Open API와 웹스크랩핑을 이용한 영화 데이터의 수집과 활용

1. Open API를 활용하기

본고는 Python 프로그래밍 언어를 통해 영화 아카이브 자료를 효율적으로 활용하는 하나의 방법을 제시하는 데 목적을 두고 있다. 따라서 이번 발표는 영화사 연구와 관련된 개인적인 문제의식을 정교화하고 연구 방법론을 고민하기보다 연구에 필요한 데이터를 수집, 활용, 응용, 공유하는 방법론에 대한 고민을 다룰 것이다. 본 발표는 크게 두 부분으로 구성되는데, 첫 번째는 영화진흥위원회에서 제공하는 `Open API`를 이용하여 영화 상영과 관련된 데이터를 수집하는 방법에 관한 것이고, 두 번째는 영상자료원의 웹페이지 스크랩핑을 이용하여 필요한 영화 자료를 수집하는 방법에 관한 것이다.

발표에 앞서 저를 비롯해서 많은 분들이 사용되는 용어의 낯섦이 있을 듯하여, 몇 가지 개념을 정리하고 가는 것이 필요할 듯하다.

> API((Application Programming Interface)

`API`란 응용 프로그램(application) 소프트웨어를 구축하고 정의하기 위한 프로토콜 세트로, 운영 체제(OS)와 소프트웨어 사이, 다바이스와 서버 사이, 혹은 웹 서비스 플랫폼(웹 서버)들 사이에서 이루어지는 커뮤니케이션을 위한 규약으로 활용된다.

PC에서 어플리케이션을 만들고 데이터를 저장하기 위해서는 운영 체제에서 제공되는 API에 따라 작업이 이루어지고, 다양한 기기에서 서버에 있는 데이터를 읽고 쓰기 위해서는 서버에서 제공하는 웹 API를 통해서 처리할 수 있다.

> REST Api

네트워크 상에서 기기들 간의 커뮤니케이션을 하도록 만든 규격 사항을 `HTTP`라고 하는데, 이와 같은 웹 API를 어떻게 만들 것인지를 정의하는 방식으로 `SOAP`(Simple Object Access Protocol)과 `REST`(REpresentational State Transfer) 가 있다. `SOAP`은 네트워크의 요청과 반응을 `XML`이라는 데이터 포맷에 저장하여 주고 받는 통신 프로토콜로서 구조화된 정보를 전송하는 데 이용된다. `REST`는 다양한 데이터 포맷으로 네트워크의 요청과 반응을 주고 받는데, 주로 `JSON`이라는 포맷을 사용하며, 데이터를 위해 리소스에 접근하는 데 이용되는 아키텍처 스타일이다.

일반적인 웹 서버와의 통신에서는 `REST` 방식이 보편적으로 많이 사용된다. `REST`는 `Post`, `Get`, `Put`, `Delete` 메소드로 구성되는데, 이는 각각 데이터를 생성, 읽기, 업데이트, 삭제하기를 의미한다. 예를 들어 `Get`을 이용해서 서버에 원하는 데이터를 `Request` (요청)하면, 서버로부터 해당 데이터를 `json`이라는 포맷으로 받아오게 된다.

> Function, Module, Package, Library

파이썬은 다중 패러다임 프로그래밍 언어로서 절차적, 객체지향적, 함수형으로 프로그램을 작성할 수 있다. 함수형 프로그래밍은 문제를 작은 조각으로 분해하여 접근하도록 이끄는데, 복잡한 변환을 수행하는 거대한 함수보다 한 가지 작업을 수행하는 작은 함수를 명시하고 작성하기가 더 쉽기 때문이다. 이는 결과적으로 프로그램이 모듈화가 되게끔 하는데, 함수를 포함하여 특정 작업을 수행하도록 만들어진 여러 기능들(변수, 클래스)을 따로 구현하여 파이썬 파일(.py)로 작성한 코드를 모듈이라 부른다. 그리고 일관된 기능적 목적 아래서 이러한 코드들을 배포하기 위해 묶어놓은 단위들을 패키지라 하며, 여러 모듈과 패키지들을 모은 상위 개념을 라이브러리라 명명한다. 그러나 이는 어디까지나 실용적 차원에서 코드를 작성하는 데 통용되는 느슨한 규정일 뿐 엄밀한 학문적 규정은 아니다. 패키지와 라이브러리의 경계는 모호하기에 본고에서는 널리 쓰이는 용법에 따라 "XX 패키지", "OO 라이브러리"라 부르기로 한다.

2. 영화진흥위원회 Kobis Open API 활용

영화관입장권통합전산망(kobis) 오픈API (이하 kobis api)를 활용하기 위해서는 해당 웹페이지에 접속하여 오픈 API가 어떤 방식으로 서비스되는지를 살펴보는 것이 중요하다.

제공 서비스에는 일별 박스오피스, 주간/주말 박스오피스, 공통코드 조회, 영화목록, 영화상세정보, 영화사목록, 영화사 상세정보, 영화인목록, 영화인 상세정보가 있다.

발표자에게 필요한 정보는 특정 년도에서 제작, 상영되었던 영화 목록이 필요한 만큼, 영화 목록 서비스를 선택하였다. 영화목록 조회 API 서비스는 REST 방식과 SOAP 방식을 모두 지원하며, 기본 요청 URL은 `http://www.kobis.or.kr/kobisopenapi/webservice/rest/movie/searchMovieList.xml`으로 나와 있는데, REST 방식으로 호출하기 위해서는 마지막 확장자를 `.xml`에서 `.json`으로 바꿔주어야 한다. 그리고 원하는 data를 서버로부터 받기 위해서는 요청 인터페이스 정보를 활용하여 요청 parameter를 입력해야 한다. 그리고 GET 방식으로 이를 호출하면 그에 맞는 응답이 이루어진다. 응답 구조는 총 16개의 값으로 구성되어 있는데, 실행을 해본 결과, 이 값들이 언제나 모두 응답되는 것은 아니고, 호출한 내용에 따라 그 응답 값이 정해지는 듯하다.

> 호출 예시

```
import urllib.request as ul

prdtyear_start = 1965
prdtyear_end = 1966
items = 10
page = 1
key = "로그인 후 받은 키 값을 입력(필수 parameter)"
url =
"http://www.kobis.or.kr/kobisopenapi/webservice/rest/movie/searchMovieList
.json?key={0}&prdtStartYear={1}&prdtEndYear={2}&itemPerPage={3}&curPage=
{4}".format(key, prdtyear_start, prdtyear_end, items, page)
print(url)
```

위의 코드를 실행하면 요청에 필요한 URL 이 출력되는데, 이를 통해 응답 받은 자료의 구조는 모두 14개 값으로 구성되어 있으며, 이를 기반으로 전체 코드를 작성하였다. 또한 본고에서는 응답 받은 자료를 보다 직관적으로 확인하고 저장하기 위해 구글 API를 이용하여 구글 스프레드시트에 호출한 자료들을 기록할 수 있도록 코드를 작성하였다.

> 모듈 import

```
import urllib.request as ul
import json
import gspread
from oauth2client.service_account import ServiceAccountCredentials
# pip install --upgrade google-api-python-client
import time
import pprint
pp = pprint.PrettyPrinter(indent=4)
```

구글 스프레드시트로 데이터를 기록하기 위해서는 구글 클라우드 플랫폼 에서 제공하는 서비스 계정이 필요하다. 서비스 계정은 해당 웹페이지에서 간단하게 만들 수 있다. 구글 API를 이용하기 위해서는 고유의 key가 필요한데, 이 또한 승인 절차 없이 json 파일로 받을 수 있다. (발표 내용 참고)

urllib.request 모듈은 URL을 여는 데 도움이 되는 함수와 클래스를 정의하고 있기에 웹 서버의 API를 통해 데이터를 호출하고 응답받기 위해 import 되었으며, json 모듈은 JSON 형식으로 받은 데이터를 파이썬 언어로 디코딩하기 위해 import 되었다.

gspread와 ServiceAccountCredentials는 구글 스프레드시트를 활용하기 위해, time과 pprint는 각각 코드의 실행을 제어하고, 코드 작성과정에 필요한 디버깅을 위해 import 하였다.

입력 값 변수 설정하기

```
# 입력값
prdtyear_start = "1960" # 제작년도 시작
prdtyear_end = "1961" # 제작년도 끝
sheet_name = "c"
```

prdtyear_start와 prdtyear_end는 kobis api에서 자료를 호출하기 위해 입력하는 parameter 값을 변수로 지정한 것이고, sheet_name은 구글 spreadsheet api를 호출할 때 필요한 시트 이름을 변수로 지정한 것이다.

Google Spreadsheet API

```
# Google Sheet API
scope = ['https://spreadsheets.google.com/feeds']
json_file_name = '/Users/seungjin/Documents/google_api/movie-
20210630.json'
credentials =
ServiceAccountCredentials.from_json_keyfile_name(json_file_name, scope)
```

```
gc = gspread.authorize(credentials)
spreadsheet_url = 'https://docs.google.com/spreadsheets/d/1RfF4-
DFF7nBZXTDfpK4O_-d3E1SWC7gZMYoR59AxvyA/edit#gid=0'
```

Google spreadsheet에 접근하기 위해서는 위와 같은 코드가 일반적으로 사용되는데, scope 변수에는 구글 스프레드시트에 접근하는 주소가 주어지며 이는 일반적으로 클라이언트들이 공통적으로 설정하는 값이다. json_file_name 변수에는 구글 API를 사용하기 위해 다운로드 받은 key파일(json 파일)이 저장된 로컬 저장소의 경로를 값으로 입력하며, spreadsheet_url에는 작성하고자 하는 스프레드시트의 URL을 값으로 입력한다.

> 영화 정보를 가져오는 함수

```
##########################################################
# 영화관입장권통합전산망 오픈API
# http://www.kobis.or.kr/kobisopenapi/homepg/main/main.do
##########################################################


def get_movie_info(prdtyear_start, prdtyear_end, items, page):
    key = "클라이언트에게 할당된 키 값"
    url =
"http://www.kobis.or.kr/kobisopenapi/webservice/rest/movie/searchMovieList
.json?key={0}&prdtStartYear={1}&prdtEndYear={2}&itemPerPage={3}&curPage=
{4}".format(key, prdtyear_start, prdtyear_end, items, page)

    request = ul.Request(url)
    response = ul.urlopen(request)
    rescode = response.getcode()
    if(rescode == 200):
        data = response.read()
        print(">>>> [Success] Result Code : {0}".format(rescode))
    else:
        print(">>>> [Fail] Result Code : {0}".format(rescode))
    result = json.loads(data)
    # pp.pprint(result)
    movie_list = result['movieListResult']['movieList']
    tot_cnt = result['movieListResult']['totCnt']
    return [movie_list, tot_cnt]
```

urllib.request (ul로 약칭) 모듈의 Request 함수를 이용하여 url 에 담긴 값을 인자로 받아 이를 kobis 웹 서버에 호출하는데, 그 내용을 request라는 변수에 저장한다. 이후 urlopen 함수에 request를 인자로 받아 서버로부터 응답을 받아오는데, 이를 response 변수에 할당하였다. 이후 서버와의 통신에 오류가 없는지 여뷰를 확인하기 위해 get.code()를 이용하여 응답코드를 받은 뒤, 이를 if 조건문으로 들여오 오류가 없을 때, 해당 데이터를 읽고 그 값을 data라는 변수에 저장되도록 만들었다. 웹 서버로부터 정상적인 값을 받고 코드가 실행되었다면, 응답 받은 데이터는 json 포

맷으로 작성되었을 것이기에 이를 파이썬 언어 형식으로 변환해주어야 한다. 따라서 `json` 모듈의 `loads` 함수를 이용하여 이를 결과 값으로 저장하였다.

서버로부터 응답 받은 json 데이터가 구체적으로 어떻게 계층화되어 있으며, 어떤 값들로 이루어졌는지를 확인하기 위해 `pp.pprint()`로 `result` 값을 풀어보았다. 데이터의 계층 구조를 파악한 결과 `movieListResult` 객체 아래 `movieList`라는 배열 자료와 `totCnt`의 값(호출한 총 영화의 수)이 문자열 형태로 주어져 있음을 확인하였다.

원하는 영화 정보는 모두 `movieList` 에 담겨 있었으나, 호출하는 데이터 수가 많을 경우 제약이 발생했기에 호출 건수를 적절히 제어하기 위한 별도의 코드가 필요했다. 따라서 호출한 데이터 전체를 카운트한 값 (`totCnt`) 도 함수가 반환하는 값에 포함하였다.

영화 총 건수 가져오기

```
a = get_movie_info(prdtyear_start, prdtyear_end, 1, 1)
tot_cnt = a[1]
print(">>> 총 영화수 : {0}".format(tot_cnt))
print('--------------------------\n')
```

조건을 만족하는 총 영화 건수를 가져온 뒤 이를 출력하는 부분이다. api 호출 건당 최대 100건만 가능하므로 page수를 파악하여 호출건수 제어하기 위해 이러한 과정이 필요했다. 토탈 카운트(`totCnt`) 알아야 호출할 페이지 수를 산출할 수 있기 때문이다.

구글 스프레드시트 불러오기

```
# 시트 불러오기
doc = gc.open_by_url(spreadsheet_url)
worksheet = doc.worksheet(sheet_name)
if worksheet.row_count > 3: worksheet.delete_rows(4, worksheet.row_count)
```

구글 스프레드시트에 호출한 데이터를 입력하기 위해 스프레드시트 문서를 불러오는데, `gspread` 모듈에서 지원하는 `open_by_url()`를 이용하여 지정된 문서를 열고, `worksheet()`를 이용하여 작업할 시트를 지정해 준다. 그리고 필요에 따라 스프레드 시트를 정리해주는 작업을 추가할 수 있다.

조건에 맞는 영화 데이터 가져오기

```
# 한번에 100건씩 pages 만큼 호출
```

```
items = 100 # 페이지당 데이터 수
pages = round(tot_cnt/items)
column_cnt = 14 # 구글시트 데이터 항목 컬럼 수
time.sleep(1)
tot_items = 0

for page in range(1, pages+1):
    print('>>> [INFO] Get Movie Info (page:{0})'.format(page))
    r = get_movie_info(prdtyear_start, prdtyear_end, items, page)
    movie_info_list = []
    for m in r[0]:
        tot_items = tot_items + 1
        movie_info_list.append(
            [
                m['movieCd'],
                m['movieNm'],
                m['movieNmEn'],
                m['prdtYear'],
                m['openDt'],
                m['typeNm'],
                m['prdtStatNm'],
                m['nationAlt'],
                m['genreAlt'],
                m['repNationNm'],
                m['repGenreNm'],
                '' if not m['directors'] else m['directors'][0]
['peopleNm'],
                '' if not m['companys'] else m['companys'][0]
['companyCd'],
                '' if not m['companys'] else m['companys'][0]
['companyNm'],
            ]
        )
    """
        (참고) 컬랙션형 데이터가 들어가 있는 형태.. 문서상에는 문자열이라 되어 있음
        'companys': [{'companyCd': '20060351', 'companyNm': '서울영화사'}],
        'directors': [{'peopleNm': '신상옥'}],
    """
    #pp.pprint(movie_list)
    worksheet.append_rows(movie_info_list)
    time.sleep(2)
    print(">>> [INFO] Upload Completed ... {0}/{1}\n".format(tot_items,
tot_cnt))
```

앞서 설명한 바와 같이 `get_movie_info(prdtyear_start, prdtyear_end, items, page)`의 각 인자들은 호출할 parameter 에 해당하는데, 호출하는 아이템 수가 최대 100건이기에 `items`를 100으로 설정하였다. 호출할 `pages`는 호출 조건인 `prdtyear_start`, `prdtyear_end` 에 할당된 값에 따라 달라지기에, `pages = round(tot_cnt/items)`와 같이 처리하였다.

이후 for구문으로 1 페이지에서 pages에 할당된 값까지 작업을 반복 수행하여 전체 페이지에 걸친 영화 정보를 웹 서버로부터 받아온다. 받아 온 자료들을 순차적으로 구글 스프레드시트에 기록하기 위해서는 1회에 응답 받는 데이터 100개에 대해 다시금 반복문을 실행하여 각 개별 영화에 대한 정보를 리스트형으로 구축해야 한다. 이를 수행하기 위해 movie_info_list 변수에 리스트 자료형 컨테이너를 마련해두었다.

```
for m in r[0]:
```

구문을 반복 수행하면서 movie_info_list 에는 movieCd, movieNm, movieNmEn, prdtYear, openDt 등의 키(key) 값에 해당하는 밸류(value) 값들이 리스트 자료형으로 쌓이게 된다.

```
worksheet.append_rows(movie_info_list)
```

그리고 최종적으로 gspread 모듈에서 지원하는 ppend_rows 모듈을 통해서 구글 스프레드시트에 자료를 덧붙여씀으로써 데이터 구축이 완료된다.

3. 웹 스크랩핑을 이용한 데이터 구축

> 모듈 설치 및 구글 API 설정하기

```
import requests
from bs4 import BeautifulSoup
import gspread
from oauth2client.service_account import ServiceAccountCredentials
import pprint
import time
import math

prodStartYear ='1960'
prodEndYear = '1961'
sheetname = 'kmdb_mv_censor_list_kor'


scope = ['https://spreadsheets.google.com/feeds']
json_file_name = '/Users/seungjin/Documents/google_api/movie-
20210630.json'
credentials =
ServiceAccountCredentials.from_json_keyfile_name(json_file_name, scope)
gc = gspread.authorize(credentials)
spreadsheet_url = 'https://docs.google.com/spreadsheets/d/1BtT-
```

```
SrZEuXupHA7GRJ6ABwcg2Dh_fe6lXJ-6TskYWNs/edit#gid=0'
```

기본적인 모듈 설치 및 구글 api 설정 방식은 앞서 진행하였던 과정과 동일하다. 다만 웹 스크리핑에 필요한 모듈로서 `BeautifulSoup`을 import 하였고, `urllib.request`를 대신하여 `requests` 모듈을 사용하였다.

> 작업 과정을 제어할 함수 작성

```
def get_items_cntNum(startCount, prodStartYear, prodEndYear):
    url = 'https://www.kmdb.or.kr/db/have/detailSearch/censorSearch?
_csrf=5149a247-b5d3-4711-ab75-
cd288c64cf98&collection=kmCENSOR&tabName=sojangTab&startCount=
{0}&storedPosition=&censorName=&censorNameSelect=AND&releatedMovieName=&re
leatedMovieNameSelect=AND&movieMemberName=&movieMemberType=director&movieM
emberTypeSelect=AND&companyName=&contentName=&prodStartYear=
{1}&prodEndYear={2}'.format(startCount, prodStartYear, prodEndYear)
    res = requests.get(url)
    res.raise_for_status()
    soup = BeautifulSoup(res.text, "lxml")
    # data_rows = soup.find("table", attrs={"class":"data-table medium
transform-m"}).find("tbody").find_all("tr")
    count = soup.find("div", attrs={"class":"result-block-tt
noline"}).find("span", attrs={"class":"em weighty"}).get_text()
    return(count)


items_cntNum = get_items_cntNum(0, prodStartYear, prodEndYear)
startCounts = math.floor(int(items_cntNum)/10)*10
print(">>> startCount 최대값 = {0}".format(startCounts))
print('--------------------------\n')
```

코드가 작동하는 기본적인 방식의 앞서 kobis api 와 거의 동일하다. 다만, 웹 서버에서 제공하는 api를 따라 정보를 가져오는 것이 아니라 웹 페이지 상에 있는 정보를 타겟팅하여 가져오는 것이기에, `BeautifulSoup` 모듈에서 지원하는 함수들을 사용하였다.

```
count = soup.find("div", attrs={"class":"result-block-tt
noline"}).find("span", attrs={"class":"em weighty"}).get_text()
```

해당 웹 페이지에서 발표자가 가져오고 싶어 했던 정보는 전체 자료의 갯수였는데, 위의 코드는 그 작업을 구체적으로 수행하는 부분이다. 우선 `find()`를 사용하여 `class` 라는 속성(attributes)에 해당하는 값인 `result-block-tt noline`를 찾고, 하위의 계층에 속한 테이블 속성값들 중 `em weighty` 부분을 찾아, 그 값을 `get_text()`를 사용하여 텍스트로 불러온 뒤, 이를 `count`라는 변수의 값으로 주었다.

구글 스프레드시트 문서

```
doc = gc.open_by_url(spreadsheet_url)
worksheet = doc.worksheet(sheetname)

# if worksheet.row_count > 2: worksheet.delete_rows(3,
worksheet.row_count)

pages = math.ceil(startCounts/items)
column_cnt = 5 # 구글시트 데이터 항목 컬럼 수
```

앞서의 과정과 동일하게, 구글 스프레드문서를 열어주고 작업 과정의 제어를 위해 `page` 수를 지정해놓는다.

웹 스크랩핑 코드

```
time.sleep(2)
tot_items = 0
for startCount in range(0, startCounts+10, 10):
    print('>>> [INFO] Get Movie Censor Info (startCount:
{0})'.format(startCount))
    url = 'https://www.kmdb.or.kr/db/have/detailSearch/censorSearch?
_csrf=5149a247-b5d3-4711-ab75-
cd288c64cf98&collection=kmCENSOR&tabName=sojangTab&startCount=
{0}&storedPosition=&censorName=&censorNameSelect=AND&releatedMovieName=&re
leatedMovieNameSelect=AND&movieMemberName=&movieMemberType=director&movieM
emberTypeSelect=AND&companyName=&contentName=&prodStartYear=
{1}&prodEndYear={2}'.format(startCount, prodStartYear, prodEndYear)
    res = requests.get(url)
    res.raise_for_status()
    soup = BeautifulSoup(res.text, "lxml")

    data_rows = soup.find("table", attrs={"class":"data-table medium
transform-m"}).find("tbody").find_all("tr")

    for row in data_rows:
        columns = row.find_all("td")

            # "td" 요소가 하나 또는 그 이하를 빼주는 코드 > 불필요한 공백을 없애기 위해, 각
각의 "tr"들 사이의 차이를 고려.
        if len(columns) <= 1: # 의미 없는 data는 skip
            continue

        data = [column.get_text().strip() for column in columns]
        # print(type(data))
        tot_items = tot_items + 1
        movie_censor_list = [data]
```

```
            worksheet.append_rows(movie_censor_list)
        time.sleep(10)
        print(">>> [INFO] Upload Completed ... {0}/{1}\n".format(tot_items,
    int(items_cntNum)))
            # print(movie_censor_list)
```

웹 스크랩핑을 하기 위해서는 해당 웹 페이지에 대한 분석이 필요한데, 발표자가 자료를 가져온 페이지는 한국영상자료원의 심의서류 목록 페이지이다. 가져오고 싶었던 정보는 1960년에서 1961년까지 생산된 심의서류의 갯수와 그와 관련된 정보들이었다.

위의 코드는 해당 웹페이지를 분석하여 필요한 정보들이 있는 위치를 파악한 뒤 작성된 것이다.

```
data_rows = soup.find("table", attrs={"class":"data-table medium
transform-m"}).find("tbody").find_all("tr")
```

html 문서로 작성된 웹페이지에서 각각의 태그 명을 통해 해당 정보가 있는 위치를 파악하고, 해당 태그 아래 작성된 값들을 모두 `soup` 객체로 변환한 뒤 이를 `data_rows` 변수에 할당하였다. 해당 페이지는 10개의 영화 심의자료 정보를 제공해주는데, 따라서 `data_rows`에는 10개의 테이블 데이터가 쌓여있다.

이후 반복문을 실행하여 각 항목에 있는 텍스트를 `movie_censor_list`라는 변수에 리스트 자료형으로 담은 다음, 이를 구글 스프레드시트에 쓰는 작업을 수행하였다. 여기서 `time` 모듈의 `sleep()` 값을 `10`으로 두었는데, 이는 1회에 요청할 수 있는 구글 API의 값이 정해져 있어 오류를 피하기 위해 10초로 설정해두었다.

4. 나가며

Open API 와 웹스크랩핑을 이용하여 수집한 데이터는 각각의 데이터 테이블로 존재하는데, 이를 기반으로 데이터를 활용할 수 있는 방안은 무수히 열려있다. 예를 들어, kobis api를 이용하여 수집한 영화 목록 데이터에서 '한국영화'를 분류한 다음, 이를 심의서류 자료 데이터 테이블과 비교하면서 심의서류가 남아 있지 않은 영화목록을 손쉽게 추려낼 수 있다. 또한 '서양영화'라고 판단될 수 있는 항목들을 분류한 다음, www.eiga.com 과 같이 개봉 일시가 데이터베이스화되어 있는 웹 페이지에서 상영정보를 데이터 테이블로 가져온 뒤 이를 join 하여 한국에서 해당 영화가 상영된 시점을 어느 정도 추론할 수 있다.