word2Vec.load("NaverMovie.model")
count=0
model_list=[]
model_list=model.most_similar(positive=["주인공"],topn=300)

for i in range(len(model_list)):
    temp_list=twitter.pos(model_list[i][0], norm=True, stem=True)
    if(temp_list[0][1]=='Adjective'):
        print(temp_list[0][0])
        count+=1

    if(count==5):
        break
```

```
건장하다
흉하다
유별나다
미적지근하다
전지전능하다
```

그림 28. word2vec 모듈

```
59 •  select n.idnovel, n.title, s.* from novel n left outer join aspect_story s on n.idnovel=s.novel
```

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

idnovel	title	novel	reaction1	reaction2	reaction3	reaction4	reaction5
14	웹소설만 읽으면 강해져	14	강하다	같다	좋다	어떻다	재미있다
15	수의사 진태민	15	아쉽다	들뜨다	유명하다	멋지다	괜찮다
16	이번엔 진짜 재벌!	16	성공하다	변하다	인하다	필요하다	좁다
17	백작가의 망나니가 되었다	17	특출나다	안타깝다	그만하다	조용하다	답답하다
18	나 혼자만 레벨업	18	특출나다	환생하다	탄탄하다	지루하다	변하다
19	환생표사	19	신비하다	용감하다	무능하다	간단하다	별다르다
20	명문고 EX급 조연의 리플레이	20	많다	열광하다	있다	그렇다	아쉽다
21	망나니 1황자가 되었다	21	얼떨떨하다	드물다	상당하다	미치다	끊임없다
22	달콤, 찬란한 재벌기	22	재미있다	인하다	힘들다	달콤하다	성하다

그림 29. <그림 28> 코드를 이용한 분석 결과 예시

다. 수집한 데이터와 분석 결과를 저장할 DB 제작

1) Logical-ERD

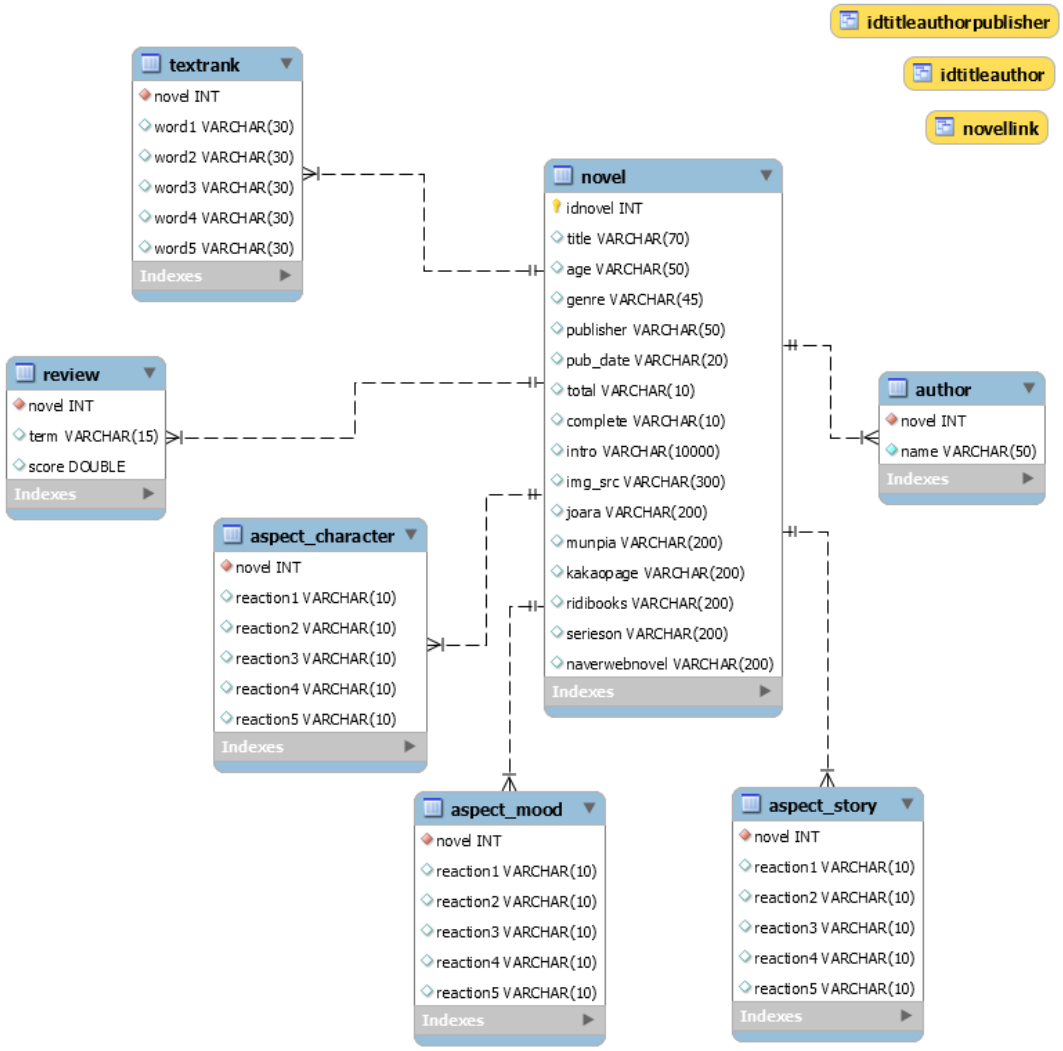

그림 30. Logical-ERD

[작품 테이블] 작품의 기본 정보가 저장되는 테이블로 작품이 삽입될 때마다 작품 번호를 자동으로 등록시켜주고 작품제목, 연령 등급, 장르, 출판사, 출간일, 총 연재 화수, 완결여부, 작품 소개, 표지 사진 링크와 각 연재처 링크를 속성으로 가진다. 연재처의 경우 지정한 연재처 6곳 중 연재하지 않는 곳은 Null로, 연재하는 곳은 연재 링크를 넣는다.

[저자 테이블] 작품 테이블의 작품 번호를 외래키로 가지는 작품과 작가의 이름을 속성으로 가지며 여러 작가가 한 작품을 집필하는 경우가 있기 때문에 저자 테이블은 따로 만들어 외래키로 메인 테이블인 작품 테이블과 연결시켜주었다.

분석 결과의 경우 총 3가지의 분석을 하기 때문에 textrank, 감성분석의 review, aspect 기본 3개의 테이블로 구성되어 있다. [댓글긍부정도 테이블]은 review테이블로 댓글과 리뷰의 긍부정 양상을 저장하는 테이블로 작품 테이블의 작품 번호를 외래키로 가지고 기간과 긍부정 비율을 속성으로 가진다. [리뷰 단어 테이블]은 textrank 테이블로 리뷰 속 빈출 단어를 저장하는 테이블이다. 단어와 분석 결과를 속성으로 가진다. [aspect 테이블]은 캐릭터, 분위기, 스토리 3개의 속성을 각 테이블로 나누어 저장했다. 각 테이블

에는 각 키워드를 설명하는 단어 최대 5개를 속성으로 가진다.

2) DB 구축

- 구성한 ERD를 바탕으로 MySQL 워크벤치를 이용하여 테이블을 구성하였다.
- 작품 테이블에서 age(연령등급)는 [전체 연령가, 12세 연령가, 15세 연령가, 19세 연령가] 4가지로 나뉜다. pub_date(출간일)은 [YYYY-MM-DD]와 같은 양식으로 삽입한다. complete(완결여부)는 [0: 미완결, 1: 완결]로 나타낸다.
- 위와 같이 테이블을 구성한 다음 <그림 31>과 같이 파이썬의 pymysql 모듈을 이용하여 csv파일로 저장한 작품 정보를 데이터베이스에 삽입했다.
- 작가 테이블의 경우, 여러 작가가 참여한 작품도 있기에 작품테이블에 같이 작성할 경우, 데이터 중복이 일어날 수 있다. 이로 인해 작가 테이블을 분리한 정규화를 진행했다.

```
CREATE TABLE `novel` (
  `idnovel` int NOT NULL AUTO_INCREMENT,
  `title` varchar(50) NOT NULL,
  `age` varchar(50) DEFAULT NULL,
  `genre` varchar(45) DEFAULT NULL,
  `publisher` varchar(50) DEFAULT NULL,
  `pub_date` varchar(20) DEFAULT NULL,
  `total` varchar(10) DEFAULT NULL,
  `complete` varchar(10) DEFAULT NULL,
  `intro` varchar(10000) DEFAULT NULL,
  `img_src` varchar(200) DEFAULT NULL,
  `joara` varchar(200) DEFAULT NULL,
  `munpia` varchar(200) DEFAULT NULL,
  `kakaopage` varchar(200) DEFAULT NULL,
  `ridibooks` varchar(200) DEFAULT NULL,
  `serieson` varchar(200) DEFAULT NULL,
  `naverwebnovel` varchar(200) DEFAULT NULL,
  PRIMARY KEY (`idnovel`)
)
```

그림 31. webnoveldb 中 [novel] 테이블

```
import pymysql
db = pymysql.connect(
    host='127.0.0.1',
    port=3306,
    user='root',
    passwd='root',
    db='webnoveldb',
    charset='utf8')
```

```
# DictCursor는 딕셔너리 형태로 결과를 반환
cursor = db.cursor(pymysql.cursors.DictCursor)
```

```
# 테이블 확인
sql="select * from novel;"
cursor.execute(sql)
result = cursor.fetchall()
```

```
# 삽입 INSERT
sql = '''insert into novel (title, age, publisher, pub_date, total, complete, intro, img_src) values ();'''

cursor.execute(sql)
db.commit()
```

그림 31. Pymysql을 이용한 DB 삽입

idnovel	title	age	genre	publisher	pub_date	total	complete
5	내 안에 마교있다	전체연령가	판타지 무협소설	라온EBM	2019-10-11	275	0
6	내 머릿속에 2000년 마법 역사	전체연령가	퓨전판타지	JC미디어	2019-07-14	196	1
7	이세계 천재 화가	전체연령가	퓨전판타지	㈜ 마야마루 출판사	2019-09-20	200	0
8	회귀의 전생	전체연령가	현대판타지	주식회사케이더	2019-03-28	525	1
9	내가 키운 S급들	전체연령가	퓨전판타지	제이플러스	2018-08-21	499	0
10	재능이 자꾸 늘어	전체연령가	현대판타지	파피루스	2020-03-06	164	0
11	아포칼립스의 고인물	전체연령가	퓨전판타지	제이플러스	2019-11-29	224	0
12	로그인하자마자 VIP	전체연령가	퓨전판타지	㈜인타임	2019-12-09	260	0
13	태왕이 되는 방법	전체연령가	퓨전판타지	㈜ 마야마루 출판사	2019-09-20	234	0
14	웹소설만 읽으면 강해져	전체연령가	퓨전판타지	제이플러스	2019-04-29	318	1
15	수의사 진태민	전체연령가	현대판타지	문피아	2020-04-20	207	0
16	이번엔 진짜 재벌!	전체연령가	현대판타지	JC미디어	2020-04-29	232	0
17	백작가의 망나니가 되었다	전체연령가	퓨전판타지	도서출판 청어람	2018-10-02	590	0
18	나 혼자만 레벨업	전체연령가	현대판타지	파피루스	2018-01-10	270	1
19	환생표사	전체연령가	판타지 무협소설	KW북스	2019-04-15	282	1
20	명문고 EX급 조연의 리플레이	전체연령가	퓨전판타지	㈜인타임	2020-01-21	227	0
21	망나니 1황자가 되었다	전체연령가	퓨전판타지	그림책토리	2019-05-30	343	0
22	달콤, 찬란한 재벌기	전체연령가	현대판타지	㈜인타임	2019-11-27	255	0
23	리퍼-무공수학자	전체연령가	판타지 무협소설	CANMEDIA	2019-05-29	175	1
24	VVIP 자원 쇼퍼	전체연령가	퓨전판타지	제이플러스	2019-07-29	188	0
25	용사를 방치했더니 자동 최강	전체연령가	퓨전판타지	제이플러스	2020-02-24	197	0

idnovel	intro	img_src	joara	munpia	kakaopage	ridibooks
5	흑월대 출신으로 천마의 다섯 번째 제자가 된 서...	//img.ridicdn.net/cover/1534091821/xxlarge	NULL	NULL	NULL	https://ridibook
6	검문의 낙오자 강민혁. 그가 고자현의 마법 운...	//img.ridicdn.net/cover/3228008724/xxlarge	NULL	NULL	NULL	https://ridibook
7	"화가가 되고 싶어." 하지만 절망에 빠졌다. 가...	//img.ridicdn.net/cover/2200032164/xxlarge	NULL	NULL	NULL	https://ridibook
8	<마계대공 연대기>, <21세기 대마법사>의 김...	//img.ridicdn.net/cover/1525004030/xxlarge	NULL	NULL	NULL	https://ridibook
9	F급 헌터. 그것도 잘나가는 S급 동생 말목이나 ...	//img.ridicdn.net/cover/2065034918/xxlarge	NULL	NULL	NULL	https://ridibook
10	무능하고 한 많은 인생이 끄트머리에 다다랐다....	//img.ridicdn.net/cover/777075371/xxlarge	NULL	NULL	NULL	https://ridibook
11	세상이 게임처럼 바뀌었다. 좀비와 몬스터가 사...	//img.ridicdn.net/cover/2065034911/xxlarge	NULL	NULL	NULL	https://ridibook
12	멸망한 세상에서 종로 쇠를 두드리던 대장장이 ...	//img.ridicdn.net/cover/1377074376/xxlarge	NULL	NULL	NULL	https://ridibook
13	천 년의 전통을 자랑하면 정파를 순수하게 힘으...	//img.ridicdn.net/cover/2200033520/xxlarge	NULL	NULL	NULL	https://ridibook
14	웹소설 속 주인공을 볼 때마다 생각했다. 나에...	//img.ridicdn.net/cover/2065030978/xxlarge	NULL	NULL	NULL	https://ridibook
15	바로 그 순간, 내 눈에 동물의 혈관과 장기가 보...	//img.ridicdn.net/cover/425168347/xxlarge	NULL	NULL	NULL	https://ridibook
16	삶의 끝자락에서 돌아왔다. 20대의 청춘. 그때 ...	//img.ridicdn.net/cover/3228015489/xxlarge	NULL	NULL	NULL	https://ridibook
17	눈을 떠보니 소설 속이었다. 그것도 망나니로 ...	//img.ridicdn.net/cover/875126447/xxlarge	NULL	NULL	NULL	https://ridibook
18	「태왕을 보았다」 의 작가 추공. 이번에는 레이...	//img.ridicdn.net/cover/777034313/xxlarge	NULL	NULL	NULL	https://ridibook
19	내 꿈은 표사가 되어 멋진 말을 타고 표물을 호...	//img.ridicdn.net/cover/2057073250/xxlarge	NULL	NULL	NULL	https://ridibook
20	국민랭컴 최종장을 클리어했더니 게임 속 이름 ...	//img.ridicdn.net/cover/1377074419/xxlarge	NULL	NULL	NULL	https://ridibook
21	검으로 환생해 수백년을 살다. 말락 열다섯 망...	//img.ridicdn.net/cover/2107061824/xxlarge	NULL	NULL	NULL	https://ridibook
22	다시 살다. 하고 싶은 일을 하며 멋지게 살아가...	//img.ridicdn.net/cover/1377074411/xxlarge	NULL	NULL	NULL	https://ridibook
23	21세기 중반 한국 청년의 무림 환생기...가 아...	//img.ridicdn.net/cover/3222001104/xxlarge	NULL	NULL	NULL	https://ridibook
24	"Shopper :[명사] 물건을 사는 사람 내 능력은 ...	//img.ridicdn.net/cover/2065034912/xxlarge	NULL	NULL	NULL	https://ridibook
25	재주는 용사가 넘고 보상은 내가 받는다.	//img.ridicdn.net/cover/2065034915/xxlarge	NULL	NULL	NULL	https://ridibook

novel	name
5	초(류희운)
6	산천
7	블루피스
8	김광수
9	근서
10	달의등대
11	슬리버
12	토이카
13	홀로선별
14	그루밤
15	서건주
16	별그림자
17	유려한
18	추공
19	신갈나무
20	기월휠
21	글럼프
22	남희성
23	곽승범
24	오늘도요
25	갯장어
26	우성

그림 33. [novel] 테이블, [author] 테이블 삽입 결과

3) Analysis 결과에 대한 DB 삽입을 위한 Pseudo code (Main function)

32 lines (23 sloc) | 2.5 KB

Raw Blame

```
import module                          # When you run the program, you need 10 minutes of initial start time

def main():
   title_list=DB Open 및 Load title #제목 가져오는 부분
   content_list_daum, content_list_tistory ...etc = []              #module화된 함수를 실행시켰을때 내용 list 저장할 용도
   count_daum, list_tistory ... etc = 0               #module화된 함수를 실행시켰을때 count list 저장할 용도
   review_list_ridibooks ... etc = []               #module화된 함수를 실행시켰을때 count list 저장할 용도
   for title in range(len(title_list)):               #DB에서 가져온 title_list 각각에 대하여....
      list daum, count_ daum = Blog,Community,SNS Crawler(title)          #함수를 실행시키고 위의 list 및 count 필요하다면 review 까지 리턴 받아 저장한다.

      if title in ridibooks(can get review any...etc) :           #ridibooks와 같이 댓글을 받을 수 있는 플랫폼은 크롤러를 실행시켜 그 결과를 저장
         Execute review_list_ridibooks=ridibooks_review_crawler()

      merge list for make txt file                #저장한 list를 이용하여 content 부분만 merge하여 텍스트 파일을 만들어서 추후에 (1-2), (3)분석에서 활용

      Add all of count for (1-1) analysis
      store 'sum of count' to DB                #return 받은 count를 모두 더해 DB에 저장

      Execute Textrank(txt file 0name) for (1-2) Analysis        #merge된 txt파일을 이용하여 1-2 수행
      store 5 text to DB                #수행 결과인 5개의 txt를 list에 return받고 이것을 DB에 저장

      iftitle in ridibooks(can get review any...etc) :           #ridibooks 등의 조건이 있을때 아래의 함수를 실행시킨다. 그외에는 리뷰없으면 실행시킬 필요가 없으니
         for review in range(len(review_list_ridibooks)): # for (2) Analysis    #return 받은 review_list를 이용하여 predict를 반복적으로 실행할 것인데
            Execute dateList=predict(review)          #만들어놓은 dateList에다가 그 결과를 저장하고
         store predict to DB              #그 dateList를 DB에 저장

      Execute Aspect_Miner(txt_file name) for (3) Analysis        #merge된 txt파일을 이용하여 3 수행
      store 5 text to DB                #수행 결과인 5개의 txt를 list에 return받고 이것을 DB에 저장

   Program closed

# 작성자 : 김성종
```

그림 34. DB-분석 연동 수도코드

라. 사용자 편의성을 고려한 UI를 구성한 웹어플리케이션 제작

- Flask 모듈을 통해 파이썬으로 코드를 구성할 수 있다.
- DB 연동의 경우, 이전에 사용한 pymysql 패키지 사용법과 상당히 유사하다. Connection을 열어 DB 정보를 가져와 sql 문을 execute시켜 원하는 정보를 가져온다. 이 때 출력되는 타입은 List이고 이를 디자인에 적용시킨다.
- 기본적인 기틀은 HTML로 구조를 잡고 세부적인 디자인은 Bootstrap 라이브러리의 CSS를 이용하여 디자인했다.

```
@app.route('/db')
def select():
    db_class = mod_dbconn.Database()

    sql2 = "SELECT title, age, genre, publisher, total, intro, img_src, idnovel \
            FROM webnoveldb.novel"
    row2 = db_class.executeAll(sql2)
    print(row2)

    return render_template('db.html', resultData=row2[0])
```

그림 35. 디비-디자인 연동 코드 中

1) Main menu

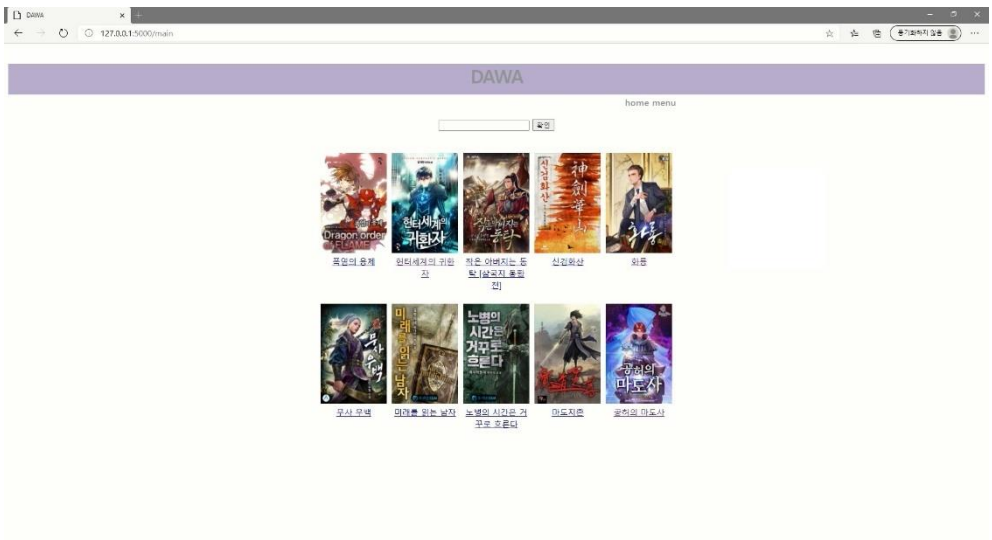

그림 36. 메인 홈 화면

- 이달의 작품 TOP 10은 감성분석 결과 이달의 스코어가 가장 좋은 10작품을 선정하여 [14]메인 페이지에 정렬된다.
- 작품 이미지를 클릭할 시, 해당 [15]작품 정보 페이지로 이동된다.

[14] http://127.0.0.1:5000/main

[15] 27페이지 <그림 34> http://127.0.0.1:5000/content/[novel ID]

- [16]메뉴 페이지에는 아이콘으로 설정한 연재처와 기본 장르 로맨스, 판타지가 메뉴로 설정되어 있다. 해당 아이콘 혹은 메뉴 클릭 시, 관련 작품 리스트로 나열된 페이지로 이동된다.

2) 컨텐츠 리스트 화면 출력

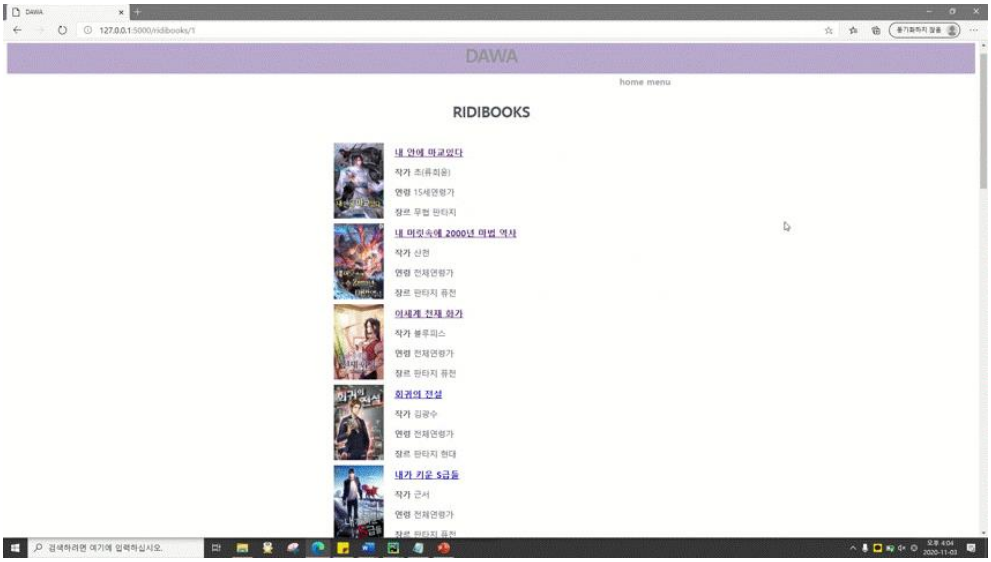

그림 37. 작품 목록 리스트 화면

- 작품 리스트 하단의 페이지의 경우, 더보기란 구현을 통해 정리하였다.
- 연재처별 작품 리스트는 제목순, 장르별 작품 리스트는 감성분석 결과 이달의 점수가 높은 순으로 정렬하였다.
-

[16] http://127.0.0.1:5000/menu

3) 검색 및 결과 화면 출력

그림 38. 작품 정보 페이지

- 현재 DAWA 웹페이지에 작품 DB 를 연동하여 페이지를 구성한다. 이 때 도메인은 작품 번호[17]를 이용하여 작품 페이지를 할당한다.
- 작품의 기본 정보 하단에는 각 분석에 대한 키워드를 최대 5개 나열하여 보여준다.
- 감성분석의 경우, 중간 기중 그래프는 해당 월의 평균 감성분석 점수를 나타내며 작품의 점수는 막대 그래프로 나타냈다. 이를 통해 평균치보다 이 작품에 대해 긍정 혹은 부정적으로 독자가 생각하고 있는지를 사용자는 알 수 있다.

[17] Ex. http://127.0.0.1:5000/content/1

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 추천 받을 수 있다.

IV. 프로젝트의 향후 계획

1. 수행 일정

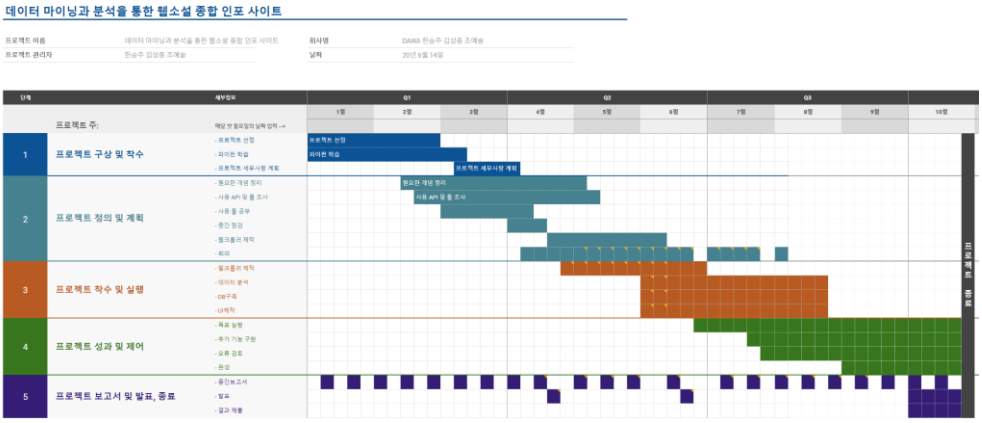

그림 39. 프로젝트 진행 계획

2. 개선 방안

- 각 연재처에서 신작품 정보를 가져올 수 있도록 크롤러를 제작하여 새로운 데이터를 지속적으로 업데이트하고 작품 제목이 아닌 작품 검색에도 분석하여 연관어를 이용한 분석으로 개선을 한다면 보다 더 정확한 분석 데이터를 수집할 수 있을 것이다.
- DB 테이블을 더 세분화하여 정규화를 한다면 독자들에게 더 자세하고 친절한 디자인으로 개선할 수 있을 것이다.
- 각 키워드가 되는 단어를 페이지로 연결시켜 단독 페이지를 만든다면 해당 작품과 비슷한 작품을 알 수 있을 것이다. 예를 들어, 장르 클릭 시, 해당 장르 작품 목록을 확인하거나 주인공 키워드에서 "멋지다"라는 반응이 나왔을 때, "멋지다"라는 반응이 나온 다른 작품을 추천해주는 등으로 발전시킬 수 있다.
- 웹페이지를 서버에 올려 구현한다면 데이터 로딩 시간을 줄일 수 있을 것이다.

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.

- 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 사용된다. 여기서 변형되지 않는 것을 어간, 변하는 부분을 어미라 칭한다. 문장에서 쓰인 단어들을 원형으로 바꿔주기 위해 용언 분석기를 사용한다.
- 용언 분석을 하기 위해서는 말뭉치를 이용할수도 있고 KoNLPy의 클래스로도 원형 변환을 할 수 있다.

```
lemmatizer.lemmatize('차가우니까')
```

```
[('차갑다', 'Adjective')]
```

그림 40. 용언 분석기를 이용한 품사의 원형 복원

2. TextRank 기법

가. PageRank

구글의 검색 엔진에 사용되는 구글이 만든 알고리즘으로 단순 계산을 반복하여 가중치를 매긴다. 각 페이지를 노드라 하고 노드를 연결하는 링크를 에지라 할 때, 각 노드로부터 가중치를 서로 넘겨 받으며 가중치가 결정된다. 이런 과정을 계속 반복하며 중요도에 따라 가중치를 부여하여 서로 간의 연관성을 나타낼 수 있도록 한다.

나. TextRank[18]

- 위의 PageRank를 텍스트에 적용될 수 있도록 단어나 문장을 노드로 설정하여 텍스트 단위로 키워드 추출이나 텍스트 요약에 쓰일 수 있다.
- TextRank(단어의 출현 빈도)의 기하 평균, 두 단어 간의 유사도와 그 길이를 곱하여, 핵심 키워드를 선정하게 된다.

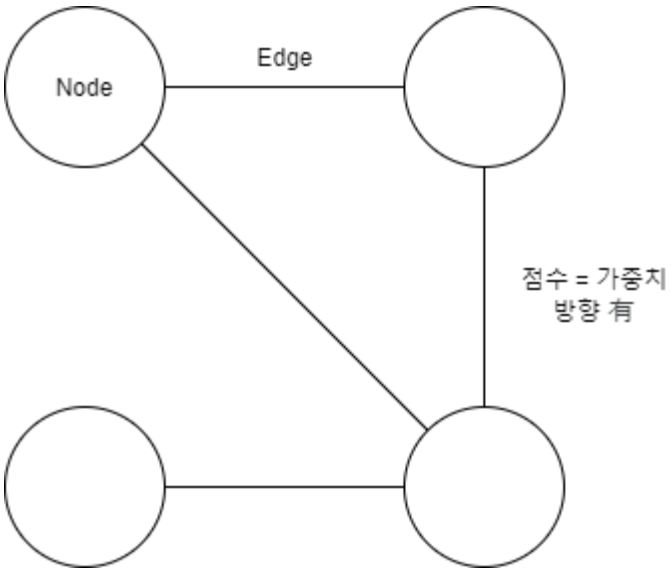

그림 41. PageRank 간단 도식

다. NSMC

네이버 영화의 리뷰 데이터 약 2만 개의 긍부정을 나타낸 말뭉치로 이 데이터를 바탕으로 긍부정에 대한 학습을 시킨 다음 이를 적용시킨다. 우리 프로젝트는 영화 리뷰는 아니지만 스토리, 캐릭터 등에 대해 유사한 형태로 긍부정 평가를 내린다는 점에서 해당 말뭉치로 학습을 시키고 이를 적용시켜도 된다고 판단하게 되었다.

[18] 참고자료 : https://bab2min.tistory.com/552

5. LSTM 알고리즘

과거의 데이터를 통해 현재의 문제를 해결하는 방식의 RNN 알고리즘의 문제는 직전의결과가 아닌 훨씬 이전의 결과를 현재의 문제에 적용한다는 것이다. 그로 인해 두 문제를 연결시키기 힘든데 이를 보완하는 알고리즘이 큰 중심 state에 gate를 만들어 이 gate에서 선택된 정보를 받아 처리하는 LSTM이다.

```
In [40]: from tensorflow.keras.layers import Embedding, Dense, LSTM
         from tensorflow.keras.models import Sequential
         from tensorflow.keras.models import load_model
         from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

In [41]: model = Sequential()
         model.add(Embedding(vocab_size, 100))
         model.add(LSTM(128))
         model.add(Dense(1, activation='sigmoid'))

In [42]: es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=4)
         mc = ModelCheckpoint('best_model.h5', monitor='val_acc', mode='max', verbose=1, save_best_only=True)

In [43]: model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
         history = model.fit(X_train, y_train, epochs=15, callbacks=[es, mc], batch_size=60, validation_split=0.2)

         Epoch 1/15
         1939/1939 [==============================] - ETA: 0s - loss: 0.3903 - acc: 0.8221
         Epoch 00001: val_acc improved from -inf to 0.84468, saving model to best_model.h5
         1939/1939 [==============================] - 47s 24ms/step - loss: 0.3903 - acc: 0.8221 - val_loss: 0.3532 - val_acc: 0.8447
         Epoch 2/15
         1938/1939 [==============================>.] - ETA: 0s - loss: 0.3274 - acc: 0.8572
         Epoch 00002: val_acc improved from 0.84468 to 0.85521, saving model to best_model.h5
         1939/1939 [==============================] - 47s 24ms/step - loss: 0.3274 - acc: 0.8573 - val_loss: 0.3347 - val_acc: 0.8552
         Epoch 3/15
         1937/1939 [==============================>.] - ETA: 0s - loss: 0.3024 - acc: 0.8721
         Epoch 00003: val_acc improved from 0.85521 to 0.85741, saving model to best_model.h5
         1939/1939 [==============================] - 46s 24ms/step - loss: 0.3023 - acc: 0.8721 - val_loss: 0.3308 - val_acc: 0.8574
         Epoch 4/15
         1937/1939 [==============================>.] - ETA: 0s - loss: 0.2844 - acc: 0.8806
         Epoch 00004: val_acc improved from 0.85741 to 0.85875, saving model to best_model.h5
         1939/1939 [==============================] - 48s 25ms/step - loss: 0.2844 - acc: 0.8806 - val_loss: 0.3266 - val_acc: 0.8587
         Epoch 5/15
         1938/1939 [==============================>.] - ETA: 0s - loss: 0.2688 - acc: 0.8890
         Epoch 00005: val_acc improved from 0.85875 to 0.85978, saving model to best_model.h5
         1939/1939 [==============================] - 48s 25ms/step - loss: 0.2687 - acc: 0.8890 - val_loss: 0.3288 - val_acc: 0.8598
         Epoch 6/15
         1938/1939 [==============================>.] - ETA: 0s - loss: 0.2543 - acc: 0.8963
         Epoch 00006: val_acc did not improve from 0.85978
         1939/1939 [==============================] - 47s 24ms/step - loss: 0.2543 - acc: 0.8963 - val_loss: 0.3380 - val_acc: 0.8583
         Epoch 7/15
         1938/1939 [==============================>.] - ETA: 0s - loss: 0.2397 - acc: 0.9034
         Epoch 00007: val_acc did not improve from 0.85978
         1939/1939 [==============================] - 49s 25ms/step - loss: 0.2397 - acc: 0.9034 - val_loss: 0.3465 - val_acc: 0.8561
         Epoch 8/15
         1939/1939 [==============================] - ETA: 0s - loss: 0.2242 - acc: 0.9108
         Epoch 00008: val_acc did not improve from 0.85978
         1939/1939 [==============================] - 48s 25ms/step - loss: 0.2242 - acc: 0.9108 - val_loss: 0.3591 - val_acc: 0.8534
         Epoch 00008: early stopping

In [51]: def sentiment_predict(new_sentence):
           new_sentence = okt.morphs(new_sentence, stem=True) # 토큰화
           new_sentence = [word for word in new_sentence if not word in stopwords] # 불용어 제거
           encoded = tokenizer.texts_to_sequences([new_sentence]) # 정수 인코딩
           pad_new = pad_sequences(encoded, maxlen = max_len) # 패딩
           score = float(loaded_model.predict(pad_new)) # 예측
           if(score > 0.5):
             print("{:.2f}% 확률로 긍정 리뷰입니다.\n".format(score * 100))
           else:
             print("{:.2f}% 확률로 부정 리뷰입니다.\n".format((1 - score) * 100))

In [52]: sentiment_predict('이 영화 개꿀잼 ㅋㅋㅋ')
         93.41% 확률로 긍정 리뷰입니다.

In [53]: sentiment_predict('이 영화 핵노잼 ㅠㅠㅠ')
         97.95% 확률로 부정 리뷰입니다.

In [54]: sentiment_predict('이딴게 영화냐 ㅉㅉ')
         99.76% 확률로 부정 리뷰입니다.

In [55]: sentiment_predict('와 개쩐다 정말 세계관 최강자들의 영화다')
         66.17% 확률로 긍정 리뷰입니다.
```

그림 42. LSTM 알고리즘을 이용한 분석 결과 예시

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)