Bogdan-Ștefan POSEDARU, PhD Candidate
bogdanposedaru@gmail.com
Bucharest University of Economic Studies, Romania

Lorena BĂTĂGAN, PhD (corresponding author)
lorena.pocatilu@ie.ase.ro
Bucharest University of Economic Studies, Romania

Răzvan BOLOGA, PhD
razvanbologa@ase.ro
Bucharest University of Economic Studies, Romania

Dimitrie-Daniel PLĂCINTĂ, PhD Candidate
daniel.placinta@csie.ase.ro
Bucharest University of Economic Studies, Romania

Corina-Marina MIREA, PhD Candidate
mireacorina17@stud.ase.ro
Bucharest University of Economic Studies, Romania

Software Architecture for Improving Scraping Systems Using Artificial Intelligence

Abstract. *This paper explores diverse innovative Web scraping techniques. It initially introduces a machine learning (ML) approach capable of adapting to changing HTML structures for continuous scraping. Following this, an automated method is detailed, specifically designed to extract data from dense web pages using subject detection and node density techniques. Lastly, the article covers a computer vision-based scraping strategy that employs object recognition and OCR to visually analyse and interpret websites. These methods seek to increase online scraping efficiency and reduce the reliance on HTML structure. The present study proposes an autonomous Web scraper system that integrates Web scraping, Artificial Intelligence (AI), and Natural Language Processing (NLP) techniques, leveraging ChatGPT's natural language understanding capabilities to achieve the desired results. The recommended JavaScript program uses NLP techniques to produce webpage lists for examination and lets users query data in natural language. By utilising cutting-edge AI and ML approaches, it promises to increase Web scraping's usability and effectiveness. The article details upcoming work, such as HTML clean-up and DOM parsing advancements, and examines the benefits of the suggested approach over the currently available tools.*

Keywords: *ChatGPT, OpenAI, Web scraper, Large Language Model (LLM), intelligent agent.*

JEL Classification: A22, A23, O33.

DOI: 10.24818/18423264/58.3.24.09

1. Introduction

Various definitions of Web scraping can be found in the literature. Considering the common elements of these definitions, Web scraping can be defined as an automated process of extracting data from web pages and making them available in a structured format. Web scraping approaches and tools were developed as an efficient solution to collect large volumes of data from the web, to overcome the limitations of manual data collection. The user must specify the kind of information that should be collected, the pages or websites to be accessed, and the output data format. Typical output options include text formats such as CSV, XML, and JSON, or storage of the data in a database.

Applications of Web scraping include indexing web pages by search engines, collecting and tracking price information from e-shops, market research, sentiment analysis of social media posts, etc. (Matta, 2020)

Collecting data through Web scraping can be subject to legal limitations as well as ethical aspects (Mitchell, 2018). Legal limitations are mostly related to copyrighted content, as well as to accessing web server resources abusively, thus causing overloads or even damage; not complying with the applicable terms of service or accessing without the proper credentials. Ethical aspects are usually related to the collection of personal information without user consent and/or compromising privacy, revealing confidential information or trade secrets, and reducing the value of the website owner's company by creating an alternative access channel to the same data (Krotov, 2018). To prevent automated data extraction, website owners may employ specific techniques (Gheorghe, Mihai & Dardala, 2018).

The traditional Web scraping approach is based on parsing the HTML and CSS coding of web pages, to find and extract the required data using special programmes or scripts (Kovacs, 2015). The scraper is usually required to also identify and follow relevant hyperlinks of pages to be analysed (Web crawling).

Scraper software is available as programming libraries and modules or as interactive point-and-click tools. The former requires the user to write code for creating the scraper programme, while the latter does not require programming, but might be less adaptable.

Limitations and challenges of traditional scraper software include:

- The website layouts may be updated periodically, thus requiring the update of the scraper configuration;
- The need to adapt the scraper configuration to the individual website layouts, especially when many different websites are targeted;
- The collected data should undergo a thorough validation and cleaning, to correct any inconsistencies or errors (Mitchell, 2018).

Recent developments have employed AI/ML techniques to overcome these limitations. The recent technological advancements represented by the advent of the LLMs may present important advantages over the previous traditional scraping approaches (Moaiad, 2021), and their analysis is very important to evaluated and

determined possible ways to improve Web scraping tasks. The present paper analyses the existing GPT tools and presents a solution that aims to overcome the limitations of the mentioned software.

The article explores advancements in Web scraping by integrating AI, specifically focusing on the use of GPT models. It starts with the by presenting the foundational aspects and challenges of conventional Web scraping techniques, but in the same time is highlighting the need for more efficient and adaptable solutions.

Subsequently, the paper reviews state-of-the-art developments and presents a detailed examination of existing GPT-powered scraping tools, such as Scrapeghost and AutoGPT, discussing their capabilities, advantages, and drawbacks.

Building on this analysis, the authors propose a novel autonomous scraper system designed to be user-friendly and highly efficient, utilising advanced AI and ML algorithms. This new system aims to automate the Web scraping process further, reducing the need for technical expertise and manual intervention.

The article presents comparative results showing the proposed system's effectiveness against existing tools, detailing the technological advancements and algorithmic strategies employed. It concludes with a discussion on potential challenges and future directions for research in AI-powered Web scraping, emphasising the impact and possibilities of this evolving technology in simplifying data extraction and enhancing accessibility.

The paper aims to present a new approach to Web scraping by integrating advanced of AI and ML technologies, specifically using GPT models. The primary objectives are as follows.

- To address existing limitations: By developing an AI-powered scraping system that overcomes the constraints of traditional methods, such as the need for constant updates and technical expertise;
- To enhance efficiency and accessibility: By proposing a system that automates the scraping process to a greater extent, making it more user-friendly and accessible to individuals without technical backgrounds;
- To demonstrate the application of GPT models: Showcasing how natural language processing capabilities of GPT models can be leveraged to improve the efficiency and effectiveness of Web scraping tools;
- To push technological boundaries: The proposed solution seeks to advance the state-of-the-art in Web scraping technology, setting a new standard for what is possible in automated data extraction.

The paper’s intent is to contribute to both the academic field and practical applications by providing a robust, efficient, and accessible tool that utilises the latest advancements in AI.

2. State-of-the-Art

Recent research (Carle, 2020) has explored the development of a robust Web scraping algorithm capable of continuously scraping websites with repetitive HTML structures, even in such cases where the structure of the website was altered. The

author describes the flaws of the traditional Web scraping approach, which in numerous cases proves to be an unfeasible solution, mainly due to the web pages' constant structure updates, and presents a solution to this issue based on a ML algorithm that identifies patterns within the HTML structure. According to the author, the limitation of such an algorithm is that it requires the presence of repetitive structures on the website to extract sufficient data for the model to use and to make the identification of these areas easier.

Earlier papers, such as (Petprasit, 2015) present the implementation of a fully automated scraping solution, which extracts relevant information from data-intensive pages, such as e-commerce websites. The author describes two algorithms that aid the process of relevant data extraction: Subject detection, based on assigning weights to specific HTML tags, and also Node density which is used to find the data-rich region in a data-intensive web page (i.e., for an e-commerce page, the data-rich region is represented by the node in the DOM tree that includes the product details). Although the experimental results are promising, the limitation of this proposed approach is that it is dependent on the subject detection technique and the link density in the data-rich area.

The study by (Dallmeier, 2021) provides a scientific effort that suggests a Web scraping strategy purely based on the visual representation of a particular website in hopes of decreasing the dependence on the structure and existence of its underlying HTML or CSS and moving toward comprehending websites in a more human-like manner. The suggested approach relies on specific Computer Vision components, such as OCR or object detection, and uses a supervised learning approach, by training the model by manually labelling specific parts of Internet Forums pages, such as author, date, subject, etc. The main advantage of this approach is that the scraper will not rely on the HTML structure, which often is updated.

Demir (2023) offers useful responses to archaeologists who analyse the three ancient cites by extracting the images through Web scraping techniques (Selenium library, Python) based on keywords as input parameters. The photos collected were further processed using the Amazon Rekognition Cloud Service cloud-based tool 'to identify the most items on the investigated cultural heritage site' using the power of deep learning features. In this manner, the scientist identified that Perge has the most columns, which is already confirmed on the ground. The study highlights the potential for Web scraping methodology, but at the same time, some limitations were observed for the recognition of buildings and incorrect location capturing from the Internet.

Karthikeyan (2019) proposes a Web scraping model for content extraction and text classification. The content is extracted from HTML files with BeautifulSoup, a Python library designed for this purpose, then a keyword matching score is calculated (KeywordProcessor library) and, based on the score, the samples are categorised. The classical text pre-processing mechanism with tokenization, stemming, and bag-of-words is applied before "selecting the best features using Logistic Regression – Recursive Feature Elimination (LR-RFE), even the LR-RFE is expensive consuming and before feeding the samples into Back-Propagation

Neural Networks (BPNN)". The BPNN model performs better than the Random Forest and Support Vector Machines models for the proposed text classification approach.

Kaur highlights in his study that the Web scraping of the news headlines from reddit.com is realised with Python Reddit API Wrapper (PRAW), where JSON responses are further processed using ML algorithms for sentiment analysis (Kaur, 2022).

Qingli Niu proposed a Web scraper tool designed to extract the required information from newspapers, websites, blogs, and images by transforming the web links into visual blocks (web page parts), detecting the structure of web pages, loading the website into the tool, extracting the data with Scrapy library and converting to JSON format with JSONify library, inserting all the data in the appropriate tags, and finally transforming the data into PDF, spreadsheet, or CSV files (Niu, 2023). The limitation of this approach is that, according to the author, it is only suitable for newspapers and blogs, and it is not a general-purpose scraping solution.

In conclusion, the scraping tools that were studied present significant drawbacks, such as the need for technical proficiency or limited applicability of the solution. As such, the goal of our research is to create a universal scraping tool, that may be used by non-technical persons, by leveraging recent advancements in the AI industry, such as LLMs. By developing such a solution, our hope is to advance the online scraping industry towards increased usability and accessibility, democratising access to data extraction.

3. Available GPT-Powered Scraper Systems

The newest developments in AI, that are applied in many domains of our life (Năstasă, 2023), and NLP are used by GPT-powered scraper systems, a new class of online scraping tools, to automate the data extraction process from websites. These tools analyse and comprehend the structure and content of web pages using GPT language models, making it easier and quicker to gather the data required for many applications, including data analysis, ML, and business intelligence. The creation of GPT-powered scraper systems has advanced in recent years, with initiatives like Scrapeghost and AutoGPT becoming more and more popular among programmers and data scientists.

When compared to traditional scraping techniques, leveraging the LLMs' natural language processing capabilities offers significant advantages for the development and maintenance of advanced scraping systems. For instance, it may accelerate the development of solutions by decreasing or eliminating the need for technical knowledge on the part of the user.

3.1 Scrapeghost

3.1.1 Description

With the advent of LLM, it is now possible to use a model to automatically extract data from a certain website without knowing the selectors in advance. One such solution is Scrapeghost, a Python library that leverages ChatGPT's power to extract the requested data (Scrapeghost, 2024). One of Scrapeghost's primary features is the Python-based schema definition. As a result, users can specify the desired level of detail for the shape of the extracted data as any Python object. The preprocessing options provided by Scrapeghost also include HTML cleaning, support for CSS and XPath selectors, and automated prompt splitting. To guarantee that the response is correct and valid, Scrapeghost also provides several postprocessing capabilities, including JSON and schema validation and a hallucination check, which ensures that the returned data does indeed exist on the given page. In addition, Scrapeghost includes cost control features that monitor token usage and provide automatic fallbacks. This feature enables users to specify a budget and stop the scraper if the budget is exceeded.

3.1.2 Limitations

One of the main limitations of Scrapeghost is the need for human input and a certain degree of technical proficiency, as to manually determine the CSS selector for the data container. This can be a difficult task, especially for complex web pages that have multiple data containers, nested containers, or dynamically changing containers. Inaccurate or improper selection of the CSS selector can result in incomplete or incorrect data extraction, rendering the app ineffective. This requirement may also pose a challenge to users with limited technical skills or experience with Web scraping tools. Another limitation of Scrapeghost is that the users need to specify a valid schema for the data to be extracted, which requires technical skills and can prove to be a tedious process. Finally, the fact that Scrapeghost can only process 4096 tokens at a time may make it more difficult for it to scrape websites with relatively large HTML structures. To scrape data from websites with many pages or those with complicated structure that need processing more tokens, the 4096-token limit can reduce the utility of Scrapeghost. Users may have to use many instances of Scrapeghost in these situations, or they may choose to use alternative Web scraping applications with higher token restrictions.

3.2 AutoGPT

3.2.1 Description

AutoGPT is an open-source solution that allows one to create and run intelligent agents (AutoGPT, 2024). It comes with many features, including Internet access for

searches and information gathering, long-term and short-term memory management, GPT-4 instances for text generation, access to popular websites and platforms, file storage and summarisation with GPT-3.5, and extensibility with plugins. AutoGPT can be a powerful tool for Web scraping, especially when coupled with other AI-assisted tools. Its ability to autonomously achieve Web scraping goals, coupled with its many features, makes it a promising development in the field. However, its limitations must also be carefully considered, and its use must be tempered by a sound understanding of applicable laws, regulations, and ethical standards. Overall, AutoGPT represents a promising step forward in the field of AI-assisted Web scraping and holds much promise for future developments in this area.

3.2.2 Limitations

AutoGPT presents several limitations that are important to consider. Firstly, AutoGPT may be quite expensive to run, so monitoring and managing one's own token usage and the associated costs is critical. Also, as an autonomous experiment, AutoGPT may generate content or take actions that are not in line with real-world business practices or legal requirements, so it is essential to ensure that any actions or decisions made based on the output of this software comply with all applicable laws, regulations, and ethical standards. AutoGPT was not created specifically for Web scraping, even though it has demonstrated promising results in automating a variety of tasks by spawning relevant agents. Because of that, when used for Web scraping, it insists on insignificant particularities like code improvement through excessive error handling. The development of the scraper modules may become more difficult and takes longer as a result. Software developers must carefully weigh the advantages of using GPT-powered automation against the expense of directing AutoGPT through the required steps to produce the desired outcome. Since its focus is on general-purpose language generation, it is understandable that it may not perform as well for specific use cases, such as Web scraping.

4. Proposed Solution – Autonomous Scraper

Through the analysis of the methods described methods and the identification of their advantages and disadvantages, our study seeks to develop a universal scraping solution that can be used by any individual, independent of their technical proficiency. To extract data from the web without requiring human intervention, for example by manually determining and providing the CSS selectors for different data found across the page, users will converse with the scraper system using natural language prompts.

4.1 Description

Our objectives include developing a JavaScript application that aims to offer a simple and effective way to scrape data from the Internet. The programme allows users to make queries using natural language while employing natural language processing algorithms to finally return a JSON file containing the scraped data.

Modern natural language processing models will be used by the app to extract the critical components of the user's query, such as the data to be scraped, to automatically determine the appropriate CSS selectors that will be used to extract the relevant information from the given web page. By correctly determining the CSS selectors, our proposed solution builds and then executes a scraper application, which extracts the required data, finally saving the data locally in JSON file format. Overall, the suggested JavaScript application is an original approach to Web scraping that makes use of advanced AI and ML algorithms, such as natural language processing. The authors aim to develop a tool that makes it easy for users to extract relevant information by integrating the most recent developments in ML and Web scraping methods. This app has the potential to be a vital resource for many academics and businesses given the rising demand for Web scraping across a wide range of fields.

4.2 Algorithm Description – Steps

The proposed algorithm consists of the following steps:

1. Provide the URL and a succinct natural language description of the data to be extracted;
2. Fetching the HTML structure of the web page available at the provided URL;
3. Clean up and reduce the size of the HTML structure, by removing unnecessary elements, such as scripts and style tags;
4. Load the HTML structure into ChatGPT by splitting it into multiple chunks (Figure 1);

```
Act like a document/text loader until you load and remember content of the next text/s or document/s.
There might be multiple files, each file is marked by name in the format ### DOCUMENT NAME.
I will send you them by chunks. Each chunk start will be noted as [START CHUNK X/TOTAL], and end of this
chunk will be noted as [END CHUNK X/TOTAL], where x is number of current chunk and TOTAL is number of
all chunks I will send you.
I will send you multiple messages with chunks, for each message just reply
OK: [CHUNK X/TOTAL], don't reply anything else, don't explain the text!
Let's begin: [START CHUNK 1/7]
```

Figure 1. ChatGPT Prompt for splitting the HTML into multiple chunks
Source: Authors' processing.

5. Ask ChatGPT to identify the appropriate selectors for the data identified within the provided HTML structure (Figure 2);

```
I want you to analyze the HTML structure that I sent you earlier and based on the identified patterns,
determine the selectors for various characteristics regarding ${prompt). It is mandatory to include the "parent
selector" which selects all ${prompt} elements from the page. It is of utmost importance to extract as much
data as possible regarding each element. Your response will consist of only an object and absolutely nothing
else. The object's keys will be the characteristics for the ${prompt} and the values will represent the CSS
selectors for extracting these characteristics.
```

Figure 2. ChatGPT Prompt for extracting CSS selectors
Source: Authors' processing.

6. Ask ChatGPT to write a scraper based on the identified CSS selectors (Figure 3);

```
Based on your HTML structure analysis, write a NodeJS function that will take the HTML string as a
parameter and will return exactly this JSON, using CSS selectors
```

Figure 3. ChatGPT Prompt to adapt a scraper based on the determined selectors
Source: Authors' processing.

7. Execute the scraper, persisting the extracted data into local JSON files.

4.3 Advantages. Comparison with pre-existent tools

The proposed GPT-powered method for autonomous Web scraping offers several advantages over Scrapeghost and AutoGPT that outweigh their limits and disadvantages. First, the suggested approach, unlike Scrapeghost, does not require the user to be aware of the selector of the data container element. Instead, it needs a natural language prompt, making it easier for users to operate and more intuitive. This streamlines the process and makes it more accessible to a larger variety of users by eliminating the need for manual entry of technical details.

In addition, unlike AutoGPT, the suggested solution does not have an excessive emphasis on unimportant particularities like code optimisation and error handling. For Web scraping, these capabilities can be cumbersome and unnecessary, although they may be useful for general-purpose AI applications. When complex HTML structures are provided, the suggested solution will prioritise data extraction rather than halting the operation or going into an endless loop, as AutoGPT occasionally does, by leveraging HTML cleanup tools and DOM analysis. Therefore, an improved and more reliable user experience will result.

In contrast to AutoGPT, the suggested solution is independent of GPT-4, the latest version of OpenAI's LLM being more expensive and less widely available than GPT-3.5. However, the suggested solution can be applied to the current GPT-3.5 architecture and is not constrained by the cost or availability of GPT-4. This makes it a more viable and available Web scraping method.

Finally, unlike AutoGPT in its experimental state, the suggested solution will not experience frequent bugs. The suggested solution will undergo extensive testing and Web scraping optimisation, ensuring a higher level of reliability and reducing the possibility of unexpected behaviour or outcomes.

4.4 Results and Contributions

To demonstrate the capabilities of the proposed solution, several empirical results are presented. As such, the presented use cases cover various data structures across different website domains, for example, extracting data regarding multiple products displayed on an e-Commerce platform, as well as extracting data about a specific company from their official website.

Using the proposed solution to extract product data from an e-Commerce website requires user intervention only for providing a succinct natural language

description of the data to be extracted and the URL for the e-Commerce platform (Figure 4).

```
Based on your HTML structure analysis, write a NodeJS function that will take the HTML string as a
parameter and will return exactly this JSON, using CSS selectors
```

Figure 4. Parameters for running the scraper
Source: Authors' processing.

The autonomous solution employs the described algorithm to retrieve and analyse the HTML, with the purpose of identifying patterns and extracting the CSS selectors (Figure 5) that are then used to autonomously develop the actual scraper code (Figure 6).

```
{
  prompt: 'books',
  selectors: '{\n' +
    "parent selector": "article.product_pod",\n'
    "title": "h3 a",\n' +
    "image url": "div.image_container img",\n' +
    "price": "p.price_color",\n' +
    "availability": "p.availability",\n' +
    "rating": "p.star-rating"\n'
  '}'
}
```

Figure 5. CSS selectors determined by the proposed solution
Source: Authors' processing.

```
const selectors = {
    "parent selector": "article.product_pod",
    "title": "h3 a",
    ...
};

(async () => {
    const browser = await puppeteer.launch();
    const page = await browser.newPage();

    await page.setUserAgent('Mozilla/5.0 (Windows NT 10.0; Win64; x64)');

    await page.goto('https://books.toscrape.com/');

    const data = await page.evaluate((selectors) => {
        const products = document.querySelectorAll(selectors["parent selector"]);

        return Array.from(products, product => ({
            title: product.querySelector(selectors["title"])?.textContent.trim() || '',
            ...
        }));
    }, selectors);
...
})();
}
```

Figure 6. Generated scraper
Source: Authors' processing.

Finally, the scraper is executed, and the following data is generated and persisted in local JSON files (Figure 7).


```
{
 "name": "A Light in the ...",
 "url": "catalogue/a-light-in-the-attic_1000/index.html",
 "image": "media/cache/2c/da/2cdad67c44b002e7ead0cc35693c0e8b.jpg",
 "price": "£51.77",
 "reducedPrice": null,
 "starRating": "Three"
},
{
 "name": "Tipping the Velvet",
 "url": "catalogue/tipping-the-velvet_999/index.html",
 "image": "media/cache/26/0c/260c6ae16bce31c8f8c95daddd9f4a1c.jpg",
 "price": "£53.74",
 "reducedPrice": null,
 "starRating": "One"
},
{
 "name": "Soumission",
 "url": "catalogue/soumission_998/index.html",
 "image": "media/cache/3e/ef/3eef99c9d9adef34639f510662022830.jpg",
 "price": "£50.10",
 "reducedPrice": null,
 "starRating": "One"
},
...
```


Figure 7. Extracted data
Source: Authors' processing.

Attempting to extract non-repetitive data, for example the contact methods of a specific company from their official website, provides the expected results (Figure 8).

```
autoScrape({
  prompt: "company name and contact methods",
  url: "https://www.one.ro/en/get-in-touch/"
})
```

Figure 8. Parameters for running the scraper
Source: Authors' processing.

The proposed solution correctly identifies the selectors, builds the scraper (Figure 9) and executes it, obtaining and persisting the required information (Figure 10).

```
const data = await page.evaluate(() => {
  const address =
    document.querySelector('footer [itemprop="address"] a')?.innerText || "";
  const telephone =
    document.querySelector('footer [itemprop="telephone"]')?.innerText || "";
  const email =
    document.querySelector('footer [itemprop="email"]')?.innerText || "";
  const facebook = document.querySelector("footer .facebook-logo")
    ? document.querySelector("footer .facebook-logo").getAttribute("href")
    : "";
  // ...

  return {
    Address: address,
    Telephone: telephone,
    Email: email,
    Facebook: facebook,
    // ...
  };
});
```

Figure 9. Main modules of the generated scraper
Source: Authors' processing.

```
{
 "Address": "One Tower, Calea Floreasca 165, 16th Floor, 1st District, Bucharest",
 "Telephone": "+40 31 225 1000",
 "Email": "office@one.ro",
 "Facebook": "https://www.facebook.com/OneUnitedProperties",
 "Instagram": "https://www.instagram.com/one_united_properties/",
 "LinkedIn": "https://www.linkedin.com/company/oneunitedproperties/",
 "Google": "https://g.page/OneUnitedProperties/review",
 "YouTube": "https://www.youtube.com/c/oneunitedproperties/",
 "TikTok": "https://www.tiktok.com/@one_united_properties"
}
```


Figure 10. Extracted data

Source: Authors' processing.

4.5 Comparison with pre-existent tools

For the above-presented use cases, our proposed solution provides satisfactory results, compared to the output provided by Scrapeghost and AutoGPT, which we will analyse in this section.

For the repetitive data structures, our proposed solution offers similar results to Scrapeghost and AutoGPT, identifying the complete list of elements and extracting multiple product characteristics. Seeing that a pure comparison with Scrapeghost is not possible, as Scrapeghost requires the user to provide the schema for the data, we chose to run our autonomous solution first, and based on the identified and extracted product characteristics, construct the schema for Scrapeghost and compare the extracted values.

As such, we notice that the data is very similar, but with some minor, but potentially impactful differences (Figure 11).


```
// Extracted by the autonomous solution
{ "title": "A Light in the ...",
  "imageUrl": "media/cache/2c/da/2cdad67c44b002e7ead0cc35693c0e8b.jpg",
  "price": 51.77,
  "availability": "In stock",
  "rating": "Three" }
// Extracted by Scrapeghost
{ "title": "A Light in the Attic",
  "imageUrl":
"https://books.toscrape.com/media/cache/2c/da/2cdad67c44b002e7ead0cc35693c0e8b.jpg",
  "price": 51.77,
  "availability": "In stock",
  "rating": "Three" }
// Extracted by AutoGPT
{ "title": "A Light in the Attic",
  "price": "\u00a351.77",
  "availability": "In stock",
  "rating": "Three" }
```


Figure 11. Comparison of extracted data

Source: Authors' processing.

First, the book's title is not completely extracted by our proposed solution. This happens because in the given HTML structure, the book's title can be identified in multiple places, and in some of these places the value is truncated. As such, our tool may benefit from the inclusion of a module of self-improvement, which should check if the extracted data is complete.

Second, the value for the book's image source is relative to the starting URL for our proposed solution, while Scrapeghost offers the full URL. Seeing as the starting URL is known beforehand, it is preferable to offer a relative value for the

image source, with the main purpose of keeping the size of the scraped data as small as possible. This is a minor difference, but it may prove to have a certain impact on the required storage capacity, as the collected data grows bigger and bigger.

Third, the value for the book's price is correctly extracted by all the three solutions, but AutoGPT lacks the capability of transforming the data to an actual number, which could prove to be a serious disadvantage when working with e-Commerce platforms.

When it comes to extracting specific, non-repetitive data from a web page, the results are identical between the data extracted by the proposed solution and by Scrapeghost (Figure 12).

Figure 12. Comparison of extracted data
Source: Authors' processing.

By analysing the output provided by our autonomous scraper, in comparison with pre-existent GPT-backed scraping systems' results, we are confident that our solution presents a satisfactory degree of accuracy and several advantages, aimed at democratising access to data extraction to users with non-technical backgrounds.

We compared the proposed solution against Scrapeghost and AutoGPT on three cases, as shown in Table 1. Here, the *accuracy* determines the percent of correct data out of the entire data provided and *completeness* determines the percent of provided data out of the entire data available.

Table 1. Comparison of proposed solution

		Case 1	Case 2	Case 3
URL		https://www.one.ro/en/get-in-touch/	https://books.toscrape.com/	https://www.cursbnr.ro
Data to be extracted		Company name and contact methods	Products data (i.e. price, title)	Foreign exchange rates
Proposed Solution	**Accuracy**	100%	100%	100%
	Completeness	100%	100%	100%

		Case 1	Case 2	Case 3
	Cost (USD)	0.063095	0.103025	0.192835
Scrapeghost	**Accuracy**	100%	100%	0%
	Completeness	100%	100%	0%
	Cost (USD)	0.037945	0.05624	0.065275
AutoGPT	**Accuracy**	100%	100%	100%
	Completeness	100%	100%	3.23%
	Cost (USD)	0.0875	0.1053	0.1325

Source: Authors' processing.

As can be seen from the Table 1, our solution consumes more tokens than Scrapeghost does but is more efficient than AutoGPT, which consumed fewer tokens only for extracting foreign exchange rates, where it provided little data, so naturally output tokens, were dramatically reduced.

The increased cost of our proposed solution, relative to other specialised scraping solutions powered by GPT, such as Scrapeghost, is an understandable disadvantage, as Scrapeghost requires a selector for the container element which includes the requested data, dramatically reducing the size, and implicitly the tokens, of the HTML structure to be analysed. However, the other analysed solution, AutoGPT, performs unnecessary extra steps in the scraping process, with reasoning self-assessment, naturally consuming more tokens.

An important advantage of our solution is that the autonomous scraper also creates and persists the complete scraper code in local storage. As such, ulterior necessities of extracting the same data, or even recurrent scraping jobs, will not need to consume any GPT tokens as the automatically generated scraper can be directly executed, unless the HTML structure has been updated.

One other important aspect resulting from the above comparison is the better performance of our proposed solution compared to pre-existent GPT-powered scraping solutions, in terms of completeness of data. As such, our solution succeeded in extracting all the available data from the three URLs that were provided, in contrast to the inferior performances obtained by Scrapeghost and ChatGPT. An interesting scenario to analyse is extracting the foreign exchange rates from a specific website, where neither Scrapeghost nor ChatGPT succeeded in extracting the complete data. One possible explanation for this result is that the provided website included some information regarding foreign exchange rates in the first part of the page, and the full list of rates at the bottom of the page. Since our proposed solution is instructed to search through the entire website for the requested data, it correctly identifies and extracts the complete list of foreign exchange rates.

We can integrate the indicators for accuracy, completeness, and cost into a composite formula to give a single score. This score can then be used to assess overall solution performance (SP). A possible way to integrate these metrics:

$SP = \sum(w_a * Accuracy\_s + w_c * Completness\_s + w_{co} * Cost\_n\_s)/nc$

where:

Accuracy_s is the Accuracy Score for each solution;

Completeness_s is the Completeness Score for each solution;

Cost_n_s is the normalised Cost Score for each solution (the lowest cost will be 100%);

w_a , w_c , w_{co} are the weights for accuracy, completeness, and cost efficiency, respectively. The sum of these weights should be 1;

nc represents the number of cases for each solution.

If we give equal importance to all three metrics, in this case we will have $w_a = 0.33$, $w_c = 0.33$ and w_{co}= 0.33. We can calculate the Solution Performance score for all of our solutions, as shown in Table 2.

Table 2. Solution Performance Scores

Application	**Score**
Proposed Solution	0.79
Scrapeghost	0.68
AutoGPT	0.63

Source: Authors' processing.

We can remark that the composite Solution Performance score for the proposed solution is 0.79, which indicates a relatively high overall solution performance, considering the defined weights and input metrics.

To further improve our solution and to address the issue of relatively higher token usage, future work on developing the proposed solution will focus on reducing the GPT tokens usage, by optimising the HTML cleanup process and implementing specific algorithms to automatically determine the selector that Scrapeghost requires as input from the user.

5. Conclusions and Future Work

The operational costs of the custom algorithm are one of its main drawbacks, especially when dealing with complex HTML structures within OpenAI prompts. We aim to create and apply a unique HTML clean-up tool for future work to reduce this problem. This tool will selectively remove extraneous information from the HTML structure, including unused JavaScript scripts and styles. By doing this, it will significantly reduce the amount of ChatGPT tokens consumed, allowing larger HTML structures to be processed more quickly and affordably.

Incipient work has been done to optimise token usage and to avoid reaching the API rate limits imposed by OpenAI, which shows promising results in identifying only a candidate element from a page with repetitive structures and determining the universal selectors by analysing the said candidate element. To achieve this task, the solution counts the total number of occurrences on the given page for a specific selector and attempts to extract a candidate element with the highest frequency

selectors. This approach assumes that pages with repetitive structures, such as e-Commerce platforms, will employ identical class names, IDs, or other attributes to the repetitive structures. Although the preliminary results are promising, there is still more work to be done to completely and seamlessly integrate this feature into our proposed autonomous scraper.

Scalability is one technical challenge that may be encountered when developing a sophisticated AI model for Web scraping. Scraping multiple websites simultaneously or large websites with numerous pages might use a lot of memory and processing power. The AI model must be able to handle a high volume of data and requests, but must operate quickly and efficiently. Additionally, it is important to consider how to handle errors and failures that can occur during the scraping process, as this can impact the accuracy and reliability of the data collected.

In the present study, we explored how AI might be used to improve the development of scraping systems while also examining the challenges and limits of conventional scraping methods. Our investigation has drawn attention to the need for novel, user-friendly, and self-sufficient methods that may overcome the required technical know-how and the difficulties involved in analysing the constantly changing HTML structure of websites.

We started by examining previous GPT-powered solutions, including Scrapeghost and AutoGPT, to identify their advantages and disadvantages, as well as areas for improvement. As such, we determined that although the novel scraping systems present significant advantages over the traditional ones, there are several issues that need to be addressed by our proposed solution, such as eliminating the need for technical proficiency to operate such a scraping system.

The paper suggests an original concept for an autonomous scraper system based on our research of previous approaches, which makes use of ChatGPT, an effective linguistic model, to allow users to naturally express the data they wish to retrieve from the web. Because our system is autonomous, handling all the stages of data extraction from the web, it does not require manual coding or interventions; thus, even individuals with little technical knowledge may use it with satisfactory results.

We believe that by democratising access to data extraction tools, our autonomous scraper system will enable users from a wide range of areas to quickly and easily find the information they need on the web without the need for expert programming knowledge or a thorough understanding of web technology. The user-friendly conversational interface offered by ChatGPT makes it possible for non-technical users to communicate their data extraction requirements clearly, and the custom-built algorithm handles all other related tasks, such as URL and selectors identification and data persistency.

Despite the potential that our proposed scraping system indicates, there are still difficulties that need to be addressed. The ability of the ChatGPT model to understand complicated queries, deal with ambiguities, and adapt to various domains must be enhanced, since these skills have a substantial impact on the system's accuracy and efficacy. By adding more checks along the way, the system should also be able to handle problems that come bundled with LLMs, such as hallucinations.

In conclusion, advances in AI, particularly in the fields of ML and natural language processing, present promising prospects for improving the development of scraping systems. Our suggested ChatGPT-based autonomous scraper system is a step toward making online scraping more approachable and user-friendly. This technology has the potential to simplify data extraction techniques with additional research and improvement, allowing users to quickly and easily access essential information from the expansive web.

References

[1] AutoGPT. (2024), *GitHub - Significant-Gravitas/AutoGPT*, https://github.com/Significant-Gravitas/AutoGPT.

[2] Carle, V. (2020), *Web Scraping Using Machine Learning,* dissertation KTH Royal Institute of Technology - School of Electrical Engineering and Computer Science.

[3] Dallmeier, E.C. (2021), *Computer Vision-based Web Scraping for Internet Forums., 7th International Conference on Optimization and Applications, Wolfenbüttel*, Germany, 1-5.

[4] Demir, N., Boyoğlu, C.S., Kayıkçı, D. (2023), *A Web scrapping and AI approach for archeologists to analyze the ancient cities. Cultural Heritage and Science*, 4(1), 01-08, https://doi.org/10.58598/cuhes.1213426.

[5] Gheorghe, M., Mihai, F.C., Dârdală, M. (2018), *Modern techniques of Web scraping for data scientists*. *Revista Romana de Interactiune Om-Calculator,* 11(1), 63-75, http://rochi.utcluj.ro/rrioc/en/rrioc-20181.html#Modern_techniques_of_Web_scraping_for_data_scientists.

[6] Karthikeyan, T., Karthik, S., Ranjith, D., Vinoth Kumar, V., Balajee, J.M. (2019), *Personalized Content Extraction and Text Classification Using Effective Web Scraping Techniques. International Journal of Web Portals*, 11(2), DOI: 10.4018/IJWP.2019070103.

[7] Kaur, P. (2022), *Sentiment analysis using Web scraping for live news data with machine learning algorithms, 2022 International Conference on Materials and Sustainable Manufacturing Technology*, 65(8), 3333-3341.

[8] Kovaks, G., Bogdanova, D., Yissupova, N., Boyko, M. (2015), *Informatics Tools, AI Models and Methods Used for Automatic Analysis of Customer Satisfaction. Studies in Informatics and Control*, 24(3), 261-270, DOI:10.24846/v30i4y202106.

[9] Krotov, V, Silva, L. (2018), *Legality and Ethics of Web Scraping. Twenty-fourth Americas Conference on Information Systems*, New Orleans, 1-5.

[10] Matta, P., Sharma, N., Sharma, D., Pant, B., Sharma, S. (2020), *Web scraping: Applications and scraping tools. International Journal of Advanced Trends in Computer Science and Engineering,* 9(5), 8202-8206, DOI: https://doi.org/10.30534/ijatcse/2020/185952020.

[11] Mitchell, R. (2018), *Web Scraping with Python: Collecting More Data from the Modern Web*. 2nd edition. Publisher(s): O'Reilly Media, Inc.

[12] Moaiad A.K. (2021), *Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application. International Journal of Advances in Soft Computing and its Applications*, 13(3), November 2021, print ISSN: 2710-1274, Online ISSN: 2074-8523.

[13] Năstasă, A., Maer Matei, M.M., Mocanu, C. (2023), *Artificial Intelligence: Friend or Foe? Experts' Concerns on European AI Act. Economic Computation and Economic Cybernetics Studies and Research*, 57(3), 5-22.

[14] Niu, Q., Kandhro, I.A., Kumar, A., Shah, S., Hasan, M., Ahmed, H.M., Liang, F. (2023), *Web Scraping Tool for Newspapers and Images Data Using Jsonify*. *Journal of Applied Science and Engineering,* 26(4), 465-474, https://doi.org/10.6180/jase.202304_26(4).0002 C.

[15] Petprasit, W., Jaiyen, S. (2015), *Web content extraction based on subject detection and node density*, 7th International Conference on Knowledge and Smart Technology (KST) Faculty of Informatics, Burapha University, Thailand, 121-125.

[16] Scrapeghost (2024), *GitHub* - jamesturk/Scrapeghost, https://github.com/jamesturk/Scrapeghost/.