2

A technique of data collection: web scraping with python

Sumit Kumar[1], Uponika Barman Roy[2]

[1]JIO, BANGALORE, KARNATAKA, INDIA; [2]TATA CONSULTANCY SERVICES LIMITED, BANGALORE, KARNATAKA, INDIA

2.1 Introduction

The internet is a rich source of information. Every day we generate data in trillions of megabytes [1]. These highly populated heterogeneous data requires security as it carries indefinite personal information. Everything that we can see and listen on the internet is a data. It can be in any format such as text, audio, video, or an image. All the websites represent data to the users; thereby, they act as a data warehouse. While walking through the life cycle of data management, the first milestone which needs to be achieved is the data collection. Web scraping is a technique which mines the data available on the internet [2]. In the following section we will see the basic functionality of web scraping along with its significance and ethics to use. The third section will give a theoretical coverage of the elements involved in web scraping along with an architectural diagram. In the fourth section, a step-by-step guided walkthrough will help the readers to comprehend how to extract data from web. This section contains a worked-out example with each code snippet followed by the explanation. In addition, it will document on the stages of transformation of the data, and it will help the readers to visualize the difference in the processed data from the original form of the data. The fifth section will enlist few real-time web scrapings in distinguished domains. Essentially, that creates an overall understanding of the multiple wide benefits of web scraping. In the sixth section, the author will conclude the discussion summarizing the topics discussed followed by the reference area.

2.2 Basics of web scraping

2.2.1 Definition

To define, "A web scraping tool is a technology solution to extract data from web sites, in a quick, efficient and automated manner, offering data in a more structured and easier to use format, either for Business-to-Business (B2B) or for Business-to-Consumer (B2C) processes" [3]. This technique takes out huge amounts of unstructured data and stores it in an structured manner. The websites hold massive amounts of data for users. Let us see an example. Consider any big e-commerce website where the users buy and sell different number of items. A user wants to buy some fresh apples from any one of the platforms. To buy, the user will definitely visit the fruits section of the application and then proceed to the varieties of apples. These days companies are using machine learning techniques to bring up recommendations for users. The one who searches for apples also gets some choice to buy some grapes with extra discounts. This catches the user's attraction to buy grapes as an add-on with apples. But what exactly runs behind the story is a simple machine learning recommendation system. Now, the question is how the system knows to recommend grapes with apples but not a soap bar? The answer is the data that is collected from the user's choice undergoes analysis before recommendation. In simple terms, it is the data.

To develop a good machine learning model, it is a primary focus to provide the model with a correct form of authenticated data. It is very important what data is being collected and how it is being formatted for the utilization. The websites from where the data gets mined is in a raw unstructured format. Web scraping not only consumes those raw data but also transforms them into a usable structured layout to feed to the machine learning models [4]. It is obvious that to collect data from a website, the site's application programming interface (API) is to be accessed. Therefore, the authentication from the website plays a major role in the process. It is not mandated that all companies permit their data to be used for building any algorithm on it.

2.2.2 Why do we need to scrape data?

There are indefinite reasons why we data needs to be scraped. The information on the internet is of various forms like text, image, media, etc. Every individual organization has their own set of requirements of data for their development or operation. To understand the requirement of scraped data better, some examples can be enlisted, such as the following:

(1) **Research and Development**: Statistical data, survey information or sensor data are the biggest data sources in the field of research. To process any analysis or develop more into a domain, exploration and experiments are the backbone. So, the sector of research and development is highly reliable on data to carry out the experiments.

(2) **Business Marketing**: The companies stack up data (customers data like phone number, email ID, recently purchased products, choice of products, reviews, etc.) from the websites to enhance on better customer facility.

(3) **Social Media Technology**: The various giant media platforms like Twitter, YouTube, and Facebook bring the user an experience of trending streaming media. They make use of very high-end search engines where the history data is the fundamental ingredient.

(4) **Recommendation Websites**: The admission portals or job listing websites provides a good example of application of web scraped data. These portals use the gathered data and recommend on the user's preference.

The above listed scopes are some of the high-end usages of web scraping but there are lot more other utilities at smaller scale where the data is gathered to fulfill organization level or personal level requirements [5].

2.2.3 Choice of programming language

Python language is deliberately chosen to discuss and demonstrate an example of web scraping. There are other helpful languages like C++, Ruby, Node.JS, R, Selenium, etc., which are used for the task [6]. All these mentioned languages have good support from their respective libraries to achieve web scraping. But to deal with internet data, the size of the data is noteworthy, and it is always big. To reach these website data, the API handling of the programming language should be flexible. So, any programming language chosen should support HTTP responses thoroughly. Even though Node.JS has an extensive HTTP library for API handling, it is limited in holding up to large amounts of data. Again, C++ sets off highly expensive to use for web related operations. Python is an easier selection due to its simplicity and vast functionalities. For any data-related programming, one of the significant aspects to keep in mind is the visualization. The data to play with is rigid and raw to analyze until they are text data. Visualization gives a better understanding of the data. There are distinct kinds of visualization libraries in Python like Matplotlib, seaborne, Pygal, etc.

The requirements have to be properly articulated to choose the best fit language for web scraping. To shortlist them as:

- Less complicated as a big size of data needs to be manipulated.
- Flexibility to float around various APIs to collect data.
- Faster and more efficient to utilize minimum CPU usage.
- Supports visualizations to data.
- Capable of generating end data in a structural format.
- Maintainability.
- Scalability.

To bring forth the above features in a coding language, Python is considered the most suitable one at the beginner's level.

2.2.4 Ethics behind web scraping

So far, what and why is discussed on web scraping. Now it's **time to start for How**!!

But let's pause a bit and rewind. There are lot of occasions where the reader is finding the buzz words like fetch the data from the internet, use a website's data, and so on. The question here is how ethical it is to pull out data from anywhere and why the websites would support it. Let us pick up the first part of the question.

The community of data scientists, data engineers, business analysts, data journalists, and everyone who relies on data, scrap them from internet. So, it is obviously not illegal, but everyone maintains basic ethics while scraping. Data scraping from the internet has some guidelines or legalities to follow, which is known as ethical web scraping [4].

Often an internet user ticks the privacy policy statement like "I accept the privacy and follow it" while web browsing. There is where he is abiding to the data policy of the website owner. Similarly, as an ethical scraper it is highly important to align to the honesty while taking out data [7,8].

- The information taken out should only be present with the scraper, not to be shared widely. The motive of web scraping should be genuine and only to bring out analysis from the data but not to prepare a duplicate copy of the collected data.
- The public APIs are the most reliable sites to scrap data from.
- The scraper should provide purpose to clear the intention of collecting data and own details to be communicated for any concern. There must be a collaborative attitude toward the website owner.
- The analysis brought up from the owner's data must end up benefiting him as well. It can help in raising the trafficking of his website. And this answers the latter half of the question.
- Likely, the website owner also to comply with certain ethics like encouraging the scrapers to utilize the data for mutual benefit.

To keep a healthy collaboration between the scraper and site owner helps to bring upon more innovation and profit to both ends.

2.3 Elements of web scraping

2.3.1 Architecture of web scraping

Fig. 2.1 gives a pictorial representation of scraping data from the web. In the next part of this section, the reader will get a detail understanding of each block of the architecture.

2.3.2 Components of web scraping

To start collecting data from websites, there are few prerequisites need to be followed. As discussed in last Section 2.2.4, data cannot be gathered from any random websites. It is always recommended to mine from public websites. From Fig. 2.1, it is observed that web scraping includes important techniques like website selection, inspection of data, etc. Let's see how these components contribute to the ultimate goal:

- **Website Selection**: The first component must be to understand the reason of data collection and then to know the nature of data required. Considering the use case

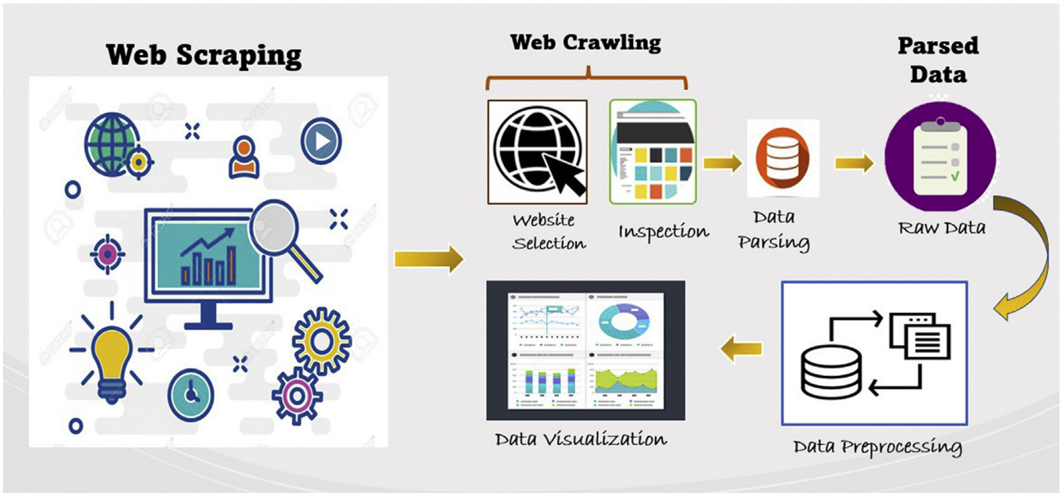


FIGURE 2.1 Architecture of web scraping.

to be price comparison of certain items among different online shopping websites, the motive should be to look for those commodities in different platforms and then gather their individual data.

- **Inspection of web data**: After selecting the website, the task is to inspect the data on the web page. While hovering around the data (required to be gathered), Ctrl + Shift + I option lets a Hypertext Markup Language (HTML) window pop at the right side of the page. This window demonstrates the HTML construction only for the inspected data.
- **Parsing**: Followed by the inspection of data, the most important part is to parse the data using the Python library. Python has a powerful library called Beautiful Soup, which parses the HTML data. By definition parsing refers to converting of machine language data to human readable data. In this case, the parsing will convert the HTML data into simple text data.
- **Data Preprocessing**: The parsing gives the data in a crude unstructured format. Python has extensive library support to format data in the desired form. This method of framing data is data preprocessing. This is a significant step as it generates a structure of the data which in the later gets used for analysis. More accurate the data is preprocessed better result can be obtained from analysis. Hence, this is considered the heart of any machine learning algorithms.
- **Data Visualization**: This is completely in the need of the user. It is a later stage operation which is done post–web scraping to understand the collected data better. Often while analysis it becomes easier to visualize the data in the form of graphs or charts. As data comprehension is an important task to make any system intelligent, so it is always suggested to be familiar with the data. In Python there

are popular visualization libraries which results the structured data into different types of statistical charts and graphs.

From Fig. 2.1, an important take away is that web scraping includes web crawling and parsing. From the above bullets, the first two points, i.e., website selection and inspection of web data is together referred as web crawling. There are automated systems which perform this web crawling task. They automatically search the website and inspect the on-demand data out from it. Although this software techniques are much faster than the manual way of scraping data, but this section demonstrates the manual method for granular understanding purpose.

Data Parsing plays the actual role in pulling out the web content into readable structure. Every web page has its own HTML framework. While inspection of data, the HTML content is accessed. The target is to crawl through the HTML tags and identify the tag responsible for the data to get scraped. Once the desired HTML tag is found, parsing comes into its way to pull it from there to the local system. In Python, the library Beautiful Soup has an interesting function to crawl on the web data for the HTML tag identification and also for parsing it. The reader will get a hands-on experience of these concepts in the implementation walk through.

2.4 An implementation walkthrough

In this section we will go through a demonstration of a real-time scraping of web data. There are different libraries which are popularly used by the scrapers to mine data.

2.4.1 Libraries for web scraping

2.4.1.1 Scrapy

This framework is primarily used for web crawling. Scrapy builds very complex systems which is capable to scrape a high dimension of data. This tool consumes notably very less memory and CPU usage. The benefit of this tool is it is an open-source framework, and it has the powerful functions to carry out the HTTP responses and their parsing. Scrapy is a Python-supported tool.

An operating system or a program when performs several tasks at one single instant, it is referred as multithread or an asynchronous system. Scrapy framework is built on an asynchronous foundation, known as Twisted, hence the pace with which it operates the program to extract data, process them and to format them is exceedingly fast. That is the main reason of selecting Scrapy for mining large scale data. It generates the structured data in different forms like CSV, JSON, XML, etc. This tool basically collects data from the websites which are designed using Java scripts [9].

Scrapy has its own command-line interface known as “Scrapy Tool,” it has different functions like fetch, crawl, check, parse, bench, etc. These are few command-line functions and alike there are other functions which help Scrapy to collect the data from the web and return into a structured format. After reading the high-end features offered by Scrapy, the readers may raise the query to justify the need of other scraping tools. To answer that, every system has their respective flaws. Scrapy is not a beginner's tool. It is used at high professional end to scrape high dimensional data from the very complex websites [10]. Due to this, the tool has configuration complexity at different operating systems.

2.4.1.2 Selenium

A web-based automation tool mainly used for developing testing scripts in the industry. Web Scraping is an add-on benefit of this tool. The basic steps to use Selenium is to install the package and import them followed by loading the browser (Google Chrome) driver. Once the configurations are satisfied, the website has to be accessed [11]. Selenium has different locators to navigate through the elements of the website. It uses elements as Tag name, Class name, IDs, XPath, CSS selectors, etc.

The above code snippet in Fig. 2.2, is sourced from GitHub [12]. In this it is seen that after loading the chrome driver, it's **get ()** function finds the website to extract data from. The **find_elements_by_class_name ()** detects the specific class (division, paragraph, table, etc.) of prime focus.

The popularity of Selenium as a web scraper is due to its ability to scrape data from the dynamic websites. There are two kinds of websites: static and dynamic. The static is the one where the data present on the website is constant whereas in the dynamic one the data keeps on changing like the geo location or navigation maps. Even Scrapy has a good potential to mine from dynamic web pages.

To compare performance between Scrapy and Selenium, it is obvious that both has individual features, but Selenium stays back of Scrapy to handle the large chunks of data. Even the multi-threading of Scrapy makes it faster. A website which has major JavaScript's depiction is better to be handled by Selenium as Scrapy suffers dependency issues to mine from those websites. Unlike Scrapy, Selenium is not a framework. It is a Python library, so it is super compatible to configure and use.

2.4.1.3 Beautiful Soup

A Python library to parse the unstructured web data to a formatted structured data. This package retrieves data out from the HTML or XML pages. Alone Beautiful Soup is not enough to access any website. It needs support exclusively from the request library of

```python
from selenium import webdriver

path = '/Users/.../chromedriver' #path of your driver file
driver = webdriver.Chrome(path)
driver.get("http://www.example.com")

driver.find_element_by_xpath('//*[@id="accept-button"]').click() #click on a button
elements = driver.find_elements_by_class_name('class-example')
data = [element.text for element in elements]

driver.quit()
```

FIGURE 2.2 A snippet code of scraping data.

Python. The request library allows easy HTTP connections with its mighty functions. Beautiful Soup employs this library to reach the website.

The snippet in Fig. 2.3, represents a web scraping with Beautiful Soup [13]. The very next step after loading the required libraries (requests and Beautiful Soup from bs4 package) is to navigate to the website.

An object of the library is created defining the HTML parsing for the specified website. There are various in-built methods to perform the operations on the data. In this section, an implementation with Beautiful Soup is worked out for the readers. Hence, a detailed step-by-step guide will help the readers to scrape data from any HTML website from scratch. Beautiful Soup has a challenge in mining large data from any complex website and it cannot stand with any JavaScript supported website. This is a best fit method to collect data from small scale static web pages.

The readers after comparing the multiple tools to scrape web data can note few inferences. The choice of the scraping tool has different aspects like size of the data, design platform of website (HTML or JavaScript), complexity of the site, dependency to other libraries, etc. The best thing of all these three tools is that all are open sources, so the cost factor nullifies.

2.4.2 Importing required libraries

In this model walk through, all the codes are built on Jupyter environment with Python language [14].

To begin the task, the necessary libraries are to be imported in the Notebook.

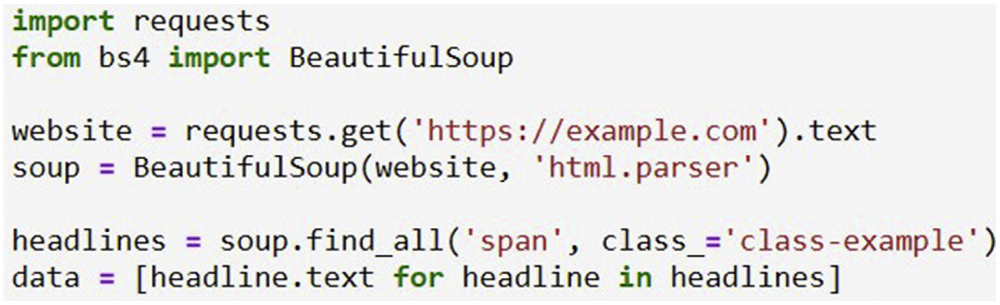
import requests
from bs4 import BeautifulSoup

As discussed in 2.4.1.3, in this implementation the library Beautiful Soup will be used. And to use that, the import of requests library is also a prerequisite.

In this implementation, a real estate website is used to scrape the data from.

2.4.3 Accessing website data

On the website, there is a list of properties representing their individual price, features (age, architecture types, etc.) and property descriptions. The task is to capture all these



```
import requests
from bs4 import BeautifulSoup

website = requests.get('https://example.com').text
soup = BeautifulSoup(website, 'html.parser')

headlines = soup.find_all('span', class_='class-example')
data = [headline.text for headline in headlines]
```

FIGURE 2.3 A code snippet with Beautiful Soup.

information in a single json file or csv file format, which further becomes a potential dataset for any prediction of the properties. There can be many facets of possible cognitive analysis and prediction once the data is available. So, the reader will see how to collect the data from the website.

The request library has a function called get () which redirects to the website. In this code sample, a header is included as a parameter to this get (). There are certain occasions to extract data from the website which needs headers to give access to their data. Once the request object is created, request.text helps in getting the text data present in the entire website.

```
url="http://www.pythonhow.com/real-estate/rock-springs-wy/
LCWYROCKSPRINGS/"
req=requests.get (url, headers= {'User-Agent': 'XYZ/3.0'})
content=req.text
print(req)
```

From the above code, the req returns the response type after communicating with the website. The response code 200 represents a successful response type from the site. Hence, the expectation is that content will return the entire HTML tagging structure of the website.

2.4.3.1 Output

```
<Response [200]>
'<!DOCTYPE html>\n<! - saved from url= (0110)http://web.archive.org/web/20160127020422/
http://www.century21.com/real-estate/rock-springs-wy/LCWYROCKSPRINGS->\n<html lang="en"
style="margin: 0px;overflow:hidden"><script async="" src="./LCWYROCKSPRINGS1_files/beacon.js">
</script>................. ............ .......<script src="http://web.archive.org/web/20151215220806/
http://chat.xtdirect.com/Chat/MasterServer/Public/Identify.php?packets=5:indentify:location,
http*X0M0M0!web%L0archive%L0org%M0web%M020160127020422%M0http*X0M0M0!www%L0century21%
L0com%M0real%K0estate%M0rock%K0springs%K0wy%M0LCWYROCKSPRINGS."type="text/javascript"
id="PCJSF_CommScriptId"></script></div></body><div></div></html>\n'
```

Here the site's HTML content is captured.

But it is complicated to analyze something so raw as well as tedious to look through the entire tags to find out the wanted section of the site data.

2.4.4 Web crawling

Web crawling helps in finding the HTML tag in charge of holding the content of the page. In BeautifulSoup library, the function find_all () traverses to that specific tag. Before using the library, an object needs to be created for the soup. And while creating the object, the function takes care of parsing the entire content of the page.

```
#Creating object for BeautifulSoup
soup=BeautifulSoup(content,'html.parser')
```

```
#Finding the HTML div type tag with class id "propertyRow" to access the list of properties #data
all=soup.find_all("div",{"class":"propertyRow"})
```

This brings the complete content of the desired section of the website where the filter is only the div class “propertyRow”. So, it will extract only the data included in the property row division.

2.4.5 Data extraction

Data extraction is all about purifying the primitive data to some meaningful information. In the discussed use case, this can be done by creating dictionaries of the respective features of each property. The data segregation happens here to send data into different categorical sections like property address, property price, locality, number of beds, area sqft., etc.

```
#Finding address
data["Address"]=item.find_all("span",{"class","propAddressCollapse"})[0].text
#Finding price
data["Price"]=item.find("h4",{"class","propPrice"}).text.replace("\n","").replace(" ","")
#Finding the count of bedrooms
data["Beds"]=item.find("span",{"class","infoBed"}).find("b").text
#Finding the sqft of the area
data["Area"]=item.find("span",{"class","infoSqFt"}).find("b").text
#Finding the number of full bathrooms
data["Full Baths"]=item.find("span",{"class","infoValueFullBath"}).find("b").text
#Finding the number of half bathrooms
data["Half Baths"]=item.find("span",{"class","infoValueHalfBath"}).find("b").text
#Printing the data
print(data)
```

2.4.5.1 Output

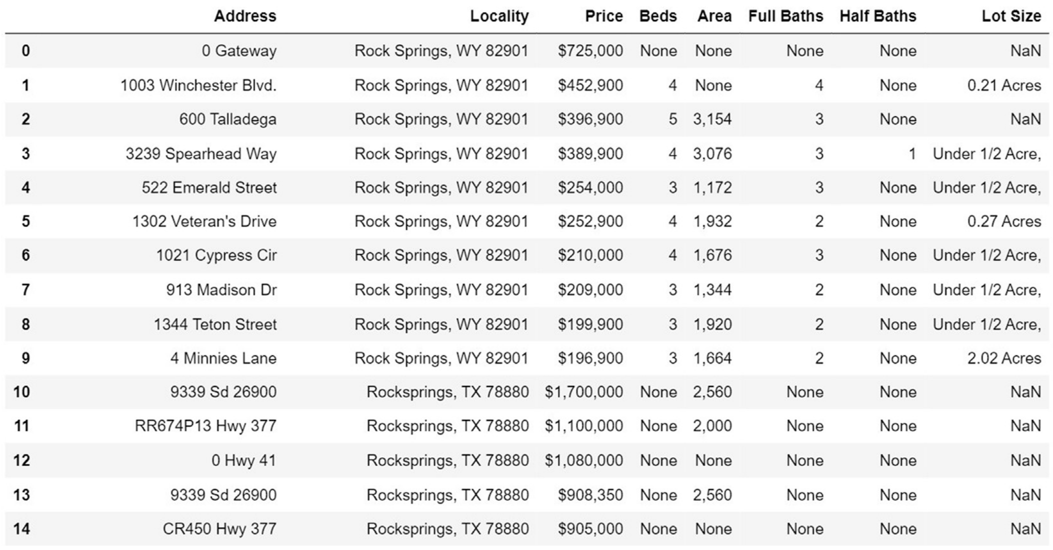
```
[{'Address': '0 Gateway',
    'Locality': 'Rock Springs, WY 82901',
    'Price': '$725,000',
    'Beds': None,
    'Area': None,
    'Full Baths': None,
    'Half Baths': None},
  {'Address': '1003 Winchester Blvd.',
    'Locality': 'Rock Springs, WY 82901',
    'Price': '$452,900',
    'Beds': '4',
  'Area': None,
  'Full Baths': '4',
  'Half Baths': None,
'Lot Size': '0.21 Acres'},
.
.
.
{'Address': '19350 E SAGUARO Drive',
  'Locality': 'Black Canyon City, AZ 85324',
  'Price': '$28,995',
  'Beds': None,
  'Area': None,
  'Full Baths': None,
  'Half Baths': None,
  'Lot Size': '0.73 Acres'},
{'Address': '20650 E Amethyst Place',
  'Locality': 'Black Canyon City, AZ 85324',
  'Price': '$15,000',
  'Beds': None,
  'Area': None,
  'Full Baths': None,
  'Half Baths': None,
'Lot Size': '0.31 Acres'}]
```

The above code appended the entire information in a dictionary which is later exported as a json file. The output shows a list of entries of the property data. Each entry represents each property with its own details like *Address, Locality, Price*, etc. The json data obtained is in a ready structured format. But there are other formats of data to export as well (Fig. 2.4).

2.4.6 Data framing

In data framing, one prepares a well formatted dataset. In Python, there is a library called Pandas, which has a functionality called Data Frame [15]. In the below snippet of code, it is shown how a pandas Data Frame gives a perfect representation of the json data in a tabular format. Again, to use this library it needs to be imported in the notebook.

```
#Importing pandas library
import pandas.
#Converting the json data into a DataFrame
df=pandas.DataFrame(data)
#Printing the dataframe
print(df)
```



	Address	Locality	Price	Beds	Area	Full Baths	Half Baths	Lot Size
0	0 Gateway	Rock Springs, WY 82901	$725,000	None	None	None	None	NaN
1	1003 Winchester Blvd.	Rock Springs, WY 82901	$452,900	4	None	4	None	0.21 Acres
2	600 Talladega	Rock Springs, WY 82901	$396,900	5	3,154	3	None	NaN
3	3239 Spearhead Way	Rock Springs, WY 82901	$389,900	4	3,076	3	1	Under 1/2 Acre,
4	522 Emerald Street	Rock Springs, WY 82901	$254,000	3	1,172	3	None	Under 1/2 Acre,
5	1302 Veteran's Drive	Rock Springs, WY 82901	$252,900	4	1,932	2	None	0.27 Acres
6	1021 Cypress Cir	Rock Springs, WY 82901	$210,000	4	1,676	3	None	Under 1/2 Acre,
7	913 Madison Dr	Rock Springs, WY 82901	$209,000	3	1,344	2	None	Under 1/2 Acre,
8	1344 Teton Street	Rock Springs, WY 82901	$199,900	3	1,920	2	None	Under 1/2 Acre,
9	4 Minnies Lane	Rock Springs, WY 82901	$196,900	3	1,664	2	None	2.02 Acres
10	9339 Sd 26900	Rocksprings, TX 78880	$1,700,000	None	2,560	None	None	NaN
11	RR674P13 Hwy 377	Rocksprings, TX 78880	$1,100,000	None	2,000	None	None	NaN
12	0 Hwy 41	Rocksprings, TX 78880	$1,080,000	None	None	None	None	NaN
13	9339 Sd 26900	Rocksprings, TX 78880	$908,350	None	2,560	None	None	NaN
14	CR450 Hwy 377	Rocksprings, TX 78880	$905,000	None	None	None	None	NaN

FIGURE 2.4 Output- scraped data in a data frame format.

2.4.6.1 Output

This above data is the first 15 entries of property on the website with their respective details. Such formatting makes a comprehensive understanding of the data for further analysis. The next step is to perform an exploration of the dataset.

The pandas library also supports to export the data into different formats like csv, xml, or json. Here in the use case, the data is exported into a csv file name "PropertyDetails."

```
#Exporting Dataframe to CSV
df.to_csv('PropertyDetails.csv', index=False)
```

2.4.7 Stages of data transformation

During the architectural discussion of web scraping in Fig. 2.1, it was noted that the data captured from the websites go through different stages of transformation. Initially it is obtained in a raw structure which is hard to analyze. So, the second stage is to imbibe the required portion of the data from the entire site's information. In the third stage, the data gets its final shape and an appropriate structure.

Before concluding with the task, it is noteworthy to understand that while scraping data the end goal is to present it in a clean structured format. This enhances the readability of the data for future predictions on it.

2.5 Web scraping in reality

In this recent competitive era of technologies, the market has the highest demand of bringing the cutting-edge automations to reach to the common people. This makes life much convenient. To add value in this process, data plays one of the key roles.

In this section a very shallow walk will be done to show the readers the various fields where scraped data is getting utilized [16]. But to mention, these applications are just for some broader picture understanding. The real utilization of data cannot be confined into one section. The applications can be enlisted as:

Price Comparisons: This is one of the biggest target zones for the utilization of web scraping. The data collected from the e-commerce sites can be compared among the competitive platforms and can deliver into some capable analysis.

Customer Sentiment Analysis: A massive usage of scraping takes place in extracting customer feedback data and analyzing their sentiments for betterment of customer service.

Newspaper data: The world is busy to spare time in reading through the pages of newspaper. To automate the process of collecting useful data from the newspaper headlines and articles improves the finance and investment workflows.

Data Science: In real-time analytics like fraudulent credit card data, customer relationship management or financial decisions are the areas to inculcate the practice of automation using data extracted from the websites.

Some utilization is always worthy to note in predictive analysis like forecasting, risk managements and business-related decision makings.

Strategic Marketing: Data is the primary ingredient to reach target customers in business. There are marketing types which has a 100% dependency on business-related statistics. The data-driven approach of marketing and the content marketing are one of these highly flourishing strategies governing the business sectors.

2.6 Conclusion

To summarize on the complete idea of web scraping, the few key takeaways are hard to miss. At first, there comes the objective of scraping data from the website. The nature of the data, the prime motive to mine and the authorization to access it are included in the objective of the web scraping. The second functionality is to discover the pieces to combine to achieve the job. The discussed architectural diagram exposes how to crawl into the websites. HTML design fetches the appropriate data out from it. A parsing concept aids the process of understanding the web data before any further operations on it. Also, there are couple of frameworks that equips significant functionalities to extract, filter and represent data. Beautiful Soup being one of the simplest libraries to scrape data at initial level from static websites. Its function in Beautiful Soup like find_all () is powerful enough to refine the data from its entire bulk. Finally, the raw form of data is barely credible to perform any analysis. Consequently, the job is to structure the data. In Python, a library called Pandas is capable of decorating the data in an orderly format.

References

[1] J. Clement, Global Digital Population, 2020 [online] Available : https://www.statista.com/statistics/617136/digital-population-worldwide/.

[2] A. Namoun, A. Alshanqiti, E. Chamudi, M.A. Rahmon, Web design scraping: enabling factors, opportunities and research directions, in: 2020 12th International Conference on Information Technology and Electrical Engineering (ICITEE), 2020, pp. 104–109, https://doi.org/10.1109/ICITEE49829.2020.9271770.

[3] O. Castrillo-Fernández, W. Scraping, Applications and Tools", European Public Sector Information Platform Topic Report No. 2015/10, December, 2015.

[4] R. Diouf, E.N. Sarr, O. Sall, B. Birregah, M. Bousso, S.N. Mbaye, Web Scraping: State-of-the-Art and Areas of Application, 2019 IEEE International Conference on Big Data (Big Data), 2019, pp. 6040–6042, https://doi.org/10.1109/BigData47090.2019.9005594.

[5] V. Singrodia, A. Mitra, S. Paul, A review on web scrapping and its applications, in: 2019 International Conference on Computer Communication and Informatics (ICCCI), 2019, pp. 1–6, https://doi.org/10.1109/ICCCI.2019.8821809.

[6] The 5 Best Programming Languages for Web Scraping. [online] Available: https://prowebscraper.com/blog/best-programming-language-for-web-scraping/.

[7] K. Turk, S. Pastrana, B. Collier, A tight scrape: methodological approaches to cybercrime research data collection in adversarial environments, in: 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), 2020, pp. 428–437, https://doi.org/10.1109/EuroSPW51379.2020.00064.

[8] James D. Ethics in Web Scraping. [online] Available: https://towardsdatascience.com/ethics-in-web-scraping-b96b18136f01.

[9] H. Wu, F. Liu, L. Zhao, Y. Shao, Data analysis and crawler application implementation based on Python, in: 2020 International Conference on Computer Network, Electronic and Automation (ICCNEA), 2020, pp. 389–393, https://doi.org/10.1109/ICCNEA50255.2020.00086.

[10] H. Yang, Design and implementation of data acquisition system based on scrapy technology, in: 2019 2nd International Conference on Safety Produce Informatization (IICSPI), 2019, pp. 417–420. https://doi.org/10.1109/IICSPI48186.2019.9096044.

[11] K. Varshney. Web Scraping Using Selenium. [online] Available: https://medium.com/@kapilvarshney/web-scraping-using-selenium-836de8677ae5.

[12] F. Andrade. Web Scraping with Beautiful Soup, Selenium or Scrapy?. [online] Available: https://gist.github.com/ifrankandrade/189e0f903773dff419f6cd39e130c906#file-web-scraping-libraries-py.

[13] T.S. Deshpande, S. Varghese, P.D. Kale, M.P. Atre, Incident classification, prediction of location and casualties, in: 2021 8th International Conference on Computing for Sustainable Global Development (INDIACom), 2021, pp. 778–781. https://doi.org/10.1109/INDIACom51348.2021.00139.

[14] A.P. Lorandi Medina, G.M. Ortigoza Capetillo, G.H. Saba, M.A.H. Pérez, P.J. García Ramírez, A simple way to bring Python to the classrooms, in: 2020 IEEE International Conference on Engineering Veracruz (ICEV), 2020, pp. 1–6. https://doi.org/10.1109/ICEV50249.2020.9289692.

[15] K. Willems. Pandas Tutorial: DataFrames in Python. [online] Available: https://www.datacamp.com/community/tutorials/pandas-tutorial-dataframe-python.

[16] H. Patel. How Web Scraping is Transforming the World with its Applications. [online] Available: https://towardsdatascience.com/https-medium-com-hiren787-patel-web-scraping-applications-a6f370d316f4.