Web Scraping Basics, Methods, and Applications

Mahmoud Embaby[*], Omar Mahmoud[µ], Mahmoud Goda[Ø] and Youssef Emad[Þ]

Department of Computer & Systems Engineering
Alexandria University
Alexandria, Egypt

[*]es-mahmoudembaby2025@alexu.edu.eg, [µ]es-OmarAbdelhady2025@alexu.edu.eg,
[Ø]es-MahmoudM.Ibrahim2025@alexu.edu.eg, [Þ]es-YoussefEmad2025@alexu.edu.eg

***Abstract* – There are over 155 million websites on the Internet. Each website contains data that concern certain users. Sometimes, we need to collect specific data from websites to analyze and extract useful information. This can be done through web scraping[1]. Collecting public data from the Internet not only provides us with new capabilities but also, allows us to measure the changes in resource volume and prices, in addition to predicting them in the future. Despite the numerous advantages, web scraping is very dangerous when used unethically by criminals to collect confidential data. This paper introduces the basics of web scraping and reviews some of the data scraping methods, approaches, data processing, and applications. Finally, it recommends a scraping approach and recommends a solution to some of the problems related to that method. Finally, it explains how this technology can improve society with an application that scrapes weather details for the Egyptian city, Alexandria. It stores the data in a file to be processed later by Machine Learning Engineers. It also sends a daily message on Telegram with the day's weather details.**

***Keywords* – Web scraping, data processing, Scrapy,**

I. Introduction

We usually use web browsers to surf the internet and collect certain information that concerns us. It is the most convenient and easy way. However, there is another way that is more efficient when trying to collect thousands of data from a dataset. With a simple block of code, we can build a program that targets webpages, reads the HTML content and parses required data then store it in a structured form. This is web scraping technology. Web scraping is valuable in many fields and is essential for many tasks to be completed. Business, marketing automation, brand monitoring, machine learning, Artificial Intelligence, and tracking changes are some of the technologies that depend on web scraping. However, we should know that not all information displayed on websites can be legally scraped without permission from the site owner. Additionally, we must validate the data that we collect before using it because it is usually written according to the desire of the site owner and can be misleading. In the current century, collecting data and gathering information is the first step for many actions. It is a step that all major companies take to keep pace with the fast-growing world.

[1] Extraction of data from a website.

Data Engineers are those who maintain the web scrapers. They build software to collect, validate and process data to be used by other people in a variety of fields. Throughout the last few years, data engineers were concerned to build libraries and frameworks for web scraping with popular programming languages to make the process as simple as possible. Finally, website analysis, website crawling, and data organizing are the three initial stages of a web scraper.

TABLE I
METHODS OF WEB SCRAPING

Method	Programming Language	Best used for
HTTP Requests	Almost all	Reading HTML or calling API
BeautifulSoup	Python	Parsing HTML and data extraction
Selenium	Python & Java	Scraping with a web-browser
Scrapy	Python	Complex web scraping projects
Octoparse	-	Simple tasks – no programming
BrightData	-	Readily available large datasets

II. Helpful Hints

A. *Static vs Dynamic Websites*

Websites are either static or dynamic. It is a way to refer to how content is loaded on a website. A static website is one that has fixed content. When you load a page, it doesn't need to make any external requests in the background to display data. Contrary, a dynamic website is the one that needs to call one or more APIs before it loads to retrieve data.

B. *Common Data Formats*

- Excel Spreadsheet
- CSV (Comma Separated Value)
- JSON (JavaScript Object Notation)
- XML (Application Programming Interface)
- MySQL (Relational database management system)

C. *What is an API*

APIs are mechanisms that enable two software components to communicate with each other using a set of definitions and protocols. For example, the weather bureau's software system contains daily weather data. The weather app on your phone "talks" to this system via APIs and shows you daily weather updates on your phone. [1]

III. Related Work

Data Engineers have always been concerned to find the best methods for extracting the data needed. Most of the data present on the internet are unstructured, so Data Engineers had to build solutions to extract useful insights from it. Fortunately, many web scraping tools are available today and can be used easily. However, there are some factors that we should consider before choosing a suitable tool for our project.

A. *Scalability*

When building a web scraper tool, we should put in mind that the data we are looking for will inevitably increase with time. Therefore, building a scalable scraping infrastructure results in having a reliable tool that can scrape large datasets at the same speed it scrapes smaller ones without takedown in performance. If we decide to create a web scraper for scraping data from social media websites like Facebook, we should invest in the scalability of our system. Looking at figure 1, we find that number of social network users worldwide is rapidly increasing. So, a web scraper that works today is not expected to run with the same performance next year unless it is scalable.

B. *Data Pipelines*

A suitable web scraping tool is one that gives users the opportunity to choose one from several pipelines. Different applications of data require different data formats. Generally, data files should be CSV, JSON, or XML. An ideal solution is to allow users to create custom pipelines to fulfill their requirements.

C. *Handling Anti-Scraping*

Some websites have anti-scraping measures that disallow a normal web scraper from accessing them. When building a solution to extract data from these websites, we should implement a method to bypass the blocking without violating the website terms.

D. *Settings and Customizations*

Usually, complex projects require building a new web scraping tool to fulfill the project needs. However, engineers managed to build tools whose settings can be changed when needed. When building a new web scraping tool, it is important to make it customizable and it is a good practice to create a settings file so that people without a programming background can modify the web scrapers to work exactly as they want them to do.

Many companies are now offering web scraping services and tools to their clients. The number of orders is unprecedentedly high. Our need for data is increasing with time. There are many available tools that even non-programmers can use to perform simple actions of web scraping. There are also professional web scrapers frameworks that depend on certain programming languages. An example of a web scraping tool that does not require any programming is Octoparse. This tool is very handy when the required data is displayed on structured web pages. On the other side, when working on a complicated project, developers tend to use tools and frameworks that depend on a certain programming language. This enables them to solve problems creatively without a limitation and allows them to have a very custom web scraping tool.

Web scraping has many uses on personal and professional levels. Some of the important uses are social media analysis, scraping is used to get information from social media sites like how people interact with certain products or news and know their preferences. Machine learning, because it needs a large amount of data to improve machine experience to work more efficiently, so you need to scrape millions of sites to collect suitable data for your models. SEO[2] monitoring is organizing sites according to their number of visits to get optimized visibility and ranking for search engines. This requires continuous scraping to update the ranking. There is a long list for the usage of web scraping.

People that do not know the importance of web scraping deny it and are strict when talking about technology. That is [3]because a lot of criminals misuse the web scraping tools for personal benefit by violating websites terms and the privacy of users. Most of social media websites forbid data extraction from their users' profiles and add anti-scraping measures. However, some unethical users build web scraping tools that evade all anti-scraping measures and collect users' data without their permission. This results in having a list of confidential data that can be sold or used for disapproved actions like sending spam to a list of emails. Although the data is public, but it was scraped in an illegal way. That was one of the disadvantages of web scraping. Another disadvantage is the difficulty of usage for new users especially those who do not have any programming background or how a website page is structured. Finally, some web scraping tools take much time until they run at their best performance. This happens because they take time to become familiar with the core usage.

Scrapy is one of the pioneering web scraping frameworks for web scraping with Python. It has many features that allow developers to complete their projects with the least effort. However, it is limited to scraping static websites. A lot of websites load pages [1] DOM after making external calls to certain APIs to retrieve the required data. If the APIs are confidential, then the ideal solution would be to use a web browser that loads the web page in the background and returns the page source. A library that can be used in this case is Selenium. It lets developers control a driver browser, visit websites, load pages and parse them. It is a good library that is

[2] search engine optimization

useful in small to medium-scale projects. However, it doesn't provide many features like Scrapy. Additionally, it is not asynchronous and can be very slow when running without collaborating libraries that support our web scraping tool to process data and save it to an external file. We were interested to find a solution to this problem and integrate Selenium library with Scrapy framework to have the best combination for scraping dynamic websites. We were able to build this tool by creating a Downloader Middleware for scrapy. When a spider starts, several web drivers are created and are ready to process requests and return the response. On making a request, Scrapy Engine sends it to the Downloader Middleware, which sends it to one of the free drivers. As soon as the driver finishes loading the page, the response is returned to the Engine for processing. This technique combined the good features of scrapy with the dynamic page loading feature of selenium.

The solution of integrating selenium with scrapy framework was helpful during scraping a complicated crowdfunding website called Kickstarter. It contains thousands of crowdfunding projects, and our goal was to extract information related to them. The website loads data dynamically. It has a standard HTML template for all project pages, the project data is loaded from an external confidential API that we could not use. The data was accessible with Selenium driver. However, we did not want to lose the advantages that Scrapy provides. Using the new Downloader Middleware enabled us to successfully scrape the data we wanted, processing and saving it with the best efficiency and high reliability.

During the Kickstarter project, we wanted to store the extracted data in a MySQL database. However, scrapy does not have a data pipeline for MySQL. Therefore, we built a new pipeline that connects to our MySQL server when a spider runs and executes a query to store the new data in the database when new data is processed.

Finally, we worked on another project related to web scraping and data analysis. Our goal was to be able to predict upcoming changes in weather in Alexandria, Egypt, and send warnings when an abnormal change occurs so that we can take the actions needed to avoid disasters. We built a new web scraper using Python programming language. Basically, it scrapes data every day from openweathermap.org website, stores it in a MySQL database, analyze the current day weather, and compares it with previous data stored in our database to predict upcoming changes. If the upcoming change exceeds a specific range, the software sends an email to warn us.

IV. Scraping Methods

There are two techniques of scraping, manual or automatic extraction. Manual extraction refers to the copying and pasting of the content from a website. It is not suitable for most of the cases, but it is best for the sites that have strong anti-scraping measures like bot detection and captchas. However, automatic extraction refers to using software to extract the data from sites automatically[1]. It is used for parsing data from structured forms like HTML. It can also be used to parse PDFs with standard formats. There are too many tools for automatic data extraction. Some of them require a programming background and some do not require it. Octoparse is an example of a tool that does not require a programming background. It is a simple software that you can install on Windows and record instructions in a simple browser simulator. It lets you scrape large amounts of data without having to do any coding. Some of the alternatives to Octoparse are Mozenda, Parse Hub, Vaazo, Portia, and Komodo Edit.

Some projects' requirements are beyond the ready-to-use web scraping tools that do not require coding. These projects depend on programming for extracting the data. There are many programming languages that can be used for data scraping. Famous ones are Python, Java, PHP, JavaScript, and Go Programming Language. Each of these languages has several frameworks and packages that make the work easier for developers working on web scraping projects. Refer to table (II) for a list of top libraries used in the mentioned languages. New tools and methods for web scraping are being developed every period and it is expected that the progress stays for a long time because the application for data is increasing along with the desire for having a structured dataset containing data that concerns us.

TABLE II
Web Scraping Packages

Python	Java	JavaScript	PHP	Golang
Scrapy	Apache Nutch	Axios	Goutte	Colly
Requests	Jsoup	Nightmare	cURL	Ferret
lxml	Jaunt	Cheerio	Requests	Gocrawl
Selenium			Buzz	Hakrawler

Scrapy is an open-source python framework that was originally created to extract data from websites using APIs.[3] It supports Python 2 and Python 3 and works on Windows, Linux, and MacOS. It is famous for its reliability. Scrapy is well-documented, scalable, and highly customizable. Its code

architecture is designed using spiders[4]. Writing spiders for a scrapy project is simple but very efficient. With scrapy, we can build web scrapers that are optimized to use low memory and CPU, and make parallel requests to many websites at the same time. Scrapy has default pipelines for data, we can store extracted data in many formats like CSV, JSON, XML, Pickle, and Marshal. Additionally, it has built-in services for logging, stats collection, sending emails, and finally a telnet console. Looking at figure 1, we find that the main components of scrapy are: Scrapy Engine, Scheduler, Downloader, Spiders, Item Pipeline, Downloader middle-wares, Spider middle-wares, and event-driven networking. This tool is maintained by Zyte company and contributions to improve the tool are available for developers through the scrapy GitHub repository. Finally, developers using scrapy can customize everything without changing the source code of the framework. This feature empowers them to build innovative solutions to complicated projects. Scrapy was first released in 2008 and is still maintained until now. All these features make scrapy the perfect option for many projects and that is why it is famous.

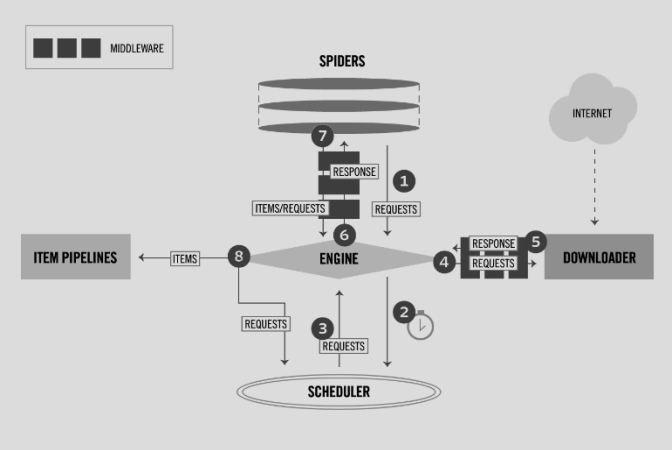

Figure 1 - Scrapy Architecture Overview

A. Data Flow in Scrapy

The engine receives the initial request from the spider, schedules the request in the Scheduler, receives the next request from the Scheduler, and sends it back to the Downloader passing through the Downloader Middlewares. The downloader middleware returns the response to the engine and the engine sends the processed item to Item Pipelines.

B. Components of Scrapy

B.1. Scrapy Engine

It is responsible[5] for controlling the data flow between all components of the system and triggering events when certain actions occur.

B.2. Scheduler

It schedules when a request should be processed. It receives the requests from the engine and stores them in a queue, then sends them back when the engine requests.

B.3. Spiders

They are custom classes written by the user with all instructions on how to scrape the website and what data should be extracted.

B.4. Item Pipeline

It is responsible for processing the data extracted by the spiders. It mainly cleans the data and validates it, then stores it in a structured data file.

B.5. Downloader middlewares

They are hooks between the Engine and the Downloader. They process requests when they pass from the Engine to the Downloader, and responses that pass from the Downloader to the Engine.

B.6. Spider middlewares

They are hooks between the Engine and the Spiders. They process spider input and output.

B.2. Event-driven networking

Scrapy uses Twisted, a popular event-driven networking framework for Python. So, it implements asynchronous code for concurrency.

V. Results and Discussion

There are many tools for data extraction that are used by millions of people. Scrapy is an ideal web scraping framework for a user with a programming background seeking to automate the process of extracting data from the internet. It is a reliable method that can be customized to fulfill the requirements of most of our projects. It is very well documented and is optimized to use the least memory and CPU giving the highest performance if set up correctly.

The scrapy framework is limited to scraping static websites. So, to extract data from dynamic websites we must use a library that can open a web browser and load the pages we need to scrape. Selenium is the best for this situation. Integrating the Selenium library with the Scrapy framework results in a reliable combination that can efficiently extract data from dynamic websites with keeping the good features that scrapy offers.

[4] Classes defined by the developer with instructions of how to scrape a certain website and how to extract data from pages.

Scrapy has built-in pipelines for saving data to the famous file formats like CSV and JSON. However, we wanted our data to be stored in a MySQL server directly. We had two options, the less efficient one was to save the data as CSV and import it to MySQL or build a new pipeline for Scrapy that connects to the server and executes queries to push extracted data directly to our MySQL server.

Finally, we managed to build an application for web scraping and data analysis with Scrapy to scrape weather data from openweathermap.com. Data is stored in MySQL and every day the software predicts the next day's weather. If an unexpected change occurs, it sends a warning to emails.

VI. Conclusions

Web scraping is important. It is the driving wheel for many applications and technologies. It is a form of using software to automate tasks where data is extracted from the Internet by a robot, processed and validated then saved in well-structured files to be used in other projects. There are two main methods for automated web scraping, a no-coding method for basic web scraping. This method can be used by people with no programming background. The other method is based on writing blocks of codes of instructions to create custom solutions for our web scraping project. If we decide to build a web scraping project by writing codes, we have many options. First, is the programming language. There are many programming languages that support web scraping and have powerful web scraping tools. Second, frameworks pave the road for developers to build reliable and efficient web scraping codes with the least effort. One of these frameworks in Python is Scrapy, it has a lot of features that developers like. Scalability, CPU and memory optimization, speed, and customization are some of what distinguishes Scrapy. Scrapy is not a good option when scraping dynamic web pages, and it does not have a data pipeline for MySQL. In this paper, we were able to create new Downloader Middleware to integrate Selenium, which is a browser-based web scraping tool, and Scrapy to have a powerful combination capable to extract data from websites that load data dynamically. We were also able to create a new data pipeline so that the extracted data is processed and stored in MySQL server directly by the spider.

VII. References

[1]E. Uzun, "A Novel Web Scraping Approach Using the Additional Information Obtained From Web Pages", IEEE Access, vol. 8, pp. 61726-61740, 2020. Available: 10.1109/access.2020.2984503.

[2]K. Henrys, "Importance of Web Scraping in E-Commerce and E-Marketing", SSRN Electronic Journal, 2020. Available: 10.2139/ssrn.3769593.

[3]"Scrapy 2.6 documentation", Docs.scrapy.org. [Online]. Available: https://docs.scrapy.org/en/latest/. [Accessed: 29- Apr- 2022].

[4]M. Greenfield, "Number of social media users 2025", Statista, 2022. [Online]. Available: https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/. [Accessed: 29- Apr- 2022].

[5]"Web Scraping With Scrapy", ScrapFly, 2022. [Online]. Available: https://scrapfly.io/blog/web-scraping-with-scrapy/. [Accessed: 29- Apr- 2022].

[6]H. Kühnemann, "Anwendungen des Web Scraping in der amtlichen Statistik", *AStA Wirtschafts- und Sozialstatistisches Archiv*, vol. 15, no. 1, pp. 5-25, 2021. Available: 10.1007/s11943-021-00280-5.

[7]M. Khder, "Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application", *International Journal of Advances in Soft Computing and its Applications*, vol. 13, no. 3, pp. 145-168, 2021. Available: 10.15849/ijasca.211128.11.

VIII. Plagiarism Percentage

Plagiarism Percentage:

Uniqueness: 94%
Plagiarized: 6%
Plagiarism Report is attached with other documents.