

Topic:

Web Scraping and Web Crawling

Date : 7th Feb 2019

Table of Contents

Aim	3
Part 1 : Compare Data scraping and Data crawling	3
What Is Web Scraping?	3
Why Scrap the Web?	3
Challenges of Web Scraping	4
Hidden Websites	4
Dynamic Websites	4
What is a Web Crawler ?	5
Data Scraping vs. Data Crawling :	6
Part 2 : Implementation	8
Steps	9
Challenges Encountered	10
Code	11
Data Collected	13
Generalization to other Departments	14
Tools Used	16
Github link of Project	17
Conclusion	17

Aim

Compare Data scraping and Data crawling. Implement web scraping without using any built-in libraries /function. Give all steps of working and execution procedure

Part 1 : Compare Data scraping and Data crawling

What Is Web Scraping?

Web scraping is the process of gathering information from the Internet. Even copy-pasting the lyrics of your favorite song is a form of web scraping! However, the words “web scraping” usually refer to a process that involves automation.

A **web scraper** is a systematic, well-defined process of extracting specific data about a topic. For instance, if you need to extract the prices of products from an e-commerce website, you can design a custom scraper to pull this information from the correct source.

Why Scrap the Web?

When you try to get the information you want manually, you might spend a lot of time clicking, scrolling, and searching. This is especially true if you need large amounts of data from websites that are regularly updated with new content.

Manual web scraping can take a lot of time and repetition.

Challenges of Web Scraping

1. The Web has grown organically out of many sources. It combines a ton of different technologies, styles, and personalities, and it continues to grow to this day.
2. One challenge is variety. Every website is different. While you'll encounter general structures that tend to repeat themselves, each website is unique and will need its own personal treatment.
3. Another challenge is durability. Websites constantly change.. The first time you run your script, it works flawlessly. But when you run the same script only a short while later, you run into a discouraging and lengthy stack of tracebacks!

However, there are a few more challenging situations you might encounter when you're scraping websites.

Hidden Websites

Some pages contain information that's hidden behind a login. That means you'll need an account to be able to see (and scrape) anything from the page. The process to make an HTTP request from your Python script is different than how you access a page from your browser. That means that just because you can log in to the page through your browser, that doesn't mean you'll be able to scrape it with your Python script.

Dynamic Websites

Static sites are easier to work with because the server sends you an HTML page that already contains all the information as a response. You can parse an HTML response with BeautifulSoup and begin to pick out the relevant data.

On the other hand, with a dynamic website the server might not send back any HTML at all. Instead, you'll receive JavaScript code as a response. This will look completely different from what you saw when you inspected the page with your browser's developer tools.

What is a Web Crawler ?

A **web crawler**, also known as a ‘**spider**’ has a more generic approach! You can define a web crawler as a bot that systematically scans the Internet for indexing and pulling content/information. It follows internal links on web pages. In general, a “crawler” navigates web pages on its own, at times even without a clearly defined end goal. Hence, it is more like an exploratory search of the content on the Web. Search engines such as Google, Bing, and others often employ web crawlers to extract content for a URL or for other links, get URLs of these links and other purposes.

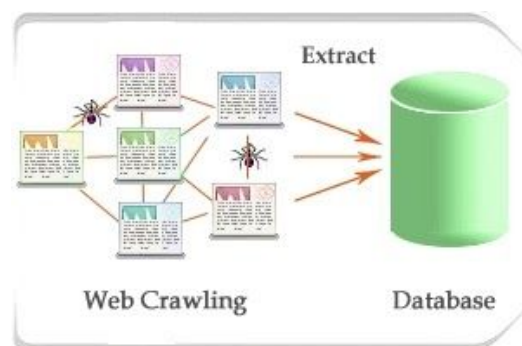
However, it is important to note that web scraping and crawling are not mutually exclusive activities. While web crawling creates a copy of the content, web scraping extracts specific data for analysis, or to create something new.

However, in order to scrape data from the web, you would first have to conduct some sort of web crawling to index and find the information you need. On the other hand, data crawling also involves a certain degree of scraping, like saving all the keywords, the images and the URLs of the web page.

Data Scraping vs. Data Crawling :

Crawling usually refers to dealing with large data-sets where you develop your own crawlers (or bots) which crawl to the deepest of the web pages. Data scraping on the other hand refers to retrieving information from any source (not necessarily the web). It's more often the case that irrespective of the approaches involved, we refer to extracting data from the web as scraping (or harvesting) and that's a serious misconception.

1. Scraping data does not necessarily involve the web. Data scraping could refer to extracting information from a local machine, a database, or even if it is from the internet, a mere "Save as" link on the page is also a subset of the data scraping universe. Crawling on the other hand differs immensely in scale as well as in range. Firstly, crawling = web crawling which means on the web, we can only "crawl" data. Programs that perform this incredible job are called crawl agents or bots or spiders (please leave the other spider in spider man's world). Some web spiders are algorithmically designed to reach the maximum depth of a page and crawl them iteratively (did we ever say scrape?).
2. Web is an open world and the quintessential practising platform of our right to freedom. Thus a lot of content gets created and then duplicated. For instance, the same blog might be posted on different pages and our spiders don't understand that. Hence, data deduplication (affectionately dedup) is an integral part of data crawling. This is done to achieve two things- keep our clients happy by not



flooding their machines with the same data more than once, and saving our own servers some space. However, dedup is not necessarily a part of data scraping.

3. One of the most challenging things in the web crawling space is to deal with coordination of successive crawls. Our spiders have to be polite with the servers that they hit so that they don't piss them off and this creates an interesting situation to handle. Over a period of time, our intelligent spiders have to get more intelligent (and not crazy!) and learn to know when and how much to hit a server in order to crawl data on its web pages while complying with its politeness policies.
4. Finally, different crawl agents are used to crawl different websites and hence you need to ensure they don't conflict with each other in the process. This situation never arises when you intend to just scrape data.

Summary:

Data scraping	Data Crawling
Involves extracting data from various sources including web	Refers to downloading pages from the web
Can be done at any scale	Mostly done at a large scale
Deduplication is not necessarily a part	Deduplication is an essential part
Needs crawl agent and parser	Needs only crawl agent

Therefore In short,

Scraping represents a very superficial node of crawling. which we call extraction and that again requires few algorithms and some automation in place.

Part 2 : Implementation

Target : <http://www.vjti.ac.in/index.php/academics/civil>

Aim :

To Scrape Faculty Details from VJTI website.

Firstly we will scrape Civil faculty details and then we will generalize it to other departments

Degree Staff							
Sr. No.	Designation	Photo	Name	Extension	Qualification	Specialization	Email Address
1	Associate Professor		Dr. P P Bhawe (BIODATA)	141	Ph.D	Environmental Engineering	ppbhawe@ci.vjti.ac.in
2	Associate Professor & Head		Dr. S. Y. Mhaske (My WebPage)	138	Ph.D	Geospatial Technology	symhaske@ci.vjti.ac.in
3	Associate Professor		Dr. A. S. Wayal (BIODATA)	135 & 140	Ph.D	Water resource Engineering	aswayal@ci.vjti.ac.in
4	Assistant Professor		P.S. Chaudhari (BIODATA)	146	M.Tech	Water resource Engineering	pschaudhari@ci.vjti.ac.in
5	Assistant Professor		Sameer Sayyed (BIODATA)	336	M.E.	Environmental Engineering	susayyad@ci.vjti.ac.in
6	Assistant Professor		Dr. V .B. Varekar (BIODATA)	148	PhD	Environmental Engineering	vbvarekar@ci.vjti.ac.in

We have not used any lib/built-in func to scrape websites directly.

Steps

1. Select the website you want to scrape. In our case it is the VJTI Civil department website. We will work with the url
['http://www.vjti.ac.in/index.php/academics/civil'](http://www.vjti.ac.in/index.php/academics/civil)
2. We want to extract data of professors in the Civil Department. The Fields we want to extract are 'Name', 'Designation', 'Qualification', 'Specialization', 'Email' and 'Extension'.
3. Request the server for contents of the page using PyQt5 Web Engine. Save the response received in a variable named '**response**'.
4. Parse response.text by creating a BeautifulSoup object, and assign this object to html_soup. The 'html.parser' argument indicates that we want to do the parsing using Python's built-in HTML parser.
5. The data we require resides in tabular form in the webpage. The unique distinguishing class for our data is 'sidebar'. So we extract all table rows having **class** as '**sidebar**' and store it in a **list** named '**faculty_containers**'.
6. The data inside the 'tr' element does not have any unique identifier and are all span elements. So we visually analyzed the webpage and located the positions of our data, and accessed them using the corresponding list indexes. As an example, name is the 4th column, so we access it as '**faculty_containers[i].findAll('td')[3].span.text**'
7. We used regex for identifying and extracting email ids from the table data.
8. This extracted information is stored in lists, made for one column each.
9. We created a dataframe using pandas library and added all the lists as dataframe columns. This data frame is created to store the data extracted in a csv file format.
10. So, the final extracted data is stored in 'Civil_Department.csv' file.

Challenges Encountered

Dynamic Scrapping

To protect email ids of faculties from spambots, VJTI website does not include email in html, It loads that data through JavaScript.

Reference :

```
<span
  style="font-size: 10pt; line-height: 115%; font-family: arial,helvetica,sans-serif; color: #003366;"
><span style="line-height: 115%;">
  <script type="text/javascript">
    <!--
    var prefix = '&#109;a' + '&i&#108;' + '&#116;o';
    var path = 'hr' + '&ef' + '=';
    var addy45623 = 'symh&#97;sk&#101;' + '&#64;';
    addy45623 = addy45623 + '&c&#105;' + '&#46;' + 'vjt&#105;' + '&#46;' + '&#97;c' + '&#46;' + '&#105;n';
    document.write('<a ' + path + '\&' + prefix + ':' + addy45623 + '\&');
    document.write(addy45623);
    document.write('<\&');
    //-->\n
  </script>
  <script type="text/javascript">
    <!--
    document.write('<span style=\&#39;display: none;\&#39;>');
    //-->
  </script>
  This email address is being protected from spambots. You need JavaScript
  enabled to view it.
  <script type="text/javascript">
    <!--
    document.write('</&');
    document.write('&span>');
    //-->
  </script></span
></span
```

To get away with it, we have used PyQt5 library to open the page as it would be opened in a JavaScript enabled browser then I get the source code from it and perform normal BeautifulSoup code.

Reference : [Cannot scrap protected email from website](#)

Code

```
# -*- coding: utf-8 -*-
"""
Created on Thu Feb  6 22:22:00 2020

@author: TheHead
"""

from bs4 import BeautifulSoup
import re #for regex

#As email are protected by js in vjti website, we have to load page in client side
first
#Dynamic Javascript Scraping
import sys
from PyQt5.QtWidgets import QApplication
from PyQt5.QtCore import QUrl
from PyQt5.QtWebEngineWidgets import QWebEnginePage

class Client(QWebEnginePage):
    def __init__(self,url):
        self.app = QApplication(sys.argv)
        QWebEnginePage.__init__(self)

        self.html=""
        self.loadFinished.connect(self.on_page_load)

        self.load(QUrl(url))
        self.app.exec_()

    def on_page_load(self):
        self.html=self.toHtml(self.Callable)
        print("In on_page_load \n \t HTML: ",self.html)

    def Callable(self,html_str):
        print("In Callable \n \t HTML_STR: ",len(html_str))
        self.html=html_str
        print("In Callable \n \t HTML_STR: ",len(self.html))
        self.app.quit()

url = 'http://www.vjti.ac.in/index.php/academics/civil'
response= Client(url)
```

```

html_soup = BeautifulSoup(response.html, 'html.parser')
type(html_soup)
#
faculty_containers = html_soup.find_all('tr', class_ = 'SideBar')
#print(faculty_containers[9])

print(len(faculty_containers))

names = []
Designation = []
Extension = []
Qualification = []
Specialization = []
Email = []

#https://stackoverflow.com/questions/27606265/python-beautifulsoup-extract-text-from-table-cell

# Extract data from individual Faculty container
i=0;
for container in faculty_containers:

    if container.find('img') is not None:
        # The names
        name = container.findAll('td')[3].span.text
        names.append(name)
        # The Designation
        _Designation = container.findAll('td')[1].span.text
        Designation.append(_Designation)
        # The Extension
        _Extension = container.findAll('td')[4].span.text
        Extension.append(_Extension)
        # The Qualification
        _Qualification = container.findAll('td')[5].span.text
        Qualification.append(_Qualification)
        # The Specialization
        _Specialization = container.findAll('td')[6].span.text
        Specialization.append(_Specialization)
        # The Email
        # Use of regex
        _Email= container.findAll('a',text = re.compile('[\w\.-]+@[\w\.-]+'))[0].text
        Email.append(_Email)

```

```

import pandas as pd
test_df = pd.DataFrame({'Name': names,
'Designation': Designation,
'Qualification': Qualification,
'Specialization':Specialization,
'Email': Email,
'Extension': Extension
})
print(test_df)
test_df
test_df.to_csv('Civil_Department.csv',encoding='utf-8-sig')

```


Data Collected

Name	Designation	Qualification	Specialization	Email	Extension
Dr. P P Bhawe	Associate Professor	Ph.D	Environmental Engineering	ppbhawe@ci.vjti.ac.in	141
Dr. S. Y. Mhaske (My WebPe	Associate Professor & Head	Ph.D	Geospatial Technology	symhaske@ci.vjti.ac.in	138
Dr. A. S. Wayal	Associate Professor	Ph.D	Water resource Engineering	aswayal@ci.vjti.ac.in	135 & 140
P.S. Chaudhari	Assistant Professor	M.Tech	Water resource Engineering	pschaudhari@ci.vjti.ac.in	146
Sameer Sayyed	Assistant Professor	M.E.	Environmental Engineering	susayyad@ci.vjti.ac.in	336
Dr. V .B. Varekar	Assistant Professor	PhD	Environmental Engineering	vbvarekar@ci.vjti.ac.in	148
Mrs. V. S. Bhangale	Lecturer (Selection Grade)	M.E.	Environmental Engineering	vsbhangale@ci.vjti.ac.in	145
A. P. Gorkar	Lecturer	M.E.	Water Resource Engineering	apgorkar@ci.vjti.ac.in	149
C R Bhole	Lecturer & I/C Head	M.E.	Construction Technology & M	crbhole@ci.vjti.ac.in	144
Mrs. S A Sagole	Lecturer	M.E.	Geotechnical Engineering	sasagole@ci.vjti.ac.in	148

Generalization to other Departments

We tried to generalize web scraping to all departments, but they don't have common semantics. For example


Civil

Degree Staff							
Sr. No.	Designation	Photo	Name	Extension	Qualification	Specialization	Email Address
1	Associate Professor		Dr. P. P. Bhav (BIODATA)	141	Ph.D	Environmental Engineering	ppbhav@ci.vjti.ac.in

Computer and IT

Degree Staff (Computers)						
Sr. No.	Designation	Name	Email	Phone	Qualification	[Ms]Specialisation
1	Professor	Dr. B. B. Meshram (BIODATA)	bbmeshram@ce.vjti.ac.in	24198157	PhD	Digital Forensic ; Information Security ; Data Mining and Data Warehouse; Cloud Computing

Electrical

Degree Staff (Electronics)							
Sr. No.		Designation	Name	EMail ID	Extension	Qualification	Specialisation
1		Professor Dean Student and Alumni	Dr. R. D. Daruwala	rdaruwala@el.vjti.ac.in	24198190 24198171	PhD	Computer Architecture, Advanced Embedded Systems, Biomedical Instrumentation, Computer Networks, QOS in 4G Networks

1. Designation is first in civil & comp while the photo is first in electrical
2. There is no column for a photo in comp
3. Specialisation in electrical is written as Specialization in civil and [MS] Specialization in comp
4. Extension in Civil and Electrical is written as Phone in Comp

We can solve this naming conflict by hard-coding but there are conflicts in the structure of HTML too like “Extension” in in civil while in <td> for electrical.

See screenshots on the next page for reference
Electrical :

The screenshot displays the website for the Electrical Engineering department. It features a table of faculty members with their profiles, designations, names, email addresses, extensions, qualifications, and specializations. The faculty list includes:

Sr. No.	Designation	Name	Email ID	Extension	Qualification	Specialization
1.	Professor (senior) (Senior and Junior)	Dr. R. D. Dasgupta	rdasgupta@vjil.ac.in	2419039 2419035	Ph.D.	Computer Architecture, Advanced Embedded Systems, Biomedical Instrumentation, Computer Networks, QoS in 4G Networks
2.	Professor & Head Physics, Mathematics Humanities and Management Passing Officer: EC	Dr. M. S. Penu	mpenus@vjil.ac.in	2419819	Ph.D.	Biomedical Instrumentation, Signal and Image Processing
3.	Professor TEQIP II Co-ordinator	Dr. R. N. Anjan	rnanjan@vjil.ac.in	2419034	Ph.D.	Image Processing, Computer Networks, Error Correcting Codes
4.	Department Head/Dean Research Development & Consultancy Professor	Dr. Parik A. S. Kati	parik@vjil.ac.in	2419038	Ph.D.	Dynamics & Control, Robotics, Cyber Physical Systems
5.	Assoc Prof	Anura Kumar	anurakumar@vjil.ac.in	2419030	M.E.	Wireless comm. & Networking

The browser's developer console shows several JavaScript errors, including "Uncaught runtime.lastError: The message port closed before a response was received." and "Uncaught runtime.lastError: The message port closed before a response was received."

Civil :

The screenshot displays the website for the Civil Engineering department. It includes an overview of the department, a list of diploma courses, and a table of faculty members. The faculty list includes:

Sr. No.	Designation	Name	Email ID	Extension	Qualification	Specialization
1.	Associate Professor	Dr. P. P. Shree (Associate)	ppshree@vjil.ac.in	2419031	Ph.D.	Environmental Engineering
2.	Associate Professor & Head	Dr. S. V. Mahesh (My Mahesh)	svmahesh@vjil.ac.in	2419032	Ph.D.	Thermal Technology
3.	Associate Professor	Dr. A. S. Anjan (Associate)	asanjan@vjil.ac.in	2419034 73.33 + 112.67	Ph.D.	Water resource Engineering
4.	Associate Professor	P. S. Chaitanya (Associate)	pschaitanya@vjil.ac.in	2419035	Ph.D.	Water resource Engineering
5.	Associate Professor	Samir Dasgupta	samir@vjil.ac.in	2419036	Ph.D.	Environmental Engineering

The browser's developer console shows several JavaScript errors, including "Failed to load resource: the server responded with a status of 404 (Not Found)" and "Failed to load resource: the server responded with a status of 404 (Not Found)".

As each department page has a different HTML structure and semantics, for each department one will have to write a different scrapper. **Generalization is not possible**

Tools Used

1. Python :

It has the most elaborate and supportive ecosystem when it comes to web scraping. But as mentioned in the Aim we have not used any lib directly Focused on scrapping like “Scrapy”.

We have used combination of following lib and wrote our own logic with it:

- Beautiful Soup: Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work. ***It's a parser not Scrapper.***
- Requests:
The requests module allows you to send HTTP requests using Python. The HTTP request returns a Response Object with all the response data (content, encoding, status, etc).
- PyQt:
To open the page as it would be opened in a JavaScript enabled browser

2. Google Chrome Developer Tools

Chrome DevTools is a set of web developer tools built directly into the Google Chrome browser. DevTools can help you edit pages on-the-fly and diagnose problems quickly, which ultimately helps you build better websites, faster.

Github link of Project

https://github.com/abhishekvispute/WebScraper/tree/master/Extra%20-%20VJTI_Website_Scraper

Conclusion

1. At first we learned what web scraping is followed by crawling. Then we saw Difference between them.
2. We Implemented Web Scraping using Python on VJTI website to scrap faculty details.
3. We learned how to tackle dynamic websites.
4. We saw how variable websites are, like VJTI department pages.