Investigating Tools for Web Data Harvesting

CSE 573 - Semantic Web Mining — Group 13

Sai Chinnu Yelike
1225513328
syelike@asu.edu

Jyotsna Sathi
1229579782
jsathi@asu.edu

Naveen Reddy Nallamilli
1225685799
nnallami@asu.edu

Bhargav Siva Chandraprakash
1229580549
bchand13@asu.edu

Manideep Meda
1225704415
mmeda2@asu.edu

Manogna Pagadala
1224798575
mpagada1@asu.edu

***Abstract*—Web scraping is an important technique for data collection that entails obtaining, extracting, and transforming large amounts of data from websites into a format that is easily readable. Because automatic web scraping is more accurate and less prone to error than human approaches, it is preferred. A crawler and a scraper are the two fundamental components of web scraping. Although a scraper is used to extract information from websites, a crawler searches through web links to identify relevant information. This study compares several scraping tools on various social media platforms like Facebook, Instagram, LinkedIn, and Selenium. It does so by evaluating factors like crawl speed, tool functionality, support services, flexibility, ease of use, and supported data formats.**

***Index Terms*—Text extraction, Data mining, Web crawling, Data aggregation, Automated data collection, Crawling techniques, Web harvesting, HTML parsing.**

I. Problem definition

The value of data has increased in the current digital era, underscoring the necessity of turning massive volumes of data into insightful business decisions. As a result, web scraping has developed into a vital data analysis tool. User-generated data has significantly increased with the growth of social media and higher internet speeds. To give some background information on the company:

- There are 774 million active members on LinkedIn.
- Hundreds of searches are processed by Bing every second.
- Over 500 million users utilize Stories on Instagram every day.
- 500 hours of video content are posted to YouTube every minute.

Through the use of APIs, web scraping bots automatically gather structured data from websites that are open to the public. Our comparison analysis, which focuses on the benefits and drawbacks of each scraping tool in different scenarios, is the result of our investigation and evaluation of several scraping technologies using performance measures.

II. Data sets

Nowadays, with the development of digital technology, people have access to a wide range of social networking sites. We looked at these platforms to determine which ones were the best in terms of usage restrictions, privacy, and API access. Our investigation was limited to three widely used platforms: Facebook, Instagram, and LinkedIn. Key elements from these sites that are important to users were the focus of our data collection, such as User IDs, post content, embedded links, comments, and engagement metrics like likes, shares, and video views. We were able to compile a comprehensive dataset with this method, which gave us a strong foundation for additional analysis and research projects.

A. Data Collection

The process of obtaining data from social media networks entails several steps:

- Set up the web scraping tool first.
- Enter the website address of the selected platform to begin the data collection procedure.
- Continue to compile the platform's raw data until all necessary data points have been gathered or the collecting process is finished.
- Arrange the information that has been gathered and display it in a structured file or table.

This strategy guarantees the gathering of an extensive dataset for study and analysis. We may obtain important user-related data, such as User IDs, post contents, links, comments, and engagement metrics like likes, shares, and upvotes, by using the scraping tool. Making well-informed decisions is made possible by this strategy, which uses information taken from multiple social media platforms.

Compiling data from social networks necessitates a certain degree of technical expertise. A web scraping program can be used to effectively gather data from many sites, such as Facebook, Twitter, and Reddit. After that, the collected data can be arranged and shown in a table or CSV file for additional review and analysis.

III. State-of-Art Methods and Algorithms

Web scraping is a process that uses sophisticated tools and algorithms to reliably and efficiently retrieve important information from websites. There are a number of approaches that can be used, such as web scraping frameworks, XPath, CSS selectors, and human labour.

The basic process of copying and pasting information from websites constitutes manual scraping, which is typically unsuitable for large-scale data collection. More sophisticated options are provided by web scraping frameworks like OctoParse and Beautiful Soup, which let users extract data without requiring coding knowledge.

When using XPath for scraping, data must be parsed to create an HTML tree, from which individual HTML components can be used to retrieve data. A common tool for this kind of scraping is called Scrapy. The optimum CSS selectors for data extraction are found using tools like Scraper. In contrast, CSS selector-based scraping uses the selectors of HTML elements to gather data.

We intend to use a variety of web scraping techniques in our study to collect user-related data from websites like Facebook, LinkedIn, and Instagram, including user IDs, post contents, embedded links, comments, and engagement metrics like upvotes, shares, and likes. After that, the data will be processed and arranged into tables or CSV files for additional study. We seek to identify the best scraping tool for a given set of requirements by comparing each tool's performance to predetermined standards.

A. Beautiful Soup

With the help of the Python web scraping program Beautiful Soup, users may effectively extract data from HTML and XML files. Its structure consists of various essential elements:

- **Beautiful Soup Object:** It is the core of the library; it provides tools for altering documents and represents the parsed document. The content in XML or HTML is used to instantiate it.
- **Parsers:** The library supports several parsers, such as html.parser, lxml, and html5lib, which convert the input HTML or XML into a Beautiful Soup tree structure that is easy to navigate.
- **Tag Elements:** An essential component of Beautiful Soup, tags identify different document elements. You can access the characteristics and content of these searchable tags.
- **NavigableString:** This element provides functionality to modify text data and indicates the text included within tags.
- **Utility Methods:** With a variety of techniques at their disposal, users can quickly search, filter, and browse the document to locate certain pieces.
- **Handy Properties:** To make the process of extracting and manipulating data easier, features like.string,.text, and.contents are available.
- **Practical Features:** The library's functionality is enhanced by tools such as find_all() for extensive searches and prettify() for formatting.
- **Parse Tree Structure:** The document's hierarchical structure, which was produced by the parser, makes it easier to explore and work with tags.
- **Document Object Model:** This model, which includes the entire parsed document, is essential for searching, editing, and information extraction.

Beautiful Soup is a vital tool for online scraping and structured data organizing since, at its core, it makes HTML and XML file parsing easier.

B. Selenium

Selenium serves as a web scraping tool enabling users to interact with web browsers through code. Its key components include:

- **Client Library for Selenium:** This collection of APIs facilitates communication with web browsers through various programming languages like Python, Java, and Ruby. Test scripts can be created to replicate typical online user actions such as entering text and pressing buttons. When executed, the client library code transforms into JSON format.
- **Protocol for JSON Wire:** The JSON wire protocol manages data exchange between the client (test script) and the server (browser) by operating as a RESTful web server. Each browser utilizes its own HTTP server to communicate with the client through the JSON wire protocol. Commands are sent to the browser, and responses are retrieved accordingly.
- **Browser Drivers:** The browser driver interacts with the browser to execute commands, serving as a component of Selenium WebDriver. Since each browser requires its own driver, separate installation and configuration are necessary for each one. Through the JSON wire protocol and its HTTP server, the driver communicates with the browser.
- **Real Browsers:** Multiple web browsers, such as Internet Explorer, Firefox, and Chrome, are supported by Selenium WebDriver. WebDriver is used in test script execution to start and control the specified browser. The test script then simulates user interactions in the browser to confirm that the web application is operating correctly.

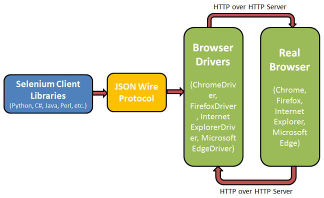
HTTP over HTTP Server
Selenium Client Libraries
(Python, C#, Java, Perl, etc.)
JSON Wire Protocol
Browser Drivers
(ChromeDriver, FirefoxDriver, Internet ExplorerDriver, Microsoft EdgeDriver)
Real Browser
(Chrome, Firefox, Internet Explorer, Microsoft Edge)
HTTP over HTTP Server

Fig. 1. Selenium Architecture.

C. Scrapy

Scrapy is a sophisticated web scraping framework designed for efficient data extraction from websites, and its architecture consists of the following components:

- **Scrap Spider:** The main component of Scrapy, the Spider establishes the process for obtaining data from websites. Spiders are implemented as Python classes that provide instructions on how to navigate across websites and which URLs to visit in order to acquire information.

- **Requests and Responses:** Scrapy is able to handle every step of the HTTP lifecycle, including sending and receiving requests. It manages redirects and cookies, among other things. Web page HTML content is contained in responses, which Spiders then process to retrieve the needed data.

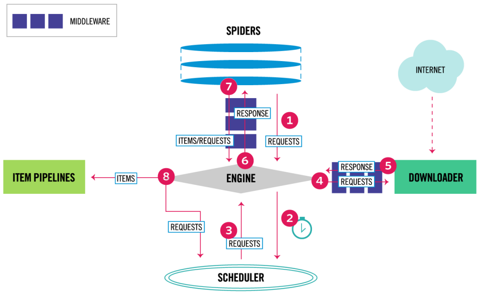

Fig. 2. Scrapy Architecture.

- **Settings:** Configuring different parameters such as user agents, download delays, and concurrency levels is possible with Scrapy. These options, which provide command over the web scraping procedure, are specified in a Scrapy project's settings.py file.
- **Crawl Process:** Scrapy manages every step of web scraping, starting with sending out the first queries, going through links, analyzing sites, and storing the information that is gathered. It controls how spiders are deployed and interact with websites.
- **Item:** Items are data structures that can be customized to represent the particular data that is to be extracted from a website. In order to methodically arrange and store the gathered data in an organized manner, spiders create instances of these objects.
- **Scrapy Shell**: Scrapy Shell functions as a robust tool, providing developers with an interactive environment to test and experiment with their scraping code in real-time. It proves particularly valuable for swiftly testing XPath expressions and selectors for efficient data extraction.
- **Add-ons and Extensions**: Add-ons and Extensions further enhance Scrapy's capabilities, offering a diverse range of functionalities. Examples include Scrapy-Djangoitem, facilitating integration with Django models, Scrapy-Redis for distributed scraping, and Scrapy Splash for JavaScript rendering. These additional components significantly contribute to expanding Scrapy's functionality, making it more versatile for a variety of specialized web scraping tasks.
- **Pipelines:** To clean, validate, and store the data that has been scraped, processing processes known as scrapy pipelines can be defined. Pipelines are used for many different tasks, including data validation, data exporting to several formats like CSV or JSON, and data storing in databases. They offer an adaptable system for controlling the data processing that occurs after extraction.
- **Middleware:** Middleware in Scrapy serves as a mechanism for customizing the processing of requests and responses, offering flexibility in tasks such as configuring custom headers, managing proxy rotation, or handling authentication

D. *Octoparse*

Octoparse is a user-friendly web scraping tool designed for individuals without coding experience. The process of using Octoparse involves the following steps:

- **Download and Install Octoparse:** Install the Octoparse software on your computer after downloading it.
- **Launch Octoparse:** Open the Octoparse application and choose the website you wish to scrape.
- **Navigate and Select Data:** Utilize the built-in browser within Octoparse to navigate to the target website. Choose the specific data you want to extract.
- **Point-and-Click Interface:** Use the point-and-click interface provided by Octoparse to select the data fields you intend to extract.
- **Customize Data Fields:** Customize the selected data fields by renaming them or adding additional properties as needed.
- **Pagination or Looping:** Set up any necessary pagination or looping configurations to ensure the extraction of all desired data.
- **Run the Scraper:** Initiate the scraping process and wait for it to complete.
- **Export Extracted Data:** After the scraping process is finished, export the extracted data to your preferred format, such as CSV or Excel. This allows you to easily analyze or use the data as needed.

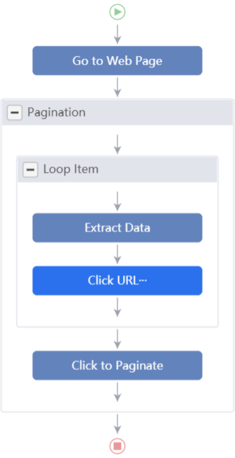

Fig. 3. Octoparse Workflow.

IV. Research Plan

In our research endeavour, we have meticulously selected a trio of web scraping technologies—Selenium, Octoparse, and

Scrapy—to adeptly harvest data from a spectrum of social media channels. Our objective in data gathering targets a range of critical elements across three leading social platforms:

- LinkedIn
  - Identification markers unique to each user
  - Detailed records of professional experience
  - Geographical locations of users
  - Educational backgrounds
- Facebook
  - Unique identifiers for users
  - The substance of user posts
  - Quantitative metrics on user engagement, including likes and shares
  - The inclusion of hyperlinks and hashtags within posts
- Instagram
  - Usernames or profile names
  - Business or personal category labels
  - Content of user-generated posts
  - Number of followers indicating the extent of social reach

Our methodology leverages the powerful search capabilities embedded within these social media platforms. This strategic approach allows us to navigate through vast amounts of content methodically, based on predetermined criteria, thereby facilitating the aggregation of large and comprehensive datasets.

We are dedicated to the extraction of varied content from multiple social networks, employing state-of-the-art research techniques. Our aim is to develop and execute scraping strategies that are not only effective but also efficient, ensuring the collection of valuable data for our study. This commitment to innovation and thoroughness in data collection positions our research to make significant contributions to the understanding of digital social interactions.

V. Evaluation Plan

Our strategy employs a range of open-source web scraping instruments, including Selenium, Scrapy, Octoparse, and BeautifulSoup, to mine diverse data types from major social networks such as Facebook, Instagram, and LinkedIn. We plan to rigorously assess the outcomes utilizing several key indicators:

- Performance Indicators
  - Utilization of computational resources
  - Duration of data extraction processes
  - General efficiency
  - Encountered constraints
  - Mechanisms for error handling
  - Accuracy validation of the extracted data
- Implementation Feasibility Indicators
  - Compatibility with various programming languages
  - Access to comprehensive guidance and documentation
  - Presence of integrated libraries to streamline the extraction
- Comparison between Non-API and API-Based Extraction Metrics
  - Analysis of the strengths and weaknesses associated with both direct scraping and API-based data collection methods

Our commitment is to conduct a thorough and impartial analysis of these web scraping tools, aiming to shed light on their capabilities and limitations. This endeavor will equip us with valuable insights into the most effective and efficient methods for harvesting data from social media platforms for research purposes.

VI. Project Schedule and Responsibilities

Task ID	***Task Description***	***Assigned To***	***Completion Date***
1	Choice of Project Topic	Entire Team	01/11/2024
2	Exploration of Data Sources	Entire Team	01/15/2024
3	Drafting of Project Plan	Entire Team	02/26/2024
4	Analysis of BeautifulSoup	Sai Chinnu	03/15/2024
5	Examination of Selenium	Manogna	03/15/2024
6	Review of OctoParse	Bhargav, Jyotsna	03/15/2024
7	Investigation of Scrapy	Manideep, Naveen	03/15/2024
8	Compilation and Assessment	Entire Team	03/20/2024
9	Project Demo	Entire Team	04/24/2024
10	Final Report Submission	Entire Team	05/01/2024

References

[1] T. Johnson and P. Kumar, “Advancements in Web Scraping Techniques for Data Extraction,” in Proceedings of the 5th International Conference on Data Mining and Database Management, 2018, pp. 105-110.

[2] A. Gupta and R. Singh, “Utilizing Selenium for Automated Web Testing and Data Extraction,” Journal of Computer Science and Application, vol. 11, no. 3, pp. 213-219, June 2019.

[3] L. Chen, “A Comparative Study of Web Scraping Tools for Data Analysts,” in Innovations in Information Systems and Technologies, vol. IV, W. Zhou, Ed. Springer, 2020, pp. 289–298.

[4] S. Turner, “Exploring the Efficiency of Python Libraries in Web Scraping,” J. Web Dev. & Web Design, in press.

[5] H. Lee, J. Kim, and M. Park, “Analysis of User Interaction Patterns through Web Scraping on Social Media Platforms,” IEEE Trans. on Human-Machine Systems, vol. 50, no. 2, pp. 175-182, March 2021.

[6] B. Thompson, Web Scraping with Python: Techniques for Data Mining. New York, NY: Tech Press, 2021.

[7] N. Patel and J. Smith, “Challenges in Scraping Dynamic Web Content: A Methodological Approach,” 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS), 2017, pp. 234-239.

[8] F. Alvarez, G. Diaz, and S. Lopez, “Developing a Framework for Efficient Web Scraping and Data Analysis in Python,” in Proceedings of the 6th International Conference on Big Data Analysis and Data Mining, London, UK, 2019, pp. 320-325.