# DAWA

## 데이터 마이닝을 이용한 웹소설 종합 인포 웹 어플리케이션

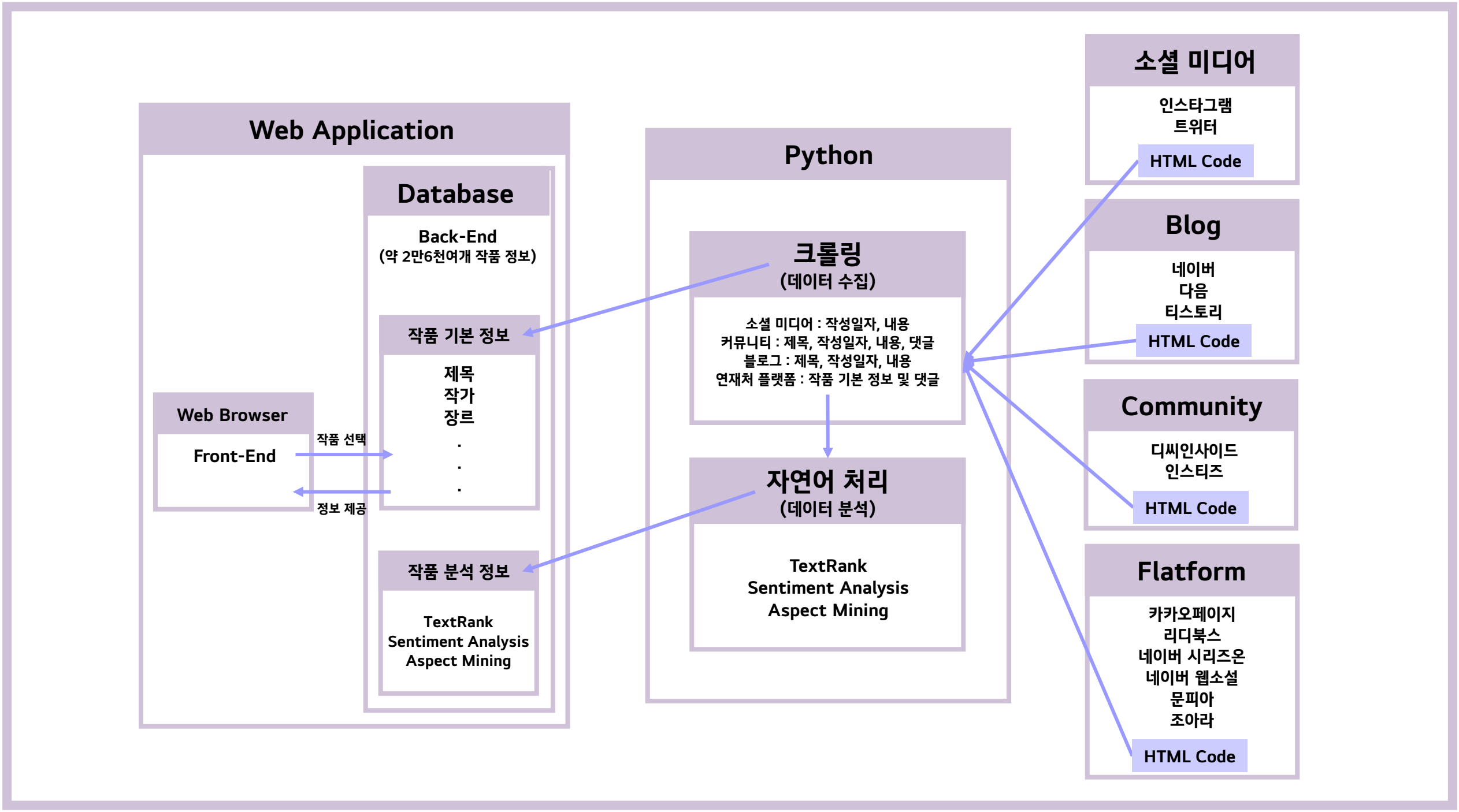

팀장 : 2015722084 한승주

팀원 : 2015722083 김성종

2017202067 조예슬

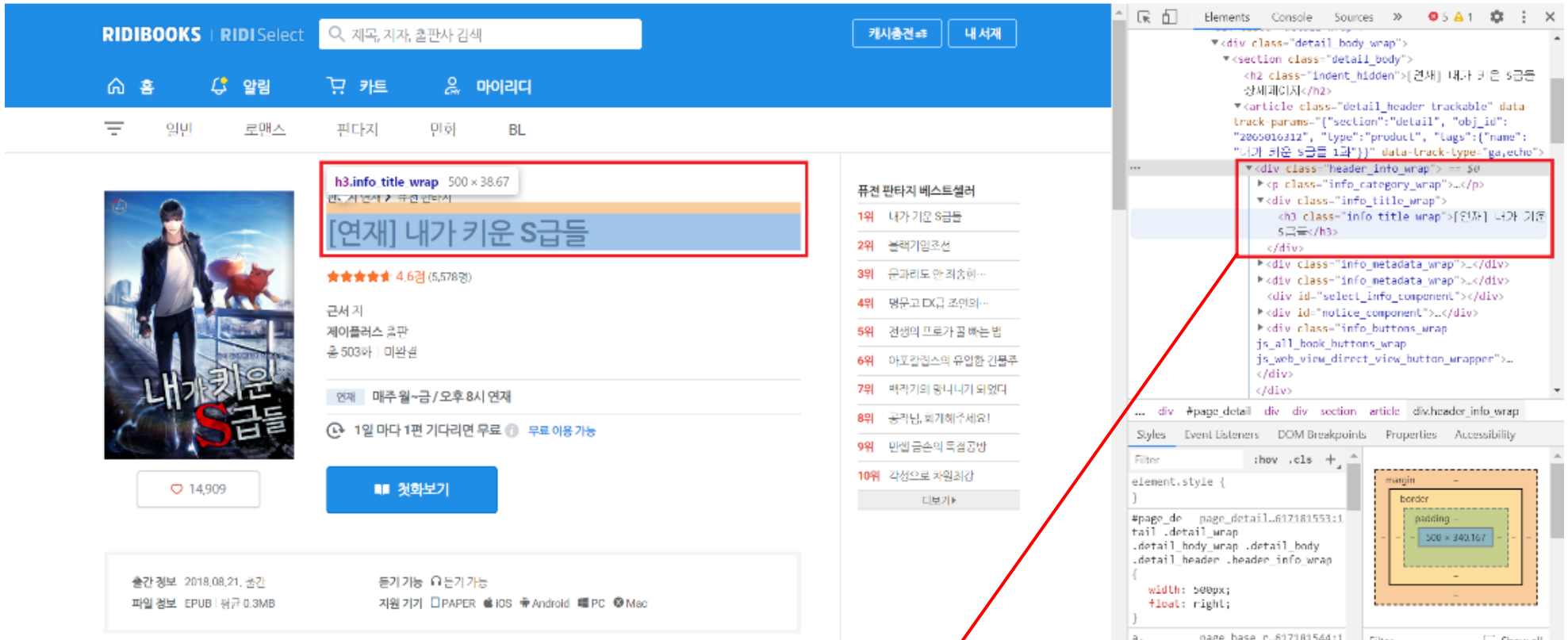
# 목차

# Ⅰ. DAWA의 구조와 설계

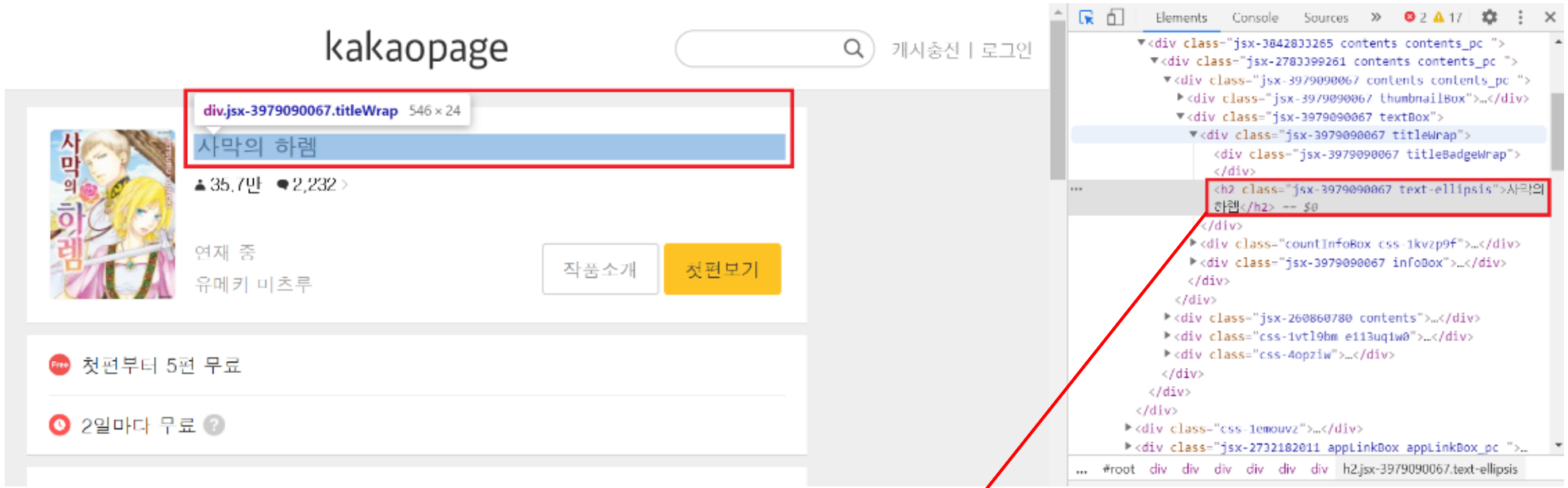

Web Application
Database
Back-End
(약 2만6천여개 작품 정보)
작품 기본 정보
제목
작가
장르
.
.
.
작품 분석 정보
TextRank
Sentiment Analysis
Aspect Mining
Web Browser
Front-End
작품 선택
정보 제공
Python
크롤링
(데이터 수집)
소셜 미디어 : 작성일자, 내용
커뮤니티 : 제목, 작성일자, 내용, 댓글
블로그 : 제목, 작성일자, 내용
연재처 플랫폼 : 작품 기본 정보 및 댓글
자연어 처리
(데이터 분석)
TextRank
Sentiment Analysis
Aspect Mining
소셜 미디어
인스타그램
트위터
HTML Code
Blog
네이버
다음
티스토리
HTML Code
Community
디씨인사이드
인스티즈
HTML Code
Flatform
카카오페이지
리디북스
네이버 시리즈온
네이버 웹소설
문피아
조아라
HTML Code

# Ⅱ. DAWA의 단계별 개발 전략

# 1. 데이터 마이닝




```
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
```




```
#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",""))
```

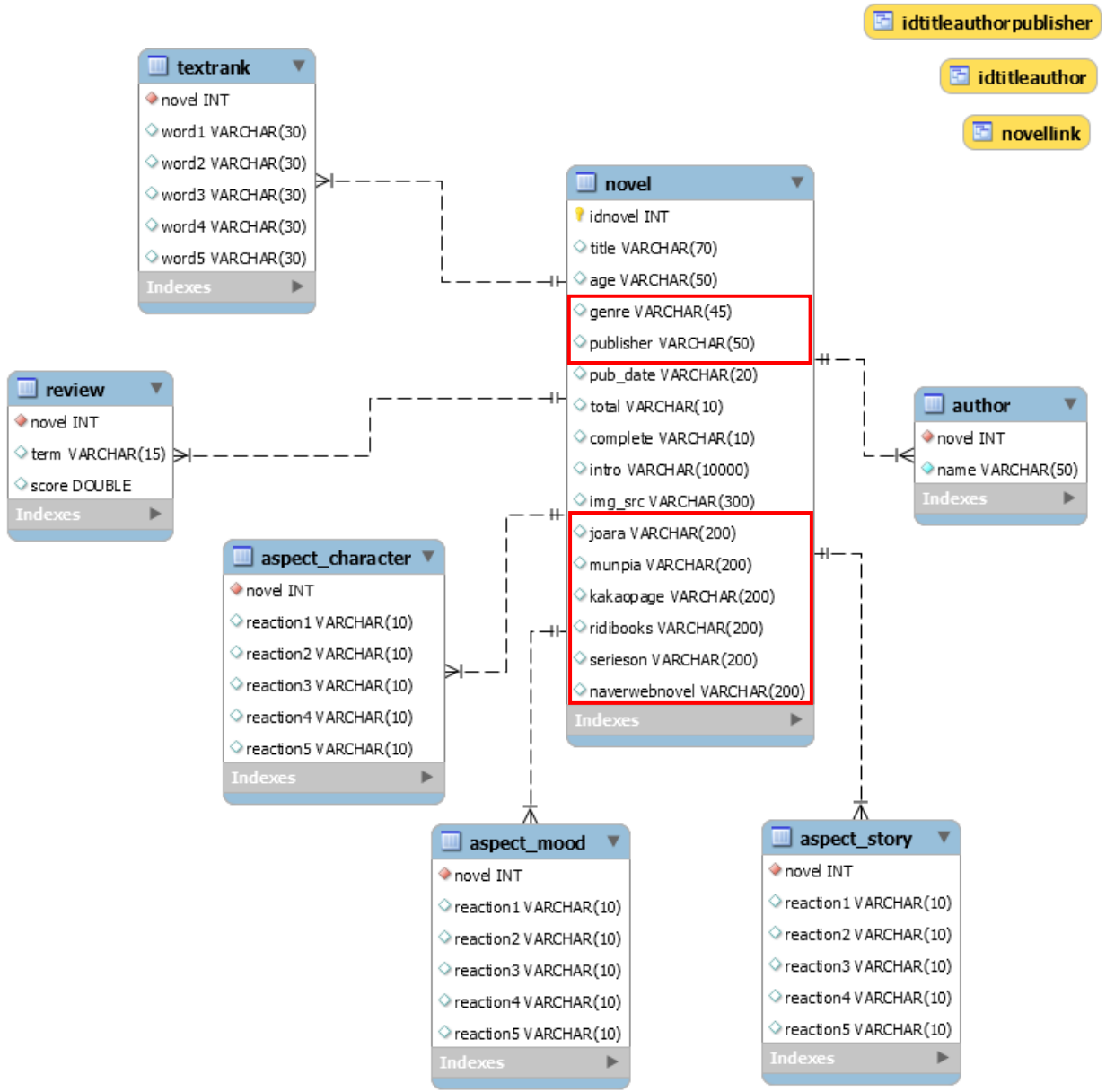
```python
def kakaopage_crawling(search_name):

    tmp = search_name.split('[')[0]
    search_name_nonspace=search_name.replace(" ","")

    url = "https://page.kakao.com/search?word=" + parse.quote(tmp)

    r=requests.get(url)
    c=r.content
    bs_obj = bs4.BeautifulSoup(c, "html.parser")

    boxes = bs_obj.findAll("div", {"class": "css-c09e5i"})


    for box in boxes:
        is_novel=box.find("div",{"class":"css-vurnku"}).text
        if("소설" in is_novel):
            parent=box.parent.parent
            title=parent.find("div", {"class": "text-ellipsis css-11602e0"}).text.replace(" ","")
            #if("단행본" in title):
            #    continue
            if(search_name_nonspace not in title):
                continue
            else:
                main=parent.parent.parent
                monopoly=True
        else:
            continue
    title=main.find("div", {"class": "text-ellipsis css-11602e0"}).text

    if '[' in title:
        print("작품이름 : " + title)
        #exclusive=title.split('[')[1][:-1]
        #print("독점여부 : " + exclusive)
    else:
        print("작품이름 : " + title)
        #exclusive='미독점'
        #print("독점여부 : " + exclusive)
```

...중략

```python
    #작품의 링크
    url="https://page.kakao.com" + newPage
    href=url
    driver.get(url)
    bs_obj = bs4.BeautifulSoup(driver.page_source, "html.parser")
    time.sleep(1)
    #pub_date=driver.find_element_by_xpath('//*[@id="root"]/div[3]/div/div/div[3]/ul/li[1]/div[2]/dl/dd').text
    #print("출간일 : " + pub_date)
    #time.sleep(0.2)

    #if complete!="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",""))

    #else:
    #    print("연재 요일 : 연재 종료")
    #time.sleep(0.2)

    total=driver.find_element_by_xpath('//*[@id="root"]/div[3]/div/div/div[3]/div[1]/span[1]')
    total=total.text.replace('전체',"").replace(' ',"")
    print("연재 수 : "+ total)
    time.sleep(0.2)

    pub_date=driver.find_element_by_xpath('//*[@id="root"]/div[3]/div/div/div[3]/ul/li[1]/div[2]/dl/dd').text
    print("최초 업데이트일 : " + pub_date)
    time.sleep(0.2)

    time.sleep(2)

    driver.find_element_by_xpath('//*[@id="root"]/div[3]/div/div/div[1]/div[2]/div[3]/div[2]/button[1]').click()
    #driver.set_window_size(1400,1000)
    publisher=None
    try:
        publisher=driver.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]').text
    except:
        pass
```

...중략

# 2. DB


idtitleauthorpublisher
idtitleauthor
novellink
textrank
novel INT
word1 VARCHAR(30)
word2 VARCHAR(30)
word3 VARCHAR(30)
word4 VARCHAR(30)
word5 VARCHAR(30)
Indexes
novel
idnovel INT
title VARCHAR(70)
age VARCHAR(50)
genre VARCHAR(45)
publisher VARCHAR(50)
pub_date VARCHAR(20)
total VARCHAR(10)
complete VARCHAR(10)
intro VARCHAR(10000)
img_src VARCHAR(300)
joara VARCHAR(200)
munpia VARCHAR(200)
kakaopage VARCHAR(200)
ridibooks VARCHAR(200)
serieson VARCHAR(200)
naverwebnovel VARCHAR(200)
Indexes
review
novel INT
term VARCHAR(15)
score DOUBLE
Indexes
author
novel INT
name VARCHAR(50)
Indexes
aspect_character
novel INT
reaction1 VARCHAR(10)
reaction2 VARCHAR(10)
reaction3 VARCHAR(10)
reaction4 VARCHAR(10)
reaction5 VARCHAR(10)
Indexes
aspect_mood
novel INT
reaction1 VARCHAR(10)
reaction2 VARCHAR(10)
reaction3 VARCHAR(10)
reaction4 VARCHAR(10)
reaction5 VARCHAR(10)
Indexes
aspect_story
novel INT
reaction1 VARCHAR(10)
reaction2 VARCHAR(10)
reaction3 VARCHAR(10)
reaction4 VARCHAR(10)
reaction5 VARCHAR(10)
Indexes

# 3. 분석

3-1. TextRank

3-2. Sentimental Analysis

3.3 Aspect Mining

# 3-1. TextRank

In [5]:
```
from krwordrank.word import KRWordRank

min_count = 5   # 단어의 최소 출현 빈도수 (그래프 생성 시)
max_length = 10 # 단어의 최대 길이
wordrank_extractor = KRWordRank(min_count=min_count, max_length=max_length)

beta = 0.85    # PageRank의 decaying factor beta
max_iter = 10
f = open("구르미 그린 달빛.txt", 'r',encoding='utf-8')
texts = f.read().split('.')
f.close()
keywords, rank, graph = wordrank_extractor.extract(texts, beta, max_iter)

for word, r in sorted(keywords.items(), key=lambda x:x[1], reverse=True)[:30]:
        print('%8s:\t%.4f' % (word, r))
```

```
    그린:	9.5049
    달빛:	8.0521
   드라마:	7.8838
   구르미:	5.6222
    라온:	5.5058
    소설:	3.9484
   윤이수:	3.8799
      ^^:	3.3569
   하지만:	3.3507
    너무:	3.2487
   로맨스:	3.2356
   그리고:	2.9492
    정말:	2.7040
    이영:	2.6532
    다시:	2.5127
```

```python
def extract(self, ratio=0.1):
    ranks = self.rank()
    cand = sorted(ranks, key=ranks.get, reverse=True)[:int(len(ranks) * ratio)]
    pairness = {}
    startOf = {}
    tuples = {}
    for k in cand:
        tuples[(k,)] = self.getI(k) * ranks[k]
        for l in cand:
            if k == l: continue
            pmi = self.getPMI(k, l)
            if pmi: pairness[k, l] = pmi

    for (k, l) in sorted(pairness, key=pairness.get, reverse=True):
        # print(k[0], l[0], pairness[k, l])
        if k not in startOf: startOf[k] = (k, l)

    for (k, l), v in pairness.items():
        pmis = v
        rs = ranks[k] * ranks[l]
        path = (k, l)
        tuples[path] = pmis / (len(path) - 1) * rs ** (1 / len(path)) * len(path)
        last = l
        while last in startOf and len(path) < 7:
            if last in path: break
            pmis += pairness[startOf[last]]
            last = startOf[last][1]
            rs *= ranks[last]
            path += (last,)
            tuples[path] = pmis / (len(path) - 1) * rs ** (1 / len(path)) * len(path)

    used = set()
    both = {}
    for k in sorted(tuples, key=tuples.get, reverse=True):
        if used.intersection(set(k)): continue
        both[k] = tuples[k]
        for w in k: used.add(w)

    # for k in cand:
    #     if k not in used or True: both[k] = ranks[k] * self.getI(k)

    return both
```

```python
def hot_topic_analyzer(work_name):
    tr = TextRank(window=5, coef=1)
    #print('Load...')
    stopword = set([('있', 'VV'), ('하', 'VV'), ('되', 'VV'), ('없', 'VV')])
    file_name= 'Crawling\\' + work_name + '.txt'
    tr.load(RawTaggerReader(file_name), lambda w: w not in stopword and (w[1] in ('NNG', 'NNP', 'VV', 'VA')))
    #print('Build...')
    tr.build()
    kw = tr.extract(0.1)
    title = work_name
    title_nonspace = title.replace(' ', '')
    count = 0
    word_list=[]
    for k in sorted(kw, key=kw.get, reverse=True):
        temp = ("%s%g" % (k, kw[k])).split('0')[0]
        if 'VV' in temp:
            continue
        if 'VA' in temp:
            continue
```

```python
from Hot_Topic_Analyzer import hot_topic_analyzer

hot_topic_analyzer('구르미 그린 달빛')
```

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

# 3-2. Sentiment Analysis

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
[illegible]
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
```

테스트 정확도: 0.8543

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.

# 3-3. Aspect Mining

```
#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
```

```
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 | 뻔하다 | 인하다 | 필요하다 | 젊다 |
| 17 | 백작가의 망나니가 되었다 | 17 | 특출나다 | 안타깝다 | 그만하다 | 조용하다 | 답답하다 |
| 18 | 나 혼자만 레벨업 | 18 | 특출나다 | 환상하다 | 탄탄하다 | 지루하다 | 뻔하다 |
| 19 | 환생표사 | 19 | 신비하다 | 용감하다 | 무능하다 | 간단하다 | 별다르다 |
| 20 | 명문고 EX급 조연의 리플레이 | 20 | 많다 | 열광하다 | 있다 | 그렇다 | 아쉽다 |
| 21 | 망나니 1왕자가 되었다 | 21 | 얼떨떨하다 | 드물다 | 상당하다 | 미치다 | 끊임없다 |
| 22 | 달콤, 찬란한 재벌기 | 22 | 재미있다 | 인하다 | 힘들다 | 달콤하다 | 성하다 |

# Ⅲ . 웹 어플리케이션 시연영상

DAWA

127.0.0.1:5000/main

동기화하지 않음

# DAWA

home menu

확인

[흑염의 용제](#)

[헌터세계의 귀환자](#)


[잘은 아버지는 등락 [심국시 동환전]](#)

[신검화산](#)


[화룡](#)

[무시 우백](#)

[미래를 읽는 남자](#)

[노병의 시간은 거꾸로 흐른다](#)


[미도지존](#)

[공허의 미드시](#)

검색하려면 여기에 입력하십시오.

오후 4:04
2020-11-03

DAWA

127.0.0.1:5000/main


작가 근석

연령 전체연령가

출간일 2018-08-21

장르 판타지 퓨전

출판 제이플미디어

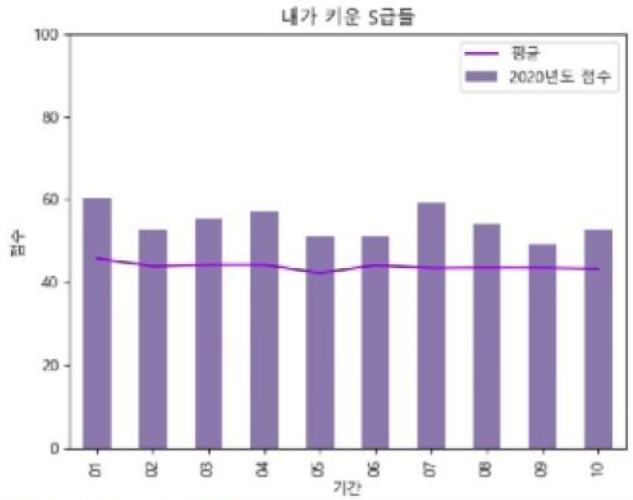
연재화수 499

미완결

연재처

소개 「급 헌터. 그것도 살아가는 S급 동생 발목이나 잡는 쓸모없고 찌질한 「급 헌. 개판 된 인생 대충 살다가 겁국 동생 목숨까지 잡아먹고 회귀한 내게 주어진 칭호, '완벽한 양육자'. 그래, 이번에는 니대지 말고 양전히 잘난놈들 뒷바라지나 해 주지. 라고 생각했는네, S급들이 좀 이상하다.

| keyword | | | | | |
|---|---|---|---|---|---|
| 캐릭터 | #유신 | #주인공 | #닌신안 | #프로필카드 | |
| 분위기 | #약하다 | #아니다 | #우신하다 | #특이하다 | #가깝다 |
| 스토리 | #무겁다 | #입다 | #싸릿하다 | #지겹다 | #구질구질하다 |
| | #애매하다 | #지루하다 | #다름없다 | #수많다 | #그렇다 |

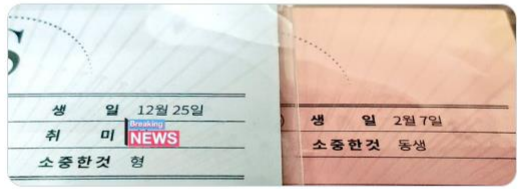



Top Latest People Photos Videos

자렌🔥🍑 @JAL_fn · Aug 29
프로필 카드 도착
😍😍😍😍😍😍

1 5 23

자렌🔥🍑 @JAL_fn · Aug 29
이건 내스급을 읽고 있는 누구나 알고 있는 사실이라 네타는 아닌 것 같아서 살짝 올려 봄.
하지만 보면 볼 수록 이렇게 서로에게 미쳐있는 놀늦 덕분에 오타쿠 가슴이 웅장해 진다.
한유현 취미도 정말 엄청나니까 아직 신청 안 하신분들 프로필 카드 꼭 신청하셔요 9




1 38 50

🍑야호(유행진)🌻🍋 @yaho_ns · Aug 14
리디 내스급 프로필 카드팩 증정 이벤트 정리
1. 무조건 판타지 카테고리내 내스급 유료분 1화 포함+타 연재 작품= 3만원 이상 구매
2. 당연히 내스급 만으로 3만원 넘어도 OKKK
3. 작품은 모두 다운로드 완료 해야함
4. 다하고 굿즈 신청 폼 꼭 작성하기
(타래에 이벤트 페이지 링크)

1 293 118

Show this thread

예림이🌊가 보고싶은 하현 @mm_waning · Aug 14
내스급 프로필 카드 굿즈.. 8월 14일부터 31일까지고 방법은 내스급 1화 유료 구매 후(필수!) 스급 포함 판타지'연재'작품 3만원이상 사면 전원 증정ㅠㅠㅠ(다운로드까지 해야..!) 공개 안된 울놔현 프로필 있음ㅠㅠㅠ(일러스트는 없는듯..) 빨리 구매해야.. 1차 배송때 받아볼 수 있는..것...

검색하려면 여기에 입력하십시오.

오후 4:05
2020-11-03


연령 전체연령가

출간일 2018-08-21

장르 판타지 퓨전

출판 제이플미디어

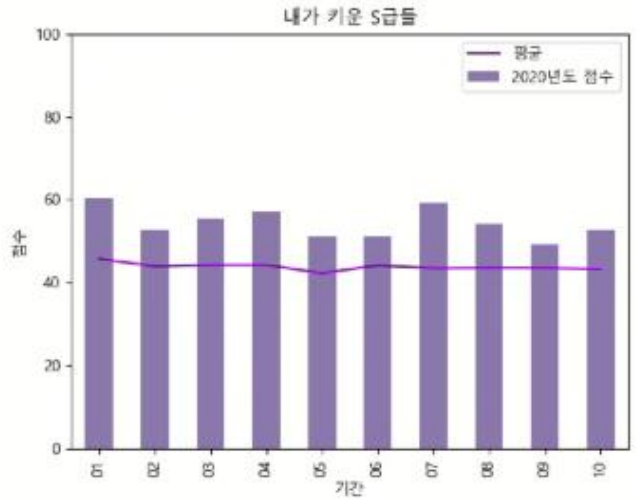
연재화수 499

미완결

연재처

소개 「급 헌터. 그것도 실나기는 S급 동생 발목이나 잡는 쓸모없고 찌질한 「급 형. 개판 된 인생 대충 살다가 김국 동생 목숨까지 잡아먹고 회귀한 내게 주어진 칭호, '완벽한 양육자'. 그래. 이번에는 니대지 말고 양전히 잘난놈들 뒷바라지나 해 주지. 라고 생각했는데, S급들이 좀 이상하다.

| | | | | | |
|---|---|---|---|---|---|
| keyword | #급헌터 | #유진 | #주인공 | #넌진안 | #프로필키드 |
| 캐릭터 | #약하다 | #아니다 | #우신하다 | #특이하다 | #기깝다 |
| 분위기 | #무겁다 | #밉다 | #싸릿하다 | #시겁다 | #구질구질하다 |
| 스토리 | #애매하다 | #지루하다 | #다름없다 | #수많다 | #그렇다 |




검색하려면 여기에 입력하십시오.
오후 4:05
2020-11-03

# Thank You.