네이버웹툰추천모델

웹툰 시놉시스의
TF-IDF 데이터를 이용한 웹툰 추천모델구현

데이터사이언스개론 프로젝트

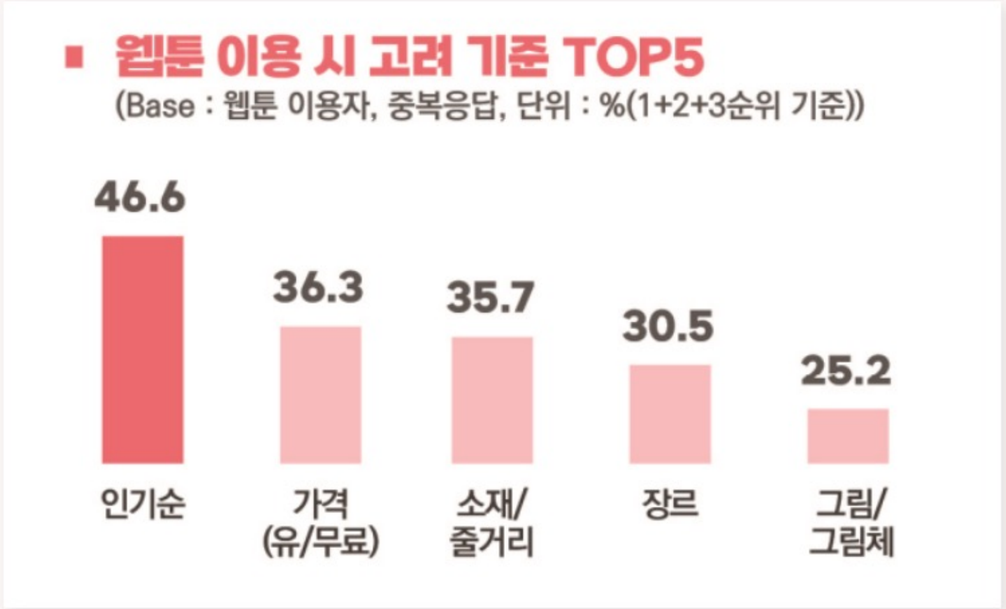
Member

201921367 송여경

201921981 이승연

1 About project

주제 선정 이유



한국콘텐츠진흥원, 2021 만화·웹툰 이용자 실태조사

한국콘텐츠진흥원 2021 만화이용자 실태조사에 따르면,
웹툰 선택 기준 중 '소재와 줄거리'가 35.7%로
높은 비중을 차지하고 있음을 알 수 있다.
그럼에도 대부분의 웹툰 플랫폼에서는
장르별로만 웹툰을 나눠주고,
비슷한 시놉시스로 묶어주는 기능은 특별히 없다.
따라서 이번 프로젝트에서 장르가 아닌
시놉시스가 비슷한 웹툰을 추천하는 기능을 구현하고자 하였다.

2 About dataset

Naver Webtoon Dataset 네이버 웹툰 데이터셋

기존 데이터셋의 문제점

(1) 최신 웹툰이 업데이트 되지 않아, 현재 연재중인 웹툰 추천의 어려움

(2) 제목의 오류 ex) '치즈인더트랩'인 경우 '치즈인더...' 으로 표기된 오류가 많았음.

따라서 네이버웹툰에서 서비스되는 연재중웹툰 584편의 정보를 직접 크롤링하여 csv파일로 추출하였다.

id	title	author	day	genre	story
0	참교육	채용택, 한가람	월	액션	무너진 교권을 지키기
1	뷰티풀 군바리	설이, 윤성원	월	드라마	'여자도 군대에 간다면
2	퀘스트지상주의	박태준 만화회사	월	드라마	[외모지상주의], [싸움
3	장씨세가 호위무사	김인호, 조형근	월	무협/사극	'당신이 부른 것이오.
4	윈드브레이커	조용석	월	스포츠	혼자서 자전거를 즐겨
5	팔이피플	매미, 희세	월	드라마	<마스크걸>, <위대한
6	퍼니게임	배진수	월	스릴러	<머니게임>,<파이게
7	신화급 귀속 아이템을 손에 넣었다	정선율, 헤스	월	판타지	D급 무투계 레이더로

2 About dataset

웹 크롤링?

Web상에 존재하는 Contents를 수집하는 작업 (프로그래밍으로 자동화 가능)

Selenium등 브라우저를 프로그래밍으로 조작해서, 필요한 데이터만 추출하는 기법

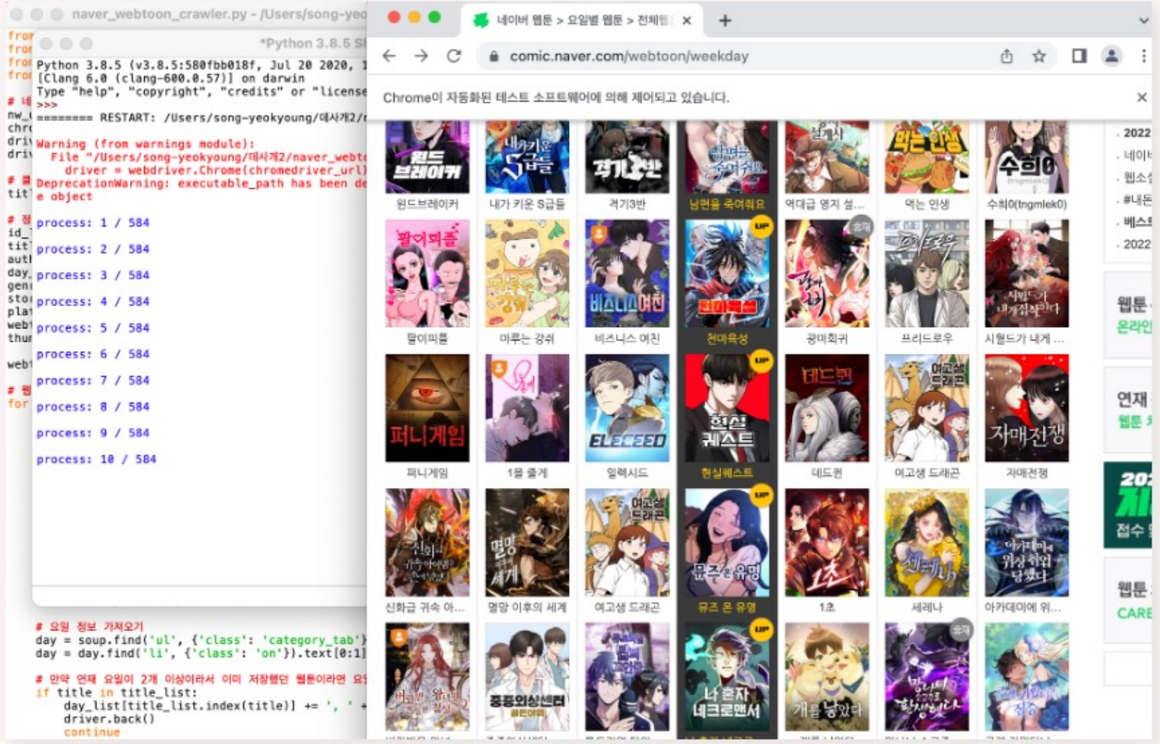
```
day = day.find('li', {'class': 'on'}).text[0:1]

# 만약 연재 요일이 2개 이상이라서 이미 저장했던 웹툰이라면 요일만 추가하고 넘어가기
if title in title_list:
    day_list[title_list.index(title)] += ', ' + day
    driver.back()
    continue

# 나머지 정보 수집하기
thumbnail_url = soup.find('div', {'class': 'thumb'}).find('a').find('img')['src']
author = soup.find('span', {'class': 'wrt_nm'}).text[8:]
author = author.replace(' / ', ', ')
genre = soup.find('span', {'class': 'genre'}).text.split(", ")[1]
story = soup.find('div', {'class': 'detail'}).find('p').text

# 정보들을 리스트에 담기
id_list.append(webtoon_id)
title_list.append(title)
author_list.append(author)
day_list.append(day)
genre_list.append(genre)
story_list.append(story)
platform_list.append("네이버")
webtoon_url_list.append(driver.current_url)
thumbnail_url_list.append(thumbnail_url)

# 뒤로 가기
driver.back()
webtoon_id += 1
sleep(0.5)
```

```
import pandas as pd

total_data = pd.DataFrame()

total_data['id'] = id_list
total_data['title'] = title_list
total_data['author'] = author_list
total_data['day'] = day_list
total_data['genre'] = genre_list
total_data['story'] = story_list
total_data['platform'] = platform_list
total_data['webtoon_url'] = webtoon_url_list
total_data['thumbnail_url'] = thumbnail_url_list

# 따로 인덱스를 생성하지 않고 id를 인덱스로 정하기
total_data.set_index('id', inplace=True)

# CSV 파일로 저장하기
total_data.to_csv("네이버 웹툰 정보.csv", encoding='utf-8-sig')
```

2 About dataset

웹 크롤링과정 캡처

네이버웹툰 웹사이트 https://comic.naver.com/webtoon/weekday 에 접근하여 정부를 수집하였음.



3 Data Analysis Environment

kaggle

python

데이터셋 참조 및 추출

python

데이터분석

4 Data structure

데이터셋 원본의 초반 데이터 구조는 아래와 같다.

```
[157] df.columns
```

```
Index(['id', 'title', 'author', 'genre', 'description'], dtype='object')
```

```
[158] df.index
```

```
RangeIndex(start=0, stop=570, step=1)
```

```
[156] df = pd.read_csv('/content/sample_data/nw.csv')
      df.head()
```

	id	title	author	genre	description
0	10000	참교육	채용택, 한가람	액션	무너진 교권을 지키기 위해 교권보호국 소속 나화진의 참교육이 시작된다!<부활남> 채...
1	10001	뷰티풀 군바리	설이, 윤성원	드라마	'여자도 군대에 간다면?'본격 여자도 군대 가는 만화!
2	10002	퀘스트지상주의	박태준 만화회사	드라마	[외모지상주의], [싸움독학], [인생존망]과 세계관을 공유하는 작품!공부, 싸움,...
3	10003	장씨세가 호위무사	김인호, 조형근	무협/사극	'당신이 부른 것이오. 나란 사람을... '은둔고수 광휘. 호위무사 되다.웹소설 원...
4	10004	윈드브레이커	조용석	스포츠	혼자서 자전거를 즐겨타던 모범생 조자현.원치 않게 자전거 크루의 일에 자꾸 휘말리게...

5 Data Analysis

❶ 장르 별 작품 수 EDA분석

장르(genre) 별로 작품의 분포가 어떻게 나타나는지 분석하였다.

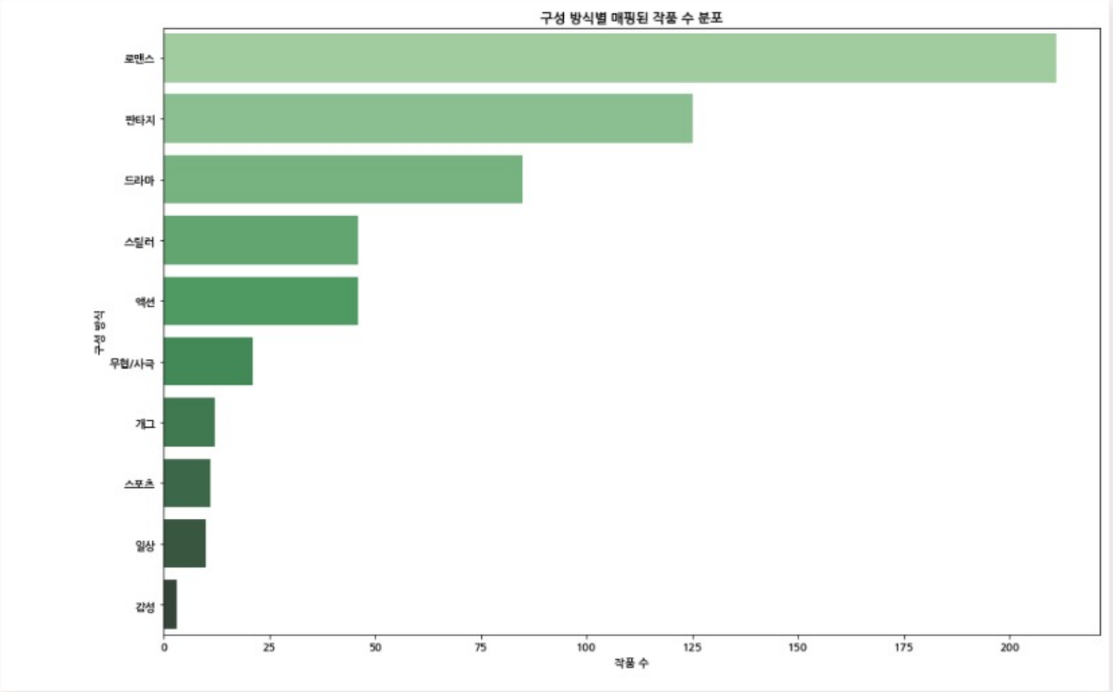
```
gnr_count = df.groupby('genre').id.nunique().reset_index(name='gnr_cnt')
gnr_list_desc = gnr_count.sort_values('gnr_cnt', ascending=False).genre

# plotting
plt.figure(figsize=(15, 10))
gnr_cnt_plot = sns.barplot(x='gnr_cnt', y='genre', data=gnr_count, order=gnr_list_desc,
gnr_cnt_plot.set_title('구성 방식별 매핑된 작품 수 분포')
gnr_cnt_plot.set_ylabel('구성 방식')
gnr_cnt_plot.set_xlabel('작품 수')

plt.show()
```

4 Data Analysis

❶ 장르 별 작품 수

분석결과, 네이버 웹툰 전체 데이터중 장르가 로맨스에 해당되는 웹툰이 전제 584개중 200개 가량으로 가장 많았고,

판타지와 드라마 장르가 그 뒤를 이어 분포하였다.



5 TF-IDF Analysis

❷ TF-IDF (Term Frequency - Inverse Document Frequency)?

정보 검색과 텍스트 마이닝에서 이용하는 가중치로,
여러 문서로 이루어진 문서군이 있을 때 어떤 단어가 특정 문서 내에서 얼마나 중요한 것인지를 나타내는 통계적 수치.
문서의 핵심어를 추출하거나, 문서들 사이의 비슷한 정도를 구하는 등의 용도로 사용할 수 있다.

TF(term frequency	'단어빈도', 특정한 단어가 문서 내에 얼마나 자주 등장하는지를 나타내는 값. 이 값이 높을수록 문서에서 중요하다고 생각할 수 있다.
DF (document frequency	'문서빈도', 특정한 단어 자체가 문서군 내에서 자주 사용되는지를 나타내는 값. 이것은 그 단어가 흔하게 등장한다는 것을 의미한다.
IDF (inverse document frequency	'역문서빈도', 문서 빈도의 역수 TF-IDF값은 TF와 IDF를 곱한 값이다.

5 TF-IDF Analysis

❷ 자연어 처리 (형태소단위의 토큰화) 및 Stopword 설정

텍스트 분석을 위해 텍스트를 분석의 단위가 되는 토큰(token)으로 토큰화(tokenization)하였다.

한국어 텍스트의 경우 형태소(morpheme)단위로 토큰화한다.

이때 사용할 것이 형태소분석기 인데,

본 프로젝트에서는 **konlpy 형태소 분석기**를 사용하였다.

konply는 Python에서 사용할 수 있는 오픈소스 형태소 분석기로, 기존에 공개된 꼬꼬마(Kkma), 코모란(Komoran), 한나눔(Hannanum), 트위터(Twitter), 메카브(Mecab)를 한 번에 설치하고 동일한 방법으로 쓸 수 있게 해준다.

```
[173] from konlpy.tag import Kkma, Hannanum, Komoran
      from konlpy.utils import pprint

      hannanum = Hannanum()
      kkma = Kkma()
      komoran = Komoran()
```

```
def get_voca(doclist):
    doc_nouns_list = []

    for doc in doclist:
        nouns = hannanum.nouns(doc)
        doc_nouns = ''

        for noun in nouns:
            if noun not in stop_words:
                doc_nouns += noun + ' '

        doc_nouns_list.append(doc_nouns)
    return doc_nouns_list
```

5 TF-IDF Analysis

❷ 자연어 처리 (형태소단위의 토큰화) 및 Stopword 설정

Stop Words 는 문서에서 단어장을 생성할 때 무시할 수 있는 단어를 의미한다.

보통 영어의 관사나 접속사, 한국어의 조사 등이 여기에 해당된다. stop_words 인수로 조절할 수 있다.

본 프로젝트의 웹툰 시놉시스텍스트에서 다음과 같은 단어는 stopword로 처리하여 무시할 수 있도록 설정하였다.

1) 웹툰, 웹툰판 등과 같이 특별하게 결측처리 해야 할 단어

2) 은,는,을,를 등의 조사

3) 사람,소년,소녀와 같은 자주 등장하는 단어

```
[241] #웹툰 시놉시스 텍스트에 대한 stopword 설정
      stop_words = set("를 의 웹 툰 웹툰 웹툰판 이번 들 등 수  이 부 판 뿐 그 것 나 그 그녀 속 시작 속 작가 신작".split())
      stop_words |= set("이야기 데 전 후 두 앞 뒤 그들 때문 사람 두 신작 한 자신 만화 소년 소녀 ".split())
```

5 TF-IDF Analysis

2 단어에 대한 유사도분석

유사도 분석을 위한 코드는 다음과 같다. 유사도에 맞추어 행렬을 생성한다.

```python
def recomend_by_word(doclist, result):
    # 결과 웹툰 식별을 위한 딕셔너리
    id_dict = {i: id for i, id in enumerate(doclist['id'])}

    description = doclist['description']
    voca = get_voca(description)

    # TF-IDF
    tfidf_vectorizer = TfidfVectorizer(min_df=1)
    tfidf_matrix = tfidf_vectorizer.fit_transform(voca)
    document_distances = tfidf_matrix * tfidf_matrix.T

    print('유사도 분석을 위한 ' + str(document_distances.get_shape()[0]) +
          'x' + str(document_distances.get_shape()[1]) + 'matrix를 만들었습니다.', end='->')
    n = document_distances.get_shape()[0]
    cnt = 0
    for i in range(n):
        a = document_distances[i].toarray().T
        similarity = set()
        for j in range(n):
            if a[j][0] > 0.15 and j != i:
                similarity.add(id_dict[j]) # 웹툰번호로 저장
        if len(similarity) > 0:
            result[id_dict[i]] = result.get(id_dict[i], set([])) | set(similarity)
            cnt += 1
    print(f'{cnt}개 웹툰에 대한 추천')
    return result
```

```
[265] result = recomend_by_word(df[['id', 'description']], {})
8초
      유사도 분석을 위한 570x570matrix를 만들었습니다.->263개 웹툰에 대한 추천

[ ] print(result)
0초
      {10001: {10482, 10323, 10204}, 10002: {10468, 10085}, 10008: {10537, 10476, 10534}, 10010: {10146, 10428, 10036, 10218},
```

5 TF-IDF Analysis

❷ 분석된 유사도에 기반한 추천

앞서 분석된 유사도에 기반하여 웹툰별로 생성된 행렬에서 추천할 웹툰 갯수를 출력한다.

```
for gnr_name in all_gnr:
    print(f'in {gnr_name}...', end=' ')
    doclist = df[df['genre']==gnr_name][['id', 'description']]
    recomend_by_word(doclist, result)
```

```
in 판타지... 유사도 분석을 위한 125x125matrix를 만들었습니다.->32개 웹툰에 대한 추천
in 로맨스... 유사도 분석을 위한 211x211matrix를 만들었습니다.->68개 웹툰에 대한 추천
in 스포츠... 유사도 분석을 위한 11x11matrix를 만들었습니다.->0개 웹툰에 대한 추천
in 무협/사극... 유사도 분석을 위한 21x21matrix를 만들었습니다.->0개 웹툰에 대한 추천
in 스릴러... 유사도 분석을 위한 46x46matrix를 만들었습니다.->4개 웹툰에 대한 추천
in 일상... 유사도 분석을 위한 10x10matrix를 만들었습니다.->0개 웹툰에 대한 추천
in 감성... 유사도 분석을 위한 3x3matrix를 만들었습니다.->0개 웹툰에 대한 추천
in 개그... 유사도 분석을 위한 12x12matrix를 만들었습니다.->0개 웹툰에 대한 추천
in 액션... 유사도 분석을 위한 46x46matrix를 만들었습니다.->5개 웹툰에 대한 추천
in 드라마... 유사도 분석을 위한 85x85matrix를 만들었습니다.->22개 웹툰에 대한 추천
```

5 TF-IDF Analysis

❷ 웹툰 제목으로 표기하여 유사도 높은 웹툰끼리 출력

고유 ID값으로 출력되는 웹툰명을 다시 제목으로 표기하여 유사도 높은 웹툰끼리 출력한다.

```
[249] # 웹툰 번호 제목으로 표기해서 출력
      def show_result(result):
          i = 1
          for key, value in result.items():
              print(i, df[df['id']==key].title.item(), end='\n➡ ')
              print(' / '.join([df[df['id']==v].title.item() for v in value]))
              print('---'*15)
              i += 1
```

```
show_result(result)
```

```
---------------------------------------------
24 연애 연기대상
➡ 완벽한 부부는 없다
---------------------------------------------
25 퇴근 후에 만나요
➡ 나쁜사람 / 나의 불편한 상사 / 가족같은 XX / 연애고수 / 순수한 동거생활 / THE 런웨이
---------------------------------------------
26 원작은 완결난 지 한참 됐습니다만
➡ 폭군 남편과 이혼하겠습니다
---------------------------------------------
27 우산 없는 애
➡ 연놈
---------------------------------------------
28 오늘의 비너스
➡ 존잘주의
---------------------------------------------
29 메리의 불타는 행복회로
➡ 똑 닮은 딸 / 보통아이 / 해귀
```

5 TF-IDF Analysis

❷ 콘솔창에 웹툰제목을 직접 입력하여 검색

검색기능 구현이 가능할지 확인하기 위해 콘솔창에 웹툰제목을 직접 입력하는 코드를 작성하였다.

```
[250] def find_recommend(result, title):
          print(f'웹툰 [{title}]과 비슷한 다른 웹툰은...', end=' ')
          others = result.get(df[df['title']==title].id.item(), [])
          if not others:
              print('아직 없습니다.')
              return
          print(' / '.join([df[df['id']==s].title.item() for s in others]))
```

5 TF-IDF Analysis

❷ 콘솔창에 웹툰제목을 직접 입력하여 검색

검색하여 나오는 결과의 예시는 다음과 같다.

```
find_recommend(result, '또다시, 계약 부부')
```

```
웹툰 [또다시, 계약 부부]과 비슷한 다른 웹툰은... 이 결혼, 새로고침 / 아마도, 굿모닝 / 호랑신랑뎐 / 반드시 해피엔딩 / 호랑이 들어와요
```

```
find_recommend(result, '이 짝사랑은 억울하다!')
```

```
웹툰 [이 짝사랑은 억울하다!]과 비슷한 다른 웹툰은... 우리 무슨 사이야?
```

```
find_recommend(result, '개를 낳았다')
```

```
웹툰 [개를 낳았다]과 비슷한 다른 웹툰은... 대신 심부름을 해다오 / 마루는 강쥐
```

데이터사이언스개론 프로젝트

THANK YOU

6조 송여경 이승연