Course1: Foundation of Information

Part – B [Marks 35]

Project-1: Using web scraping to build a database of movie related information from: The Movie Database (TMDB) movie data

Problem statement:

A common business requirement in the context of information gathering is to extract and filter relevant data from web pages that host this information. However, access to information spread over several web pages, hosted potentially on multiple websites is a cumbersome process and we cannot rely on manual procedures to execute this task. In this project, you will employ a programmatic approach to access, parse and extract relevant information from a website of interest.

Objective:

The project's goal is to extract data (from a chosen number of pages) from The Movie Database website (https://www.themoviedb.org/) into a tabular data format so that further analysis (e.g., details about a movie's genre, cast, and user rating) can be facilitated.

To execute this project, you will have to read the documentation links provided against each task in the assignment and adapt the code examples provided in the documentation for the task at hand

Pre-requisites :

Tools:
Jupyter Notebook or Google Colab or Microsoft Visual Studio IDE

Languages: Python, HTML

Libraries: requests, beautifulSoup, pandas

Task Questions (35 Marks)

1. **Establish a connection to the webpage - "https://www.themoviedb.org/movie" - and provide the following details (4 marks)**

    a. Import the requests library (https://requests.readthedocs.io/en/latest/) and formulate a get request to download the contents of the webpage ("https://www.themoviedb.org/movie") (1 mark)
    b. Verify the status code of the request and confirm that the request was executed appropriately (https://requests.readthedocs.io/en/latest/user/quickstart/#response-status-code) (1 mark)
    c. Print the contents of the page obtained from the response and save it in a variable (https://requests.readthedocs.io/en/latest/user/quickstart/#response-content) (1 mark)
    d. Infer the type of the variable created in part 1c and display the first 200 characters of the content from the server's response (1 Mark)

Note: While pulling request from the given link if you address "HTTP Status code Error 403". It is a client error response code that indicates that the server understands the request made by the client, but refuses to authorize it. Reasons could be Incorrect or missing authentication credentials, Insufficient permissions, IP blocking or Resource is no longer available.
Specifying a user agent string is important because some servers may serve different content or limit access to certain resources based on the user agent string. Please copy the user agent string to pull the request

```
needed_headers = {'User-Agent': "Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36"}

response = requests.get(("https://www.themoviedb.org/movie"),headers = needed_headers)
```

2. **Parse the content of HTML response using the BeautifulSoup library and execute the tasks specified in the guidelines mentioned below (6 marks)**

    a. From the BeautifulSoup library (**bs4**) import the **BeautifulSoup** class. Pass the contents of the webpage obtained from step 1c as an argument to create an instance of the **BeautifulSoup** class (2 Marks)

Hint : Use " html.parser " as another argument of the BeautifulSoup constructor function (https://www.crummy.com/software/BeautifulSoup/bs4/doc/#making-the-soup)

b. Extract the title of the parsed web page content using an appropriate method or attribute of the document object created in part 2a (https://www.crummy.com/software/BeautifulSoup/bs4/doc/#navigating-using-tag-names) (1 Mark)

c. Write a user defined function to generalize the task presented in Q2a to any URL that retrieves the content of the webpage. Your function should take a URL string as an input and return a correctly formulated **BeautifulSoup** instance as the output. In your function definition, ensure that appropriate exceptions are raised to the user (through status codes) if they pass in malformed/incorrect URLs. Write two test cases for your function - one with a working URL and another with an URL that gets a 404 response. (3 marks)
Note: *While pulling the request please specify a user agent string as showed in Q.1.*

3. Extract the content of the webpage - https://www.themoviedb.org/movie - that hosts a current dated listing of popular movies. (5 Marks)

a. Write a function call to the user defined function created in 2c with the url https://www.themoviedb.org/movie as an input and store the response in a variable (1 mark)

Apply appropriate methods from the BeautifulSoup library *(https://www.crummy.com/software/BeautifulSoup/bs4/doc/#making-the-soup)* on the variable created in part 3a to answer the following questions:

b. Print the HTML content associated with the first movie displayed on the web page using appropriate HTML tags to access this listing on the object created in part 3a (1 mark)
c. Display the name of the first movie using appropriate HTML tags to access this listing on the object created in part 3a (1 mark)
d. Display the user rating of the first movie by using appropriate HTML tags to access this listing on the object created in part 3a (1 mark)
e. For the first movie, extract the part of the url following the string "https://www.themoviedb.org/" using the appropriate HTML tags to extract this portion on the object created in part 3a (do not use built-in string methods). (1 mark)
For example, if the first movie on the web page had the URL https://www.themoviedb.org/movie/779782 " your output should be **movie/779782**

Hint : Use F12 key (Console) in google chrome browser to get the details on HTML components (such as tags, classes) of a webpage. This method of "inspecting" HTML components of interest is widely

used by front end developers. Refer to the support documentation attached with this project for further information

4. **Write user defined functions for each subsection below (i.e., Q4 a, Q4b, Q4c, Q4d, and Q4e) to return (10 marks)**

    a. Titles of all the movies on the page as a Python list (2 marks)
    b. User ratings of all the movies on the page as a Python list (2 marks)
    c. HTML content of all the individual pages of movies collected into a Python list. (2 marks)
       (Hint : For each of the movies listed on the page, extract the end url as obtained in q3e)
    d. Genres of all the movies on the page as a Python list (2 marks)
    e. Cast of all the movies on the page as a Python list (2 marks)

    Note :

    A) Input to the user defined functions :
    - For Q4a, Q4b, Q4c: the response object created in Q3a
    - For Q4d, Q4e: the list output from Q4c

    B) Note that some movies might not have a user rating. Your function should be able to parse the numeric value of rating when it exists. When the rating is not available, the value "not rated" should be appended to the list created in Q4b

5. **Write an user defined function that returns a pandas data frame with following data: (5 marks)**

    a. Titles of the movies listed on the page
    b. User ratings of the movies listed on the page
    c. Genres of the movies listed on the page
    d. Cast of the movies listed on the page

    Hint : The above information can be obtained by making function calls to the user defined functions in Q4a,b,d,e

    Note:
    Input to the user defined function :

    - The response object created in Q3a
    - The list output from Q4c

6. **Scraping the data and combining the dataframes (5 marks)**

(a) Write a function that scrapes data (mentioned in Q5) from page number 1, 2, 3, 4 and 5 on the URL https://www.themoviedb.org/movie and returns 5 data frames which can be exported to csv file by calling the functions defined in Q3a, Q4c and Q5 (3 marks)

Hint : use the string " ?page={ } " concatenated to the string " https://www.themoviedb.org/movie " which extracts the data from desired page no passed to page value

(b) Combine the data obtained from dataframes in Q6(a) (2 marks)

Hint : refer to data integration of week4 to utilize the appropriate method available in pandas library to combine dataframes

HAPPY LEARNING!