

Scraping Google Scholar Data Using Cloud Computing Techniques

Nagham A. sultan
Department of Computer Science
University of Mosul
Mosul, Iraq
Naghamajeel@uomosul.edu.iq

Dhuha B. Abdullah
Department of Computer Science
University of Mosul
Mosul, Iraq
prof.dhuha_basheer@uomosul.edu.iq

Abstract—The current revolution in Internet technologies has led to a tremendous amount of data on the world wide web (www). Dealing with such big data is not an easy task in terms of collection and analysis. Moreover, the unstructured nature of most of this data makes it more difficult to collect and analyze. Therefore, scraping and crawling techniques are considered the most common types of data collection from the web. In this paper, a scraper is developed based on two methods to scrape data from one of the most complicated platforms on the web, which is Google scholar. These two methods are Beautiful Soup and SERP API. The proposed scraper is divided into three parts: the web scraper finds the needed links on the internet, the data is extracted from the source links, and lastly the data is stored in a CSV file. For implementing the scraper, Python language is used. The results showed that the designed scraper is efficient in retrieving data from Google scholar. Also, it was found that Python is a suitable and efficient language for scraping necessary data from a particular website of interest due to the abundance of library resources.

Keywords—*Web Scraping, Crawling, Information Extraction Python, Web Technology, Implementing Web Scrape, API.*

I. INTRODUCTION

The scientific community has embraced Google Scholar. Google Scholar Citations and Google Scholar Metrics have quickly grown due in part to their promises of providing universal and almost simple access to scientific literature as well as the perception that it better covers the most scientific contributions than other traditional multidisciplinary databases [1]. The introduction of Google Scholar citations and Google scholar metrics marked a revolution in the scientific information market by allowing access to most of the documents available on the web. The launch of Google scholar was in 2004 (a novel search engine focused on retrieving any type of academic material along with its citations). Moreover, it poses a fresh threat to the established bibliometric indexes and databases provided by Elsevier (Scopus and SJR) and Thomson Reuters (Web of Science and JCR), breaking their monopoly and becoming a significant rival; The new products from Google scholar portray a world where moral and social conundrums could have significant repercussions for the field of science and research evaluation [2]. It is a matter of fact, that any type of research, whether academic, marketing, or scientific, requires data. People may desire to gather and analyze information from different websites [3]. The material on the many websites in the category is presented in a variety of methods. You might not be able to see all of the data on a single page. The information may be spread across numerous pages and sections. Most websites do not allow you to save a copy of the data they

display to your local storage[4]. One of the most frequent and common ways to save the data displayed on the website is to manually copy and paste it into a local file on a PC. This is a tedious task that can take a long time [5]. In terms of information that consumers require, a Web page comprises useless content such as menus, advertising, banners, footers, and sitemaps, as well as necessary content such as title, summary, main text, pricing, and description. With the rise of unnecessary content on web pages, it's become more important than ever to remove it and extract the necessary content for text processing applications like search engines, question-answering systems, recommendation systems, trend detection/monitoring, sentiment analysis, and e-commerce market monitoring [6].

Web scraping is a method for extracting data from various websites into a single spreadsheet or database, making it easier to analyze and visualize the information. The goal of this research is to provide an overview of online scraping strategies and applications for extracting data from websites [4]. Web scraping software connects to the internet via protocols (such as HTTP/HTTPS) or web browsers. The majority of the data extraction is automated and done by software or script. As a result, web crawling is another term for it. The web scrapper's actions can be compared to data copying, but the result is in the format that we require[7]. However, obtaining that information is difficult. The majority of information on the internet is presented as an HTML document, which is a diagram designed to convey information to humans rather than machines, and such data is classified as unstructured data. Every year, the amount of data on the internet grows exponentially, and much of it is unstructured. Information churning is difficult with unstructured data since it does not follow any data model. Our focus is the extraction of a significant volume of data from the internet and its storage for future work. Web Scraping Software is a software solution that allows for such extraction [8].

The majority of web scraper software downloads or grabs the HTML page from the website. It will begin analyzing and processing the data after the download is completed. As a result, a web scraper is an important part of scraping because it assists with retrieving, processing, and storage. The data from the uninstaller can be saved in any format required (e.g. .xml .pdf .txt .csv). A web scraper usually extracts data to create new data. The scraping procedure is shown in Fig. 1. Website pages are created using mark-up languages (HTML and XHTML) and provide sufficient material for users in the form of text. Because most automated tools are designed for humans, they are unable to utilize this information. This is where the term "web scraper" came from. The majority of

websites lack a proper API for extracting data. As a result, we use a web crawler. Only a few sites, such as Amazon, Twitter, and Google, offer free API to users[7]. HTML elements and data between these elements make up web pages. Web scraping is the process of obtaining specific data from these aspects for use in other applications such as monitoring online pricing changes, assessing product reviews, tracking online presence, gathering articles, and so on.

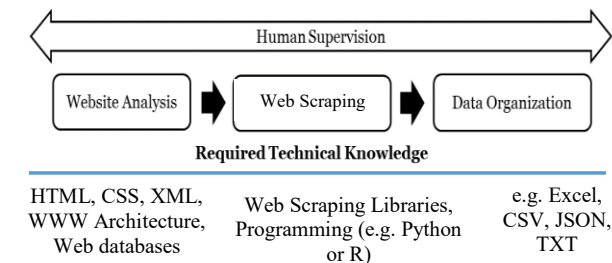


Fig. 1: Scraping process

For these applications, the internet provides a great supply of "big data." The majority of "big data" studies [2]. The purpose of collecting researchers' data is to obtain this information for its importance in evaluating universities and researchers. The scraper can be executed periodically to update data in every period.

LITERATURE REVIEW:

[9] presented a paper to scrape data from www using the Python programming language and its virtual libraries to extract data from www. There was no specific database and there for the goal was to develop and program a web crawler to fetch information from the Internet and filter the data for useful and graphical purposes for users. Accordingly, a web crawler was implemented to collect information from www and download it to the user's computer, but they encountered some errors such as spam engines that prevent major search engines from publishing their ranking algorithms. [10] proposed a scraper using web crawler technology for Maonan Residential Information About Maonan Nationality in China The aim was to protect and develop the culture of local residential architecture and the environment. As a result, the Maonan ethnic group was found which has unique local customs and cultural details. [11] presented a computer-aided literature review, using automated text aggregation with manual qualitative analysis, and a bibliometric sentiment analysis study of 6,996 papers. They searched the history of sentiment analysis, evaluated the impact and trends of sentiment analysis through citation and bibliometric study, identified sentiment analysis communities by finding the most popular place of publication, discovered what research topics were investigated in sentiment analysis, and reviewed the most cited original works and literature reviews in Analysis feelings. We found that the science of sentiment analysis and opinion mining has deep roots in studies conducted on public opinion analysis at the beginning of the twentieth century.

[8] presented a paper entitled Cloud-based web scraping for big data applications, Selenium was used as a web scraping tool (an open-source software suite) and the goal was to build a web scraper for efficient data collection. And as a result, the skimmer was effective in collecting data and analyzing the

scale and performance of the scraper. Other works in the literature tried to analyze authors publishing tendencies using data that was collected using a special purpose scraper that targets Google scholar data [12]. Google Scholar data was also collected by the authors of [13]. They analyzed the Google scholar citation network by collecting data using the R language. Finally, other contributions in the literature tried to scrape and analyze Google scholar data such as [14-16]. According to the literature, it can be seen that most of the works do not provide technical details about the scraping process. This issue is considered a lack in the literature. Hence, this work comes with a contribution that deals with the aforementioned issue and provides a detailed design of a scraper that is considered efficient in scraping Google scholar data. Also, this work presents the detailed steps of building a scraper and how to overcome the most frequent issues in such designs.

II. BACKGROUND

The Difference between web scraping and web crawling is it Many authors and programmers will use both words interchangeably, so the distinction between "web scraping" and "web crawling" is hazy. In general, the term "crawler" refers to a program's ability to traverse websites on its own, possibly without a clear end goal or purpose, continually investigating what a site or the internet has to offer. Search engines like Google rely on web crawlers to retrieve material for a URL, evaluate that page for further links, and retrieve the URLs for those links, among other things [17]. Web scraping (also known as screen scraping or information scraping) is the automated process of accessing web documents and extracting certain, pre-defined information, such as prices, before converting and saving them into a structured format. Web crawling, on the other hand, entails accessing and indexing web content via hyperlinks; hence, only the URL is extracted, but no specific information. Instead, the whole content is accessible via a URL, but it is rarely archived. Google and other search engines cruise the web, analyze the material, and collect all of the links they find to match the search query. Price comparison tools also use crawlers (or spiders). Shopbots are programs that browse websites to gather pricing information from several sellers to get the best deal. See Table I for a conceptual contrast between scraping and crawling [18].

A. Websites scraper

Web scraping is a technique for obtaining useful and interesting text from web pages. The majority of current research on this topic focuses on automated web data extraction. These investigations establish a DOM tree before accessing the essential data through it throughout the extraction process. Depending on the data structure of the DOM Tree, the construction process of this tree increases the time cost [6]. It is a technique for extracting data from a variety of websites. Web data extraction, web data scraping, web harvesting, and screen scraping are other terms for the same thing. In general, websites include unstructured data, which can be retrieved and extracted using web scraping, then transformed into a usable format like CSV and stored in an external database [4]. The diversity of websites is one of the most difficult aspects of gathering data through them. The researchers refer to the layout as well as the facts. Since every

website has a distinct layout, utilizes various HTML IDs (or doesn't) to select fields, and so on, it's difficult to create a one-size-fits-all scraper. As if that weren't enough, many websites' designs change regularly. The scraper will stop operating if this happens. The only alternative in these instances is to revisit your code and modify it to changes on the target website [19]. The entire web scraping process can be divided into several steps and summarized as follows [20], [21]:

TABLE I. DISTINCTION BETWEEN WEB SCRAPING AND WEB CRAWLING

	<i>Web scraping</i>	<i>Web Crawling</i>
Process	Requesting and collecting information from web documents automatically	Finding and fetching hyperlinks repeatedly, starting with a list of initial URLs
Target information	Predetermined information on selected websites	Depending on the search request, URLs to access various types of information
Output	Data in an organized format was downloaded.	Database-stored indexed hyperlinks
Use	Data gathering (e.g. price series)	Speculative requests (e.g. search engines, price comparison tools)

Phase 1: Collect data

To begin, choose the websites from which the data will be accessed in this stage. The HTTP protocol, which is used to send and receive requests from a web server, can then be used to fetch the data. During this phase, libraries like curl, and traverse can be used to send an HTTP GET request to a website's URL (desired location), and an HTML document will be returned as a response.

Phase 2: Information Extraction

The next step would be to extract our required information from the website after retrieving the desired HTML pages. This can be accomplished using a variety of techniques, including HTML parsing, DOM parsing, XPath, and Text pattern matching, which will be covered in detail in the following section.

Phase-3: Data Transformation

The data will be in an unstructured format once the required information is collected from the relevant places (URL). It can then be converted to a structured format like CSV, spreadsheet, or pdf for display or storage. Fig. 2 summarizes the processes outlined previously in the web scraping process.

Many practical applications of having access to and acquiring data on the web fall under the category of data science. The following is a list of some intriguing real-world applications [17], [22]:

- Many of Google's products have profited from the company's primary business of site crawling. For example, Google Translate uses text from the internet to train and improve itself. For competition analysis, banks and other financial institutions use web scraping. Banks, for example, regularly monitor competitors' websites to get a sense of where branches are opening or closing or to track loan rates given - all of which is useful information that can be included in internal models and projections.

- One researcher was able to build a deep learning network to predict whether an image would be regarded as "beautiful" using scraped photographs from Tinder and Instagram along with their "likes." Smartphone manufacturers are already using such models in their camera apps to assist you to improve your photos.
- Social scientists scrape social media sites to track public opinion and political preferences.

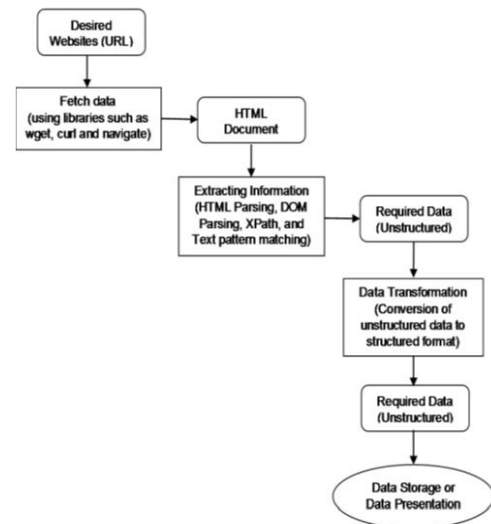


Fig. 2: Web Scraping Process Overview

The importance of web scraping can summarize use cases [22]:

- ✓ when the website from which you wish to extract data lacks a public API and there are no similar third-party APIs that give the same set of data.
- ✓ Even on their paid tier, the API does not reveal all of the data you want to acquire, whereas the website does.
- ✓ Any key business operation includes web scraping/crawling.
- ✓ If an API is available, the free tier is rate limited, which means you can only use it a specific number of times. Although the subscription tier of the API is prohibitively expensive for your intended use case, accessing the website is free.

Especially, if you don't get the data in a "clean" form or have access to a web API that can serve the data you need in JSON, XML, or some other common format, the dataset you need to build your visualization will often Encode them in HTML, in the form of tables, headers, ordered and unordered lists of content, and the like. we can cut and paste if the data collection is small enough, but aside from the boring and inevitable human error, this solution won't scale. However, even if the data is hidden inside HTML blocks, these blocks usually have a recursive structure. These two facts enable us to define where that data in the HTML is in a formal style that a machine can understand. We may then manually extract it on a small scale using specialized tools such as requests and BeautifulSoup, or in bulk using Scrapy [23].

There are various techniques for implementing a web scraper, the best of which can be chosen based on a programmer's

requirements. These approaches are utilized throughout the extraction process, and they are largely separated into two categories: manual web scraping and automated web scraping, as shown above. This section examines some of the most important strategies in each area and provides options for choosing the best one [4], [20], [24]:

- Manual Web Scraping

Copying and pasting is a time-honored method. Traditional copy-pasting is a manual method of extracting data from a webpage. The necessary information would then be gathered from the group and organized into a structured format. This is sometimes the best strategy for smaller amounts of data. This technique includes a lot of human work; therefore, it may be tedious and error-prone when creating huge datasets.

- Automated Web Scraping

Parsing HTML. Parsing is the process of analyzing a string of symbols in natural language, computer languages, or data structures that follow grammar rules. Parsing a document usually produces a tree of nodes that represent the document's structure. After retrieving the HTML document, parsing creates a tree of nodes from which relevant information such as page heading, title, and paragraphs can be extracted by detecting HTML nodes. To parse HTML pages and retrieve and transform page content, semi-structured data query languages such as XQuery and HTQL can be used.

B. Type of scrapers

Web scraping can be done with a variety of libraries. However, as the most widely used languages in data science, Python and R contain a large number of built-in libraries that make retrieving meaningful information from websites much easier.

1) Web Scraping in Python Using BeautifulSoup Library

Beautiful Soup is a Python language package that parses and extracts data from HTML strings. It includes several HTML parsers that let us extract data even from badly structured HTML. We can use the Requests package to fetch the HTML page, and once it's on our local machine, we can start experimenting with lovely soup objects to extract valuable data. Web pages change frequently, which makes learning web scraping challenging because the web page must remain consistent [22]. BeautifulSoup has grown in popularity as a result of its ease of use, however, it is slower than lxml. One of the most significant benefits is that it can be used on any sort of website. This can be used in conjunction with requests to complete the fetch and extraction processes. It can be coupled with the lxml parser to speed up the process [20].

BeautifulSoup and lxml are the two most popular lightweight scrapers in Python. Their core select syntax is different, yet they can use each other's parsers, which is perplexing. Although it appears the XML is significantly faster, BeautifulSoup may be more powerful when dealing with badly configured HTML. I found lxml to be sufficiently powerful, and the path-based syntax to be more robust and intuitive [23]. Python will be used because it has a simple syntax, is widely used (it comes pre-installed on many operating systems), and has some good modules for sending HTTP requests and parsing HTML answers that will make our life much easier [25]. The first thing we need to do is

make sure that requests and BeautifulSoup, the two fundamental packages we'll be using, are installed. If you already have python and pip installed on your machine, you should be able to install the packages using the following command:

```
pip install requests beautifulsoup4
```

If that command doesn't work, double-check that you have Python installed. We need also to run the following command from the command line to check that pip is installed:

```
easy_install pip
```

We're ready to get started once you've installed python, pip, and those packages.

2) SERP API

SERP API is a real-time API that allows users to get search results from Google, Bing, Yahoo, Yandex, eBay, Walmart, YouTube, Home Depot, LinkedIn, and other sites. It solves problems like renting proxies, handling captchas, and digesting rich structured data. Search engine results can be arranged with SERP API google (search, maps, jobs, shopping, scholar). It allows us to quickly obtain real-time search engine data. Get parsed information from both organic and paid results, as well: Many features are included, including [26]:

- ✓ precise Locations
- ✓ Real Results in Real-Time
- ✓ Legal Shield in the United States
- ✓ Results in JSON

The Google Scholar API, which allows users to scrape SERP results from a Google Scholar search query, is one of the most essential APIs. The following endpoint can be used to access the API:

/search? Engine=google scholar. A user may query utilizing a GET request.

This feature includes four options for searching for material in Google Scholar [20]:

Author API: users can scrape author results from the Google Scholar Author search page using the Author API.

Author Articles API: The Author Articles API enables users to scrape the results of a Google Scholar Author search for articles. SerpApi is capable of scraping, extracting, and interpreting this data. We add article results to our JSON output whenever we come across them.

Co-Authors API: The Author Co-Authors API lets you scrape the co-author's results from a Google Scholar Author search. Name, link, author id, affiliations, email, and thumbnail results are all possible to extract.

Author Cited By API: cited author the cited results of a Google Scholar Author search can be scraped using the by API. Table and graph results are both possible to extract. Citations, h index, and i10 index findings are all extracted from table data. We extract year and citation results from the graph results.

Author Citation API: A user can scrape the citation results of a Google Scholar Author search using the Author Citation API. Title, link, resources, authors, publication date, journal, description, total citation results, and more can all be extracted.

III. IMPLEMENTATION AND RESULTS

To achieve the objectives of this research, many experiments have been applied using the previously described methods,

and the results reached by will researchers will be presented sequentially and results will be discussed to determine the pros and cons of each method of obtaining information from the web, noting that this information will be of high value to other researchers. Those who wish to work in this field will facilitate the process of choosing the appropriate method according to their requirements for the information they wish to obtain. The methods used are as follows:

A. Beautiful Soup

In this method, the researcher's page on Google Scholar was determined using its link and then using the Beautiful Soup to obtain the information available on the researcher's page, and then it was possible to print the results or store them in a CSV file format.

General Steps for using BeautifulSoup in python are:

- *Getting data from the webpage Google Scholar*
- *get data within a specific element*
- *Getting the data by looking into each tr.*
- *print(data)*

the results obtained after the implementation step were shown in Fig. 3 which the print research title sorted as citation number in the researcher's google scholar website was selected above.

name, and then the search query keyword also must selected which presents a key search at the google scholar webpage, finally using the SERP API to obtain the information available on google scholar page, and then it was possible to print the results or store it to a CSV file format.

General Steps for using SERP API in python are:

- *Get the API Secret Key first. We must first create an account on SerpApi to acquire the Secret key to access the API.*
- *Google Search Results with Python Module.*
- *Using Params, construct a Search Query.*
- *Gather information from the outcomes.*
- *The code must be completed.*

Figure 5: Shows the result of applying these parameters in our Python code:

```
params =
{"engine": "google scholar",
 "q": "big data",
 "Api_key": "user API key"}
```

Whereas:

Engine: exemplify a search engine.

Q: The parameter specifies the search query. In your query, you can also employ helpers.

API key: After signing up, SERP API provides users with a private API Key.

```
===== RESTART: C:\Users\LEGION\Desktop\BeautifulSoup.py =====
['Collaboration networks: university of mosul case study', 'Ethical hacking imple
mentation for lime worm ransomware detection', 'Network Centralities-Based Ap
proach for Evaluating Interdisciplinary Collaboration', 'normalized cut algorit
hm on curvelet coefficient for digital image segmentation']
```



Fig. 3: Specific researcher information

To ensure results are obtained, Fig. 4 explains search results on the google scholar website, after comparing between two results, it will be clear that our code works perfectly with obtained results.

[illegible]

Fig. 5: Successful Downloading of File

To ensure results are obtained, Fig. 6 explains search results on the google scholar website, after comparing between two results, it will be clear that our code works perfectly with obtained results. Through all the experiments that were performed and implemented in this research and the results and experiments that were reviewed at the beginning of this paragraph, it is possible to note the differences, advantages, and disadvantages of each of the two methods used in this research. We notice a clear difference in the requirements for implementation in both ways, as the first method requires a link to the researcher's page on the Google Scholar website, while the second method requires only specifying the method and search word through which the dependent information will be extracted and printed or stored in the way the programmer needs for the proposed system. On the other hand, which is the way the two methods work, we notice a difference in the mechanism of work, as the second method provides documentation of a complete classification of the functions that can be used in this way, the results desired by the programmer, and thus the second method overcomes the

Nagham ajeel

assistant lecture, university of mosul
Verified email at uomosul.edu.iq
data structure image processing network

TITLE

Collaboration networks: university of mosul case study
[Al-Raddad, Journal of Computer Sciences and Mathematics 14 \(1\), 117-133](#)

Ethical hacking implementation for lime worm ransomware detection
[Al-Raddad, 2020 68th International Engineering Conference 'Sustainable Technology and ...](#)

Network Centrality-Based Approach for Evaluating Interdisciplinary Collaboration
[Al-Raddad, 2020 68th International Engineering Conference 'Sustainable Technology and ...](#)

normalized cut algorithm on curvelet coefficient for digital image segmentation
[Al-Raddad, 2020 68th International Engineering Conference 'Sustainable Technology and ...](#)

Fig. 4: google scholar window results

Through all the experiments that were performed and implemented in this research and the results and experiments that were reviewed at the beginning of this paragraph, it is possible to note the differences, advantages, and disadvantages of each of the two methods used in this research.

B. SERP API

In this method, the search engine must be selected which must be Google Scholar in this research by determining its

first in ease of use and identification of results. What should also be mentioned is the difference in the amount of information and results between the two methods. It was noted that the second method provides more information and

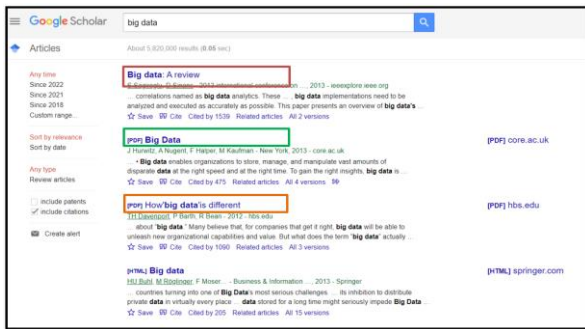


Fig. 6: google scholar results

details than the first depend on the amount and type of data that was extracted from a particular web page. Therefore, given the importance of Google Scholar, the researcher's point of view in this research is that the second method, which is SERP API, is better and easier to use than the beautiful soup method. After all this, we do not negate the existence of limitations in both ways, which the two methods cannot search for a specific organization or university which presents a great important feature for any company or organization that decides to perform web scraping and made getting statistical information about it was a complex problem.

IV. CONCLUSION AND FUTURE WORK

In this work, a Google scholar scraper is developed to scrape data from Google scholar. The proposed scraper is based on two techniques; Beautiful Soup and SERP API. The proposed scraper is divided into three parts: the web scraper finds the needed links on the internet, the data is extracted from the source links, and lastly the data is stored in a CSV file. Python programming language was used to implement the proposed scraper. According to the implementation results, the proposed scraper was efficient in retrieving data from Google scholar. Moreover, it was observed that Python was highly suitable for scraping data due to a large number of tools and libraries that can be used for scraping purposes. The implementation of this work showed several obstacles such as captcha appears, storage issues, and the necessity for intensive processing data extraction reliability, and calculation capacity. Scraping can be broken down into web renderer automation, parser, filter, and formatting of results, as well as Storage. Similarly, managing large-scale scraping requests falls under the distributed computing umbrella. BeautifulSoup, Requests, and Scrappy are some of the libraries that can be used to create scrapers. As future work, it is planned to scrape other websites and measure the performance of the proposed approach under other similar platforms such as IEEE Xplore digital library, ACM, and Academia.

ACKNOWLEDGMENT:

The authors are grateful to the Computer Science department at the University of Mosul for the support in overcoming the issues during this work.

REFERENCES

- [1] D. López - Cózar, E., N. Robinson - García, and D. Torres - Salinas, The G oogle scholar experiment: How to index false papers and manipulate bibliometric indicators. Journal of the Association for Information Science and Technology, 2014. 65(3): p. 446-454.
- [2] E.D López-Cózar,, N. Robinson-Garcia, and D. Torres-Salinas, Manipulating Google Scholar citations and Google Scholar Metrics: Simple, easy, and tempting. arXiv preprint arXiv:1212.0638, 2012.
- [3] B. Mahmood, Z. Younis, and W. Hadeed, Analyzing Iraqi Social Settings After ISIS: Individual Interactions in Social Networks. American Behavioral Scientist, 2018. 62(3): p. 300-319.
- [4] D.S. Sirisuriya,, A comparative study on web scraping. 2015.
- [5] R.B. Penman,, T. Baldwin, and D. Martinez, Web scraping made simple with site scraper. Penman Web Scraping, 2009: p. 1-10.
- [6] E. Uzun, , A novel web scraping approach using the additional information obtained from web pages. IEEE Access, 2020. 8: p. 61726-61740.
- [7] V. Bhateja, S.C. Satapathy, and H. Satori, Embedded Systems and Artificial Intelligence: Proceedings of ESAI 2019, Fez, Morocco. Vol. 1076. 2020: Springer Nature.
- [8] R.S. Chaulagain, et al. Cloud-based web scraping for big data applications. in 2017 IEEE International Conference on Smart Cloud (SmartCloud). 2017: IEEE.
- [9] F.J.M. Shamrat, et al., An effective implementation of web crawling technology to retrieve data from the world wide web (www). International Journal of Scientific & Technology Research, 2020. 9(01): p. 1252-1256.
- [10] Z. Jian, and L. Qin, The application of big data network crawler technology for architectural culture and environment protection. Concurrency and Computation: Practice and Experience, 2022. 34(9): p. e5769.
- [11] M.V. Mäntylä, D. Graziotin, and M. Kuuttila, The evolution of sentiment analysis—A review of research topics, venues, and top cited papers. Computer Science Review, 2018. 27: p. 16-32.
- [12] D.B. Abdullah, and B. Mahmood. Network-Based Bibliometric Method for Analyzing Collaboration and Publishing Tendencies. in 2020 6th International Engineering Conference "Sustainable Technology and Development"(IEC). 2020: IEEE.
- [13] A.J. Mohammed, T.M. Hasan, and B. Mahmood. Citation Networks Iraqi Universities Case Study. in 2020 3rd International Conference on Engineering Technology and its Applications (IICETA). 2020: IEEE.
- [14] B.M. Mahmood,, et al., Collaboration networks: university of Mosul case study. AL-Rafidain Journal of Computer Sciences and Mathematics, 2020. 14(1): p. 117-133.
- [15] N.A. Sultan,, et al. Network Centralities-Based Approach for Evaluating Interdisciplinary Collaboration. in 2020 6th International Engineering Conference "Sustainable Technology and Development"(IEC). 2020: IEEE.
- [16] B. Mahmood, et al., Measuring scientific collaboration in co-authorship networks. IAES International Journal of Artificial Intelligence, 2021. 10(4): p. 1103.
- [17] S.V. Broucke, and B. Baesens, Practical Web Scraping for Data Science: best practices and examples with Python. 2017: CreateSpace.
- [18] J. Hillen, Web scraping for food price research. British Food Journal, 2019.
- [19] G. Hajba, Website Scraping with Python: Using BeautifulSoup. 2018, USA.: O'Reilly Media.
- [20] P. Thota, and E. Ramez. Web Scraping of COVID-19 News Stories to Create Datasets for Sentiment and Emotion Analysis. in the 14th Pervasive Technologies Related to Assistive Environments Conference. 2021.
- [21] E. Persson, , Evaluating tools and techniques for web scraping. 2019.
- [22] J.M. Patel, Getting Structured Data from the Internet: Running Web Crawlers/Scrapers on a Big Data Production Scale. 2020: Springer.
- [23] K. Dale, , Data visualization with python and javascript: scrape, clean, explore & transform your data. 2016: " O'Reilly Media, Inc.".
- [24] A.V Saurkar,, K.G. Pathare, and S.A. Gode, An overview on web scraping techniques and tools. International Journal on Future Revolution in Computer Science & Communication Engineering, 2018. 4(4): p. 363-367.
- [25] H. Brody, The Ultimate Guide to Web Scraping. 2017, Lean Publishing. "SerpApi: Google Search API." SerpApi, serpapi.com. Accessed 5 July 2022.