DSC 540- Milestone-3 WebScraping

Kannur, Gyan

Instructor Catherine Williams


```
import pandas as pd
import requests
import traceback
from bs4 import BeautifulSoup
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

Speaking about WebScraping the box office hits, the most complicated part of this is inspecting the webpage source code to determine what to grab and what to ignore. I started off to create a dataset from the weekend performance page of Box Office Mojo, a great datasource for box office performance data.

> Generally tabular data that is visible on the page will be put into 'tr' tags. With some of the code below I am exploring edge cases in the ouput of the page. When there is a special occasion for the weekend, in this case thanksgiving, there is a different format and structure that is displayed.

```
# Request to website and download HTML contents
scrape_website='https://www.boxofficemojo.com/weekend/by-year/'

weekend_data = pd.DataFrame()
#Get last 10 years of box office collections
years = list(range(2013,2024,1))
years
#remove later
weekend_data = pd.read_pickle("raw_web_scrapped_df_pickle")
```

Writing a function which scrapes the box office contents

```
def scrape_for_year(year):

    url = f'https://www.boxofficemojo.com/weekend/by-year/{year}/'
    response=""
    try:
        response = requests.get(url)
```

```
    except Exception as ex:
        print(f"Exception occurred {ex}")
        traceback.print_exc()
    content = response.text
    ## Reading webpage using beatifulsoup method available in bs4
    soup = BeautifulSoup(content)
    ## Find the tabular data that is visible on the page will be put
into 'tr' tags
    rows = soup.findAll('tr')

    appended_data = []
    for row in rows:
        data_row = {}
        data_row['year'] = yr
        data = row.findAll('td')
        if len(data) == 0:
            continue
        if len(data[0].findAll('span')) > 0:
            #Data for special weekend
            data_row['occasion'] = data[0].findAll('span')[0].text
            data_row['date'] = data[0].findAll('a')[0].text
        else:
            data_row['occasion'] = "normal weekend"
            data_row['date'] = data[0].text
        data_row['top10_gross'] = data[1].text
        data_row['top10_wow_change'] = data[2].text
        data_row['overall_gross'] = data[3].text
        data_row['overall_wow_change'] = data[4].text
        data_row['num_releases'] = data[5].text
        data_row['top_release'] = data[6].text
        data_row['week_no'] = data[10].text
        appended_data.append(data_row)
    return appended_data
```

Creating Dataframe from the appended yearly data

```
for yr in years:
    result = scrape_for_year(yr)
    scrapped_df = pd.DataFrame(result, columns =
['date','occasion','year', 'top10_gross', 'top10_wow_change',
'overall_gross', 'overall_wow_change', 'num_releases', 'top_release',
'week_no'])
    weekend_data =
pd.concat([weekend_data,scrapped_df],ignore_index=True)

#remove later
#weekend_data.to_pickle("raw_web_scrapped_df_pickle")
weekend_data.shape
```

```
(1354, 10)
```

```
weekend_data.head()
```

```
           date             occasion  year   top10_gross
top10_wow_change  \
0     Dec 27-29      normal weekend  2013  $167,837,974
+24.5%
1     Dec 20-22      normal weekend  2013  $134,837,792            -
2.6%
2     Dec 13-15      normal weekend  2013  $138,369,003
+64.8%
3       Dec 6-8   Post-Thanksgiving  2013   $83,941,456           -
55.7%
4  Nov 29-Dec 1  Thanksgiving 3-Day  2013  $189,483,142           -
11.7%

  overall_gross overall_wow_change num_releases  \
0  $197,177,755             +37.3%           81
1  $143,571,365              -2.8%           80
2  $147,702,714             +56.2%           94
3   $94,535,353             -54.6%           96
4  $208,125,032              -8.1%          104

                           top_release week_no
0  The Hobbit: The Desolation of Smaug      52
1  The Hobbit: The Desolation of Smaug      51
2  The Hobbit: The Desolation of Smaug      50
3                              Frozen      49
4      The Hunger Games: Catching Fire      48
```

```
weekend_data.columns
```

```
Index(['date', 'occasion', 'year', 'top10_gross', 'top10_wow_change',
       'overall_gross', 'overall_wow_change', 'num_releases',
'top_release',
       'week_no'],
      dtype='object')
```

Step 2: Removing null rows from the dataset

```
## Calculating number of null rows present in the dataset
weekend_data.occasion.value_counts()
```

```
occasion
normal weekend                    932
COVID-19 Pandemic                 128
Indig. Peoples' Day wknd           20
Labor Day wknd                     20
MLK wknd                           20
Thanksgiving 3-Day                 18
Thanksgiving 4-Day                 18
Thanksgiving 5-Day                 18
```

```
Memorial Day wknd                                18
Easter wknd                                      18
Post-Thanksgiving                                18
Presidents' Day wknd                             16
New Year's long wknd                             10
July 4th long wknd                               10
Christmas long wknd                              10
World Cup (Russia)                               10
World Cup (Brazil)                               10
Sochi Olympics                                    6
PyeongChang Olympics                              6
Rio Olympics                                      6
COVID-19 PandemicMemorial Day wknd                4
World Cup (Qatar)                                 4
COVID-19 PandemicEaster wknd                      4
Mayweather/Pacquiao Fight                         2
Thanksgiving 4-DayWorld Cup (Qatar)               2
Thanksgiving 3-DayWorld Cup (Qatar)               2
Post-ThanksgivingWorld Cup (Qatar)                2
COVID-19 PandemicMLK wknd                         2
COVID-19 PandemicPresidents' Day wknd             2
COVID-19 PandemicIndig. Peoples' Day wknd         2
COVID-19 PandemicLabor Day wknd                   2
COVID-19 PandemicThanksgiving 5-Day               2
COVID-19 PandemicThanksgiving 4-Day               2
COVID-19 PandemicThanksgiving 3-Day               2
COVID-19 PandemicPost-Thanksgiving                2
Presidents' Day wkndPyeongChang Olympics          2
Presidents' Day wkndSochi Olympics                2
Thanksgiving 5-DayWorld Cup (Qatar)               2
Name: count, dtype: int64
```

Step 3: Remove dollar signs and comma convert to integer

```
weekend_data['top10_gross'] =
weekend_data['top10_gross'].replace('[$,]', '',
regex=True).astype(int)
weekend_data['overall_gross'] =
weekend_data['overall_gross'].replace('[$,]', '',
regex=True).astype(int)
```

Step 4: Create some new columns in millions for each gross column

```
# Create the column in millions
weekend_data['top_10_gross_in_millions'] = weekend_data['top10_gross']
/ 1000000
weekend_data['top_10_gross_in_millions'] =
weekend_data['top_10_gross_in_millions'].apply(lambda x: f"{x:.2f}")

weekend_data['overall_gross_in_millions'] =
```

```
weekend_data['overall_gross'] / 1000000
weekend_data['overall_gross_in_millions'] =
weekend_data['overall_gross_in_millions'].apply(lambda x: f"{x:.2f}")

weekend_data.head()
```

```
           date            occasion  year  top10_gross
top10_wow_change  \
0     Dec 27-29      normal weekend  2013    167837974
+24.5%
1     Dec 20-22      normal weekend  2013    134837792             -
2.6%
2     Dec 13-15      normal weekend  2013    138369003
+64.8%
3       Dec 6-8   Post-Thanksgiving  2013     83941456           -
55.7%
4  Nov 29-Dec 1  Thanksgiving 3-Day  2013    189483142           -
11.7%

   overall_gross overall_wow_change num_releases  \
0      197177755             +37.3%           81
1      143571365              -2.8%           80
2      147702714             +56.2%           94
3       94535353             -54.6%           96
4      208125032              -8.1%          104

                            top_release week_no
top_10_gross_in_millions  \
0  The Hobbit: The Desolation of Smaug      52
167.84
1  The Hobbit: The Desolation of Smaug      51
134.84
2  The Hobbit: The Desolation of Smaug      50
138.37
3                               Frozen      49
83.94
4      The Hunger Games: Catching Fire      48
189.48

  overall_gross_in_millions
0                    197.18
1                    143.57
2                    147.70
3                     94.54
4                    208.13
```

Step 5: Convert gross columns to float and movie name to lower case

```
weekend_data['top_10_gross_in_millions'] =
weekend_data['top_10_gross_in_millions'].astype(float)
```

```
weekend_data['overall_gross_in_millions'] =
weekend_data['overall_gross_in_millions'].astype(float)
weekend_data['top_release'] = weekend_data.top_release.str.lower()
weekend_data["num_releases"] =
weekend_data.num_releases.astype('int64')
```

Step 6: Replace Headers with more meaningful names

```
weekend_data.rename(columns={'top_release': 'movie_title'},
inplace=True)
```

Step 7: Replace arrange Headers

```
new_col_order=['movie_title', 'year' , 'occasion',
'top_10_gross_in_millions', 'overall_gross_in_millions',
'top10_wow_change','overall_wow_change','top10_gross',
'overall_gross',  'num_releases',
               'week_no','date']

for i,col in enumerate(new_col_order):
    tmp = weekend_data[col]
    weekend_data.drop(labels=[col],axis=1,inplace=True)
    weekend_data.insert(i,col,tmp)

weekend_data.head()
```

```
                          movie_title  year            occasion  \
0  the hobbit: the desolation of smaug  2013      normal weekend
1  the hobbit: the desolation of smaug  2013      normal weekend
2  the hobbit: the desolation of smaug  2013      normal weekend
3                              frozen  2013   Post-Thanksgiving
4      the hunger games: catching fire  2013  Thanksgiving 3-Day

   top_10_gross_in_millions  overall_gross_in_millions
top10_wow_change  \
0                    167.84                     197.18
+24.5%
1                    134.84                     143.57             -
2.6%
2                    138.37                     147.70
+64.8%
3                     83.94                      94.54           -
55.7%
4                    189.48                     208.13           -
11.7%

  overall_wow_change  top10_gross  overall_gross  num_releases week_no
\
0             +37.3%    167837974      197177755            81      52

1              -2.8%    134837792      143571365            80      51
```

```
2             +56.2%    138369003      147702714             94       50

3             -54.6%     83941456       94535353             96       49

4              -8.1%    189483142      208125032            104       48
```

```
           date
0     Dec 27-29
1     Dec 20-22
2     Dec 13-15
3       Dec 6-8
4  Nov 29-Dec 1
```

Gather the list of unique movie titles and store it for future milestones!

```
scraped_movie_list=weekend_data['movie_title'].unique()
len(scraped_movie_list)
```

```
353
```

```
weekend_data.to_csv(r'./project_datasets/clean-
webscraped.csv',index=False)
```

WebScraping Ethical Implications and Assumptions:

WebScraping Ethical Implications and Assumptions:

Are there any legal or regulatory guidelines for your data or project topic?

The budget data is the source for judging how well the movie performed which impacts the movie industry. Scraping copyrighted or sensitive data can lead to legal consequences and fortunately there was nothing very significant in my project except for the budget details.

As the website allows scraping, the guidelines follows some of the best practices for scraping data. It should be noted that excessive requests may overload servers, hence one should perform a fetch all instead of fetch often.

Did you make any assumptions in cleaning/transforming the data? The wow change and top wow change are opinioned and not based on any logic. I do not plan to use that as part of my analysis.

How was your data sourced / verified for credibility? boxofficemojo is a well known website.

Was your data acquired in an ethical way?

Take the root of the url, in this case https://www.boxofficemojo.com and add '/robots.txt' to the end. This will come up with a page that shows what type of web scraping is allowed or disallowed.

/#robots.txt for BoxOfficeMojo

User-agent: *

Allow: /

Thankfully Box Office Mojo allows all and we should not have issues with scraping this website.