

중간보고서

Vol. 17

프로젝트 명 : 데이터 마이닝을 이용한 웹소셜 종합 인포 웹 어플리케이션

지도 교수 : 컴퓨터정보공학부 이기훈
팀 장 : 컴퓨터정보공학부 한승주
팀 원 : 컴퓨터정보공학부 김성종
컴퓨터정보공학부 조예슬

2020. 10. 16



광운대학교
KwangWoon University

목 차

I. 프로젝트의 개요

1. 배경 및 필요성
2. 목표
3. 개발 내용

II. 프로젝트의 내용

1. 설계 및 개발의 내용
2. 역할 분담
3. 최종 결과물

III. 프로젝트의 활용 및 기여

1. 프로젝트 결과물의 활용
2. 프로젝트 결과물의 기여

IV. 프로젝트의 향후 계획

1. 수행 일정
2. 개선 방안

V. 별첨

VI. 참고문헌

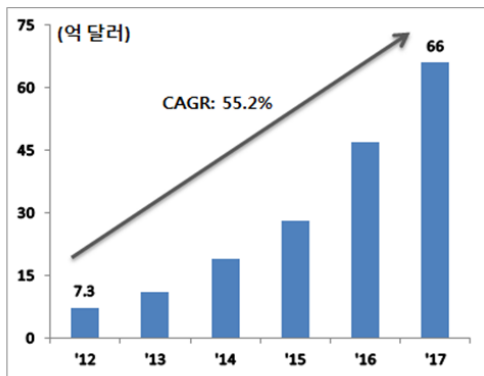
I. 프로젝트의 개요

1. 배경 및 필요성

가. 시장 성장

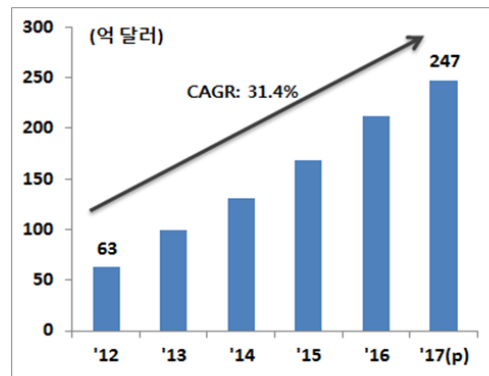
현대에 이르며 미디어 시장은 많은 변화가 이루어졌다. 미디어는 물론 미디어 플랫폼도 다양해지며 현대인들은 더 쉽고 간편하게 빠르고 편리하게 미디어를 접할 수 있게 되었다. CD나 카세트 같은 아날로그식의 접근은 음원 다운로드를 넘어 스트리밍으로 디지털화되었고 영화와 TV 미디어와 같은 영상물도 마찬가지로 영화관, TV를 넘어 다운로드와 VOD 시스템의 활성화를 지나 '넷플릭스', '왓챠플레이', '홀루' 등 스트리밍 플랫폼으로 넘어오며 가히 스트리밍의 시대가 도래했다고 할 수 있다. 이런 변화는 출판 업계도 마찬가지였다. 만화와 소설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

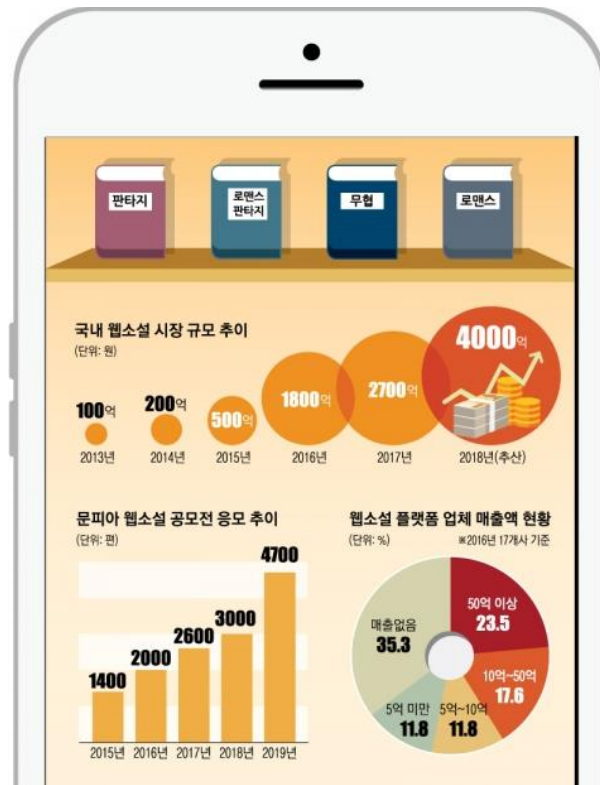


그림 2. 국내 웹소설 시장 규모 추이 (출처 : 서울신문, 2019.05)²

이렇게 커진 웹소설 시장을 방증하듯 독점 연재도 추진하며 인지도 높은 플랫폼만 약 6곳, 그 외에도 여러 작품의 연재 서비스를 제공하는 곳까지 합하면 그 수는 10 곳을 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

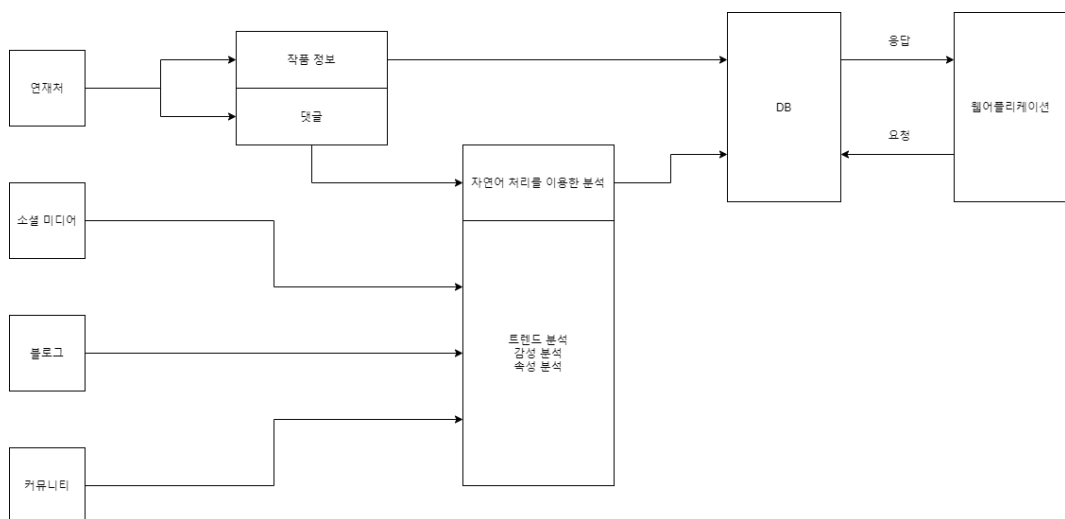


그림 3. 시스템 구조

- 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)

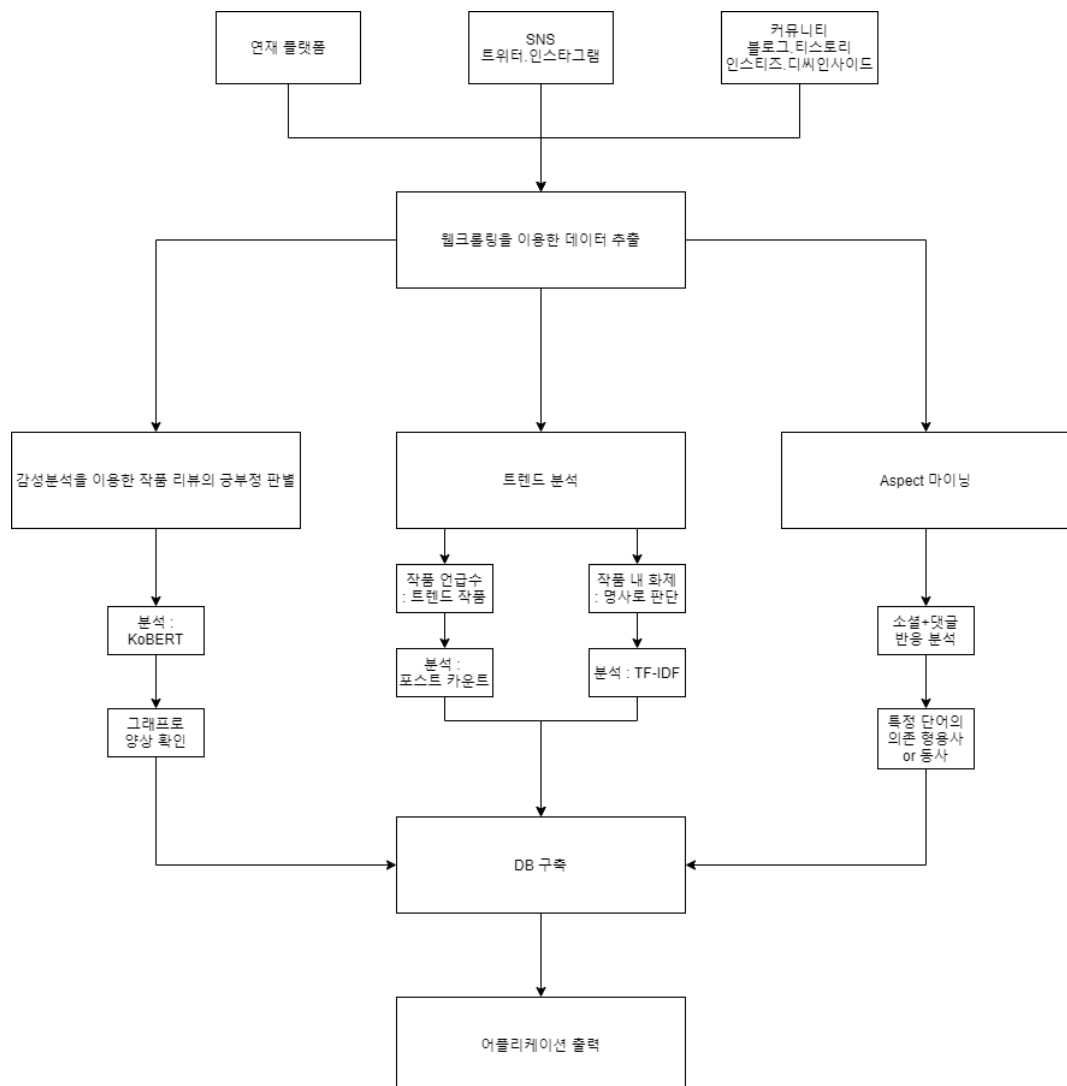


그림 4. Flow Chart

- **[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 단순히 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.

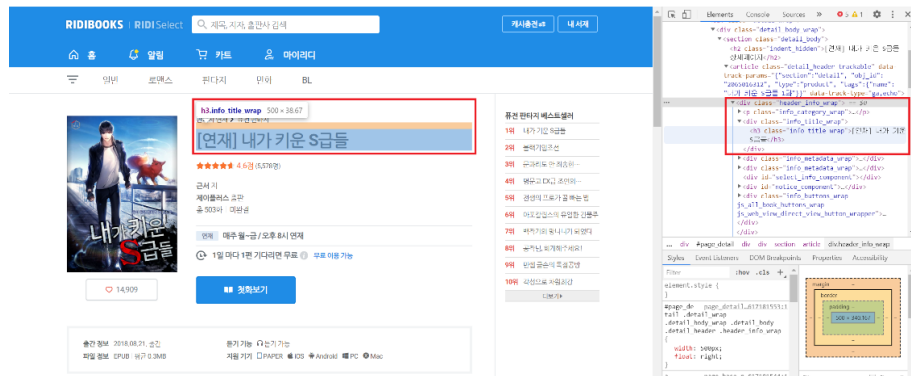


그림 5. 웹사이트 속 HTML 코드 예시

```
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 > h3")
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** 라이브러리를 이용한다.

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

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

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

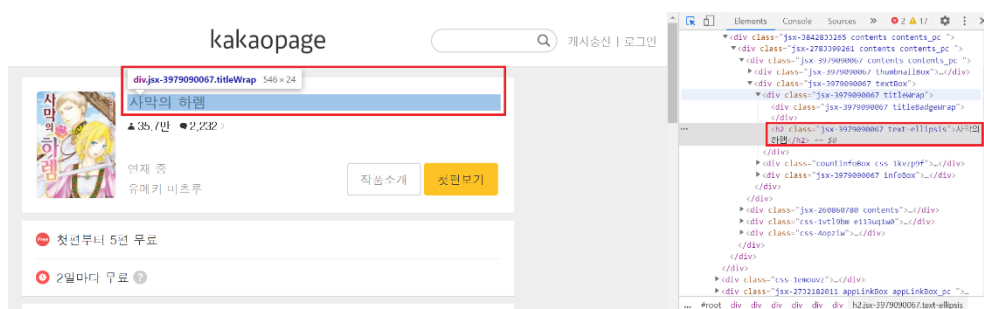


그림 7. 자바스크립트로 렌더링된 웹페이지

```
#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**)]

- 각 사이트와 페이지별로 링크를 재귀적으로 검색하여 필요한 정보를 수집한다.

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** 간격으로 설정하고 추이를 살펴본다. **트위터, 인스타그램, 디씨인사이드, 인스티즈**의 게시물 수를 카운트한다.

[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 선정하고 결과물의 메인 화면에 보여준다.

[2] 작품 내에서 어떤 것이 화제가 되고 있는가

작품에 대해 독자들이 어떤 얘기를 나누는지 1달을 주기로 텍스트 분석을 통해 알아본다. 커뮤니티 및 소셜 미디어의 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 **TextRank 기법**을 이용해 핵심 어구 추출을 통해 분석을 진행한다.

[다] 이 분석을 하기 위한 몇 가지 방법을 생각했다. 첫 번째로 단순히 KoNLPy³를 이용하여 형태소 분석을 하고 여기서 순수히 빈도를 계산하는 BOW⁴방법, 두 번째로 단어의 맥락을 고려한 정확도를 위해 TF-IDF⁵방식을 이용하여 벡터화하는 방식, 세 번째로 **TextRank 기법**⁶을 이용하여 핵심 어구를 추출하는 방법이 있다. 정확성과 사용 방법을 고려하여 **TextRank 기법**을 사용한 분석을 선택했다.

[라] TF-IDF는 단어의 빈도와 단어의 출현의 역수를 가지고 계산하는 방식인데 이 방식은 모든 글을 하나의 텍스트 파일로 파일화한 후 처리하여 단어의 출현이 1이 될 수밖에 없는 문제를 발생시킨다. 이러한 문제로 인해 TextRank 알고리즘을 이용한 분석으로 방법을 변경하는 것으로 결정했다.

³ 별첨 [1] 항목 참조

⁴ 별첨 [2] 항목 참조

⁵ 별첨 [2] 항목 참조

⁶ 별첨 [4] 항목 참조

[마] 위와 같은 분석을 통해 나온 결과에서 **상위 5개의 데이터를 시각화**하여 유저에게 제공할 것이다.

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

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

[1] 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.

[2] 위와 같은 분석을 위한 첫 번째 방법으로 **KoBERT를 활용한 감성 분석**⁷이 있다. 두 번째로는 **텐서플로우와 케라스**를 가지고 **NSMC**⁸를 이용하여 학습을 시킨 뒤 **LSTM 알고리즘**⁹을 이용하여 분석하는 방법이 있다. 이번 프로젝트에서는 후자를 이용하여 분석할 계획이다.

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 반응을 분석한다.

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

⁷ 별첨 [3] 항목 참조

⁸ Naver Sentiment Movie Corpus : <https://github.com/e9t/nsmc>

⁹ 별첨 [5] 항목 참조

[2] 먼저 **군집분석**을 진행한다. 웹소셜 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. 여기에 동시 출현 연관어 분석¹⁰과 형태소 분석을 통해 명사를 표현하는 동사나 형용사에 대해서 벡터를 구성하고 용언 분석기(lemmatize)를 활용하여 '원형'에 대해 count를 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.

4) 웹사이트 UI 제작

- 사람들은 보통 3-5개 이상의 단계를 거쳐 결론에 도달하게 되면 그 인터페이스는 복잡하다고 느낀다. 이러한 점을 적용하여 이번 프로젝트에서 보여주고자 하는 결과물을 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 결과를 보여줄 것이다. 해당

¹⁰ 별첨 [6] 항목 참조

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 사용자가 검색하고 싶은 작품을 검색해 볼 수 있도록 구성하였다.

- 작품 페이지에서는 수집한 작품의 기본 정보와 함께 TextRank 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 Flask)** : 마이크로 웹 프레임워크로 데이터 수집부터 분석, DB 조작에 사용하던 언어인 파이썬을 이용하여 결과물을 구현이 수월할 것이라 생각하여 선정하였다.

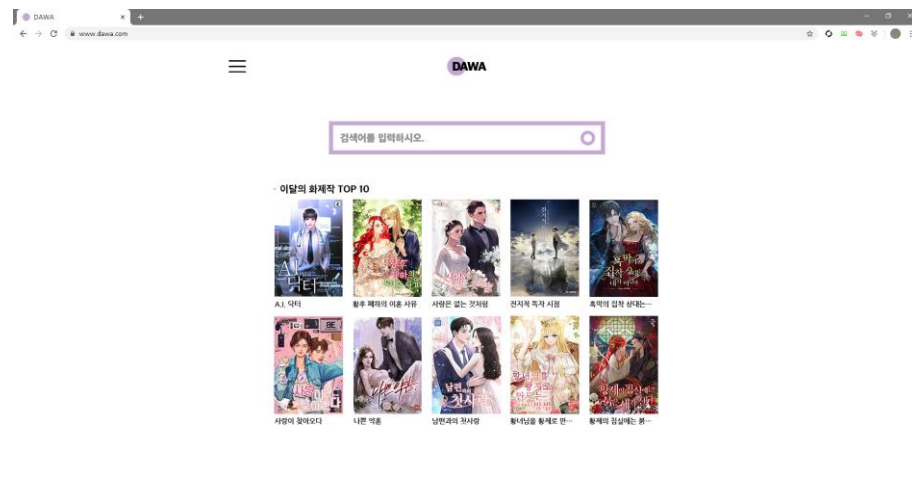


그림 9. 메인화면 예시

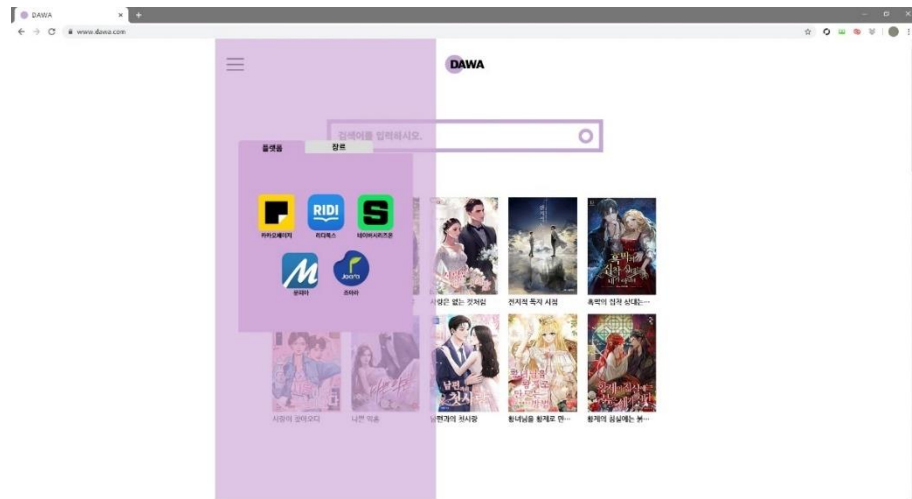


그림 10. 메뉴 화면 디자인 예시

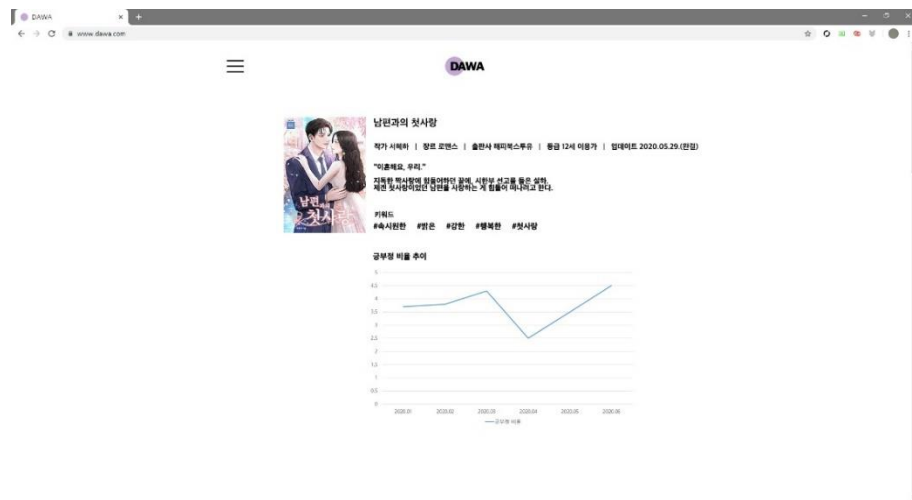


Figure 1. 작품 정보 화면 디자인 예시

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 [김성중, 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 구축

한다.

- 웹 애플리케이션 [조예슬, 김성중] : 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.
김성중 [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(디시인사이드, 인스티즈) 데이터를 수집하는 코드를 작성한다. 웹페이지 구현[조장] : 회의를 통해 구상한 UI의 구체화와 JavaScript, 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] K
#[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]/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 11. 카카오페이지 작품 정보 크롤링 중

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 12. 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 13. 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: 로맨

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 휘타에게 자신을 맡기게 되는데.... 사랑하는 사람을 위해 영혼마저 팔아버린 그들의 가슴 시리도록 아픈 사랑 이야기가 펼쳐집니다.

Process finished with exit code 0

그림 14. 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_list'})
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 15. 네이버 시리즈온 작품 정보 크롤링 중

제목: 입술이 너무해
화수: 106
저자: 갯너
그림: PM
장르: 로맨스
출판사: 와이엠티북스
등급: 전체 이용가
최근 업데이트: 2018.09.21.
완결여부: 완결
별점: 9.8
소개: “키스하면 뽀빠?” 남자가 되는 병에 걸린 지 7년, 그 남자와의 하룻밤은 서연을 다시 여자로 만들었다. 머리카락은 길어지고, 가슴은 볼록해지고, 입술은 더욱더 새빨갳게 피어났다. “예쁜데요.” “네?” “입술 예뻐네.” 성형 대폭발, 심쿵 주의, 치명적으로 섹시한 직진남의 꿀 떨어지는 막강 돌미엄이 시작됐다. 운명에 얽힌 세계 최고 달달한 썸쟁과 더 달달한 끈적끈적 연애! 예측불허 입술 로맨스.

그림 16. 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 > str")
author=author_list[0].text

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

```

그림 17. 조아라 작품 정보 크롤링 중

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

그림 18. 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 19. 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 : 와이키카진
 작품등록일 : 2016.10.13 11:00
 작품등록일 : 2020.04.28 23:43
 추천수 : 3,826,385
 조회수 : 108,882

그림 20. 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 21. 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 연재이메일: 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 22. 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 가능하기에 오픈소스 API를 이용해 2019년 1월 1일부터 약 1년 간의 데이터를 수집하여 분석할 것이다.
- 크롤링에 사용한 API twint¹¹는 기간 설정이 가능하며 설정한 기간에 설정한 키워드를 작성한 트윗을 수집했다.

```
# 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 트윗을 list 타입으로 저장

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

그림 23. 기간에 따른 트위터 검색 정보 추출

<그림23> 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.

¹¹ <https://github.com/twintproject/twint>

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

그림 24. 인스타그램 크롤링 중 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 25. 인스타그램 크롤링 중 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 가을가을하기 때문에 외모모...	2020-04-11
1	https://www.instagram.com/p/B7ivmF3Jr6a/	choinuri17	리셋팅 레이디. 이셀라 에반스 낙서사실 읽은지는 좀 돼서 가을가을하기 때문에 외모모...	2020-01-20
2	https://www.instagram.com/p/ByxvfoFIVk/	gilllip	..이번에도 저와 결혼해 주시겠습니까?..."이번이 두번째로 최악인 청혼이예요....	2019-06-16
3	https://www.instagram.com/p/B8BazVend_w/	raaaheen	2020.01. [완독] 리셋팅레이디회귀를 토폰. 회귀도 토폰도 새롭게 해석한 소설...	2020-02-01
4	https://www.instagram.com/p/B8ekVDkHyMx/	ilikehouse1	#리셋팅레이디 #리디북스 #로맨스판타지소설 #자서전 #회귀를 #피해를 #완결 #완결...	2020-02-13
5	https://www.instagram.com/p/Bj8_8GIHh_y/	happy_hji	#리셋팅레이디 등장인물들마저도 깨달음이 취져. 시은경. 밑바닥 인생에서 온 되는 일이면...	2018-06-13
6	https://www.instagram.com/p/CAIvqHlgSO6/	u_u9oo	한줄 감상 : 불닭볶음면보다 매운 로맨스릴러.....🔥.『리셋팅 레이디』, 자서전...	2020-05-14
7	https://www.instagram.com/p/B73RjhnQwX/	roommr_0202	캐런은 117세 생일을 맞이하여 살인자가 되기로 결심했다.#리셋팅레이디	2020-01-29

그림 26. 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 때문이다.

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

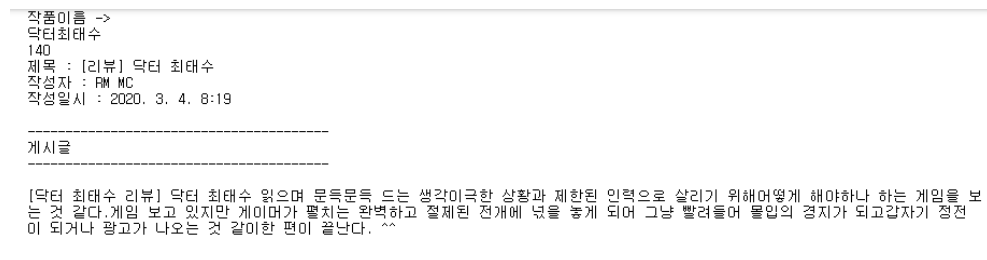


그림 27. 네이버 블로그 크롤링 - 추출 결과 예시

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

- 네이버 블로그와 마찬가지로 검색어 결과로 나온 글들의 링크를 통해 작성글 제목, 작성자, 작성 일시를 출력한다.

- 게시물마다 구성 방식이 달라 이에 따라 여러 버전의 크롤링 코드를 구성하였다.

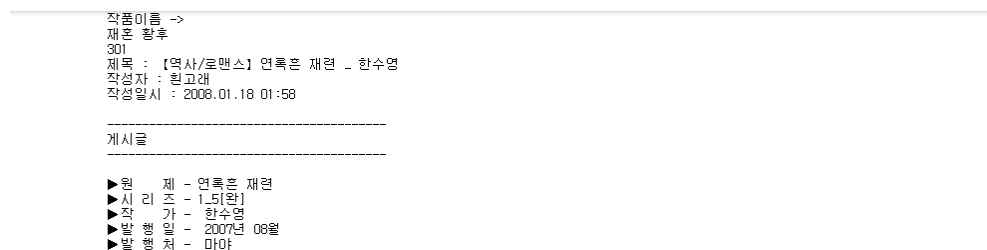


그림 28. 다음 블로그 크롤링 결과 예시

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_url 을 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.

```
제목: 예의를 좀 아는 그랜저 마스터 연예인은?  
글쓴이: 없음  
날짜: 20.05.12  
조회수: -  
추천수: -  
-----  
제목: 장갈 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] 장갈 통합 공지 0.8
```

차단 및 삭제 대상글의 문종소설 제목 말 안 하고 튀기소설과 전혀 무관한 억박이나 잡담 (막줄에 소설이야기 알아두는 것도 포함)참독 어그로타겟 올 피오기특히 특정 소설 얘기 없는 처1,2년, TS 발골은 댓글로 호응해 주는 것도 차단함 (글쓰는 7일 차단, 댓글 단 3일 차단)특정 소설 얘기 없는 백합,보림종 글장차의 성인 글(차단)박시 뇌졸중언급이런 소설 없나? 등등의 제목으로 소설과 관련없는 글 올리려고 쓰는 글 (차단)소설이랑 관련 없는 섹스 만화(3일 차단)----- 영문간 소설 관련 없는 발골들은 최소 1일 차단 하겠음 ----- 영문간 연독록,구매수 언급하는 글들 최소 1일 차단 하겠음 -----작가 홍보는 15화 이상 쓴 글만 해당 글 제외 작가 티 내지 마셈차단 단어,아이피http s://gall.dcinside.com//mgallery/board/view/?id=genrenovel&no=698038

```
제목: 장마갈 신문고  
글쓴이: Qwer1234a  
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] 장마갈 신문고앱에서 작성
```

환장 호출됩니다 댓글달 때 링크 뒤에 글자 붙이지 말아주세요 기본적으로 새벽반이긴 한데 새벽 아날 때 사용해도 상관 없습니다 올
은 예) <https://m.dcinside.com/board/genrenovel/708312> 삭제 좀 틀린 예) <https://m.dcinside.com/board/genrenovel/708312삭제> 좀

그림 29. 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.

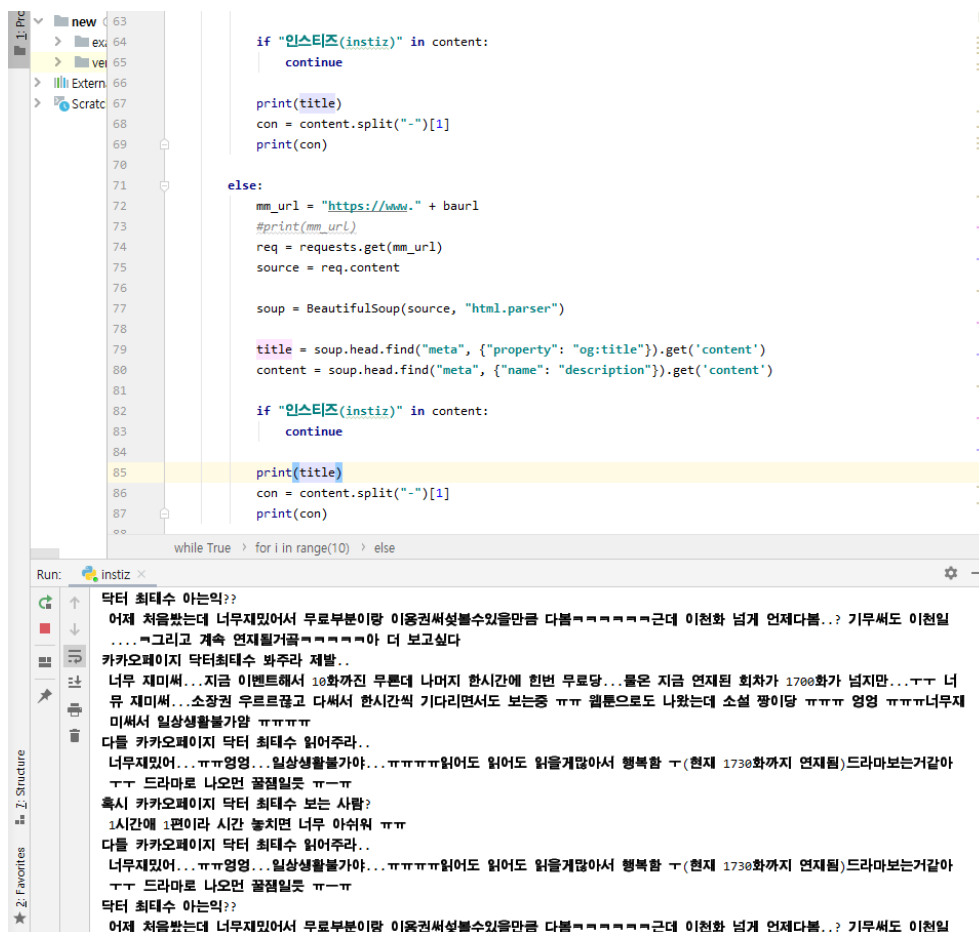


그림 30. 인스티즈 크롤링 코드 中 / 크롤링 결과

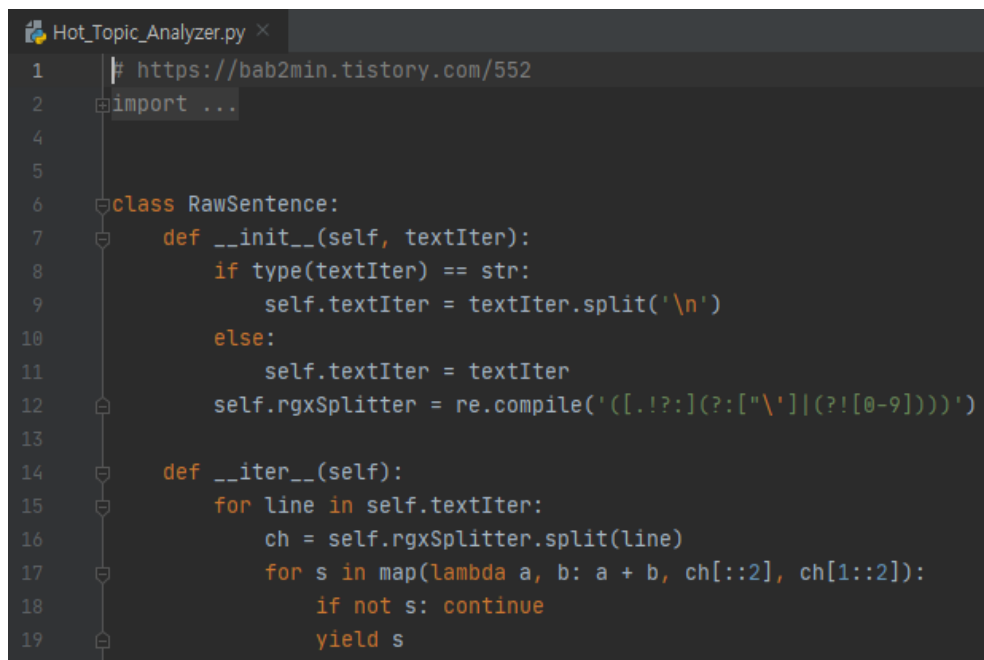
나. 데이터 분석을 위한 전처리과정과 분석 코드 제작

1) 트렌드 분석 : TextRank 알고리즘을 활용하여 핵심 키워드를 추출

- 간단한 예시 텍스트를 시험해 본 결과 작품에 대한 키워드는 추출 가능하지만 작품과 관련된 이슈와 관련한 키워드는 추출이 되지 않을 수도 있다고 추측된다.
- 이 알고리즘을 이용하면 핵심 문장 또한 추출해낼 수 있는데 독자들의 핵심 반응을 추출하는데 활용할 수 있을 것으로 기대되는 것으로 보아 추가적인 분석 적용도 생각해 볼 수 있을 것이다.

[가] 핵심 키워드 추출

- 작품 이름을 검색시, 두 단어 간의 유사도에 따라 n-gram을 만족하도록 하였다.
- 데이터베이스에 저장할 5개의 단어를 얻을 수 있도록 코드를 수정하였다.



```
Hot_Topic_Analyzer.py x
1 # https://bab2min.tistory.com/552
2 import ...
3
4
5
6 class RawSentence:
7     def __init__(self, textIter):
8         if type(textIter) == str:
9             self.textIter = textIter.split('\n')
10        else:
11            self.textIter = textIter
12        self.rgxSplitter = re.compile('[.!?:](?:"\'|(?![0-9]))')
13
14    def __iter__(self):
15        for line in self.textIter:
16            ch = self.rgxSplitter.split(line)
17            for s in map(lambda a, b: a + b, ch[::2], ch[1::2]):
18                if not s: continue
19                yield s
```

그림 31. TextRank를 이용한 키워드 추출

```
In [2]: from Hot_Topic_Analyzer import hot_topic_analyzer
        hot_topic_analyzer('구르미 그린 달빛')
```

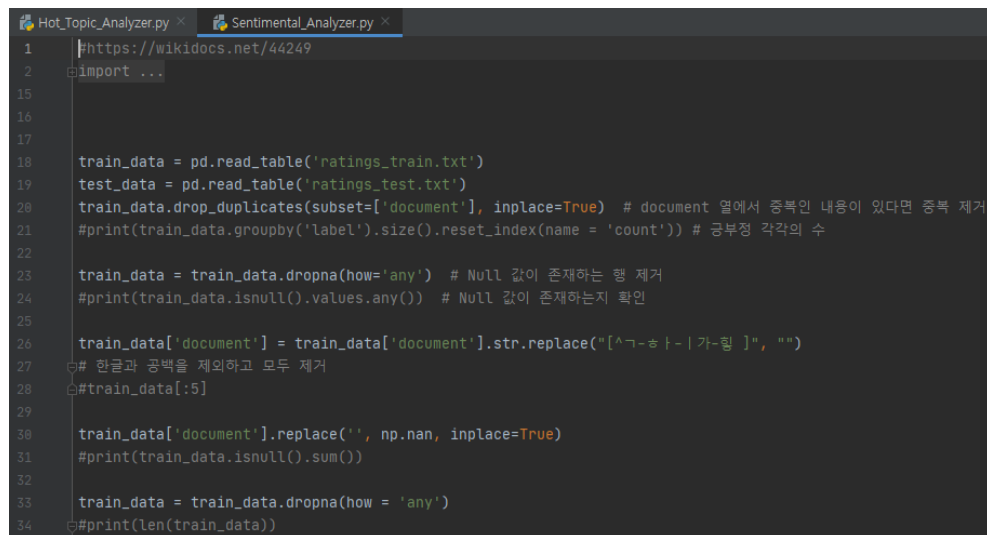
```
Load...
Build...
라온
윤 이수
드라마
병 연
화초 저하
```

그림 32. <그림31> 분석 코드를 이용한 결과 예시

추후 이를 활용한 코드 작성 완료 시 보완 예정이다.

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

- 오피니언 마이닝(감성 분석)을 이용하여 리뷰의 긍부정을 판단하고 평가에 대한 점수를 기간에 대한 그래프로 표현하여 시간이 지남에 따라 작품의 평가 변화 양상을 확인한다.
- KoBERT 의 경우 Windows 환경보다는 Linux 혹은 구글에서 제공하는 Colab, Transformer 등에서 보다 더 정확한 결과를 도출하지만 다른 팀원들과 동일한 환경에서의 작업을 위해 KoBERT 대신 TensorFlow 와 Keras 를 활용한 분석으로 변경하였다. 변경된 방법의 분석은 KoBERT 보다 조금은 떨어진 85.5% 정도의 정확도를 가지긴 하지만 기존 방식이 긍부정의 비율을 보여준다면 현재 활용하고 있는 방식은 긍정의 확률을 보여주기 때문에 변칙적인 확률을 보여줄 수 있어 기존에 우리 프로젝트에서 계획했던 방향에 더 부합한 결과를 보여준다.
- 머신러닝을 위해 수행해야할 코드가 많아 해당 부분을 모듈화함으로써 import 로 불러와 자동으로 모델링을 구성하고 predict 함수를 사용하여 결과를 볼 수 있도록 구성하였다.



```
1 | https://wikidocs.net/44249
2 | import ...
15
16
17
18 | train_data = pd.read_table('ratings_train.txt')
19 | test_data = pd.read_table('ratings_test.txt')
20 | train_data.drop_duplicates(subset=['document'], inplace=True) # document 열에서 중복된 내용이 있다면 중복 제거
21 | #print(train_data.groupby('label').size().reset_index(name = 'count')) # 긍부정 각각의 수
22
23 | train_data = train_data.dropna(how='any') # Null 값이 존재하는 행 제거
24 | #print(train_data.isnull().values.any()) # Null 값이 존재하는지 확인
25
26 | train_data['document'] = train_data['document'].str.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣]", "")
27 | # 한글과 공백을 제외하고 모두 제거
28 | #train_data[:5]
29
30 | train_data['document'].replace('', np.nan, inplace=True)
31 | #print(train_data.isnull().sum())
32
33 | train_data = train_data.dropna(how = 'any')
34 | #print(len(train_data))
```

그림 33. 감성분석 코드

```
In [2]: import Sentimental_Analyzer as SA
#약 10분 소요

C:\Users\maruk\Anaconda3\lib\site-packages\numpy\core\asarray.py:83: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray
    return array(a, dtype, copy=False, order=order)

Epoch 1/15
1937/1939 [=====>.] - ETA: 0s - loss: 0.3894 - acc: 0.8236
Epoch 00001: val_acc improved from -inf to 0.84393, saving model to best_model.h5
1939/1939 [=====] - 54s 23ms/step - loss: 0.3894 - acc: 0.8236 - val_loss: 0.3534 - val_acc: 0.8439
Epoch 2/15
1939/1939 [=====] - ETA: 0s - loss: 0.3281 - acc: 0.8574
Epoch 00002: val_acc improved from 0.84393 to 0.84792, saving model to best_model.h5
1939/1939 [=====] - 49s 25ms/step - loss: 0.3281 - acc: 0.8574 - val_loss: 0.3437 - val_acc: 0.8479
Epoch 3/15
1937/1939 [=====>.] - ETA: 0s - loss: 0.3027 - acc: 0.8706
Epoch 00003: val_acc improved from 0.84792 to 0.85765, saving model to best_model.h5
1939/1939 [=====] - 51s 26ms/step - loss: 0.3027 - acc: 0.8706 - val_loss: 0.3319 - val_acc: 0.8576
Epoch 4/15
1937/1939 [=====>.] - ETA: 0s - loss: 0.2838 - acc: 0.8814
Epoch 00004: val_acc improved from 0.85765 to 0.85951, saving model to best_model.h5
1939/1939 [=====] - 51s 26ms/step - loss: 0.2838 - acc: 0.8814 - val_loss: 0.3253 - val_acc: 0.8595
Epoch 5/15
1937/1939 [=====>.] - ETA: 0s - loss: 0.2673 - acc: 0.8894
Epoch 00005: val_acc did not improve from 0.85951
1939/1939 [=====] - 52s 27ms/step - loss: 0.2672 - acc: 0.8894 - val_loss: 0.3298 - val_acc: 0.8587
Epoch 6/15
1938/1939 [=====>.] - ETA: 0s - loss: 0.2524 - acc: 0.8974
Epoch 00006: val_acc did not improve from 0.85951
1939/1939 [=====] - 49s 25ms/step - loss: 0.2524 - acc: 0.8974 - val_loss: 0.3383 - val_acc: 0.8570
Epoch 7/15
1937/1939 [=====>.] - ETA: 0s - loss: 0.2377 - acc: 0.9040
Epoch 00007: val_acc did not improve from 0.85951
1939/1939 [=====] - 48s 25ms/step - loss: 0.2376 - acc: 0.9040 - val_loss: 0.3437 - val_acc: 0.8549
Epoch 8/15
1938/1939 [=====>.] - ETA: 0s - loss: 0.2222 - acc: 0.9110
Epoch 00008: val_acc did not improve from 0.85951
1939/1939 [=====] - 48s 25ms/step - loss: 0.2221 - acc: 0.9111 - val_loss: 0.3647 - val_acc: 0.8494
Epoch 00009: early stopping
1532/1532 [=====] - 6s 4ms/step - loss: 0.3321 - acc: 0.8540

테스트 정확도: 0.8540

In [3]: SA.predict('이 영화 개꿀잼 ㅋㅋㅋ')
30
93.51% 긍정도를 보이는 리뷰입니다.

In [5]: SA.predict('이 영화 핵노잼 ㅠㅠ')
30
1.10% 긍정도를 보이는 리뷰입니다.
```

그림 34. <그림 33> 코드를 이용한 리뷰 분석 결과 예시

```
def make_date_list():
    date_array = [['2010-01-01', 0, 0], ['2010-02-01', 0, 0], ['2010-03-01', 0, 0], ['2010-04-01', 0, 0],
                  ['2010-05-01', 0, 0], ['2010-06-01', 0, 0],
                  ['2010-07-01', 0, 0], ['2010-08-01', 0, 0], ['2010-09-01', 0, 0], ['2010-10-01', 0, 0],
                  ['2010-11-01', 0, 0], ['2010-12-01', 0, 0],
                  ['2011-01-01', 0, 0], ['2011-02-01', 0, 0], ['2011-03-01', 0, 0], ['2011-04-01', 0, 0],
                  ['2011-05-01', 0, 0], ['2011-06-01', 0, 0],
                  ['2011-07-01', 0, 0], ['2011-08-01', 0, 0], ['2011-09-01', 0, 0], ['2011-10-01', 0, 0],
                  ['2011-11-01', 0, 0], ['2011-12-01', 0, 0],
                  ['2012-01-01', 0, 0], ['2012-02-01', 0, 0], ['2012-03-01', 0, 0], ['2012-04-01', 0, 0],
                  ['2012-05-01', 0, 0], ['2012-06-01', 0, 0],
                  ['2012-07-01', 0, 0], ['2012-08-01', 0, 0], ['2012-09-01', 0, 0], ['2012-10-01', 0, 0],

    def date_list_pos(date):
        num = re.compile('^ 0-9+')
        date = num.sub('', date).replace(' ', '') # 한글과 띄어쓰기를 제외한 모든 부분을 제거
        year = int(date[2:4])
        month = int(date[4:6])
        pos = (year - 10) * 12 + (month - 1)

    return pos
```

그림 35. 기간별 분석을 위한 date_list.py

추후 이를 활용한 코드 작성 완료 시 보완 예정이다.

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

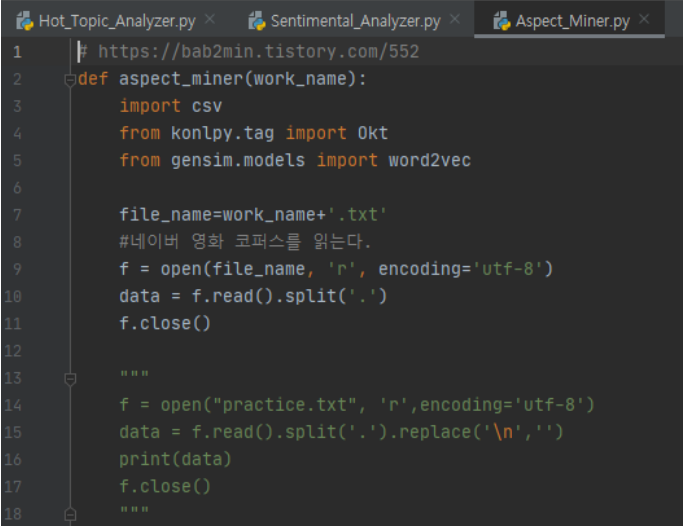
- Aspect Mining 이란 속성(Asspect)을 나타내는 특정 키워드와 관련된 원소(Element), 즉 속성 값을 추출하는 기법이다. 웹소셜에서 주요 요소인 '주인공', '스토리', '분위기'를 키워드라고 한다면 속성 값은 주인공은 '밝다', 스토리는 '진부하다', 분위기는 '칙칙하다'의 형용사와 동사들이다.

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

[가] 군집분석

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

- word2vec 을 수행하기 위해 사전에 처리해야 할 코드를 모듈로 작성하면 작품의 글이 담긴 txt 파일로 word2vec 을 수행하고 해당 작품의 aspect 를 추출하도록 코드를 수정하였다.



```
Hot_Topic_Analyzer.py x Sentimental_Analyzer.py x Aspect_Miner.py x
1 | https://bab2min.tistory.com/552
2 def aspect_miner(work_name):
3     import csv
4     from konlpy.tag import Okt
5     from gensim.models import word2vec
6
7     file_name=work_name+'.txt'
8     #네이버 영화 코퍼스를 읽는다.
9     f = open(file_name, 'r', encoding='utf-8')
10    data = f.read().split('.')
11    f.close()
12
13    """
14    f = open("practice.txt", 'r', encoding='utf-8')
15    data = f.read().split('.').replace('\n', '')
16    print(data)
17    f.close()
18    """
```

그림 36. word2vec 모듈

```
In [6]: from Aspect_Miner import aspect_miner
aspect_miner('닥터 최태수')

Word2Vec Modeling finished
있다
재밌다
뛰어나다
같다
어떻다
```

그림 37. <그림 36> 코드를 이용한 분석 결과 예시

추후 이를 활용한 코드 작성 완료 시 보완 예정이다.

다. 수집한 데이터와 분석 결과를 저장할 DB 제작

1) Logical-ERD

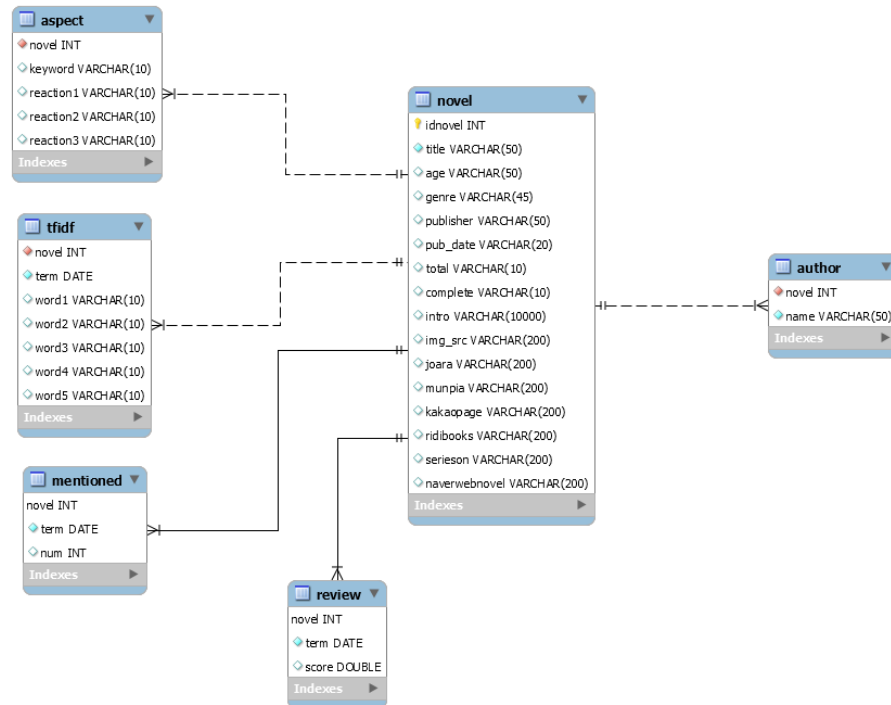


그림 38. Logical-ERD

[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 Null 로, 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 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. [댓글공부정도 테이블]은 댓글과 리뷰의 공부정 양상을 저장하는 테이블로 기간과 공부정도 비율을 속성으로 가진다. [리뷰 단어 테이블]은 리뷰 속 빈출 단어를 저장하는 테이블로 단어와 분석 결과를 속성으로 가진다. [aspect 테이블]은 기준 속성 단어와 그에 대한 반응 키워드를 속성으로 가진다.

2) DB 구축

위에서 구성한 ERD 를 바탕으로 MySQL 워크벤치를 이용하여 테이블을 구성하였다.

작품 테이블에서 age(연령등급)는 [전체 연령가, 15세 연령가, 19세 연령가] 3가지로 나타낸다. pub_date(출간일)은 [YYYY-MM-DD]와 같은 양식으로 삽입하고 추후 date 타입으로 변경한다. complete(완결여부)는 [0: 미완결, 1: 완결]로 나타낸다.

위와 같이 테이블을 구성한 다음 <그림 46>과 같이 파이썬의 pymysql 모듈을 이용하여 csv 파일로 저장한 작품 정보를 데이터베이스에 삽입했다.

```

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 39. 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 38. Pymysql을 이용한 DB 삽입

id	title	author	genre	publisher	date	date	total	complete	rate	url	score	margin	isbn	isbn
5	내 인생의 가장 아름다운 날	전지현	소설	문학	2019-07-11	275	0	0	0	전지현의 첫 장편소설. 전지현의 대표 단편이 실려 있다.	4.5	0.0	9781100000000	9781100000000
6	내 인생의 가장 아름다운 날	전지현	소설	문학	2019-07-14	108	1	1	1	전지현의 첫 장편소설. 전지현의 대표 단편이 실려 있다.	4.5	0.0	9781100000000	9781100000000
7	내 인생의 가장 아름다운 날	전지현	소설	문학	2019-07-14	108	1	1	1	전지현의 첫 장편소설. 전지현의 대표 단편이 실려 있다.	4.5	0.0	9781100000000	9781100000000
8	내 인생의 가장 아름다운 날	전지현	소설	문학	2019-07-14	108	1	1	1	전지현의 첫 장편소설. 전지현의 대표 단편이 실려 있다.	4.5	0.0	9781100000000	9781100000000
9	내 인생의 가장 아름다운 날	전지현	소설	문학	2019-07-14	108	1	1	1	전지현의 첫 장편소설. 전지현의 대표 단편이 실려 있다.	4.5	0.0	9781100000000	9781100000000
10	내 인생의 가장 아름다운 날	전지현	소설	문학	2019-07-14	108	1	1	1	전지현의 첫 장편소설. 전지현의 대표 단편이 실려 있다.	4.5	0.0	9781100000000	9781100000000
11	내 인생의 가장 아름다운 날	전지현	소설	문학	2019-07-14	108	1	1	1	전지현의 첫 장편소설. 전지현의 대표 단편이 실려 있다.	4.5	0.0	9781100000000	9781100000000
12	내 인생의 가장 아름다운 날	전지현	소설	문학	2019-07-14	108	1	1	1	전지현의 첫 장편소설. 전지현의 대표 단편이 실려 있다.	4.5	0.0	9781100000000	9781100000000
13	내 인생의 가장 아름다운 날	전지현	소설	문학	2019-07-14	108	1	1	1	전지현의 첫 장편소설. 전지현의 대표 단편이 실려 있다.	4.5	0.0	9781100000000	9781100000000
14	내 인생의 가장 아름다운 날	전지현	소설	문학	2019-07-14	108	1	1	1	전지현의 첫 장편소설. 전지현의 대표 단편이 실려 있다.	4.5	0.0	9781100000000	9781100000000
15	내 인생의 가장 아름다운 날	전지현	소설	문학	2019-07-14	108	1	1	1	전지현의 첫 장편소설. 전지현의 대표 단편이 실려 있다.	4.5	0.0	9781100000000	9781100000000
16	내 인생의 가장 아름다운 날	전지현	소설	문학	2019-07-14	108	1	1	1	전지현의 첫 장편소설. 전지현의 대표 단편이 실려 있다.	4.5	0.0	9781100000000	9781100000000
17	내 인생의 가장 아름다운 날	전지현	소설	문학	2019-07-14	108	1	1	1	전지현의 첫 장편소설. 전지현의 대표 단편이 실려 있다.	4.5	0.0	9781100000000	9781100000000
18	내 인생의 가장 아름다운 날	전지현	소설	문학	2019-07-14	108	1	1	1	전지현의 첫 장편소설. 전지현의 대표 단편이 실려 있다.	4.5	0.0	9781100000000	9781100000000
19	내 인생의 가장 아름다운 날	전지현	소설	문학	2019-07-14	108	1	1	1	전지현의 첫 장편소설. 전지현의 대표 단편이 실려 있다.	4.5	0.0	9781100000000	9781100000000
20	내 인생의 가장 아름다운 날	전지현	소설	문학	2019-07-14	108	1	1	1	전지현의 첫 장편소설. 전지현의 대표 단편이 실려 있다.	4.5	0.0	9781100000000	9781100000000
21	내 인생의 가장 아름다운 날	전지현	소설	문학	2019-07-14	108	1	1	1	전지현의 첫 장편소설. 전지현의 대표 단편이 실려 있다.	4.5	0.0	9781100000000	9781100000000
22	내 인생의 가장 아름다운 날	전지현	소설	문학	2019-07-14	108	1	1	1	전지현의 첫 장편소설. 전지현의 대표 단편이 실려 있다.	4.5	0.0	9781100000000	9781100000000
23	내 인생의 가장 아름다운 날	전지현	소설	문학	2019-07-14	108	1	1	1	전지현의 첫 장편소설. 전지현의 대표 단편이 실려 있다.	4.5	0.0	9781100000000	9781100000000
24	내 인생의 가장 아름다운 날	전지현	소설	문학	2019-07-14	108	1	1	1	전지현의 첫 장편소설. 전지현의 대표 단편이 실려 있다.	4.5	0.0	9781100000000	9781100000000
25	내 인생의 가장 아름다운 날	전지현	소설	문학	2019-07-14	108	1	1	1	전지현의 첫 장편소설. 전지현의 대표 단편이 실려 있다.	4.5	0.0	9781100000000	9781100000000
26	내 인생의 가장 아름다운 날	전지현	소설	문학	2019-07-14	108	1	1	1	전지현의 첫 장편소설. 전지현의 대표 단편이 실려 있다.	4.5	0.0	9781100000000	9781100000000

novel	name
5	초(류희운)
6	산전
7	블루피스
8	김광수
9	근서
10	달의등대
11	슬리버
12	로이카
13	홀로선별
14	그루발
15	서건주
16	별그림자
17	유려한
18	추공
19	신갈나무
20	기원할
21	클럼프
22	남회선
23	곽승범
24	오늘도요
25	갯장어
26	우성

그림 39. [novel] 테이블, [author] 테이블 삽입 결과

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

```

32 lines (23 sloc) | 2.5 KB
Raw Blame

1 import module # When you run the program, you need 10 minutes of initial start time
2
3 def main():
4     title_list=DB Open 및 Load title #제목 가져오는 부분
5     content_list_daum, content_list_tistory ...etc = [] #module화된 함수를 실행시켰을때 내용 list 저장할 용도
6     count_daum, list_tistory ... etc = 0 #module화된 함수를 실행시켰을때 count list 저장할 용도
7     review_list_ridibooks ... etc = [] #module화된 함수를 실행시켰을때 count list 저장할 용도
8     for title in range(len(title_list)): #DB에서 가져온 title_list 각각에 대하여...
9         list daum, count_daum = Blog,Community,SMS Crawler(title) #함수를 실행시키고 위의 list 및 count 필요하다면 review 까지 리턴 받아 저장한다.
10
11     if title in ridibooks(can get review any...etc) : #ridibooks와 같이 댓글을 받을 수 있는 플랫폼은 크롤러를 실행시켜 그 결과를 저장
12         Execute review_list_ridibooks=ridibooks_review_crawler()
13
14     merge list for make txt file #저장한 list를 이용하여 content 부분만 merge하여 텍스트 파일을 만들어서 후후에 (1-2), (3)분석에서 활용
15
16     Add all of count for (1-1) analysis
17     store 'sum of count' to DB #return 받은 count를 모두 더해 DB에 저장
18
19     Execute Textrank(txt file 0name) for (1-2) Analysis #merge된 txt파일을 이용하여 1-2 수행
20     store 5 text to DB #수행 결과인 5개의 txt를 list에 return받고 이것을 DB에 저장
21
22     if title in ridibooks(can get review any...etc) : #ridibooks 등의 조건이 있을때 아래의 함수를 실행시킨다. 그외에는 리뷰 없으면 실행시킬 필요가 없으니
23     for review in range(len(review_list_ridibooks)): # for (2) Analysis #return 받은 review_list를 이용하여 predict를 반복적으로 실행할 것인데
24         Execute datelist=predict(review) #만들어놓은 datelist에다가 그 결과를 저장하고
25         store predict to DB #그 datelist를 DB에 저장
26
27     Execute Aspect_Miner(txt_file name) for (3) Analysis #merge된 txt파일을 이용하여 3 수행
28     store 5 text to DB #수행 결과인 5개의 txt를 list에 return받고 이것을 DB에 저장
29
30     Program closed
31
32 # 작성자 : 김성중

```

라. 사용자 편의성을 고려한 UI를 구성한 웹어플리케이션 제작

- Flask 모듈을 통해 파이썬으로 코드를 구성할 수 있다.
- Flask 패키지를 통해 flask 모듈을 import한다.
- flask 객체를 생성하고 정의한 run 함수를 이용하여 웹페이지를 구성한다.

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def run():
    return 'hello world'

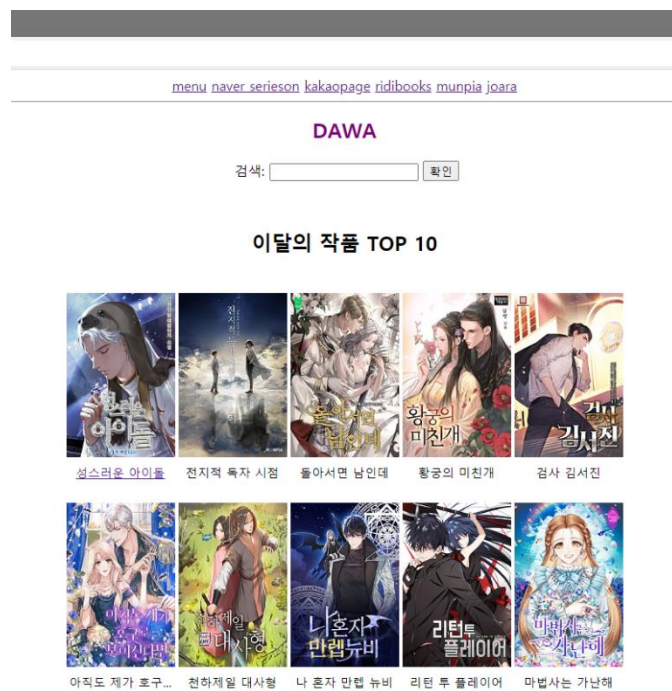
if __name__ == "__main__":
    app.run()
```

hello world

127.0.0.1:5000

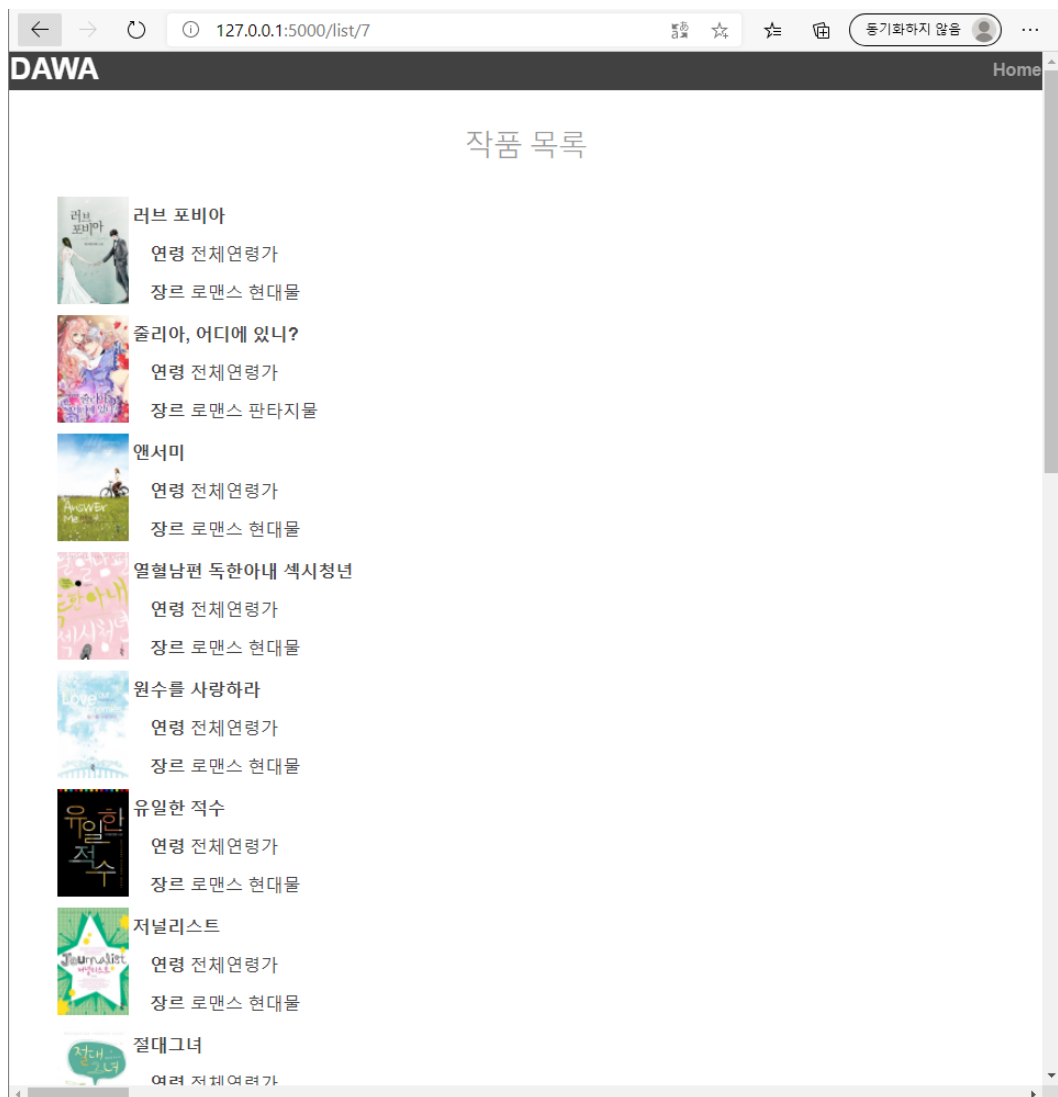
그림 40. Flask 예제

1) Main menu

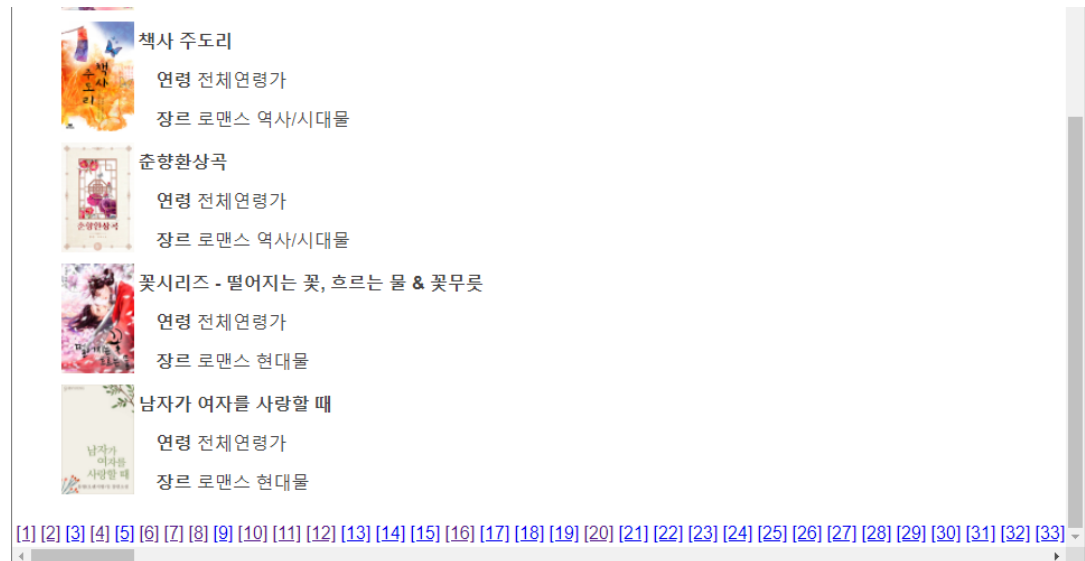


DAWA의 메인 페이지(그림1)입니다. 이달의 작품 TOP 10을 분석해 메인 페이지에 정렬되게 만들며 작품 이미지를 클릭할 시 대표 연재처 페이지와 연결되게 설정해 두었으며(그림2) 제목을 클릭할 시 해당 작품에 대한 소개 및 분석 페이지로 넘어가게(그림3) 됩니다. 헤더로 설정해 둔 플랫폼 이름들은 해당 플랫폼으로 넘어갈 수 있게 도와 주며 아이콘으로 설정하는 것이 좋을지 글자로 설정하는 것이 좋을지에 대해서는 디자인적인 부분을 고려해 결정할 예정입니다.

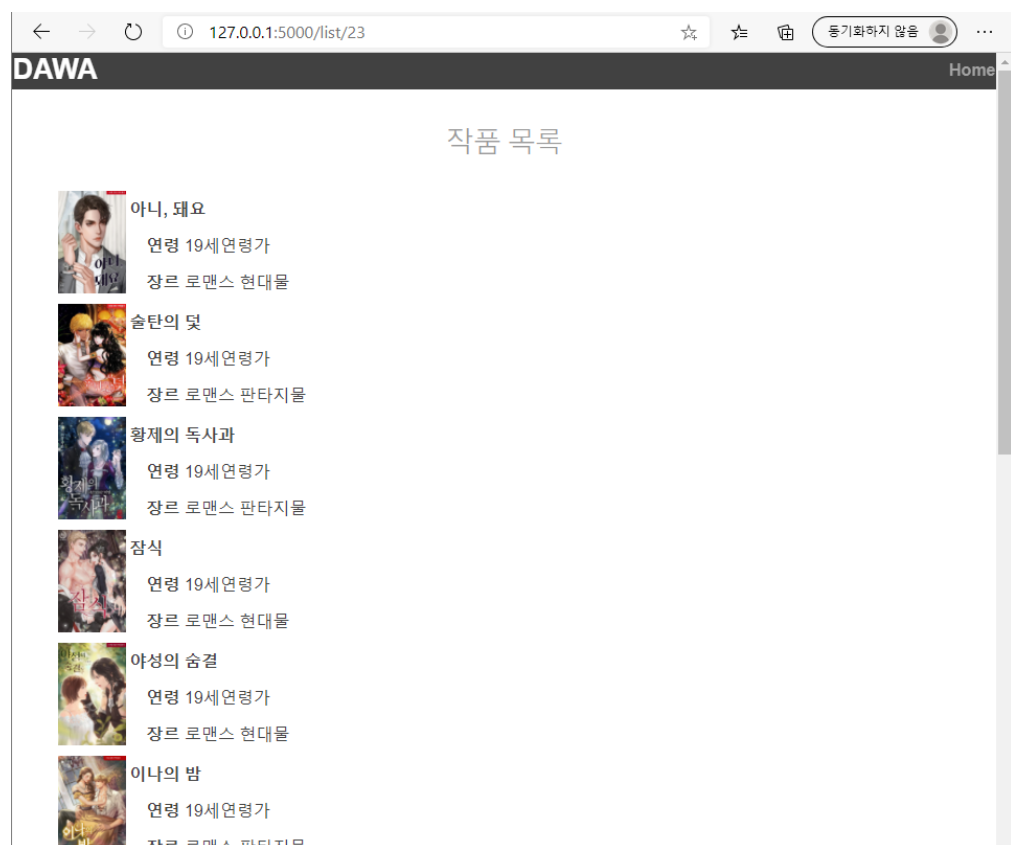
2) 콘텐츠 리스트 화면 출력



http:메인링크/list/(숫자) 에 접속 시 현재 플랫폼별 연재 중인 혹은 완결된 작품 리스트를 볼 수 있게 합니다. 작품 제목, 작품 표지, 연령, 장르를 출력되게 하였으나 가독성과 이미지를 위해 수정을 고려 중에 있습니다.



현재 20개의 작품을 한 페이지에 담았으나 pagination 의 숫자가 방대하게 늘어나는 문제가 발생해 5개 혹은 10개 단위로 끊어 paginate 되게 할 것입니다.



/list/7이 23으로 바뀐 것을 확인할 수 있는데요. Paginate 된 숫자를 누르면 해당 페이지로 이동하며 할당된 리스트를 출력하게끔 하였습니다.

현재 해당 작품들을 작품 소개 칸 하이퍼링크로 연결시키지는 못하였으나 다음 주 중으로 구현 예정 중에 있으며 연결될 작품 소개 칸은 3)에서 소개드릴 형식과 같습니다.

3) 검색 및 결과 화면 출력



현재 DAWA 페이지와 DB 을 연동해 작품별로 페이지를 할당해 주었습니다.
/content/1로 접속할 경우 다음과 같은 작품이,



/content/1088로 접속할 경우 다음과 같은 작품이 출력됩니다. 앞서 설명드린
바와 같이 리스트로 출력된 작품에 하이퍼링크를 달아 해당 페이지로 이동할 수
있게끔 구현할 것이며 검색을 통해 작품을 찾을 수 있는 기능도 함께 구현할
예정입니다.

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

데이터 마이닝과 분석을 통한 헬스케어 종합 인포 사이트

프로젝트 이름	데이터 마이닝과 분석을 통한 헬스케어 종합 인포 사이트	회사명	DBMS 인공지능 사업부
프로젝트 관리자	인공지능 사업부	날짜	2015 6월 14일

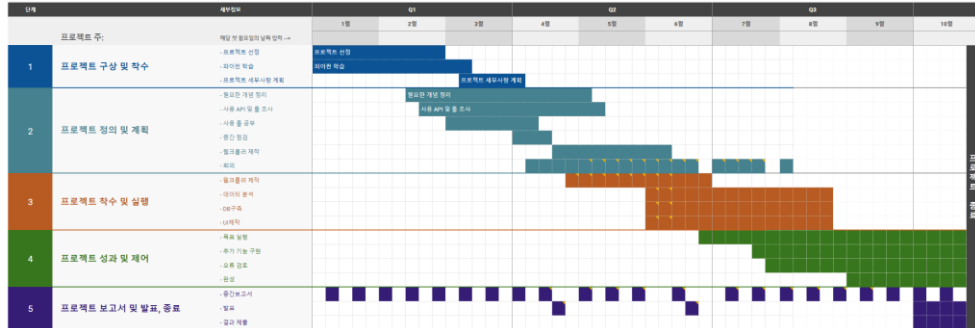


그림 41. 프로젝트 진행 계획

- 플라스크를 이용한 웹페이지 구축 중에 있습니다.

2. 개선 방안

- 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.

- 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')
```

```
[('차가다', 'Adjective')]
```

그림 44. 용언 분석기를 이용한 품사의 원형 복원

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.

나. TF-IDF

<수식 1>에서 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. IDF값이 클수록 해당 단어가 문서의 연관성이 크다는 것을 의미한다.

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

수식 1. TF-IDF 가중치 계산

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 하냐에 따라 정확도가 달라진다.

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.

다. 딥러닝 기반 감성 분석

KoBERT : 2019년 9월 SKT¹²에서 오픈소스로 제공된 딥러닝 모델로 BERT에서 한국어 성능의 한계를 극복하고자 개발했다고 한다. 수백만 개의 한국어 문장으로 이루어진 말뭉치를 학습시켜 기존 감성 분석에서 활용된 Fast-Text나 구글에서 제공하고 있는 BERT에 비해 한국어에 최적화되어 있다. '네이버 영화 리뷰 말뭉치'를 이용해 pre-train을 했을 때 90%의 정확도를 보여준다.

- BERT¹³는 구글에서 개발한 자연언어를 양방향으로 학습하는 모델이다. BERT의 모델링 방식은 대량의 말뭉치로 미리 학습을 시킨 뒤 분류를 원하는 데이터를 넣으면 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 알려져 있다.

4. TextRank 기법

가. PageRank

- 구글이 만든 알고리즘으로 각각의 페이지를 노드라 하고 페이지와 페이지 사이 연결 링크를 에지로 한 그래프를 가지고 각 노드와 에지의 가중치를 통해 최종적으로 각 노드, 즉 각 페이지의 가중치를 구하는 알고리즘이다.
- 이 알고리즘은 하이퍼링크 구조를 가지는 문서에 다른 문서와 비교했을 때 중요도에 따라 가중치를 부여하여 서로 간의 연관성을 나타낼 수 있도록 한다.

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

¹³ <https://github.com/google-research/bert>

나. TextRank¹⁴

위의 PageRank를 한 텍스트에 적용하게 되면 노드를 단어로 설정하고 연결 링크를 문장에 적용시킬 수 있다. 그렇게 해당 단어와 동시에 한 문장에서 출현하여 연관성이 높은 단어를 파악할 수 있다. TextRank(단어의 출현 빈도)의 기하 평균, 두 단어 간의 유사도와 그 길이를 곱하여, 핵심 키워드를 선정하게 된다.

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.

라. TextRank 적용 예시

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.

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

    def load(self, sentenceliter, 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 sentenceliter:
            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 42. 키워드 추출을 위한 Text Rank Class 생성

¹⁴ 참고자료 : <https://bab2min.tistory.com/552>

이 클래스를 활용하여 외계행성에 대한 나무위키 페이지의 데이터를 가져와 핵심 키워드와 핵심 문장을 추출해보았다.


```

In [15]: tr = TextRank(window=5, coef=1)
print('Load...')
stopword = set(['있', 'VV'], ('하', 'VV'), ('되', 'VV'), ('없', 'VV')])
tr.load(RawTaggerReader('test2.txt'), lambda w: w not in stopword and (w[1] in ('NNG', 'NNP', 'VV', 'VA'))))
print('Build...')
tr.build()
kw = tr.extract(0.1)
for k in sorted(kw, key=kw.get, reverse=True):
    print("%s\t\t%g" % (k, kw[k]))

Load...
Build...
가스 행성 3.416085994155906
암석 행성 3.0106208860477417
가스 행성 2.4228342211456226
외계 행성 대부분 2.199690669831413
(가스, 'NNG'), (행성, 'NNG')
0.200046

In [14]: tr = TextRank()
print('Load...')
from konlpy.tag import Komoran
tagger = Komoran()
stopword = set(['있', 'VV'], ('하', 'VV'), ('되', 'VV')])
tr.loadSents(RawSentenceReader('test2.txt'), lambda sent: filter(lambda x: x not in stopword and x[1] in ('NNG', 'NNP', 'VV', 'VA'), tagger.tokenize(sent)))
print('Build...')
tr.build()
ranks = tr.rank()
for k in sorted(ranks, key=ranks.get, reverse=True)[:100]:
    print("%s\t\t%g" % (k, ranks[k]))
print(tr.summarize(0.2))

Load...
Build...
2 0.09243363545450946 행성계 2840개에서 행성 3796개.
0 0.07180532982473248 외계 행성(外行星) 또는 계외 행성(系外行星)은 태양계 밖의 행성으로, 태양이 아닌 다른 항성 주위를 공전하고 있는 행성이다.
5 0.06708175840119866 [5] 발견된 외계 행성들 중 지구와 가장 가까운 것은 프록시마 b이다.
1 0.06213274717329027 지금까지 3800여 개의 외계 행성이 발견되었으며(2018년 6월 23일 기준:
16 0.061672538482577546 [8] [9] [10] 갈색 왜성을 도는 외계 행성들도 있으며 어떤 항성에도 속박되지 않고 우주를 떠도는 행성도 있다.
14 0.05461506550300031 [6] 상대적으로 가벼운 지구질량 수 배 정도의 외계 행성들도 많이 발견되었으며 통계적 연구결과 이를 암석형 외계 행성의 수는 가스 행성보다 많은 것으로 보인다.
10 0.05027593582461892 관측 기술의 향상 덕분에 이후 외계 행성의 발견 속도는 상승했다.
12 0.04978326968936733 확인된 외계 행성 대부분은 목성 또는 해왕성 정도 덩치의 가스 행성으로 추측되나 가스 행성이 외계 행성들 중 대부분을 차지한다는 의미는 아니다.
15 0.04777777777777778 [7] 최근 연구에 따르면, 1천여 개의 외계 행성들이 발견된 이후에 이들 중 일부는 행성이 아닌 암석 행성일 수도 있다.

```

그림 46. 핵심 키워드 추출 예시

그 결과 <그림 46>과 같이 외계 행성을 설명하는 몇몇의 키워드가 정확히 설정되었다.

5. LSTM 알고리즘

가. RNN

RNN은 과거의 데이터를 통해 현재의 문제를 해결하는 방식의 알고리즘이다. 이 알고리즘은 이 전의 데이터를 가진다면 이를 통해 현재의 문제에 대한 해결 방식을 알려줘 리스트나 연속적인 이벤트에 효과적이다.

나. LSTM¹⁵

그러나 RNN은 직전의 결과가 아닌 훨씬 그 이전의 결과를 현재의 문제에 적용해야 한다면 두 문제를 연결시키는데 힘들어한다는 단점이 있다. 이를 보완한 알고리즘이

¹⁵ 참조 : <https://brunch.co.kr/@chris-song/9>

LSTM이다. LSTM은 큰 중심 스테이트에 게이트를 만들고 이 게이트에서 선택된 정보를 받아 처리를 한다.

```
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: 0.8906
Epoch 00004: val_acc improved from 0.85741 to 0.85978, saving model to best_model.h5
1939/1939 [=====] - 46s 25ms/step - loss: 0.2844 - acc: 0.8906 - val_loss: 0.3266 - val_acc: 0.8597
Epoch 5/15
1938/1939 [=====] - ETA: 0s - loss: 0.2688 - acc: 0.8990
Epoch 00005: val_acc improved from 0.85978 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 [=====] - 48s 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) # 동문화
new_sentence = [word for word in new_sentence if not word in stopwords] # 불용어 제거
encoded = tokenizer.texts_to_sequences([new_sentence]) # 결구 인코딩
pad_new = pad_sequences(encoded, maxlen = max_len) # 패딩
score = float(model.predict(pad_new)) # 예측
if(score > 0.5):
    print("{:.2f}% 확률로 긍정 리뷰입니다.ㄴ".format(score * 100))
else:
    print("{:.2f}% 확률로 부정 리뷰입니다.ㄴ".format((1 - score) * 100))

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

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

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

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

그림 43. LSTM 알고리즘을 이용한 분석 결과 예시

6. 동시 출현 기반 연관어 분석

단어를 중심으로 어떤 단어가 연관이 있는지 파악하는 분석 기법이다.

```
pos_review=(glob.glob("C:\Users\정지훈\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[i][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 48. 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", "넋풀릭스"))
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), ('월', 0.6065816283226013), ('기결', 0.6044149398803711), ('이따', 0.6040357947349548), ('알다', 0.6020412445068359), ('이러다가', 0.6007715463638306), ('거양', 0.6000221967697144), ('도리', 0.5988008975982666), ('혹', 0.5978043079376221)]
[('개꿀', 0.6190359592437744), ('역시', 0.6100665926933289), ('월', 0.6065816283226013), ('기결', 0.6044149398803711), ('이따', 0.6040357947349548), ('알다', 0.6020412445068359), ('이러다가', 0.6007715463638306), ('거양', 0.6000221967697144), ('도리', 0.5988008975982666), ('혹', 0.5978043079376221)]
[('ㅈㅈㅈㅈ', 0.4821982681751251), ('엎', 0.46736279129981995), ('줄아하다', 0.45916640758514404), ('추광', 0.44668009877204895), ('헨즈', 0.4345983564853668)]
```

그림 49. 어근 변경 후 유사성 판단 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)