JavaScript Web Scraping Tool for Extraction Information from Agriculture Websites

Mariya Zhekova[1,*], and *Emir* Yumer[1]

[1]University of Food Technology, 26 Maritza Blvd, Plovdiv 4002, Bulgaria

Abstract. Extracting information from an information platform, site or system is possible if the information is structured or annotated in a way that is convenient for subsequent analysis and data processing, decision making and reasoning. The goal of this paper is to review and categorize various techniques, tools, and libraries for extracting information from unstructured web content (platforms, sites, systems), and to develop a JavaScript application that crawls and extracts data from dynamic web pages without the need to browse, read and search the page content. The paper presents an implementation of a particular JavaScript web scraper that retrieves a list of news headlines from the official European Union Agriculture and Rural Development website without the need for the content of the document to be read by users. The web scraper is configured to extract the searched content directly from the source HTML code of the document, regardless of whether the information is explicit or implicit. It also searches all pages related to the document. Finally exports data in a proper format. The benefits of such a tool for extracting web content from source code are related to saving time, manual labour and means of generating quality content in the biotech and agriculture industry.

1 Introduction

Nowadays, data is growing at breakneck speed. They can be collected from various sources – text documents, social networks, image galleries, geographic data, maps, networks, tables, web services, databases, etc. In web information systems, user-system interaction takes place through the graphical user interface of the system. This interaction is possible using reference requests through the menus, fields, controls, elements and tools of the system's graphical user interface.

The extraction of information from an information platform, site or system is possible if the information is structured, arranged and annotated in a way convenient for subsequent analysis and data processing, decision-making and justification. But there is another way to extract unstructured web content and store it in a structured format – this technique is called *web scraping*.

The goal of this paper is to introduce and categorize various techniques, tools and libraries for extracting information from unstructured web content, as well as to develop a JavaScript application that crawls and scrapes data from dynamic web pages (in our case – the official European Union Agriculture and Rural Development website), without having to browse, read and search the contents of the pages.

2 Materials and Methods

2.1. Web scraping techniques

Web scraping is widely used in practice to obtain data and its subsequent analysis without the need to write additional program code. Software-built scrapers can extract data from dynamic sites that change their content over time. And since in this case, it is a programmed web scraper, it can also have many other functionalities such as searching for content that is hidden in the web page, i.e. one that becomes visible when certain conditions occur, such as an activated URL link, a button press, a mouse click or hold, etc. However, this is not a problem for web scraping as it extracts the content directly from the HTML source code of the document, whether the information is explicit or hidden. The web scraper can be set to extract information even after scrolling the entire page content. Since there is information that is loaded when the user has reached the end of the web page, the script can adjust the wait time, scrolling method and time to get the required data.

While web scraping is the extraction of data from one or more websites, web crawling is the crawling (searching) of web content to find or discover URLs or links within it. Linked pages and page link tracking (references) could also be crawled using web scraping to crawl the site in depth. Besides all text content (using tags like div, span, p, text, ul, etc. and values in their attributes) and available hyperlinks, the web scraper can extract images, videos and files in various

* Corresponding author: m_jekova@uft-plovdiv.bg

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

multimedia formats (ppt, pdf, jpg, xls, doc). Once the required content is obtained, it can be exported in various formats – text, CSV, excel, html, etc. In data mining projects, the two approaches – web crawling and scraping – are successfully combined.

The difference between web scraping and web crawling is that web scraping is used to extract information, while web crawling is used to index content. In web scraping, it is unnecessary to visit all web pages, while in web crawling, each page is crawled up to the last line. Web scraping is done on both small and large scale, while web crawling is mostly used on large-scale projects. The scope is also different, web scraping is used in machine learning, marketing, and commerce, and web crawling is used in search engines.

The web crawler can traverse the information on the web page by itself, and the search engine is inseparable from the web crawler. The most important role of the web crawler is to crawl the big data of the Internet, find effective information, and store the needed information data in the local database [1].

2.2. Web scraping tools

The data and information on websites are very often hidden behind the complex, hierarchical structure of the documents that make them up. The information in them is dynamic and change requires continuous work on the part of developers of information retrieval applications as well.

There are four main approaches to retrieving information using web tools: off-the-shelf (external) scrapers, built-in web robots, application programming interfaces (APIs), and ready-made datasets. There are other solutions for non-professionals, the so-called web content extractors (environments, editors, extensions) with which specific content can be obtained without writing code (coding).

2.2.1 Off-the-shelf web scrapers

Some of the better-known off-the-shelf web scraping tools are Bright Data, Smartproxy, Oxylabs, Octoparse, ParseHub, etc. Two of the tools are reviewed, the others have similar functions and work on the same principle.

Octoparse is a tool for extracting texts, videos and images from websites [2] that interacts with each element on the page and with it the user can design his data extraction workflow. This makes Octoparse a personal assistant in forming a specific user task. Contains over 100 free, pre-built templates to pull information from Amazon, Booking, Google Maps, Yelp, eBay, etc. Users select a template and enter keywords or URLs for their chosen data fields on a site. They can reformat the received data, create a task list, set up a base/parent process, and speed up the extraction process. [2]. Octoparse is a paid tool, but it also has a free version. It supports a cloud service and offers a high data browsing speed, as well as various formats for exporting the received data.

ParseHub is a web scraper that collects data from websites built on AJAX technology, JavaScript, etc. An advanced machine learning technology turns the content of web documents into data [2]. ParseHub handles searching web forms, opening drop-down menus, entering websites, clicking on maps, and managing sites with infinite scrolling, tabs, and pop-ups to retrieve searched data. It is a fast and easy-to-start tool that handles the toughest navigations and extractions and requires no coding. It supports scheduled launch, works with conditions and regular expressions, selectors and attributes, extracts data from tables, maps and images, etc. Using a set of keywords, ParseHub automatically searches each document and its related pages. It exports the data in Excel or CSV [3].

2.2.2 Built-in (internal) web scrapers

Embedded web scrapers are created by programmers using programming languages such as Python, JavaScript, and Ruby and their built-in libraries, e.g. Beautiful Soup, Scrapy and Selenium for Python, Puppeteer and Nightmare for JavaScript, and Nokogiri for Ruby. Python is the tool used to create professional web scrapers – programs for searching and extracting data from web documents/sites/platforms.

Beautiful Soup is a Python web scraping library that extracts data from HTML and XML files. It provides methods for navigating, searching, and modifying a parse tree: a set of tools for traversing a document and retrieving the information sought [4]. Beautiful Soup parses and traverses the DOM tree of web pages and can find all hyperlinks in them or all links whose URL matches "sample.com" or return text from an input element of class = "second" etc. It works fast and saves developers time and money.

Puppeteer is a Node.js library that can be used for web scanning, automation and testing purposes. It was originally developed by Google to help developers automate their workflows. Puppeteer can now be used to take screenshots, create PDF files, navigate pages, click buttons and links, fill out forms, and can also be used for testing purposes [5].

Nokogiri makes working with XML and HTML from Ruby easy and painless. It provides a sensible, easy-to-understand API for reading, writing, modifying and querying documents [6].

2.2.3 Web content extractors

Web Scraper is a free data scraper with an easy-to-use interface [7]. It doesn't require installation, knowledge or experience with programming languages such as PHP, Python, etc. It extracts data from dynamic sites built with JavaScript and with multiple levels of hierarchy, successfully navigating between individual levels – categories and subcategories, automated pagination and product pages. Web Scraper has a modular structure of selectors that give instructions on how to crawl the target site and what data to retrieve. It offers the following features – extracts diverse content

(text, images, URLs, etc.), deletes data from multiple pages, extracts data from dynamic pages (JavaScrit + AJAX), infinite scrolling, stores the data and export to Excel, CSV, XLSX and JSON [7]. The tool handles the task of extracting product data from product pages in online stores (such as prices, description, images, etc.), generating up-to-date information (about potential customers – email, phone, address), crawling the content of news sites, forums, networks, monitor and analyse user sentiment and reactions, monitor and collect information about prices, availability, customers and suppliers of competitive forms, etc. [7].

Instant Data Scraper is an automated tool for extracting data from web pages. It uses heuristic AI analysis of HTML document structure to discover data to extract [8]. Features of Instant Data Scraper – detect data with AI, detect dynamic data, customize delay and maximum waiting time and desired crawl speed, support pagination, auto-navigation via links and buttons, support infinite scrolling, export received data to electronic tables in Excel or CSV format [8].

3 Web scraping implementation using JavaScript

The paper presents a web scraper developed in JavaScript that retrieves a list of news headlines from the official website of the European Union Agriculture and Rural Development without the need for the content of the document to be read by users.

The following approach was used to create our web scraper:

Step 1. Create an appropriate DOM tree.

Step 2. Retrieve the requested data.

Step 3. Visualization of the results.

The resulting data set is subject to organization, cleaning, ordering, and normalization, applying algorithms, building hypotheses, making decisions, drawing conclusions and obtaining a final result.

The interface of the source web document (Fig. 1) https://agriculture.ec.europa.eu/news_en, presents news titles in tabular form, with descriptions for title, date, category (news, announcement, article) and download and review link. In it, block (div) elements with classes page-content, news-list, news-date and news-title contain the content of the pages, a list of news, dates and titles, respectively.

When the web scraper is started, the functions getScrapeData() and scrapeData() are called sequentially. The function getScrapeData() creates the DOM tree corresponding to the design of the document and then accumulates all elements of the news-list class in the news variable. Using a loop, for each news from news-list, its name (title) and date (date) are taken and added to the array newsData (Fig. 2).

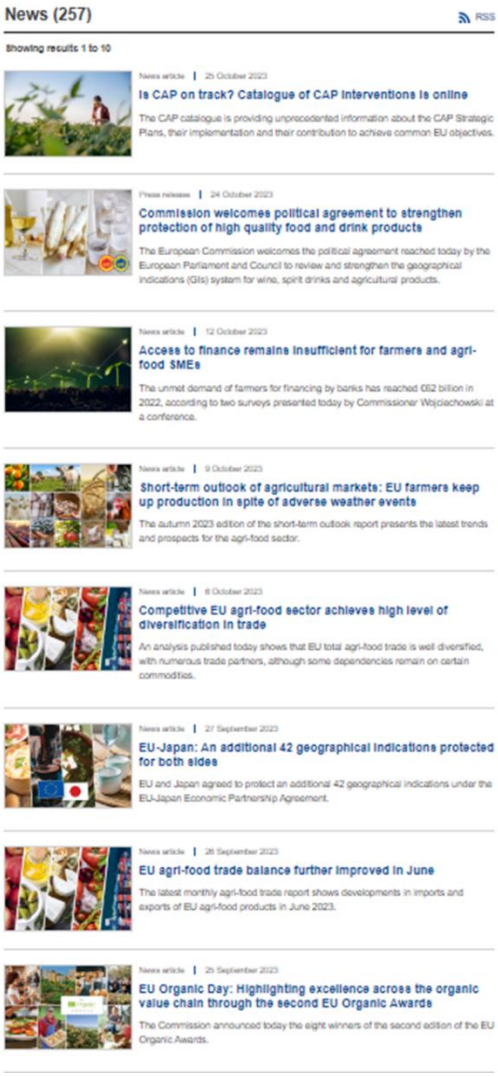

Fig. 1. News page – https://agriculture.ec.europa.eu/news_en

```
async function scrapeData() {
    const proxyUrl =
'https://api.allorigins.win/raw?url=';
    const url = 'http://
agriculture.ec.europa.eu/news /';
    const response = await fetch(proxyUrl + url);
    const html = await response.text();
    const parser = new DOMParser();
    const doc = parser.parseFromString(html,
'text/html');
    const news = doc.querySelectorAll('.news-list);
    const newsData = [];
    news.forEach(article => {
        const title = news.querySelector(".news-
title").innerText;
        const date = news.querySelector(".news-
date").innerText;
        newsData.push({title, date });
    });
    return newsData;
}
```

Fig. 2. Programming code of scrapeData()

Depending on the result that getScrapeData() returned, the scrapeData() function starts adding block elements (div) for each of the found news in the news-list (Fig. 3). In the front-end part of the scraper, HTML code is added, which adds paragraphs in which it puts the date and title of the news found.

```
Async function getScrapeData() {
    const newsList =
document.getElementsByClassName("newsList")[0];
    const newsData = await scrapeData();
    newsData.forEach(news => {
        const div = document.createElement('div');
        div.classList.add("news");
        div.innerHTML = `
            <p class="news-title">${news.title}</p>
            <p class="news-date"> ${news.date}</p>`;
        newsList.appendChild(div);
    }); }

function generateTable(btn) {
    let table = document.getElementById("table");
    let row = document.getElementById("row").value;
    let column =
document.getElementById("column").value;
    let learnSection =
document.querySelector('.learn-section');

    if(table.innerHTML == ""){
        learnSection.style.height = 100 + 'vh';
        btn.innerText = "Clear Table";
        table.setAttribute("border", "1");
        for (let I = 0; I < row; i++) {
            let tr = document.createElement("tr");
            for (let j = 0; j < column; j++) {
                let td =
document.createElement("td");
                td.innerText =
Math.floor(Math.random() * 10 + i);
                tr.appendChild(td);
            }
            table.appendChild(tr);
        }
    }
    else {
        table.innerHTML = '';
        learnSection.style.height = '';
        btn.innerText = "Generate Table";
    }
}
```

Fig. 3. Functions getScrapeData() and generateTable()

4 Results

The result of extraction is a list of headlines (Fig. 4).

Fig. 4. The result of Web scraping extraction

This example shows the possibility of crawling an HTML document and retrieving a list of news headlines from the official website of the European Union Agriculture and Rural Development without the need for the document content to be read by users.

To create the web scraper, a hybrid approach was used according to the described methodology. Initially, the corresponding DOM tree was created for the target web document, after which the searched data was extracted from it. Finally, the process ends with a visualization of the obtained results. The resulting data set is subject to organising, cleaning, ordering, and normalization, applying algorithms, building hypotheses, making decisions, drawing conclusions and obtaining a final result.

In addition to generating content with web scraping, data can also be collected about the mood and reactions of users, their emotions, likes, and views – factors that form the so-called content model based on user reactions.

5 Conclusion

The technique for crawling web documents and creating the corresponding DOM tree increases the time cost proportionally, depending on the hierarchical structure of the content. At the same time, scraping data from the web reduces the manual work of extracting and organizing information and provides an easy and fast way to collect data, convert it to a specific format and store it in a repository.

With web scraping, analysis and manipulation of the obtained data and subsequent statistical processing, strategies formation, as well as decision-making and their justification are possible. It also offers the

possibility of collecting and storing the extracted data and structuring them in large databases.

The benefits of such an automated program for extracting web content from source code are related to saving time, manual labour and means of generating quality content in the biotech and agriculture industry. Besides the listed advantages, the presented technique also has disadvantages. One of them is the need for additional annotation of the extracted structured data.

References

1. L. Yu, Y. Li, Q. Zeng, Y. Sun, Y. Bian & W. He. Summary of web crawler technology research. IOP Publishing Ltd. J. Phys.: Conf. Ser. **1449**, 012036 (2020)
2. Octoparse, URL: https://www.octoparse.com
3. ParseHub, URL: https://www.parsehub.com/
4. Beautiful Soap documentation, https://www.crummy.com/software/BeautifulSoup/
5. Puppeteer, https://www.headspin.io/blog/testing-with-puppeteer-a-complete-guide
6. Nokogiri documentation, https://nokogiri.org/
7. Webscraper, URL: https://webscraper.io/
8. Instant data scraper documentation: https://chrome.google.com/webstore/detail/instant-data-scraper
9. N.K. Rao, B. Naseeba, N.P. Challa, S. Chakrvarthi. Web scraping (IMDB) using Python, Telematique. **21**, 235 (2022)
10. H. Nigam, P. Biswas. Web Scraping: From Tools to Related Legislation and Implementation Using Python. In *Innovative data communication technologies and application*. Lecture Notes on Data Eng. Commun. Technol. (vol 59. Springer, Singapore, 2021)