web scraper - goodreads

Oct 30, 2019

Contents:

1 Documentation for the Code **1**

1.1 HelperUtils.py . . . 1
1.2 FilePicking.py . . . 1
1.3 GenreScraper.py . . . 2
1.4 MainBookScraper.py . . . 4
1.5 SiteNavigator.py . . . 5
1.6 BookReviews.py . . . 5
1.7 book_review_visualization.py . . . 7
1.8 review_rating_calculation.py . . . 7

2 Indices and tables **9**

Python Module Index **11**

Index **13**

CHAPTER 1

Documentation for the Code

1.1 HelperUtils.py

- *extract_book_name_from_root_url(root_url)*
- *check_if_file_exists_otherwise_handle(file_path)*
- *data_for_book_exists_current_date(book_data_folder)*

HelperUtils.**check_if_file_exists_otherwise_handle**(*file_path*)
This function checks if the path file exists in the path If it doesnt then returns false and if it exists then deletes it and returns true

Parameters **file_path** (*str*) – path of the file to check

Returns bool flag indicating whether file exists or not

HelperUtils.**data_for_book_exists_current_date**(*book_data_folder*)
Checks whether a folder exists and if its creation is the current date or not

Parameters **book_data_folder** (*str*) – folder name/partial path

Returns bool flag indicating whether folder exists or not

1.2 FilePicking.py

- *save_obj(obj, name, directory)*
- *load_obj(name, directory)*
- *load_latest_obj(name, directory)*

FileUtil.FilePicking.**load_latest_obj**(*name*, *directory*)
Loads the latest .pkl file from a book directory

Parameters

- **name** (*str*) – name with which to save the .pkl file
- **directory** (*str*) – name of the directory where the .pkl file is saved

FileUtil.FilePicking.**load_obj**(*name*, *directory*)
Loads the .pkl file named *name* from *directory*

Parameters

- **name** (*str*) – name with which to load the .pkl file
- **directory** (*str*) – name of the directory where the .pkl file is saved

FileUtil.FilePicking.**save_obj**(*obj*, *name*, *directory*, *json_save_needed*)
Functions checks if the individual book directory exists as of current date If it does'nt exist then it creates it otherwise it deletes the older directory and recreates it Pickles and saves the *obj* with the name *name* in dir *directory* Also saves the data as json in the same folder based on *json_save_needed*

Parameters

- **obj** (*python object*) – this is being pickled
- **name** (*str*) – name with which to save the .pkl file
- **directory** (*str*) – name of the directory where the .pkl file is saved
- **json_save_needed** (*bool*) – indicates whether data needs to be saved as json or not

---

1.3 GenreScraper.py

Functions:

- *_create_main_parser(genre)*
- *_build_book_details_map((sci_fi_book_details, book_index, book_details))*
- *_retrieve_book_name(book_block, class_name)*
- *_retrieve_book_URL_and_image_URL(book_block)*
- *_retrieve_author_name(book_block, class_name)*
- *_retrieve_number_of_times_shelved(book_block)*
- *_retrieve_rating_published_details(book_block)*
- *retriveSciFiBookList(genre)*

GenreScraper.**_create_main_parser**(*genre*)
Creates the bs4 parser from the goodreads URL. This is to scrape details of most popular books in a genre

Parameters **genre** (*str*) – Genre to scrape book details

Returns parser

Return type bs4 object

GenreScraper.**_build_book_details_map**(*sci_fi_book_details*, *book_index*, *book_details*)
Build the dictionary containing the details of the book. This is then further pickled for persistent storage

Parameters

- **sci_fi_book_details** (*dict*) – dictionary to store book details
- **book_index** (*int*) – counter variable for the dictionary *sci_fi_book_details*
- **book_details** (*list*) – List containing the different details about the book
- **needs to be stored in the dict sci_fi_book_details** (*that*) –

Returns book details

Return type dict

GenreScraper.**_retrieve_book_name**(*book_block*, *class_name*)
Finds the book name which is in a link <a> tag

Parameters

- **book_block** (*bs4*) – represents the bs4 for an individual book on the webpage
- **class_name** (*str*) – the CSS class name needed to extract the book name

Returns name of the book

Return type str

GenreScraper.**_retrieve_book_URL_and_image_URL**(*book_block*)
Finds the book URL (which we use in :mod: *web_scraper_goodreads_root.BookReviews*) and book image URL

Parameters **book_block** (*bs4*) – represents the bs4 for an individual book on the webpage

Returns book url and book image url

Return type list

GenreScraper.**_retrieve_author_name**(*book_block*, *class_name*)
Finds the author details of the book Author details are: -author name -Goodreads author URL -is author on goodreads or not

Parameters

- **book_block** (*bs4*) – represents the bs4 for an individual book on the webpage
- **class_name** (*str*) – the CSS class name needed to extract the book name

Returns author details

Return type list

GenreScraper.**_retrieve_number_of_times_shelved**(*book_block*)
Finds the number of times a book has been shlved by people Shelving is just a way for users of goodreads to categorize a book

Parameters **book_block** (*bs4*) – represents the bs4 for an individual book on the webpage

Returns number of times shelved

Return type int

GenreScraper.**_retrieve_rating_published_details**(*book_block*)
Finds rating details of the book Rating details include: -average rating -number of ratings -year the book was published

Parameters **book_block** (*bs4*) – represents the bs4 for an individual book on the webpage

Returns rating details

Return type list

GenreScraper.**retriveSciFiBookList**(*genre*)

Main entry into *GenreScraper* This function does the following:

1. Creates the bs4 parser - *_create_main_parser(genre)*
2. Gets a list of bs4 for each of the books
3. Loops through each book, retrives the book details and stores in dict *sci_fi_book_details*

Parameters **genre** (*str*) – genre to scrape details about

Returns book details from a particular genre

Return type dict

1.4 MainBookScraper.py

Functions:

- *generate_book_review_images()* : Scrapes goodreads.com for reviews and visualizes data

MainBookScraper.**generate_book_review_images**(*genre*)

Does the following:

1. Scrapes goodreads.com to get list of most popular book & details for input *genre*
2. Saves details to pickle file
3. Loads the latest pickle file data
4. **Loops through the book list to:**
    - extract book name from book URL
    - scrape book review details
    - save details to pickle file
    - load latest pickle file data
    - visualize review likes data

Args: genre (str): book genre to process

Note: When there is a timeout exception during scraping, *generate_book_review_images* function will retry upto *FAILURE_THRESHOLD* from web_scraper_goodreads_root.CommonConstants.Constants times before skipping the book

---

1.5 SiteNavigator.py

Functions:

- *_init(root_url)*
- *get_html_code_for_first_page(root_url, new_book)*
- *get_html_code_for_other_pages(root_url)*

SiteNavigator.**_init**(*root_url*)

Creates and initializes selenium object for traversal. Running via chromedriver.exe. For this code to work ensure chromedriver.exe is on the path. Download from here Making this a headless selenium object via chrome_options

Parameters **root_url** (*str*) – used to initialize selenium object

Returns selenium instance

SiteNavigator.**get_html_code_for_first_page**(*root_url*, *new_book*)

Visits the first page in a book and returns the HTML code for the book review part The *new_book* indicator helps to ensure we recreate the selenium driver instance for every book

Parameters

- **root_url** (*str*) – used to initialize selenium object
- **new_book** (*bool*) – indicates whether its a new book or not

Returns html code of the book reviews

SiteNavigator.**get_html_code_for_other_pages**(*root_url*)

Visits the other review pages and returns the html code Cliking on the a review page at the bottom performs an Ajax call which returns a Element.update() which in turn updates the "reviews" id with HTML code

The way we check whether the review data has loaded is by adding a dummy id in the HTML and waiting till its not present anymore after the Ajax call

Parameters **root_url** (*str*) – used to initialize selenium object

Returns html code of the book reviews

1.6 BookReviews.py

- *_create_book_review_scraper_from_source(html_source)*
- *_retrieve_review_rating(book_review_tag)*
- *_retrieve_review_likes(first_page_book_review_tag)*
- *_retrieve_review_date(first_page_book_review_tag)*
- *_build_review_rating_map(book_review_details, book_review_index, key, value)*
- *_retrieve_book_review_details_per_page(book_review_details, root_book_review_tags, book_review_index)*
- *retrieve_book_review_details(book_url, new_book)*

BookReviews.**_create_book_review_scraper_from_source**(*html_source*)

Creates the bs4 parser from the HTML source

Parameters **html_source** (*str*) – html source of the book

Returns bs4 parser

BookReviews.**_retrieve_review_rating**(*book_review_tag*)
Retrieves the rating given by the review Maps to in integer value from *web_scraper_goodreads_root.CommonConstants.Constants*

Parameters **book_review_tag** (*bs4*) – represents the bs4 instance for a particulat review

Returns review rating

BookReviews.**_retrieve_review_likes**(*first_page_book_review_tag*)
Retrieves the number of likes on the review

Parameters **first_page_book_review_tag** (*bs4*) – represents the bs4 instance for a particulat review

Returns review likes

BookReviews.**_retrieve_review_date**(*first_page_book_review_tag*)
Retrieves the review date

Parameters **first_page_book_review_tag** (*bs4*) – represents the bs4 instance for a particulat review

Returns date when the review was posted

BookReviews.**_build_review_rating_map**(*book_review_details*, *book_review_index*, *key*, *value*)
Build the dict *book_review_details*

Parameters

- **book_review_details** (*dict*) – the book review details
- **book_review_index** (*int*) – counter variable for the books
- **key** (*str*) – key used in dict *book_review_details*
- **value** (*str*) – value used in dict *book_review_details*

Returns details of the reviews for a book

BookReviews.**_retrieve_book_review_details_per_page**(*book_review_details*, *root_book_review_tags*, *book_review_index*)
This functions is the main driver function for all other functions in this file Retrieves details of the book which include: - rating of the book by the review - likes on the review - date of the review

Parameters

- **book_review_details** (*dict*) – the book review details
- **root_book_review_tags** (*bs4*) – bs4 instance for a particular review
- **book_review_index** (*int*) – counter variable for the books

Returns: - details of the reviews - which book number was processed

BookReviews.**retrieve_book_review_details**(*book_url*, *new_book*)
Main entry function into this file's code Also handles the progress bar Basically this function scrapes review data from the first page and then visits each of the review pages and scrapes review data from them

Parameters

- **book_url** (*str*) – URL of the book
- **new_book** (*bool*) – indicates whether its a new book or not

Returns review details of the book

1.7 book_review_visualization.py

- *_build_color_scatter_plot_array(book_review)*
- *visualize_and_save_review_information(book_review, book_name)*

book_review_visualization.**_build_color_scatter_plot_array**(*book_review*)
Creates the review color list based on the review rating

Parameters **book_review** (*dict*) – details of the book review

Returns list of colors based on the review

book_review_visualization.**visualize_and_save_review_information**(*book_review*, *book_name*)
Main function which visualizes the review likes. Each review is represented as a circle, the more likes a review has the larger the circle is The color of the circle is based on how positive the review is Also saves a high res PNG image of the review visualization Normalizes the review likes via min max normalization

Parameters

- **book_review** (*dict*) – details of the book review
- **book_name** (*str*) – the name of the book whose review details are to be visualized

1.8 review_rating_calculation.py

Note: Reference - https://www.analyticsvidhya.com/blog/2019/07/introduction-online-rating-systems-bayesian-adjusted-rating/

- *_extract_review_likes_ratings(book_review)*
- *_calculate_simple_avg_review_rating(review_ratings)*
- *_convert_to_bayesian_adj_rating(review_likes, review_ratings)*
- *_calculate_bayesian_adj_rating(bayesian_adj_ratings)*
- *_build_ratings_list(processed_book_review_info)*
- *quicksort(arr_to_be_sorted, start, end)*
- *_process_reviews()*

review_rating_calculation.**_extract_review_likes_ratings**(*book_review*)
Extracts the likes and ratings from the dict (pkl file)

Parameters **book_review** (*dict*) – details of a book

Returns likes and ratings

Return type 2 lists

review_rating_calculation.**_calculate_simple_avg_review_rating**(*review_ratings*)
Calculates the average rating

Parameters **review_ratings** (*list*) – ratings from the reviews

Returns average rating

Return type float

review_rating_calculation.**_convert_to_bayesian_adj_rating**(*review_likes*, *review_ratings*)

Calculates the Bayesian Adjusted ratings from the goodreads ratings and likes on the ratings

Parameters

- **review_likes** (*list*) – likes from the reviews
- **review_ratings** (*list*) – ratings from the reviews

Returns bayesian adjusted ratings

Return type list

review_rating_calculation.**_calculate_bayesian_adj_rating**(*bayesian_adj_ratings*)

Calculates the average bayesian adjusted rating

Parameters **bayesian_adj_ratings** (*list*) – bayesian adjusted ratings from the reviews

Returns average bayesian adjusted rating

Return type float

review_rating_calculation.**_build_ratings_list**(*processed_book_review_info*)

Build a list rating details from a python dict

Parameters **processed_book_review_info** (*dict*) – processed review details

Returns processed review details but as a list

Return type list

review_rating_calculation.**quicksort**(*arr_to_be_sorted*, *start*, *end*)

Quicksort implementation to sort a list in ascending order Complexity - n * logn

Parameters

- **arr_to_be_sorted** (*list*) – the input list to be sorted
- **start** (*int*) – the start index
- **end** (*int*) – the end index

review_rating_calculation.**_process_reviews**()

Main code to start processing the review details Ensure sci-fi-books-list_YYYY-MM-DD.pkl file is present in current date otherwise run MainBookScraper to get it We are making sure to add 1 to review likes which are 0 so as to not ignore those reviews completely

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

b

book_review_visualization, 7
BookReviews, 5

f

FilePicking, 1
FileUtil.FilePicking, 1

g

GenreScraper, 2

h

HelperUtils, 1

m

MainBookScraper, 4

r

review_rating_calculation, 7

s

SiteNavigator, 5

Index

Symbols

_build_book_details_map() (*in module GenreScraper*), 2
_build_color_scatter_plot_array() (*in module book_review_visualization*), 7
_build_ratings_list() (*in module review_rating_calculation*), 8
_build_review_rating_map() (*in module BookReviews*), 6
_calculate_bayesian_adj_rating() (*in module review_rating_calculation*), 8
_calculate_simple_avg_review_rating() (*in module review_rating_calculation*), 7
_convert_to_bayesian_adj_rating() (*in module review_rating_calculation*), 8
_create_book_review_scraper_from_source() (*in module BookReviews*), 5
_create_main_parser() (*in module GenreScraper*), 2
_extract_review_likes_ratings() (*in module review_rating_calculation*), 7
_init() (*in module SiteNavigator*), 5
_process_reviews() (*in module review_rating_calculation*), 8
_retrieve_author_name() (*in module GenreScraper*), 3
_retrieve_book_URL_and_image_URL() (*in module GenreScraper*), 3
_retrieve_book_name() (*in module GenreScraper*), 3
_retrieve_book_review_details_per_page() (*in module BookReviews*), 6
_retrieve_number_of_times_shelved() (*in module GenreScraper*), 3
_retrieve_rating_published_details() (*in module GenreScraper*), 3
_retrieve_review_date() (*in module BookReviews*), 6
_retrieve_review_likes() (*in module BookReviews*), 6
_retrieve_review_rating() (*in module BookReviews*), 6

B

book_review_visualization (*module*), 7
BookReviews (*module*), 5

C

check_if_file_exists_otherwise_handle() (*in module HelperUtils*), 1

D

data_for_book_exists_current_date() (*in module HelperUtils*), 1

F

FilePicking (*module*), 1
FileUtil.FilePicking (*module*), 1

G

generate_book_review_images() (*in module MainBookScraper*), 4
GenreScraper (*module*), 2
get_html_code_for_first_page() (*in module SiteNavigator*), 5
get_html_code_for_other_pages() (*in module SiteNavigator*), 5

H

HelperUtils (*module*), 1

L

load_latest_obj() (*in module FileUtil.FilePicking*), 1
load_obj() (*in module FileUtil.FilePicking*), 2

M

MainBookScraper (*module*), 4

Q

quicksort() (*in module review_rating_calculation*), 8

R

retrieve_book_review_details() (*in module BookReviews*), 6
retriveSciFiBookList() (*in module GenreScraper*), 3
review_rating_calculation (*module*), 7

S

save_obj() (*in module FileUtil.FilePicking*), 2
SiteNavigator (*module*), 5

V

visualize_and_save_review_information() (*in module book_review_visualization*), 7