University of Warsaw
Faculty of Economic Sciences

Marian Nehrebecki, student ID: 399214
Magdalena Sobala, student ID: 403715

Scraping Goodreads with *python*

Webscraping and Social Media Scraping

Faculty of Economic Sciences, University of Warsaw
Summer Semester 2020/21

Warsaw, May 2021

Contents

1 Introduction

With a few lines of code (with different levels of complicated of course) we will hopefully be able to scrape the desired data. Our target is the Goodreads website[1] (books' rating and recommendations) and basic data on books.

1.1 Goodreads

Goodreads is an imdb of the book world. It is a place for book lovers to rate, find and review books. The website provides information on books and user ratings. It also contains a classification on best books of the century, by user ratings. This is our target.

1.2 Desired output

We decided to scrape the information on *Best books by century*. The following information on each book was scraped (Table 1.):

- title of the book,
- author of the book,
- website - link to the book's website on Goodreads,
- average rating – average rating of the book from the users,
- score,
- number of votes.

i.e. the basic information available in the overview.

Table 1. Designed dataset

#	*book title*	*author*	*website*	*average rating*	*score*	*votes*

...

2 Scrapers

We scraped the desired data from the Goodreads website using Python language and applied three different methods:

- beautiful soup (2.1),
- scrapy (2.2),
- selenium (2.3).

Goodreads is a *static* website, therefore, there is no reason to omit the *beautiful soup* scraper.

2.1 Beautiful Soup

Building a Beautiful soup ("bs4") scraper was time-consuming and not that intuitive for some specific data (some variables required looping). The structure of the page from which we wanted to obtain our output, is not very straightforward, and required some more complicated input. However, the results were satisfying.

[1] https://www.goodreads.com/

This method allowed us to store our data in one *.csv* file. Before exporting the scraped data, we stored it in *pandas* data frame and applied some simple data cleaning methods so that the output was ready for use and analysis, i.e. removing unnecessary characters and words, converting numeric variables from string to float, etc. some parts of the data cleaning process could have been executed with regex expressions built into the scraper, however, it was much easier to perform these actions directly on the data frame. Additionally, we verified in which variables there were missing observations, we checked the number of missing observations. We then added values for the missing observations.

Short technical description of the scraper

1. Import packages (bs4, request, pandas, numpy, sleep(), randint()).
2. Preparation of the storage of the scraping data.
3. Analysis of the URL.
4. Scraping multiple pages.
5. Cleaning data.
6. Checking missing data.
7. Time measurement for scraping data.

2.2 Scrapy

Generally, working with scrapy on Goodreads was quite difficult. In the end, we ended up with two different approaches for scraping with Scrapy:

1) scraping directly from the *Best Book by Century* websites – this gave a complicated output which we ended up writing a whole separate *.py* file that imports the scrapy output, converts the data into a *pandas* dataFrame and gives a .csv output; it was for sure possible to do it with iterations directly in the spider, however, it was too difficult (at least for Magda, who was responsible for the scraper);
2) scraping the data from the actual websites of each book on the list – this approach was much easier from the perspective of the Scraper, with previously prepared code, it took about 15 to 20 minutes to write, however, there was one major obstacle which we could not overcome within the scraper itself – the data available on direct book website is different than the data available on the *Best Book by Century* list – the desired output could not have been achieved that way.

Therefore, at the end, we decided to go with the first approach for the project submission, as it gave "the same" output as our, fairly easily executed, bs4 scraper.
The scraping itself did not take much time, and the output was clear, and executed with no errors, however, as mentioned above, it required a lot of further handling with pandas (splitting the data, creating lists, dataFrames and cleaning; the provided file does not take care of it all – that just shows how much cleaning it requires).

Short technical description of the scraper

1. Import packages (scrapy, pandas).
2. Analysis of the URL
3. Scraping multiple pages.

Additional file for handling the scraped data:

4. Making data readable, creating a *pandas* dataframe.
5. Cleaning data.
6. Export to *.csv*.

2.3 Selenium

When building Selenium scrapy, it should be mentioned that we are using a real browser, which is an advantage. At the same time, there are also problems related to the lack of repetition of activities. The above problems were noticeable when extending data scraping on more websites.

This method, like Beautiful Soup, allowed us to store our data in one *.csv* file. Before exporting the scraped data, we stored it in *pandas* data frame and applied some simple data cleaning methods so that the output was ready for use and analysis, i.e. removing unnecessary characters and words, converting numeric variables from string to float, etc. some parts of the data cleaning process could have been executed with regex expressions built into the scraper, however, it was much easier to perform these actions directly on the data frame. Additionally, we verified in which variables there were missing observations, we checked the number of missing observations. We then added values for the missing observations.

Short technical description of the scraper

1. Import packages (selenium, webdriver_manager, pandas, numpy, sleep(), randint()).
2. Preparation of the storage of the scraping data.
3. Analysis of the URL
4. Scraping multiple pages
5. Cleaning data
6. Checking missing data
7. Time measurement for scraping data.

3 Output description

The above section consists of the presentation of the collected data, basic statistics, obtained histograms and the measuring time of the scrapped data.

3.1 Part of the scraping data

Table 2 lists the first 10 cases from the .csv file. We present the above data after cleaning process.

Table 2. The first 10 observations in DataFrame

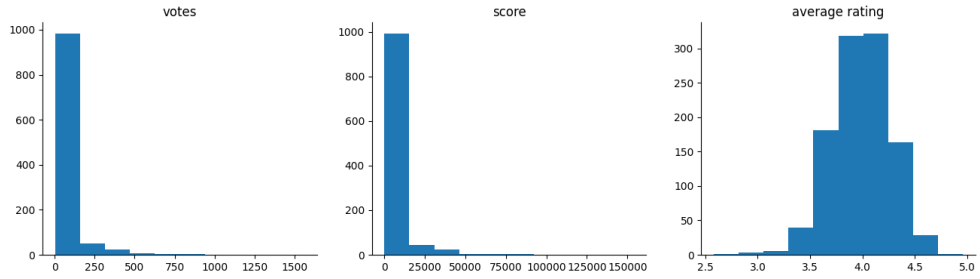
	book title	author	website	score	votes	average rating
0	Perceval, or, The Story of the Grail	Chr√©tien de Troyes	https://www.goodreads.com//book/show/397023.Perceval_or_The_Story_of_the_Grail	2968	30	3.65
1	The History of the Kings of Britain	Geoffrey of Monmouth	https://www.goodreads.com//book/show/129521.The_History_of_the_Kings_of_Britain	2860	29	3.73
2	Lancelot: The Knight of the Cart	Chr√©tien de Troyes	https://www.goodreads.com//book/show/129898.Lancelot	2466	25	3.62
3	The Lais of Marie de France	Marie de France	https://www.goodreads.com//book/show/119079.The_Lais_of_Marie_de_France	2453	25	3.87

4	Yvain, or The Knight with the Lion	Chr√©tien de Troyes	https://www.goodreads.com//book/show/143100.Yvain_or_The_Knight_with_the_Lion	2367	24	3.66
5	La Chanson de Roland	Unknown	https://www.goodreads.com//book/show/13946287-la-chanson-de-roland	2166	22	3.54
6	The Guide for the Perplexed	Maimonides	https://www.goodreads.com//book/show/761925.The_Guide_for_the_Perplexed	1577	16	4.13
7	Arthurian Romances	Chr√©tien de Troyes	https://www.goodreads.com//book/show/449589.Arthurian_Romances	1570	16	3.96
8	Hildegard von Bingens Physica: The Complete English Translation of Her Classic Work on Health and Healing	Hildegard von Bingen	https://www.goodreads.com//book/show/578007.Hildegard_von_Bingen_s_Physica	1466	15	4.19
9	The Alexiad	Anna Comnena	https://www.goodreads.com//book/show/485025.The_Alexiad	1371	14	4.03
10	The Conference of the Birds	Attar of Nishapur	https://www.goodreads.com//book/show/144870.The_Conference_of_the_Birds	1167	12	4.23

3.2 Elementary data analysis

When analyzing the data, you should analyze the distribution of the analyzed variables. For this purpose, the *matplotlib.pyplot* module was used.

Figure 1. Histograms of the variables votes, score and average rating



Based on the histogram for votes, it should be noted that the shape of the plot is not like a normal distribution, most of the values are in the interquartile's range from 2 to about 32. The distribution of score scores also does not resemble the normal distribution, the values interquartile's range from 175 to 3000 On the other hand, the average rating has a Q1 of around 3.72, while Q3 is at the level of 4.2.

Some further EDA could be done on the data, as well as some simple regressions and classification.

3.3 Comparison of the time for scraping data

Method	Average execution time of the method (in seconds)
Beautiful soup	*170*
Scrapy	*6*[2]
Selenium	228

4 Files description

As required, our files were uploaded to the joint GitHub repository in three separate folders:

[2] execution time was collected from dumped scrapy stats.

- *soup,*
- *scrapy,*
- *selenium*,

which, respectively, contain all files required to run each of the scrapers (***scrapy* folder contains the additional file for reading the data!**).

5 Division of work

- bs4 - we worked together on the bs4 scraper equally, improving the file back and forth, until we were satisfied with the obtained output,
- scrapy – Magda,
- selenium – Marian.