Design and Implementation of a Multipurpose Web Scraper for Financial and Employment Data

Samaksh Mathur[1], Priyanka Sharma[1], Ajay Bhardwaj[2], Rohit Upadhyay[1], Rohit Jangid[1], Rohit Garg [1]

[1]Department of Computer Science & Engineering, Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur, Rajasthan-302017 (India)

[2]Department of Electrical Engineering, Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur, Rajasthan-302017 (India)

Email: samakshmathur25@gmail.com, priyanka.sharma@skit.ac.in, ajay.bhardwaj@skit.ac.in, rsharma1000@gmail.com, gargrohit182@gmail.com

Received 19.05.2025 **received in revised form** 17.11.2025, **accepted** 23.11.2025

DOI: 10.47904/IJSKIT.15.2.2025.24-27

Abstract- With the increasing availability of structured and semi-structured data on websites, there is a need for efficient tools to extract such information. Web scraping, the process of automatically extracting data from web pages, has been very important in applications like financial analytics and job market studies. This study presents a powerful, multifunctional web scraper created specially to retrieve employment and stock market data. Implemented using Python and its companion tools, including Beautiful Soup and Selenium, the scraper fixes every problem caused by dynamic site content, JavaScript rendering, and CAPTCHA methods. It has been verified and tested on websites for employment portals and livestock.

Keywords– Web Scraping, Beautiful Soup, Selenium, Dynamic Content, Data Extraction.

1. INTRODUCTION

The digital era has substantially enhanced the accessibility of information through web platforms. However, converting unstructured web data into structured information for useful information is a difficult task. Even with a single website we may not be able to see all the relevant data at one place. Web scraping has become an effective option for automating this procedure [1]. Web Scraping is the technique in which people can extract data form multiple websites to a single database or excel or spreadsheets so that it is easy to analyse the data [2]. It can be used in a variety of fields, such as in finance to monitor stock market variations, price comparison, web indexing [3].

Even though in today's era online scraping technologies have developed, many of the tools are domain-specific, due to which their application in different sectors in limited. There are some methods that some websites use to prevent web scraping, such as detecting and disallowing bots from crawling (viewing) their pages. The aim of this research paper is to provides a multifunctional web scraper that can easily scraps data from a variety of domains in order to address these issues. A key characteristic is its capacity for adaptation to dynamic online environments. Also, it implements a modular architectural design and follows ethical data extraction methodologies.

2. LITERATURE REVIEW

Big data analytics helps companies to understand large amount of data and get new informations about their business [4]. It also helps in finding hidden patterns inside big data sets which is very useful for taking better decisions. Now a days, internet is full of data from websites, social media, and online platforms, so its important for companies to know how to collect and use this data. Web scraping is one of the common methods for doing this [5].

Before, scraping was simple. People use to write regular expressions to extract data from static HTML pages. But now things got changed. Most websites are made using AJAX, React and other dynamic framework, so the old scraping tricks are not working properly anymore. Tools like BeautifulSoup and Selenium became popular after they got explained in books like Automate the Boring Stuff with Python by Al Sweigart. Later, Mastering Web Scraping in Python by Michael Heydt showed how to deal with dynamic content, APIs and also how to solve CAPTCHAs and rate limits.

But still, many scrapers work only for particular type of websites. They don't work properly if the structure of website changes. So, now there is a need of general scraper which can work for different kind of websites, handle dynamic data and also avoid anti-scraping measures. This paper is about looking into such modern scraping methods and tools [6]. Despite these advances, the majority of scraping

solutions are domain-specific, limiting their generality to various data sources or changing website architectures. It is with this that a new paper is proposed: presenting a generic scraper designed in such a way as it can overcome the above by adapting to various data sources and emphasizing accuracy, reliability, and scalability. Hence, it bridges the difference between specialized tools and what is needed: a universally applicable scraping solution capable of overcoming dynamic content and anti-scraping measures [7].

3. METHODOLOGY

The Methodology section describes the systematic approach taken in the development of an overall, multi-purpose web crawler. The current project deals with two dominant domains: financial data and employment data. The methodology could be split into several critical stages to ensure complete understanding of the techniques, tools, and workflows.

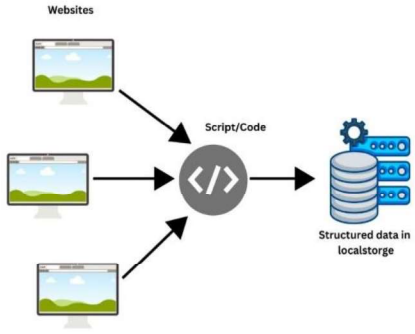

Figure 1: Web Scrapping

3.1 Data Collection

Identify trusted sources of information: This is the initial development process for the scraper. For financial data, they scoured stock market websites and trading websites. They took employment data from job portal websites and companies' career pages. They used these platforms because they had the most active users and vast amounts of data storage. The data scraping scripts were so well-configured to extract both structured and unstructured data, including HTML tables, JSON responses, and dynamic content loaded. Special attention was given to the frequency of data and relevance to keep the scraped information current.

3.2 Tools and Technologies:

The web scraper uses a very effective set of tools and technologies in extracting data from static and dynamic websites.

Python is primarily used as the programming language because it is easy to use, flexible, and has a large number of libraries. For tasks on web scraping, it is using libraries such as Beautiful Soup and Selenium. Beautiful Soup does all the parsing and extracting data from static HTML content while Selenium allows interaction with the dynamic, JavaScript-driven pages by simulating actions for the browser. HTTP libraries like Requests, for example, are employed to send requests and collect HTML content, whereas, tools like browser developer tools and network inspectors handle the AJAX requests for dynamic loading of data. The process of scraping dynamic content also happens without a graphical interface thanks to headless browsers, such as ChromeDriver.

To ensure efficient data storage and processing, relational databases such as SQLite or MySQL have been integrated along with the Python Pandas library used for organizing and analyzing scraped data. Anti-scraping measures are also countered with techniques such as user-agent rotation and CAPTCHA handling, which minimizes chances of detection. Pandas is used for data manipulation and structuring.

3.3 Data Processing

The data collected was then cleaned with maximum quality and ease. All the duplicate records and incomplete records with irrelevant values were carefully sought out and removed. In the clean-up process, the pandas and NumPy libraries for Python were used to make it easy to manipulate large datasets. For financial data, cleaning involved formatting into time-series structures, which helped in a comparative detailed analysis of the trends and forecasting. On job-related information, cleaning would involve systematic categorization considering criteria that would include role, location, and company to further break down.Proper cleaning and optimization of datasets were followed with special care toward paving the way for the further processing and analyses that were required. Techniques to tackle possible anomalies or inconsistencies included proper statistical checking and validation methods. The results, thus derived are reliable and well-structured concerning just what is actually needed to meet the exact requirements from both the financial and the employment sectors. This has, therefore, provided a wonderful basis towards deriving meaningful insights and subsequently the research objective was consequently very effective.

Figure 2: Scraping flow structure

3.1 Proposed work:

The proposed work is multipurpose in scraping the web and attempts to create one that responds to the urgent need to automate accurate and efficient data extraction from dynamic web platforms. It is going to target two of the very critical domains: financial markets and employment platforms. Financial market data- such as stock prices, trends, and even economic indicators is dynamic, time-sensitive, and sensitive to even minor changes within a timeframe, hence necessitating precise collection in due time.

The job boards have to continue updating postings and candidate profiles. Again, this requires dynamism for the scraper with complex structures of data. In using the specificity of the individual domains, the scraper is achieving easiness in the processes for gathering data while always ensuring to be reliable as well as accurate. Design Modularity, scalability, and adaptability are the key aspects used to overcome the challenges created by modern web technologies. The system of modularity ensures that all reusability comes along with components that can be easily modified for data requirements of various users.

The system's flexibility to scale allows it to manage increasing data volumes. Adaptability is helpful when dealing with dynamic content that is rendered through APIs, JavaScript, or AJAX. To make it more accurate and useful, more sophisticated methods are used, such as AJAX request handling, API integration, and browser automation with Selenium.

This study addresses changing data collecting difficulties and offers a complete and reliable automated data extraction solution, laying the groundwork for flexible applications in a variety of web settings.

3.2 Functionality:

3.2.1 Versatile Data Extraction: Data can be extracted by the scraper from a variety of sources, including dynamically produced online content and static HTML pages. Also, it can use data query languages to some parse HTML pages and to retrieve and transform page content.

3.2.2 Flexible Modular Design: This multipurpose scraper is efficient and also flexible as it is designed with modular architecture. Users can easily modify or add modules to adapt the scraper for new data sources or specific domains, all while maintaining its core functionality.

3.2.3 Efficient Data Storage and Management: To ensure that the extracted data easily connects with data analysis tools, we have provided support for exporting the data to structured formats like CSV, and JSON file. We have used Python libraries such as Pandas and NumPy to handle missing data along with irrelevant records, and duplicate entries.

3.2.4 Scalability and Adaptability: This multipurpose web scraper's basic design features are scalability and flexibility, which are important for its function in multiple applications and data-intensive scenarios.

Scalability enables handling large volumes of data without degradation of performance in a particular application, including such applications in finance, recruitment, and so on.

The adaptable architecture of this type of web scraper will allow new websites and data formats to be handled with little reconfiguration of such architecture. Such features make the scraper versatile and reliable, thus meeting the demands of dynamic and changing web environments.

3.3 Challenges and Mitigation

This project poses the following challenges:

Dynamic Content: It has been encountered that most target websites use JavaScript to load content. This was handled using Selenium to simulate a browser environment.

CAPTCHA Handling: Third-party tools and browser automation were used to bypass CAPTCHA mechanisms. Wherever possible, direct fetching with APIs has been enabled in order to avoid scraping blocks.

Anti-Scraping: Rate limiting and request headers were modified to simulate human interaction in order to minimize detection.

Ethical Considerations

Data scraping is indeed a gray area in the legality and copyright paradigm. Ethical data scraping not only has to deal with law but probes further into considerations like data authenticity, author credibility and compensation, user data consent whether public or confidential.

This study highly adhered to the legal and ethical standards set for the development and deployment of the web scraper. The goal while building this scraper was to ensure that we ventured into the domain of web crawler and scraping but stayed within the limits of sensitive topics like privacy and consent objectives. All the scraping was done respecting the robots.txt instructions of the targeted websites to not step beyond them as instructed by their developers. All the scraped data was utilised for only academic, not-for-commercial purposes. There was no activity done on that data which could possibly be violating intellectual property rights or even abusive to the data itself. Scraping from websites like social media and forums was purposefully avoided to prevent the tech from invaded any boundaries set by the development team because such actions can lead to misuse of data, potentially causing unwanted and/or massive damage (scams,frauds,stealing, etc). While confined to the policies and constraints set out by the development team, the scraper program accomplished and fulfilled its intended purpose and data gatherings goals showcasing a stellar approach by focusing on respect for legal boundaries and online resources, executed with ethical practices.

3.4 Testing and Validation

Testing and validation phase was a crucial phase in the development of our product, hence providing a framework to test the limits and capabilities of the web based scraper. It was tested against manual and semi-generated test cases to ensure the scraper functioned accurately and fetched raw data files without any corruption, loss or discrepancy.

To ensure consistent data generation volumes and smooth process flow, recognised and acclaimed market tools were used, also highlighting the limits and further improvements area for future analysis.

The web scraper testing was productive and able to help the project to accomplish its planned goals and evolve into a efficient yet powerful and scalable solution for different domains like media, finance, job market, medicine, etc.

4. RESULTS AND DISCUSSION

This multipurpose web scraper was tested on different set of websites, including online portals like Amazon and Google services like Wikipedia, to see how effective it was on scraping different kinds of data. While doing web scrapping code reusability and maintainability is very important.

Code reusability is very important because when we develop a program mostly having some common features that was used before for e.g. for a website scraping data the program needs to access the website and if a good code is written in a way using password and username you can reuse it in future project if it is gonna be run in a same situation. Scrapping involves maintenance because no matter how you make scrapper, a big change in site design or behaviour will make it difficult for program to find appropriate data on the site.

a) Using python to scrape the html tags from the websites.
b) Selenium to install chrome driver for robot.txt file.

3. CONCLUSION

Web scraping is a very useful tool in today's information world, and an essential one in various fields for any company which wants to maintain online presence which is very necessary to survive in market today, But while speaking of this we should know that there are more strict legal measures enforced and will be enforced in coming years but the rate of this market will keep on growing which makes it a very valuable skill to understand. Python is the best language to implement web scrapping due to it's fast learning curve and powerful syntax. In the future, the scraper will be able to identify contents in real time using machine learning techniques, which will help to increase its flexibility to changing data structures and formats. The outcome of this study is to offers a review on web scraping techniques and software for a multipurpose web scraper which can be used to extract data from multiple web sites and can show them at one place.

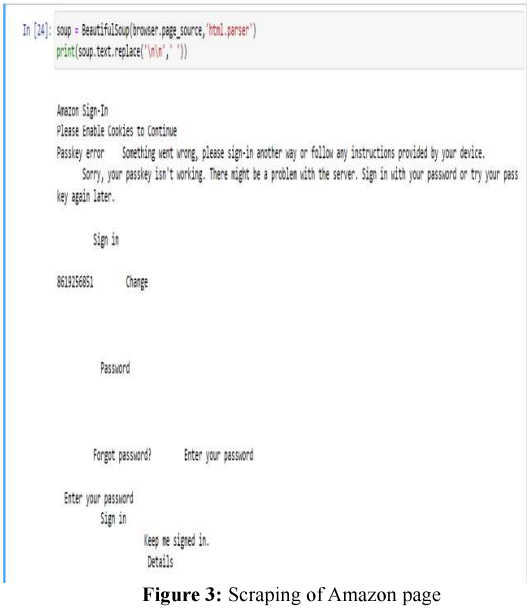

Figure 3: Scraping of Amazon page

REFERENCES

[1]. A. Sweigart, Automate the Boring Stuff with Python, (2015), San Francisco, CA, USA: No Starch Press.
[2]. M. Heydt, Mastering Web Scraping in Python, (2018), Birmingham, U.K.: Packt Publishing.
[3]. R. Mitchell, Web Scraping with Python: Collecting Data from the Modern Web, (2018), Sebastopol, CA, USA: O'Reilly Media.
[4]. K. Singh and P. Agrawal, "Challenges in web scraping: Techniques and tools," Journal of Data Science, (2021).
[5]. J. Murach, Murach's Python Programming, (2019), Fresno, CA, USA: Mike Murach & Associates.
[6]. M. Heydt, Python Web Scraping Cookbook, (2017), Sebastopol, CA, USA: O'Reilly Media.
[7]. R. Shirazi, Effective Web Scraping and Data Extraction, (2016), New York, NY, USA: Apress.