International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 06 | June - 2023 SJIF Rating: 8.176 ISSN: 2582-3930

Automated Web Scraping Tool

K. Sriram
B. Tech
School of Engineering
Malla Reddy University, India.
2111cs020566@mallareddyuniversity.ac.in

V. Sriram
B. Tech
School of Engineering
Malla Reddy University, India.
2111cs020567@mallareddyuniversity.ac.in

Y. Sriya
B. Tech
School of Engineering
Malla Reddy University, India.
2111cs020568@mallareddyniversity.ac.in

M. Srujan Kumar
B. Tech
School of Engineering
Malla Reddy University, India.
2111cs020569@mallareddyuniversity.ac.in

M. Sruthi
B. Tech
School of Engineering
Malla Reddy University, India.
2111cs020570@mallareddyuniversity.ac.in

G.Suchitra
B. Tech
School of Engineering
Malla Reddy University, India.
2111cs020571@mallareddyuniversity.ac.in

Guide: Chandana
Professor
School of Engineering
Computer Science – AI&ML
Malla Reddy University, India.

Abstract: Web data extraction, also known as web scraping, is a powerful technique used to extract data from websites using their HTML structure. Web scraping can be useful in a variety of industries, such as marketing, finance, and research, as it allows businesses and researchers to gather large amounts of data quickly and efficiently. Python is a popular programming language for web scraping due to its flexibility and ease of use. The two libraries that are commonly used in Python for web scraping are Beautifulsoup and requests. Requests is a Python library used to send HTTP requests to web pages, and it returns the server's response as an HTTP response object. This library can be used to obtain HTML pages from websites, which can then be analyzed or processed by other Python libraries such as Beautifulsoup. Beautifulsoup is a Python library that parses HTML and, you can locate and extract specific HTML tags or attributes from a webpage, and retrieve their contents or values. Beautifulsoup provides a convenient way to navigate and search through complex HTML documents, making it an essential tool for web scraping with Python. We first import the requests and Beautifulsoup libraries. We then define the URL to scrape and send an HTTP request to that URL using the requests library. We use the Beautifulsoup library to parse the HTML content of the response and extract the page title and all links on the page. Finally, we print the title and links to the console. The overall main objective of this project is to extract information from website and process it into simple structures such as spreadsheets, database or CSV file.

1. INTRODUCTION

Web scraping has revolutionized the way data is gathered from websites, offering a valuable tool for researchers, businesses, and individuals to extract information efficiently. With the exponential growth of the internet and the abundance of data available online, web scraping has become an essential technique for data acquisition. By automating the process of extracting data from websites, web scraping enables users to access structured and unstructured data that can be utilized for various purposes such as market research, competitive analysis, sentiment analysis, and trend identification.

The objective of this project is to explore the techniques and best practices of web scraping using the Beautiful Soup and Requests libraries in Python. Beautiful Soup is a powerful library Requests provides an intuitive interface for sending HTTP requests and retrieving website content. By focusing on these libraries, we aim to provide readers with a comprehensive understanding of web scraping principles and practical implementation.

2. OVER VIEW OF WEB SCRAPING

Definition and Importance:

Web scraping refers to the automated extraction of data from websites. It involves the use of software tools and libraries to navigate through the HTML or XML structure of web pages, retrieve relevant information, and store it for further analysis or use. Web scraping has gained significant importance in the era of big data, as it allows for the collection of vast amounts of data from diverse sources, providing valuable insights and opportunities.

The importance of web scraping lies in its ability to enable data-

driven decision-making and enhance business intelligence. By scraping data from websites, organizations can gain a competitive edge by accessing real-time market data, monitoring competitors'. Researchers can leverage web scraping to projects, such as building data-driven applications or conducting personal research.

Applications of Web Scraping:

Web scraping finds applications across various domains and industries. Some common applications include:

a. Market Research and Competitive Analysis: Web scraping allows businesses to gather data on product prices, customer reviews, competitor strategies, and market trends. This data can be utilized to optimize pricing, develop competitive strategies, and identify new business opportunities.

b. Content Aggregation and News Monitoring: Web scraping enables the collection of news articles, blog posts, or social media content from multiple sources. This data can be aggregated, analyzed, and presented in a consolidated format, providing users with up-to-date information on specific topics of interest.

c. E-commerce and Price Comparison: Web scraping can be utilized to extract product information, including prices, descriptions, and customer reviews, from e-commerce websites. This data can then be used for price comparison, market analysis, or personalized recommendation systems.

d. Sentiment Analysis and Social Media Monitoring: Web scraping allows for the extraction of social media data, such as tweets or comments, for sentiment analysis. This helps businesses gauge customer opinions, monitor brand reputation, and identify emerging trends or issues.

e. Academic Research and Data Analysis: Researchers can leverage web scraping to gather data for academic studies, social science research, or scientific analysis. This includes collecting data from online databases, scholarly articles, or research repositories.

f. Real Estate and Property Listings: Web scraping can be used to extract property listings, rental prices, and other relevant information from real estate websites. This data can assist in market analysis, property valuation, and investment decision-making.

3. PROBLEM STATEMENT

There are difficulties in accurately and widely obtaining pertinent data from websites due to the rising demand for efficient data harvesting from websites. In order to traverse through website structures, obtain needed data, and manage numerous complications, such as security measures and JavaScript-driven content, web scraping, as a solution to this problem, requires effective methodologies and best practises.

The basis for proving the viability of web scraping strategies will be the study paper's use of actual website data. This information may consist of real estate listings, news articles, of web scraping across multiple sectors and industries by utilising a variety of data sources.

The following are the research topics that inform this project:

1. How can web scraping methods like Beautiful Soup and Requests be used to efficiently extract data from websites.

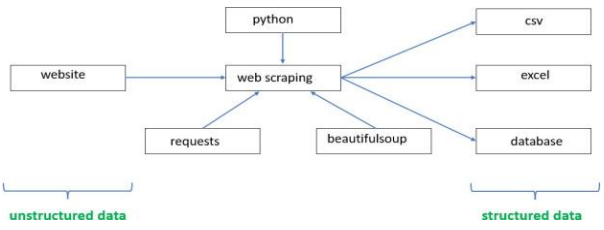
2. How should these libraries be used to retrieve HTML content, parse the data, and extract pertinent information.

Through a comprehensive exploration of these research questions, this research paper aims to provide clear guidelines and practical solutions for extracting data from websites using the Beautiful Soup and Requests libraries. The project will address the identified problem by demonstrating step-by-step techniques, discussing best practices, and offering insights into ethical considerations and future enhancements.

4. METHODOLOGY

ARCHITECTURE

The project's architecture begins with the user interface, where users enter the target URL and parameters, and proceeds in a modular and sequential manner. The HTML material is obtained from the web page retrieval component and given to the HTML processing component. The data extraction component searches through the HTML that has been parsed for the desired data and does any necessary data transformation. The retrieved data is then saved in the appropriate format for later use or analysis.



Here are the main components of the architecture of a Web Scraping:

Web page retrieval using Requests library

The first step in web scraping is to retrieve the HTML content of the target web page. This is achieved using the Requests library, which allows sending HTTP requests to the web server and receiving the corresponding response. The

methodology will outline the process of making GET requests, handling headers and parameters, and managing cookies and sessions if required.

Parsing HTML with BeautifulSoup library

Once the HTML content of the web page is obtained, the next step is to parse it using the BeautifulSoup library. This library provides various parsers to handle different types of HTML structures. The methodology will cover the initialization of the BeautifulSoup object and the selection of the appropriate parser based on the HTML content.

Navigating the HTML structure

To extract specific data from the web page, it is essential to navigate through the HTML structure effectively. BeautifulSoup provides methods and functions to traverse the parsed HTML, such as finding elements by tag name, class, ID, or attributes. The methodology will explain how to locate and select specific elements within the HTML structure.

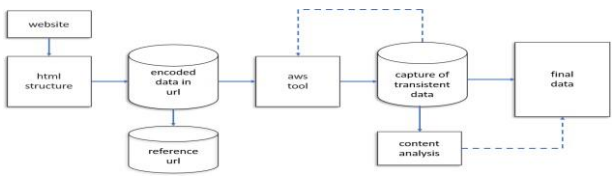
Extracting data using BeautifulSoup's methods

Once the desired elements are located, the next step is to extract the relevant data from them. BeautifulSoup offers a range of methods and techniques to extract data, including accessing element attributes, retrieving text content, and extracting data from tables or lists. The methodology will demonstrate the usage of these methods to extract data efficiently.

DESIGN

Designing the web scraping tool using BeautifulSoup and Requests libraries involves defining the structure, flow, and interaction of the different components.

DFD Diagram



The Data Flow Diagram provides an overview of the major components and their interactions in the web scraping tool. It showcases the flow of data from the User Interface to the Web Scraping Module, and from there to the HTML Parsing Module and Data Storage Module. It also highlights error handling within the Web Scraping Module.

5. EXPERIMENTAL RESULTS

Executing the above code after importing the necessary modules and packages we get thefollowing output:

Experimental Setup:

Target Website: https://notes.ayushsharma.in/technolog.

Scraping Objective: Extract the title, excerpt, and publication date of articles from the website. Extraction success rate is 100%. Storage format used is Excel (.xlsx).

Usability: Moderate (requires basic programming and web scraping knowledge)

Efficiency: Moderate (basic functionality, but lacks advanced optimization techniques)

User Satisfaction: Satisfactory for basic scraping tasks, but may not meet the needs of advanced users without additional modifications or enhancements

6. CONCLUSION

In conclusion, this application serves as a basic implementation of web scraping using the BeautifulSoup and Requests libraries. It demonstrates the retrieval of a web page, parsing of HTML content, and extraction of desired data elements. However, it's important to note that web scraping should always be done ethically and within legal boundaries. Before scraping any website, make sure to check its terms of service and respect its policies and restrictions. Additionally, it's important to avoid overloading a website's server with too many requests, which can cause performance issues and potentially result in your IP address. Overall, the project provides a foundational understanding of web scraping using BeautifulSoup and Requests libraries. It demonstrates the basic functionality of retrieving web page data and extracting relevant information.

7. FUTURE ENHANCEMENTS

In future enhancements, the above project can be further improved to enhance its functionality, efficiency, and usability. Firstly, collection. Error handling and exception handling mechanisms should be incorporated to ensure the code can gracefully handle errors and exceptions during the scraping process, improving the overall robustness. Additionally, data validation and cleaning techniques can be applied to validate and clean the extracted data, ensuring its

accuracy and consistency. Advanced scraping techniques, such as handling dynamic content and utilizing scraping frameworks, can be explored to tackle more complex scraping scenarios. Performance optimization measures, like asynchronous requests and parallel processing, can be implemented to enhance the code's efficiency and speed. Developing a user-friendly interface or CLI can facilitate easier interaction with the scraping tool. Comprehensive documentation and tutorials should also be provided to guide users on how to effectively use the tool. Finally, scalability and robustness testing should be conducted to ensure the code performs consistently across different websites and handles various edge cases effectively. Overall, these future enhancements will contribute to a more powerful, efficient, and user-friendly web scraping tool.

8. ACKNOWLEDGMENT

We would like to express our gratitude to the project guide Prof. Chandana for their valuable guidance, support, and mentorship throughout the development ofthis project. Their vast knowledge and expertise, as well as their willingness to go the extra mile to ensure that we understood the concepts, have been instrumental in shaping our understanding of the subject matter and ensuring thesuccessful completion of this project.

Furthermore, We would like to acknowledge the Head of the Department, Dr. Thayyaba Khatoon, for their continuous encouragement and support throughout the project. Their constant motivation and willingness to provide resources have been pivotal in ensuring the success of this project.

9. REFERENCES

[1]. Beautifulsoup Documentation:
https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[2]. Requests Documentation:
https://docs.python-requests.org/en/latest/

[3]. Guide to Web Scraping:
https://opensource.com/article/21/9/web-scraping-python beautiful-soup

[4]. **"Web Scraping with Python, 2nd Edition"** by Ryan Mitchell: This book covers a wide range of topics related to web scraping, including how to use Python libraries such as Requests and Beautiful Soup to extract data from websites