International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V72I1P128

Volume 72 Issue 1, 284-299, January 2024
© 2024 Seventh Sense Research Group®

Original Article

Comparative Analysis of Web Scraping Tools for Low-Resource Language Text

Navroz Kaur Kahlon[1], Williamjeet Singh[2]

[1,2]*Department of Computer Science Engineering, Punjabi University, Patiala, Punjab, India.*

[1]*Corresponding Author : kahlon.navroz3@gmail.com*

Received: 17 August 2023 Revised: 18 November 2023 Accepted: 23 December 2023 Published: 07 January 2024

Abstract *- Introduction: Over the past few years, the accessibility of information on the internet has increased the availability of data in multiple languages. Several web scraping methodologies and tools have been developed; however, the scraping of "low resource" language text has not been emphasized vigorously. Objective: This paper presents a circumstantial comparison between various scraping tools while scraping from different Punjabi language text-based websites. Methods: Three Python-based and two desktop-based commercial tools have been considered for evaluation. The evaluation framework for comparing these tools includes performance, ease of use and reliability. The resultant comparison is done based on various parameters like runtime, memory usage, GitHub metrics, complexity metrics, etc. Result: While all tools are popular and viable in scraping content from the web, python-based tools give better results in terms of performance as they are customized according to the current structure of the web page. Conclusion: The paper will be useful for readers of both programming and non-programming backgrounds, as the qualities of both types of tools are discussed in detail.*

Keywords *- Desktop tools, Evaluation parameters, Punjabi, Python, Web scraping.*

1. Introduction

The Internet is growing immensely day after day. In due course of time, the internet has become the largest repository of data in the history of mankind. Most of the data present on the web is produced and consumed by humans and is thus mostly in an unstructured format. This is where web scraping plays its role. Web scraping is a process of downloading, parsing, and organizing data from the web in an automated manner [1].

There are generally three phases in the process of web scraping: a) fetching data, b) extracting information and c) data transformation. Firstly, the desired websites are selected from where the data is to be collected. Fetching can be done using HTTP protocol, which helps send and receive requests from web servers. Once the HTML documents are received from the website, the next step is to extract the required information from the website. Extraction can be done through various methods like XPath, CSS Selectors, Regular expression matching, HTML parsing, etc. Finally, the extracted information, which is in the unstructured format, is transformed into the desirable structured format like JSON, CSV, spreadsheet, or pdf for storage or presentation [2]. There are several tools and techniques available to scrape data from websites. These techniques are normally required in the extraction phase. Thus, it can be done either manually using the traditional copy-and-paste method or can be purely automatic. For automated scraping, the most technical approach is to use libraries as formulated by different programming languages.

Furthermore, web scraping frameworks, which are complete packages consisting of functionalities appropriate for fetching and extraction phases, can be used. Additionally, desktop-based web scraping tools are also preferred by people who are not keen on programming. These tools have a graphical user interface and use a point-and-click approach to fetch and extract data from websites. Web scraping is being used in multiple areas like online media, hospitality, tourism, retail, health, education, e-commerce, academic research and many more. With the advent of more and more people on the Internet every day, the availability of data in different languages has become popular with the rise of blogs and social media. Certain "low resource" languages have also seen a significant rise on the Internet. Various research studies are available wherein authors have tried to scrape data written in "low-resource" languages from websites. Linder et al. [3] created a Swiss German text corpora consisting of more than half a million sentences by scraping data from the web. Gereme et al. [4] created a fake news detection system by scraping Amharic (African language) text from different news websites.

In this article, a comparative analysis of different web scraping tools and techniques is done on “low resource” language websites (Punjabi in this case). 100 million speakers in the world speak Punjabi. Gurumukhi's script has been used to write religious scriptures of the Sikh religion. Due to the major efflux of people of Punjab over the past many years, the Punjabi language has made a strong base in foreign lands as well. Apart from being the official language of states like Punjab and Haryana in India, it is the second most-used language in the United Kingdom and the fourth most-spoken language in Canada. As of 2017, there are 2.2 million Punjabi speakers in the United States of America. In addition, there are 20+ Punjabi print-based newspapers across India and Pakistan and 15+ in Western countries. Hence, it is evident that although it’s a low-resource language compared to English, French, or Arabic, it nevertheless represents a significant population on the globe. However, it has been observed that very little has been done with respect to web scraping for this language, and the same has been tried to be achieved in this paper.

Although less work has been done in scraping Punjabi language, certain studies have attempted to scrape this low-resource content from the web. Closer to home, Singh et al. [5] developed a named-entity recognition system for Gurmukhi text. In this, they created a dataset by scraping data from various sources of Punjabi language like newspapers. Similarly, Mahi et al. [6] used Python programming language to construct a corpus of more than 134,000 news articles in nine different new genres.Several studies have also attempted to write a review on Web scraping techniques and applications; Khder [7] has written a review on state-of-art techniques of web scraping, scraping with Python language, the importance of scraping in business intelligence, cyber security, cloud computing etc. and legal concerns related to scraping. Glez Pena et al. [8], in their study, include different web scraping methods and categorize scraping under libraries, frameworks, and desktop-based environments. The authors exemplify the process by introducing a biomedical web scraping scenario.With respect to the comparison of different web scraping techniques, Gunawan et al.[9] has compared different techniques such as Regular expression, HTML DOM and Xpath. Processing time, memory usage and data consumption are used as comparison parameters for the same.

Along similar lines, Sirisuriya et al. [10] have also compared different techniques of web scraping, which also includes various web scraping software. Although comparative parameters used by both studies help in accessing performance, none of this comparative analysis has been performed on regional language. Furthermore, a comparison between program-based scrapers and desktop-based scrapers is also lacking with respect to different performance metrics. Hence, an attempt is made in this study to assess the performance of different scraping methods when extracting Punjabi text from the web.

Tools chosen for scraping Punjabi text comprise frameworks, libraries and point and click desktop environments. Several software metrics have been selected to analyse the working of each tool. The tools and related metrics will be discussed in the next sections. Finally, the article concludes with a discussion of the result of applied metrics.

2. Tools and Techniques

There are several approaches one can use to implement web scrapers. With the enormous amount of increase in online data, the need arose to deploy a straightforward tool for extracting data [8]. Different types of web scraping methods are discussed as follows:

2.1. Libraries/ Packages

The most common approach used to implement web scraping is using libraries. Using this approach, web scraping is implemented like any other software program using any programming language. The most accepted programming languages for web scraping are Python, R, Java, Dot Net and Ruby. These programming languages use several built-in libraries to fetch HTML pages from the web and further scrape data from the relevant page.

Yang et al. used C#.Net language to create a climate data scraping tool by scraping data from USA-based climatic websites [11], and later, in a Canadian climatic study, Bonifacio et al. used the same programming language to create a free climatic data scraping tool [12]. Grubesic et al. examined issues of consumer air travel accessibility by gathering information on 156 U.S. airports using the Perl script. LWP (Library for World Wide Web in Perl), a PERL library, is used in this case to simulate the search of an online air travel consumer and scrap data from the related web pages [13]. Although certain studies like Hsu’s [14] study on digital media and Angelis et al.’s [15] study on university rankings have used scripting languages like Ruby and Prolog, respectively, for scraping, in later years, more dynamic languages like Python and R superseded above discussed programming languages. As in a journalism-based study proposed by Krebs et al., a Python package Requests [16] is used to automatically collect HTML codes of Google search engine result pages regarding trending topics. Furthermore, using Pandas [17] and XLSXWriter [18], all relevant links and their accompanying ratings were automatically retrieved and entered into a dataset. Finally, the titles of each result were collected using BeautifulSoup [19]. BeautufulSoup is a powerful Python library that is used for quick screen-scraping tasks. According to [20], BS provides simple Pythonic idioms for navigating and searching the document and extracting the relevant data from it. Further, BS used popular Python parsers like lxml and html5lib, which helps in performing fast and efficient parsing over the extracted data. Another dominant Python library that has been used by various researchers, either individually or in combination with other tools, is Selenium.

In Kreb's journalism research, Selenium was used to extract data from YouTube. In this case, a Chrome WebDriver was used to open each YouTube search result page and scroll down to the bottom of the screen to load more results. Selenium is most required when web pages are heavily laden with JavaScript (e.g., TripAdvisor, Airbnb, Expedia, etc.).

In Han and Anderson's study of hospitality, the author scraped reviews from the above-mentioned travel websites. Herein, Selenium is used to scrape data hidden behind buttons like "Read more" and "more reviews", which was not possible with BeautifulSoup and Requests libraries. Selenium emulates human browsing behaviour and scrapes data from pages with pagination and infinite scrolling features [21].

Another prominent programming language in the field of web scraping is R language. R is a statistic-focused programming language and is effectively used among statisticians and data miners. Authors have used R language to scrape legislative bills relating to different political aspects in [22] and [23]. rvest, purr, curl, and tidyverse are libraries of R developed by the R core team, which are used for web scraping in different studies [24]. Wiechetek et al. used the rvest [25] library to scrape metrics from the ResearchGate portal of 1497 researchers and, after cleaning, stored them in a CSV file [26]. Bradley et al. scraped 92 pages of a Gambling discussion forum using RStudio and rvest package [27]. rvest package makes it easy to scrape data from HTML web pages as it allows the usage of both CSS (Cascading Style Sheets) selectors and XPATH (XML path language) queries. Apart from Python and R, there are other programming languages which have been used for scraping data from the web, such as Java, Dot Net, PHP, Perl, Ruby, and Prolog. Twitter 4J, an open-source Java library, is used by Sokolova et al. to harvest tweets related to crowdfunding projects [28]. A Java-based content extraction library, Apache Tikka deleted unwanted HTML markups and extracted plain text of data scraped from climate discourse websites [29].

2.2. Frameworks

Frameworks are an alternative way of scraping data from the web. They take care of each step of the web scraping process and diminish the need to rely on different libraries. In Python, Scrapy is an influential scraping framework that uses robots as classes sourced from a BaseSpider class. Starting URLs and a parsing function designated for each web iteration are provided by the BaseSpider class [30]. According to Kumar et al., Scrapy architecture consists of three things: items, spiders and processing elements [31]. Firstly, the scheduler delivers the downloaded page to the spider for analysis, after which the data is sent to the item pipeline for post-processing, and any further link to capture is sent to the scheduler. Dongo et al. performed an exhaustive study to perform a credibility analysis of Twitter.

The author uses Twitter Scrapy, a collaborative framework with the goal of achieving an unlimited volume of tweets. In addition, the authors went into a detailed qualitative and quantitative comparison between Twitter API and Twitter Scrapy while extracting various attributes of tweets. The results of the study indicated that extraction using Scrapy is faster and extracts a greater number of tweets in comparison to Twitter API [32]. According to Jay Patel's book on Getting Structured Data from the Internet, Scrapy possesses all the features of a good scraper like robots.txt parser (explicitly lists out pages on the sites which can be scraped), crawl delay (specifies the time interval in seconds between fetching successive pages). Despite its adequate features, Scrapy performs weakly on large-scale crawls as it is more of a focused crawler [33]. For large-scale scraping, a Java-based framework named Apache Nutch is an alternative. Nutch is a highly extensible and scalable web scraper and can be deployed either on a single machine or a cluster if large data is to be extracted [34]. Other Java-specific web scraping frameworks are Web-Harvest and jARVEST. They include several features like retrieving web content using 'http', converting 'html-to-xml', and using'xpath' to extract its contents [8]. jARVEST is used by Glez-Pena to exemplify web data extraction scenarios for biomedical data.

2.3. Desktop based Environments

Extracting data from the web has become an inevitable step for any kind of data analysis. Researchers from all fields use Internet data in their respective studies, and not all have acumen in the field of programming. For that, various desk- top applications are available which meet the needs of non-programmers or beginners. These tools are empowered by a graphical user interface and work on a point-and-click basis. Users generally navigate to the intended web pages and select the elements of the page to be extracted without going into the technicalities of XPath or regular expressions. Several desktop applications have surfaced in the past few years, like Mozenda, Octoparse, Parsehub, etc. These applications are commercially distributed and thus are not completely open source. However, all applications offer a basic plan which can be used to scrape small or restricted amounts of data. In an academic networking study proposed by Jordan, the

Mozenda scraping tool was used to extract data from Academia.edu and ResearchGate websites, and the final data was converted into a common format of CSV files [35]. McGregor conducted research related to ophthalmic content on social media, for which he collected 3,785 posts from five different social media platforms using Mozenda. The data was compiled in an Excel format [36]. Etumnu et al., in an online based grocery sales study, used Octoparse, an end-user web scraping tool, to extract data related to the ground from Amazon.com and Amazon.co.uk [37]. It is also a visual tool used to extract bulk data from websites and save it in a clean, structured format.

Another scraping tool named Parsehub was used by Wilner et al. to scrape pins related to misinformation regarding breast cancer on Pinterest [38]. A sample of 838 pins was collected for further analysis. Along similar lines, a South Korean study also used Parsehub to collect data and develop a Voice Assistant application for COVID-19 updates [39]. The above-mentioned tools are easy to use and are rapidly established on any website, enabling the user to extract data with a few clicks. However, these tools lack flexibility and are difficult to manipulate or customize according to our needs. In the next section, we will try to give a detailed comparative analysis between different programming language-based libraries/ frameworks and desktop-based applications.

2.4. Databases

As discussed above, in past years various tools have been developed to scrape data from the web. These tools have managed to scrape large amounts of data and have assisted in building large databases. McGregor proposed in his study that much health-related information is available on the Internet but is not formally assessed. In his study, he analyzed the ophthalmic content of social media platforms. For this, he selected 5 platforms and scraped 3785 posts from the site using the Mozenda[40] scraping tool [36]. Another study based on travel data proposed by Grubesic et al. generates a matrix of 24,180 searches of fare information, travel times and connections for 156 airports in the United States[13]. In political studies proposed by Ganghof et al., R libraries have been used to scrape 2,645 and 1,028 legislative bills, respectively [22][23]. R libraries have also been used to scrape data of 2847 employees along with their ResearchGate metrics of Polish Universities [26]. Desktop tools like ParseHub and OctoParse have also been used to scrape massive amounts of content. Etumnu et al. [37] used OctoParse to extract consumer information from the Amazon website, and Wilner et al. [38] used ParseHub to scrape 838 breast cancer-related pins from the Internet. Apart from scraping English language content from websites, lately, efforts have been made to scrape low-resource language text from the web. Linder et al. prepared a customized web scraping tool to create a dataset of half a million sentences of Swiss German text corpus [3]. Gereme et.al. proposed a study to combat “fake-new” in low-resource languages by scraping online versions of Amharic news using Python libraries [4]. The latest study on Punjabi text proposed by Singh et al. scraped 2.64 million news articles of Gurmukhi text to perform Named Entity Recognition over it. Thus, as several tools are available to extract and maintain large datasets from the Internet, a comparative study of these tools is necessary.

3. Comparative Analysis Between Tools

In this section, we will give a qualitative and quantitative comparative analysis of different tools and methods used for web scraping.

This analysis will help the readers in making an informed decision while choosing the tool for scraping. Several comparative studies related to pulling data from the web have been done in previous years. [41] gave a comparative clarification on several web scraping techniques and software.

However, the study included several scraping methods but did not give any subjective or quantifiable review of the methods. Gheorge et al. presents a case study on collecting data from a Transportation Authority’s website to assess industry products and techniques used for scraping [42]. These comparative reviews had a limited outlook towards the performance and other features of the scraping tools. There are other studies wherein web scraping techniques have been compared for plugging out data from the web, but unfortunately, to the best of our knowledge, not much work has been done for scraping of “low resource” language text from the web. In the next subsection, we will discuss the websites to be scraped and the tools chosen for exhibiting the comparative analysis on them.

3.1. Domain and Tool Used

In this study, we will scrape Punjabi text from different websites. Punjabi is a “low resource” language, but with an increase in the multilingualism of the internet, a significant number of websites present their web information in Punjabi language. For this paper, we have chosen 3 websites: Jagbani newspaper (jagbani.punjabkesari.in), BBC News (https://www.bbc.com/punjabi) and Shiromani Gurudwara Prabhandak Committee (sgpc.net).The first two are newspaper websites from where we have scraped headlines on the homepage and description, tag words, and images of the related headlines. Third, is an organization website wherein we have scraped data present on the homepage, including important links and audio files. Tools for scraping data from the websites mentioned above have been selected from those stated in the previous section. We have chosen three programming language-based and two desktop environment-based tools to present an intricate comparative investigation of their working as data scrapers. For programming language-based, we have selected two Python libraries, Beautiful Soup and Selenium, and one framework named Scrapy for analysis. With respect to desktop-based environments, Octoparse and Parsehub are chosen for data scraping and analysis.

3.2. Scraping Task and Hardware Used

For this paper, we have performed scraping over three websites, out of which two are news websites, and the third is a general website related to a particular organization. For news websites such as Jagbani and BBC News, we scraped Headlines, news descriptions, the date and time of the news being posted, the name of the author and location of the news, and the URL of the image. As for the third website, SGPC, apart from scraping the news updates present on the right slider, we have also scraped audio links related to Mukhwaak and Mukhwaak Katha. SGPC website content is in both Punjabi and English, but for our work, we have only scraped content written in Punjabi text.

We have used the community edition of Visual Studio code 1.73.1 as the Integrated development environment. The hardware used is Intel(R) Core (TM) i5-2520M CPU, 2.50GHz and 64-bit Windows 10 operating system with 4GB RAM.

In the next subsection, we will discuss different software metrics chosen for presenting the comparison among the selected tools and the metric values of each parameter with respect to the scraping process carried out by each tool.

3.3. Evaluation Parameters

A web scraping task involves the extraction of large amounts of data from the web. An appropriate scraping process should utilize all the available resources in a judicious way. In our investigation, we have chosen parameters based on performance, features extracted and ease of use. In the following section, tasks used to measure these aspects are presented.

3.3.1. Runtime

The vital part of scraping software is the amount of time taken to extract the requested data. Especially for retailers and business holders looking for trends in largely collected data from the web, having a faster tool to perform the task is the principal point in choosing the specific tool. In our case, python tools were timed using the time[43] package of Python. The code was executed 10 times, and finally, the average run time in (ms) was calculated. The time taken to execute scraping using OctoParse and ParseHub was calculated automatically by the software itself at the end of every execution cycle. Table1 depicts the average runtime of each tool for three selected websites. The average run time for all tools has been calculated at 50 runs.

Table 1. Average runtime for each tool in milliseconds

Tools	Jagbani	BBC	SGPC
BeautifulSoup	1638	1362	2697
Selenium	16780	4595	24817
Scrapy	1137	753	2207
Octoparse	609124	56342	19611
Parsehub	16400	9800	11000

Table 2. Memory usage for each tool in megabytes

Tools	Jagbani	BBC	SGPC
BeautifulSoup	52.56	52.56	52.83
Selenium	44.24	44.25	44.26
Scrapy	6.9	6.9	6.8
Octoparse	542.08	227.57	694.04
Parsehub	715.97	831.78	786.9

3.3.2. Memory Usage

Memory, also known as physical memory or RAM, is the second parameter of evaluation. It plainly measures the amount of physical memory used by the process. For Python libraries BS and Selenium, it is measured using the *psutil*[44] package, which can be installed using a simple pip command. Further, we have used *mprofile*[45] to measure memory usage by the Scrapy framework, which is a low-overhead memory profiler. Table 2 highlights the memory usage of all 5 scraping tools while executing the scraping process over the mentioned websites.

3.3.3. JavaScript Support

As the above two parameters measure the performance of the tools, this parameter evaluates features supported by tools. In this, we evaluate whether a tool supports a particular feature or not. While scraping, there are cases where the website content is loaded dynamically via a JavaScript code. It is important to observe whether the scraper in use can scrape this sort of page. In our case, while scraping news headlines from BBC.com and Jagbani.punjabkesari.in, certain news items uploaded using a JavaScript code are displayed only after clicking the "load more" button. The challenge was to scrape the news headlines behind the "load more" button by simulating and performing similar actions to what a real user would do. Similarly, on the SGPC website, there are three tags, "Updates", "Articles", and "News", with dynamically uploaded data which could only be extracted by the tool having JavaScript support.

3.3.4. Dependencies

Dependencies are simply defined as the extra software required to run the tool. One's that are installed automatically with the tool are not considered as dependencies. Beautiful Soup requires requests [16] library to fetch the HTML page and bring it to the local computer for Beautiful Soup to perform scraping over it [33]. *The lxml package is often used in combination with the requests library, as it helps parse* the desired information requested by the user [47].

Table 3. Dependencies required for each tool

Tools	Dependencies
BeautifulSoup	Requests,lxml
Selenium	Webdriver(chrome)
Scrapy	Python
Octoparse	None
Parsehub	None

Table 4. Documentation links for each tool

Tools	Documentation links
BeautifulSoup	https://beautiful-soup-4.readthedocs.io/en/latest/
Selenium	https://www.selenium.dev/documentation/
Scrapy	https://docs.scrapy.org/en/latest/
Octoparse	https://www.octoparse.com/
Parsehub	https://www.parsehub.com/

Similarly, Selenium requires a third-party software called *WebDriver* for execution. In our instance, we loaded the desired pages from each of the four websites using chromeDriver, and it scrolled to the bottom of the screen to load more results. Scrapy, on the other hand, is free from such dependencies.

Scrapy's spider [48] subclass defines certain attributes and methods like start *requests() and parse()*, which handle generating requests on desired web pages and handling responses. Octoparse and Parsehub are standalone software and, thus, do not require any third-party dependency or software to run the scraping process. Table 3 represents the dependencies required for scraping for each tool.

3.3.5. Availability of Documentation

When having to decide which tool to use for scraping its official documentation, it plays an important role. We evaluated the availability of documentation based upon assumptions such as whether the documentation contains the necessary examples and how easy it is to navigate through it. Documentation for Beautiful Soup is very thorough, with a detailed guide for installation as well. As in our case, we dealt with Punjabi text, and the documentation also includes a subtopic related to encoding wherein sub-libraries have been specified, which helps in detecting the document encoding and converting it into Unicode.

Selenium has vast documentation as it includes a documentation range of tools and libraries that enable the automation of web drivers. On similar lines, Scrapy has thorough documentation with each and every method and class described in detail. There are several examples depicting different classes, methods, and key specific problems. The documentation also offers easy navigation. For OctoParse and ParseHub extensive tutorials are available on the official websites. Tutorials include video examples of scraping from specific web pages. ParseHub also offers certifications for web scraping beginners and intermediate level free of cost.

3.3.6. Ease of Installation

The installation process includes steps needed to install the tool and to import the tool into the code. When comparing the tools to be used for scraping, it is important to analyze how easy is the installation method or how smoothly the installation process is followed. Pip is the package manager for Python, which is used for installing BeautifulSoup, Selenium and Scrapy, through a simple pip install command. Using this command, BS can be installed completely and will be ready to use, but for Selenium, it does not come with a browser to control and thus requires downloading a WebDriver. Though Scrapy can also be installed using a pip install command, installing Scrapy is harder than BS. As it offers more robust and extensible features than BS, it is necessary to install the complete environment of Scrapy before starting to use it. Installation of OctoParse and ParseHub is by far the easiest as they are online available software and, like any other software, can be installed by using the setup.exe file.

3.3.7.GitHub Repository Statistics

GitHub is an Internet hosting provider for software development. It consists of many public repositories, and there are several statistics which give information regarding the repository itself. This includes the number of stars, watches, forks, open/closed issues, and number of commits. These statistics impact the quality of the tool from a developer's point of view. For this paper, we have collected these metrics from GitHub repositories of BS, Selenium and Scrapy. We were not able to find sufficient metrics for the free version of OctoParse and ParseHub tools. GitHub metrics for the tools can be seen in Table 5.

3.3.8. Cyclomatic Complexity & Maintainability Index

It is a quantitative measure to determine the number of linearly independent paths through a program code. It can be applied to different functions, methods, classes, and modules within a program [49]. Cyclomatic complexity has been used as a software metric by various studies to access and quantify the quality of software.

Table 5. Github metrics

Tools	Stars	Watches	Forks	Releases	Language	Commits
BeautifulSoup	172	7	55	na	Python, html	599
Selenium	25.2k	1.3k	7.3k	77 Latest Version (4,60)	C#, javascript, Python	28983
Scrapy	45.2k	1.8k	9.8k	29 Latest version (2.7.1)	Python	9499
OctoParse	31	2	8	na	na	11
Parsehub	na	na	na	na	na	na

Table 6. Cyclomatic complexity and maintainability index values

	Cyclomatic complexity			**Maintainability index**		
Tools	Jagbani	BBC	SGPC	Jagbani	BBC	SGPC
BeautifulSoup	2.5	3.0	1.0	70.36	74.52	100.00
Selenium	2.0	2.0	2.57	67.58	76.46	100.00
Scrapy	2.5	2.34	2.5	68.19	100.00	100.00

In [50], CC is used to calculate the effectiveness of software application developed for knowledge discovery of big data. Additionally, in [51], CC, amongst other metrics, is used to examine code submissions of students enrolled in data science programs. In our case, we have managed to calculate cyclomatic complexity for BS, Selenium and Scrapy codes to determine the complexity of the respective codes. Apart from CC, we have also calculated the Maintainability Index of all three programming language codes for the respective websites. MI measures the relative ease of maintaining the code. In [52], MI is used to measure the maintainability of a chatbot designed on the paradigm of natural language understanding using radon. Also, [53] has measured the MI of four open-source software, thus determining the maintainability value of software. The ease of maintaining the code is represented by an index value of 0-100 in this study. A high score indicates higher maintainability, while a low value indicates the opposite.

Table 6 presents the CC and MI values for programming language tools for the three websites. The values have been calculated using Radon[54], a Python tool for computing code metrics.In this section, we have presented the evaluation parameters of different tools. Certain parameter values have been depicted based on the scraping process applied over the mentioned websites. In the next section, we will discuss the tools based upon the parameters and their respective results.

4. Web Scraping Tools: Current Status Analysis

Before delving into the parameters indicated above, this section examines the current state of the tools detailed in the preceding section. This section discusses different web scraping investigations as well as the technologies utilized for the same. A dataset composed of 40,212 records of articles focused on COVID-19 has been extracted from Scopus, PubMed, arXiv and bioRxiv databases from January 2019 to July 2020 using Python web scraping techniques. Scrapy, a Python library, was used to scrape metadata from these scholarly articles in this study [55]. Frazier et al.'s [56] academic study employs Selenium and Chromedriver to traverse and interact with Elsevier's ScienceDirect website to extract information on North Carolina State editors. The scraper uses page elements such as HTML ID tags and XPaths to parse through page content to extract data.

Another academic study proposed by Single et al. [57] works with chemical databases to extract related entities out of chemical accident databases to find similarities between chemical accident cases. In this study, the web scraping Python package BeautifulSoup is used to extract reports of chemical accidents and new misses from the e-MARS accident database. The proposed scraping method in the study makes ontology creation a simple and efficient process.

A smart city initiative study proposed by Nicolas et al. [58] uses web scraping techniques to align citizens with project initiatives. For this, BeautifulSoup is used to extract top-down government announcements, and further extracted data is processed for similarity analysis between different topics. Finally, six categories are deduced to understand citizen alignment with government policies. Similarly, to understand the full scope of the US rental market, Boeing et al. [59] extracted eleven million rental listings from the Craigslist website using a scrapy web scraping framework. The scraper retrieves important data from the HTML page using the XPath query language and saves it in an organized fashion. Further, BeautifulSoup and the lxml Python library are used by Lunn et al. [60] to extract content from job websites.

A study proposed by Sangkaew et al.[61] aims to understand the tourist experience at local markets of Phuket by analyzing TripAdvisor reviews. The study used BeautifulSoup to extract 2,934 English reviews in the form of structured data from the HTML/XML content. A hospitality study proposed by Han et al. [21] aims to understand aims to equip hospitality researchers with tools and methods to augment their data sources and help customers make better travel choices. Selenium is used to interact with Java Script-heavy pages of travel sites such as TripAdvisor, Airbnb, and Expedia. Selenium was chosen over BeautifulSoup or Scrapy for this study. The latter tools, because they cannot scrape the hidden information unless it is clicked on, are not suitable for heavy Java script pages.

Ai et al. [62] investigate the structural nativisation of the English language by examining regionally emerging linguistic patterns in China English using large-scale corpus data acquired from an online discussion forum. Python modules urllib2 and BeautifulSoup are used to retrieve and parse each web page. Ensari et al. [63] offered a study that used web scraping techniques to acquire urban data to support urban design decisions. Because it was primarily text-based, one of the most popular real estate listing websites was scraped with BeautufulSoup.

Anglin's [64] study of analyzing school district policy variation utilized Requests and Beautiful Soup to make HTTP requests and parse HTML, respectively. In total, 3,995 documents were scraped from 5 district websites using mentioned Python packages. To understand student psychology to serve student needs, Hall et al.[65] extracted 35,248 most shared pins from Pinterest using Selenium web driver. The collected data includes associated images, the original URL, and the website from which the pin was saved. The study clarified social-emotional trends across the users. Scraping has also been used in different natural language processing concepts, such as in Franz et al. [66] topic-modelling study wherein Self-injurious thoughts and behavior related topics being discussed on digital platforms are analyzed.

For this, Beautiful Soup is used to scrape threads related to website subforums titled "Depression", "Suicide", "Friends", etc., from the Teenhelp.org website. The technique used has been helpful in identifying and quantifying topics or themes discussed by users. Similarly, to analyze online discussions of back and neck pain through medical forums, python scripts are used by Mintz et al. [67] to scrape 12,369 questions from six leading Israeli Back and Neck forums. The authors of the study have accepted scraping to be a viable tool in extracting data for analysis from such online forums.

A review of sampling methodology used in studies assessing the effectiveness of risk minimization measures in the European Union proposed by Jouaville et al. [68] uses the European Union electronic Register of Post Authorization Studies (EU PAS). In this study, Selenium is used to scrape the content from this register into an Excel file. The strength of this study is the ability to scrape a strong source, which is the EU PAS register.

Another study proposed by Keskin et al. [69] uses the Selenium library to scrape a large dataset from the online marketplace skipthegames.com. This study aims to help law enforcement agencies to combat individuals/ organizations involved in sexual trafficking activities. The website mentioned used for scraping is an online classified advertisement website. Hence, a web scraping application developed using Selenium is used to harvest 434,720 post strings and 749,503 unique picture files.

The techniques discussed above require programming knowledge to scrape data from the web; however, other software mimics human behavior to interact with web pages. Chiu et al. examined the experience of sports tourists at the Formula 1 Singapore Grand Prix. For this, 347 reviews were scraped from the Trip Advisor website using Octoparse, which is a point-and-click web scraping tool. Further, the reviews were broken into words to analyze the gender-based satisfaction of the tourists.

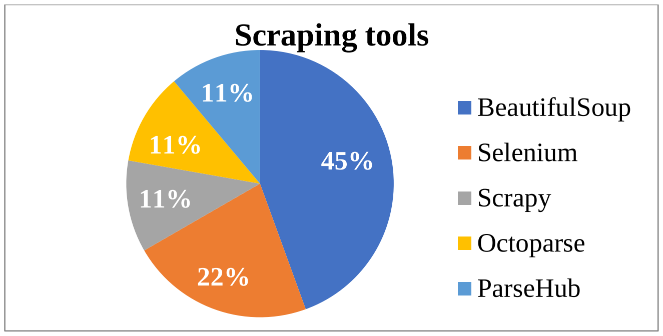

Fig. 1 Scraping tools

Another study proposed by Etumnu et al.[37] uses Octoparse to scrape data on ground coffee from Amazon.com and Amazon.co.uk. For each ground coffee product, price, title, bestseller rank, and online customer rating information were collected using this point-and-click software.

A study proposed by Molina et al. [70] explores the technological affordances of a fitness mobile app, its behavioral outcomes and goal attainment rate.A total of 682 profiles were used for analysis of the BodySpace website using another point-and-click tool, i.e. ParseHub. The program had to be run at intervals due to the page limit extraction limitation of the tool.

A voice assistant created by Primkulov et al. [39] tells statistics about COVID-19 country, city and location; it also uses ParseHub to scrape a statistics website worldmeter.com. The flexibility of the tool helps authors in scraping the contents of the website just by clicking on the objects. The importance of scraping as a means of creating global datasets for extensive study is emphasized in this section. There are various programming and automated solutions available on the market to help with this, as may be seen in Figure 1.

Automated tools or point-and-click tools are simple to use for those without programming skills, but they lack the programmatic tool's customization capability. The evaluation method of the instruments indicated above will be discussed categorically in the following section.

5. Discussion

In this section, the outcomes of the defined evaluation process are discussed. The discussion is categorized into four categories. The first one is performance, where the performance tasks such as runtime and memory usage are discussed. It is followed by the second category of features, wherein the features related to different scraping processes are discussed. The third category is ease in usage of the specific tool which deals with documentation, tutorials, dependencies, and installation segment of the tools. Last is reliability, which includes metrics provided by the GitHub repository of each tool.

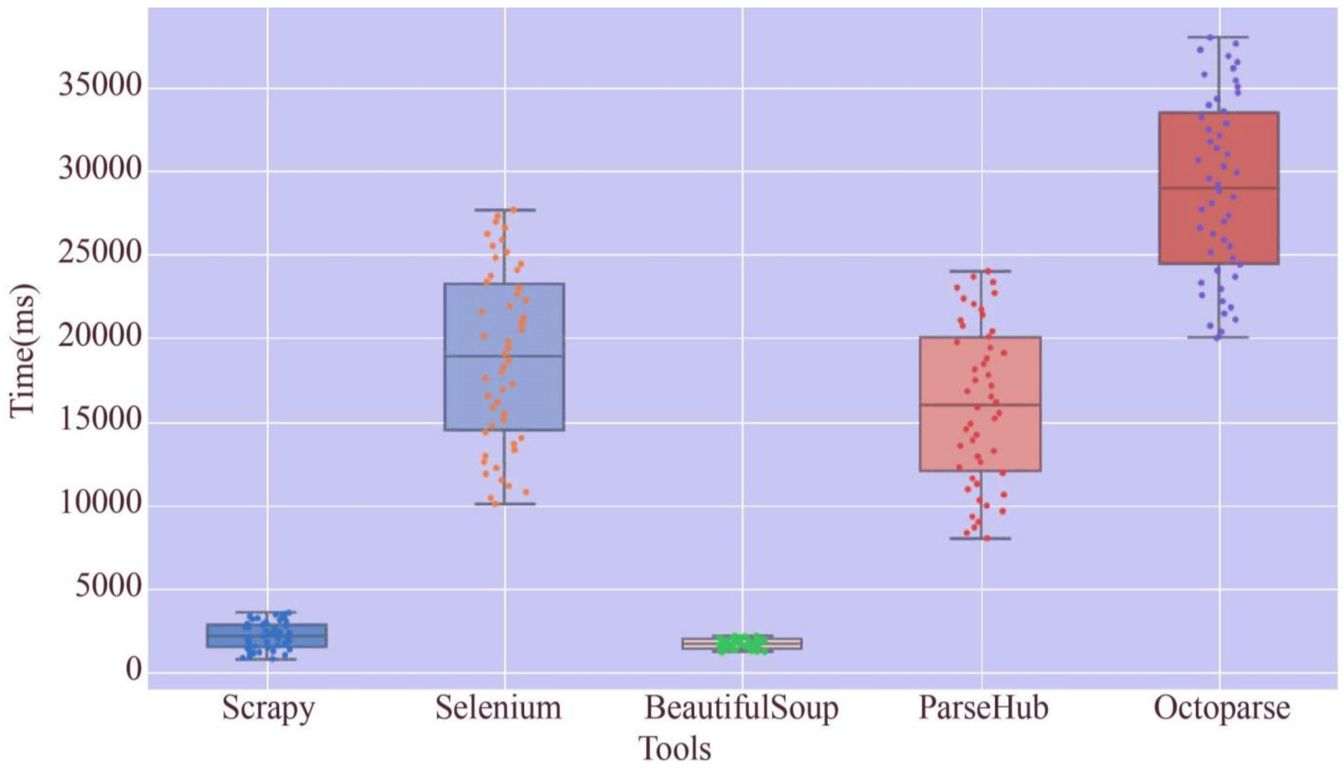

Fig. 2 Runtime graph for BBC News Punjabi

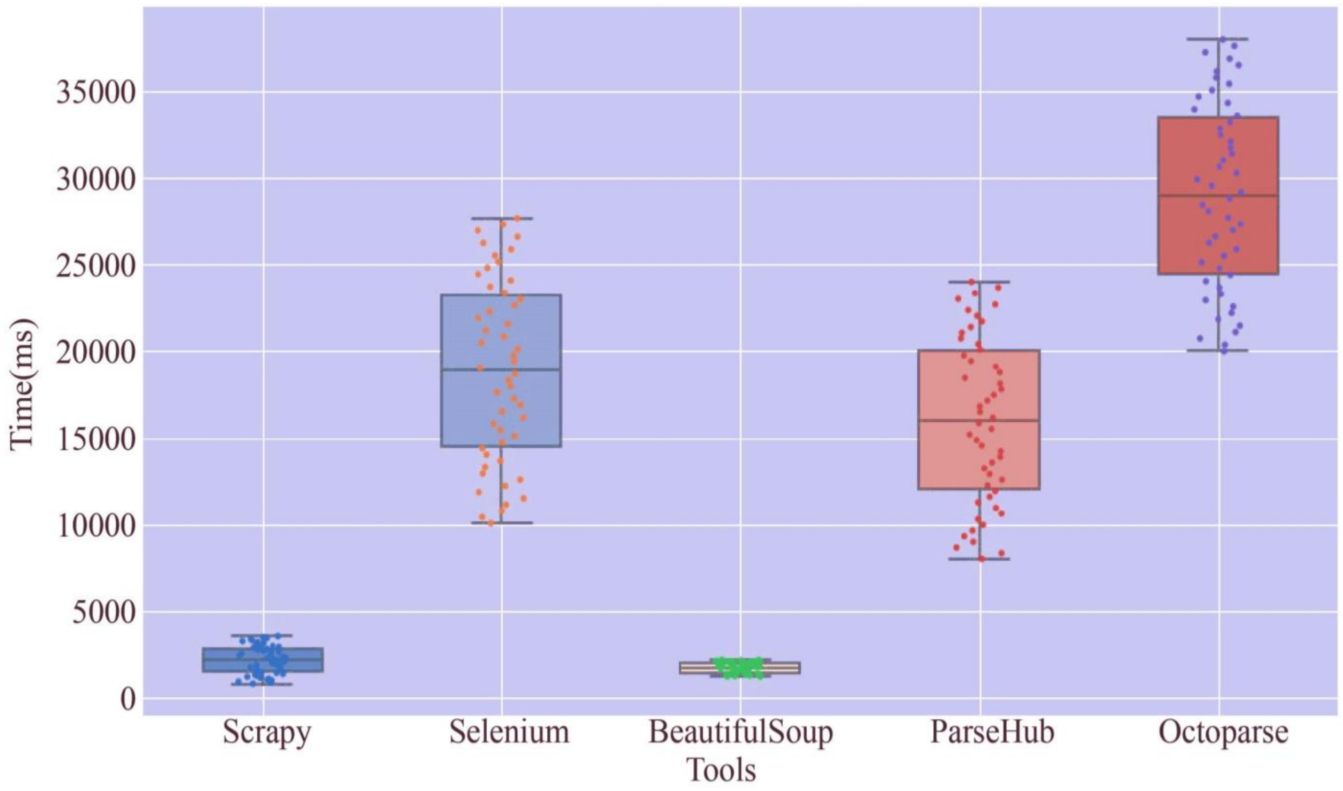

Fig. 3 Runtime graph for Jagbani news

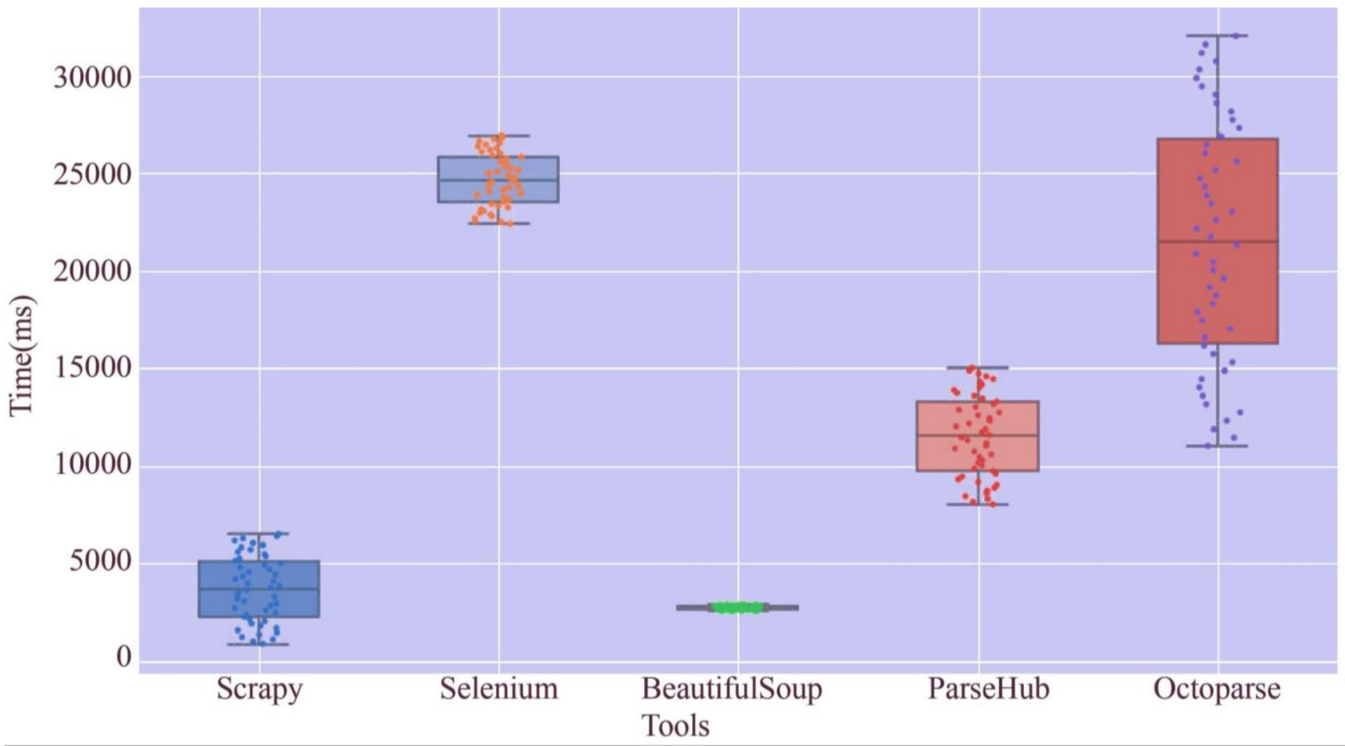

Fig. 4 Runtime graph for SGPC

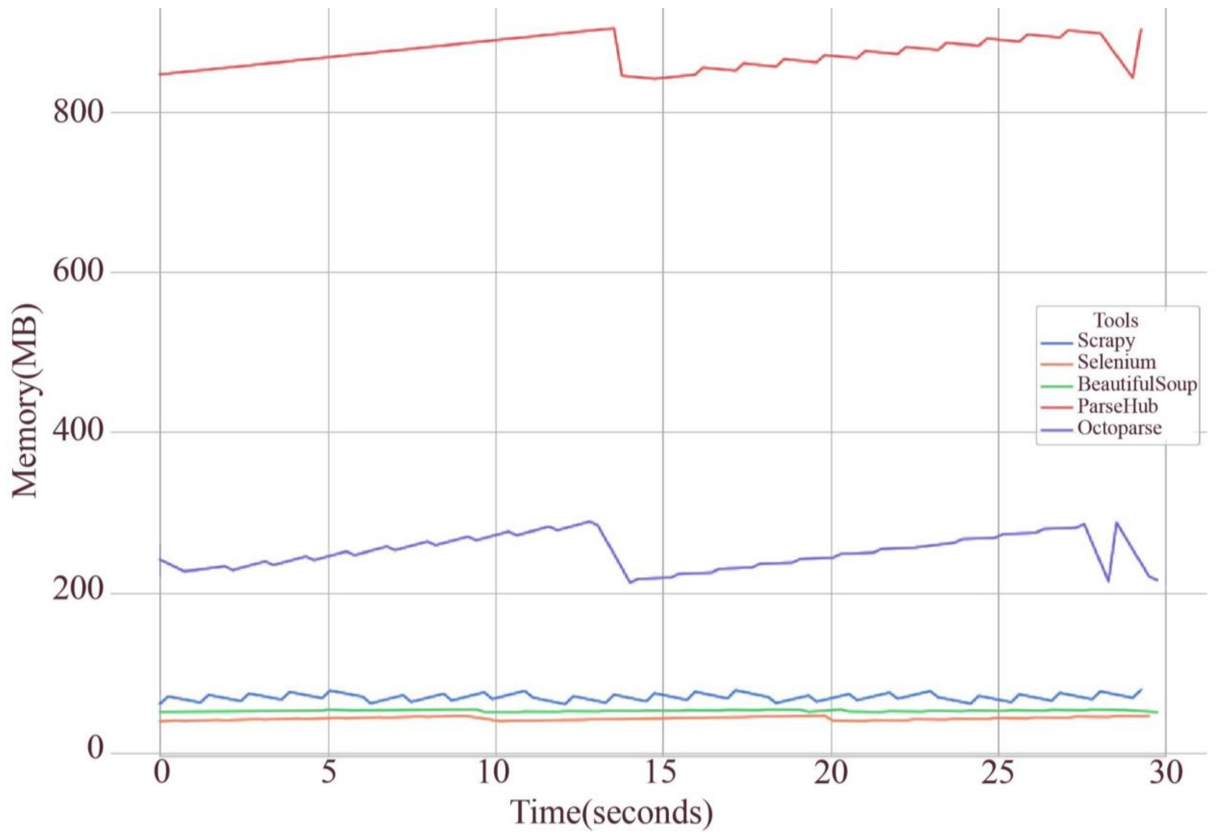

Fig. 5 Memory usage graph for BBC News Punjabi

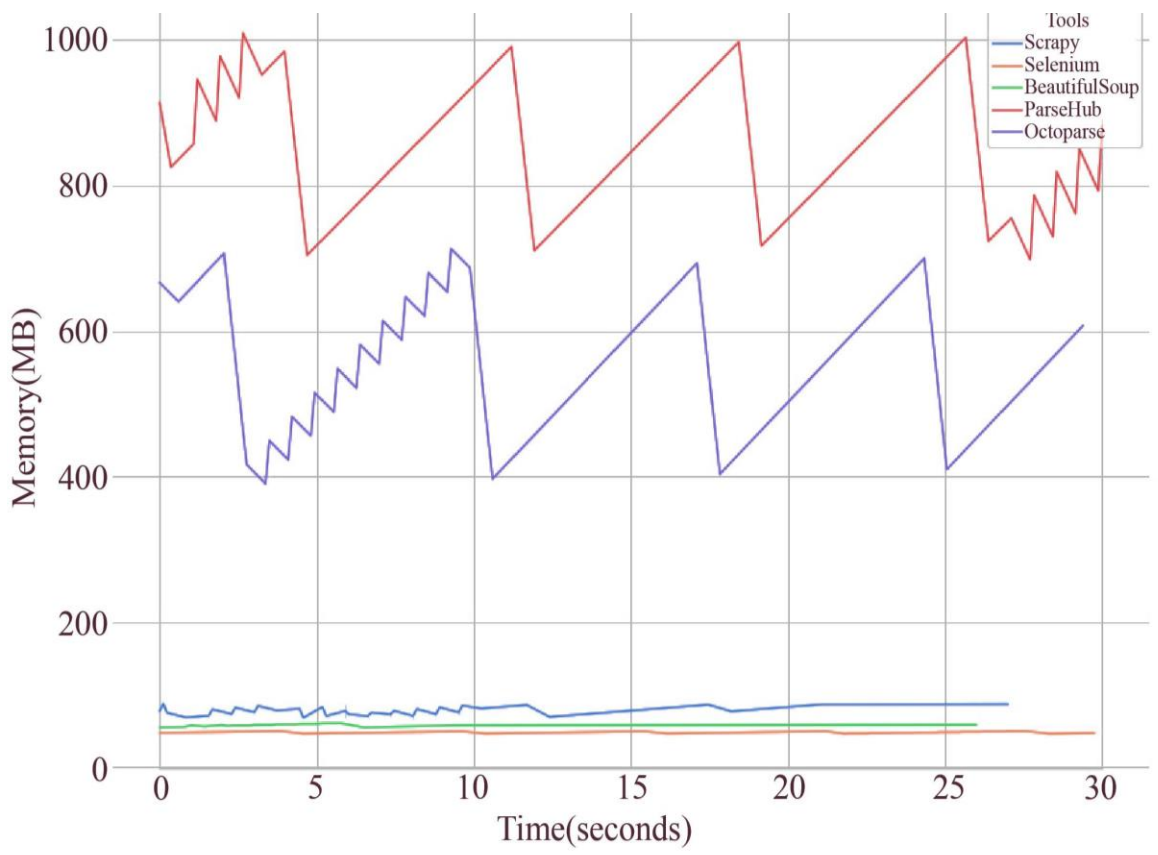

Fig. 6 Memory usage graph for Jagbani news

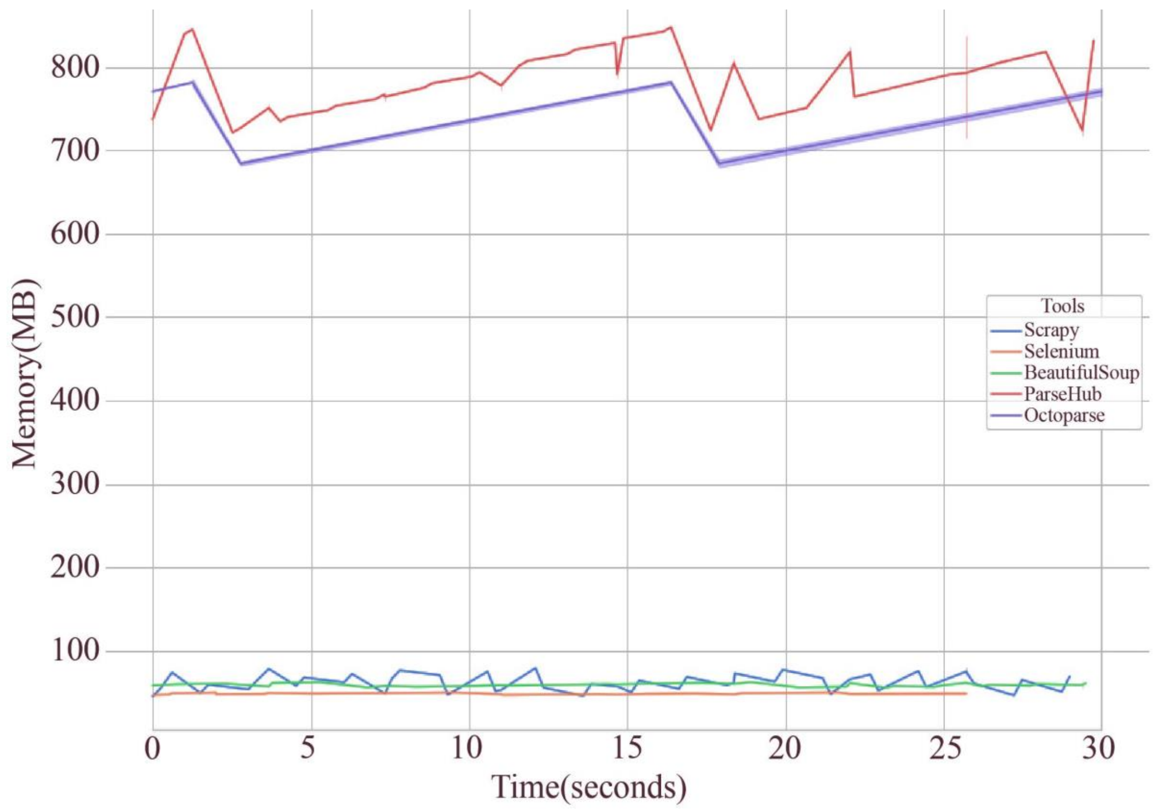

Fig. 7 Memory Usage for SGPC

5.1. Performance

In the previous section, the average time taken to scrape data from BBC, Jagbani, and SGPC websites is presented in the table. In this section, the time of the individual runs of the selected tools has been presented in the form of a box graph. It is clearly noticeable from all three graphs[9] presented in Figures 1 to 3 that amongst the Python libraries, BS4 takes the least time to perform scraping, followed by Scrapy and Selenium. The software tools, on the other hand, take significantly more time to scrape the data from the respective websites.

The second performance parameter evaluated is memory usage. In the previous section, a table depicting average memory usage values while scraping the selected websites was used. In this section, Figures 4 to 6 portray the individual memory usage values of each tool for BBC, Jagbani and SGPC websites. The wavy graphs starkly depict that memory usage of software tools is significantly greater than as done by Python tools. Thus, it is illustrated clearly from the above discussion and graphs that the free version of software tools falls short in performance over open-source Python-based libraries and frameworks.

5.2. Features

As observed in our case, BeautifulSoup library and Scrapy framework can scrape everything that is displayed on the screen but are unable to extract information (in this case, news items) hidden behind the “load more” button on Jagbani and BBC websites. On the contrary, Selenium easily emulates human behaviour and can extract dynamically loaded JavaScript code. Similarly, scrapy and BS4 were unable to scrape the JavaScript content from the home page of the SGPC website, which was thus extracted after manually clicking on the buttons. Though BS4 and Scrapy are unable to handle JavaScript-loaded pages, alternatives such as *dryscrape* [71] for BS4 and *splash* [72] with scrapy can be used to complete the desired task. Between OctoParse and ParseHub, OctoParse could perform the scraping of the Javascript content from all three websites, thus also affecting the average run time, which for all websites is more in the case of OctoParse. There was no JavaScript support provided by the free version of ParseHub tool.

5.3. Ease of Use

In our study, we have chosen three evaluation parameters to measure the ease of using the selected tools. Firstly, dependencies are checked for each tool to determine which tools can be used independently and which require third-party support to complete the scraping. Table 3 depicts the required dependencies for each tool.It is observed from the table that though software tools lag in runtime and memory usage properties, they are independent and can scrape entire websites without requiring any mediator to perform the task.

Although, the features offered by the priced plans of the software are far more advanced than those offered by the free plans.The second parameter for evaluating ease of use is documentation and tutorials. It is observed that necessary documentation is available for all the tools selected, using which we learned and practiced performing the scraping tasks. Several tutorials offered video lectures to understand and do hands-on scraping of the websites. It is noticed that available documentation and tutorials for OctoParse and ParseHub are easier to understand and implement by people from varied backgrounds.

Lastly, the ease of installation for all the tools is checked to foresee the simplicity of using them. All Python programming tools can be installed using the Pip install command, and different packages which, when required can be installed using the same command. As mentioned in the previous section, OctoParse and ParseHub can be installed like any other software.Thus, it can be concluded that where ease of use is concerned, OctoParse and ParseHub are easier than programming language tools. They are easy to learn from the available documentation, require no extra dependency, and do not involve technical installation.

In addition, the usage of software tools does not require knowledge of any programming language and can be used for extracting data from the web by people from non-technical backgrounds as well.

5.4. Reliability

We have considered two parameters to evaluate the reliability of the tools used. First is GitHub repository metrics, which is a factor which portrays how developers view the project quality. Table 5 in the previous section metric depicts values for tools. All values have been manually collected, and they depict several times the repository of the tool, which has been starred, watched, and copied by various developers.Less data was found in the GitHub repositories of OctoParse and ParseHub, which could be, as unlike programming language tools, these tools are not available for free and thus are not popular. As for the code quality metrics of the BS4, Selenium and Scrapy, we have measured CC and MI values. CC measures the complexity of the code, and according to McCabe’s ranges of complexity, the calculated CC values indicate that complexity for all three tools is not high and are, therefore, simple procedures [73]. As far as MI is concerned, the score for the SGPC website is high for all the tools, indicating ease of maintaining the code with respect to the other websites. However, for all tools, MI values are greater than 65, thus indicating that scraping codes developed using all three tools range between moderately maintainable and highly maintainable [46].

In this section, different aspects of scraping tools have been discussed under three broad categories with an aim to encompass major aspects faced by scraper while extracting data from the web. The web scraping review proposed by Khder [7] discusses different facets of web scraping and gives a comparative analysis between Python libraries like BeautufulSoup, lxml and Regular expressions; however, the analysis is only limited to performance and ease of installation and use of the tool. Sirisuriya [41], in his study, gives a review of different web scraping methods and software; however, the comparison between the methods and software is limited to operating systems and data export formats. Glez Pena et al. [8] give a larger perspective on various web scraping methods, and the comparison of the methods is also based on a wide variety of parameters. However, the study revolves around biomedical data extraction, which has been chosen as the subject of extraction and the application of just one tool, i.e. jarvest, has been exemplified for the extraction of data. Similarly, a study proposed by Gunawan et al. [9] also compares different web scraping methods on different parameters such as time, memory, and data usage; however, all tools are tested on a single website. The comparative analysis proposed in this study aims to provide users with a comprehensive view of different web scraping methods. Three websites have been chosen to test and compare the selected web scraping tools and software.

Furthermore, the exemplification is on websites based on the Punjabi Language. Hence, this study will help readers interested in working in regional languages to examine which tool to work with depending on the comparative parameters discussed in this and previous sections.

6. Conclusion and Future Work

This paper compares the most accepted web scraping tools to scrape "low-resource language" text from the web. Two Punjabi language newspaper websites and one organization website were chosen to scrape. The evaluation parameters significantly indicate the balance between selected tools for the reader to easily select an appropriate tool before scraping any kind of data. The paper stipulates that programming language tools are more customizable in comparison to desktop environment-based tools.

Among the programming language tools, BS4 is a more appropriate tool when it comes to scraping text from the kind of websites selected for this paper. However, Scrapy and BS4 are not able to scrape the JavaScript content of the pages for which Selenium is required as a necessary tool. For ParseHub and OctoParse, we have chosen free plans for both tools; thus, we have faced several limitations while scraping. Despite the limitations, both tools are automated in their work and are more suitable for people with no or limited programming background. As for the future, more tools pertaining to programming and software backgrounds can be compared. The scraping of "low-resource language" text has seen fewer contributions in the past. Thus, tools specifically calibrated for scraping Punjabi text can be created by performing relevant comparison studies among the present tools.

References

[1] Seppe vanden Broucke, and Bart Baesens, *Practical Web Scraping for Data Science*, pp. 155–172, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[2] Poojitha Thota, and Elmasri Ramez, "Web Scraping of COVID-19 News Stories to Create Datasets for Sentiment and Emotion Analysis," *Proceedings of the 14th PErvasive Technologies Related to Assistive Environments Conference*, pp. 306–314, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[3] Lucy Linder et al., "Automatic Creation of Text Corpora for Low-Resource Languages from the Internet: The Case of Swiss German," *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pp. 2706–2711, 2020. [Google Scholar] [Publisher Link]

[4] Fantahun Gereme et al., "Combating Fake News in 'Low-Resource' Languages: Amharic Fake News Detection Accompanied by Resource Crafting," *Information*, vol. 12, no. 1, pp. 1–9, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[5] Navdeep Singh et al., "DeepSpacy-NER: An Efficient Deep Learning Model for Named Entity Recognition for Punjabi Language," *Evolving Systems*, vol. 14, pp. 673-683, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[6] Gurjot Singh Mahi, and Amandeep Verma, "Development of Focused Crawlers for Building Large Punjabi News Corpus," *Journal of ICT Research and Applications*, vol. 15, no. 3, pp. 205–215, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[7] Moaiad Ahmad Khder, "Web Scraping or Web Crawling : State of Art, Techniques, Approaches and Application," *International Journal of Advances in Soft Computing and its Application*, vol. 13, no. 3, pp. 144-168, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[8] Daniel Glez-Peña et al., "Web Scraping Technologies in an API World," *Briefings in Bioinformatics*, vol. 15, no. 5, pp. 788–797, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[9] Rohmat Gunawan et al., "Comparison of Web Scraping Techniques : Regular Expression, HTML DOM and Xpath," *Proceedings of the 2018 International Conference on Industrial Enterprise and System Engineering*, vol. 2, pp. 283–287, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[10] SCM de S. Sirisuriya, "A Comparative Study on Web Scraping," *8th International Research Conference*, pp. 135–140, 2015. [Google Scholar] [Publisher Link]

[11] Y. Yang, L.T. Wilson, and J. Wang, "Development of an Automated Climatic Data Scraping, Filtering and Display System," *Computers and Electronics in Agriculture*, vol. 71, no. 1, pp. 77–87, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[12] Charmaine Bonifacio et al., "CCDST: A Free Canadian Climate Data Scraping Tool," *Computers and Geosciences*, vol. 75, pp. 13–16, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[13] Tony Grubesic, and Matthew Zook, "A Ticket to Ride: Evolving Landscapes of Air Travel Accessibility in the United States," *Journal of Transport Geography*, vol. 15, no. 6, pp. 417–430, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[14] Wendy Fangyu Hsu, "Mapping the Kominas' Sociomusical Transnation: Punk, Diaspora, and Digital Media," *Asian Journal of Communication*, vol. 23, no. 4, pp. 386–402, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[15] Lefteris Angelis, Nick Bassiliades, and Yannis Manolopoulos, "On the Necessity of Multiple University Rankings," *COLLNET Journal of Scientometrics and Information Management*, vol. 13, no. 1, pp. 11–36, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[16] Kenneth Reitz, Requests Package, Python-Requests. [Online]. Availabe: https://docs.python-requests.org/en/master/

[17] Pandas. [Online]. Availabe: https://pandas.pydata.org/

[18] McNamara, XlsWriter. [Online]. Availabe: https://xlsxwriter.readthedocs.io/

[19] Isabelle Krebs et al., "Non-Journalistic Competitors of News Media Brands on Google and Youtube : From Solid Competition to a Liquid Media Market," *Journal of Media Business Studies*, vol. 18, pp. 27-44, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[20] Beautiful Soup Documentation, Crummy. [Online]. Availabe: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[21] Saram Han, and Christopher K. Anderson, "Web Scraping for Hospitality Research: Overview, Opportunities, and Implications," *Cornell Hospitality Quarterly*, vol. 62, no. 1, pp. 89–104, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[22] Nick H.K. Or, "How Policy Agendas Change when Autocracies Liberalize: The Case of Hong Kong, 1975–2016," *Public Administration*, vol. 97, no. 4, pp. 926–941, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[23] Steffen Ganghof et al., "Do Minority Cabinets Govern More Flexibly and Inclusively? Evidence from Germany," *German Politics*, vol. 28, no. 4, pp. 541–561, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[24] Giorgia Chinazzo, "Investigating the Indoor Environmental Quality of Different Workplaces through Web-Scraping and Text-Mining of Glassdoor Reviews," *Building Research and Information*, vol. 49, no. 6, pp. 695–713, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[25] Hadley Wickham, Rvest. [Online]. Availabe: https://rvest.tidyverse.org/

[26] Lukasz Wiechetek, Kongkiti Phusavat, and Zbigniew Pastuszak, "An Analytical System for Evaluating Academia Units Based on Metrics Provided by Academic Social Network," *Expert Systems with Applications*, vol. 159, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[27] Alex Bradley, and Richard J.E. James, "Web Scraping Using R," *Advances in Methods and Practices in Psychological Science*, vol. 2, no. 3, pp. 264–270, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[28] Karina Sokolova, and Charles Perez, "The Digital Ingredients of Donation-Based Crowdfunding. A Data-Driven Study of Leetchi Projects and Social Campaigns Data-Driven Study of Leetchi Projects and Social Campaigns," *Journal of Decision Systems*, vol. 27, no. 3, pp. 146–186, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[29] Ueli Reber, "Overcoming Language Barriers: Assessing the Potential of Machine Translation and Topic Modeling for the Comparative Analysis of Multilingual Text Corpora," *Communication Methods and Measures*, vol. 13, no. 2, pp. 102–125, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[30] Vidhi Singrodia, Anirban Mitra, and Subrata Paul, "A Review on Web Scrapping and its Applications," *International Conference on Computer Communication and Informatics*, pp. 1–6, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[31] Manish Kumar, Rajesh Bhatia, and Dhavleesh Rattan, "A Survey of Web Crawlers for Information Retrieval," *Wiley Interdisciplinary Reviews Data Mining Knowledge Discovery*, vol. 7, no. 6, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[32] Irvin Dongo et al., "A Qualitative and Quantitative Comparison between Web Scraping and API Methods for Twitter Credibility Analysis," *International Journal of Web Information Systems*, vol. 17, no. 6, pp. 580–606, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[33] Jay M. Patel, *Getting Structured Data from the Internet*, pp. 1-397, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[34] Giulio Barcaroli et al., "Use of Web Scraping and Text Mining Techniques in the Istat Survey on 'Information and Communication Technology in Enterprises,'" *European Conference on Quality in Official Statistics*, pp. 1–13, 2014. [Google Scholar] [Publisher Link]

[35] Katy Jordan, "Validity, Reliability, and the Case for Participant-Centered Research: Reflections on a Multi-Platform Social Media Study," *International Journal of Human–Computer Interaction*, vol. 34, no. 10, pp. 913–921, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[36] Freia McGregor et al., "Social Media Use by Patients with Glaucoma: What Can we Learn?," *Ophthalmic Physiol. Optics*, vol. 34, no. 1, pp. 46–52, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[37] Chinonso E. Etumnu et al., "Does the Distribution of Ratings Affect Online Grocery Sales? Evidence from Amazon," *Agribusiness*, vol. 36, no. 4, pp. 501–521, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[38] Tamar Wilner, and Avery Holton, "Breast Cancer Prevention and Treatment: Misinformation on Pinterest, 2018," *American Journal of Public Health*, vol. 110, pp. S300–S304, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[39] Jiwon Ryu, and Gerard Kim, "Interchanging the Mode of Display Between Desktop and Immersive Headset for Effective and Usable On-line Learning," *International Conference on Intelligent Human Computer Interaction*, vol. 12615, pp. 218-222, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[40] Every Day Our Web Scraping Solutions Turn Millions of Web Pages into Data for Sales Marketing Finance Operations, Mozenda. [Online]. Availabe: www.mozenda.com

[41] SCM De S Sirisuriya, "*A Comparative Study on Web Scraping,*" Proceedings of 8th International Research Conference, KDU, pp. 135–140, 2015. [Google Scholar] [Publisher Link]

[42] Mihai Gheorghe, Florin-Cristian Mihai, and Marian Dârdală, "Modern Techniques of Web Scraping for Data Scientists," *Revista Romana de Interactiune Om-Calculator*, vol. 11, no. 1, pp. 63–75, 2018. [Google Scholar] [Publisher Link]

[43] Time Access and Conversions, Python:Time. [Online]. Availabe: https://docs.python.org/3/library/time.html

[44] Psutil 5.9.7, Psutil. [Online]. Availabe: https://pypi.org/project/psutil/

[45] Mprofile 0.0.15, Mprofile. [Online]. Availabe: https://pypi.org/project/mprofile/

[46] Tim Gilboy, "Maintainability Index - What is it and Where does it Fall Short?," *Sourcery*, 2022. [Online]. Available: https://sourcery.ai/blog/maintainability-index/

[47] Alex Luscombe, Kevin Dick, and Kevin Walby, "Algorithmic Thinking in the Public Interest: Navigating Technical, Legal, and Ethical Hurdles to Web Scraping in the Social Sciences," *Quality and Quantity*, vol. 56, pp. 1023-1044, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[48] Build and Run Your Web Spiders, Scrapy. [Online]. Availabe: https://scrapy.org/

[49] Christof Ebert et al., "Cyclomatic Complexity," *IEEE Software*, vol. 33, no. 6, pp. 27–29, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[50] Huy Nguyen et al., "Exploring Metrics for the Analysis of Code Submissions in an Introductory Data Science Course," *ACM 11th International Learning Analytics and Knowledge Conference*, pp. 632–638, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[51] Neelam Singh et al., "µBIGMSA-Microservice-Based Model for Big Data Knowledge Discovery: Thinking Beyond the Monoliths," *Wireless Personal Communications*, vol. 116, no. 4, pp. 2819–2833, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[52] Andri Muhyidin, Muhammad Adi Febri Setiawan, and Nurkhamid, "Developing UNYSA Chatbot as Information Services about Yogyakarta State University," *Journal of Physics: Conference Series*, vol. 1737, pp. 1-9, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[53] Anita Ganpati, Arvind Kalia, and Hardeep Singh, "A Comparative Study of Maintainability Index of Open Source Software," *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, no. 10, pp. 228–230, 2012. [Google Scholar] [Publisher Link]

[54] Radon's Documentation, Radon. [Online]. Availabe: https://radon.readthedocs.io/en/latest/

[55] Breno Santana Santos et al., "COVID-19 : A Scholarly Production Dataset Report for Research Analysis," *Data in Brief*, vol. 32, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[56] Katharine Frazier, Hilary Davis, and John Vickery, "Seeing the Forest for Trees: Tools for Analyzing Faculty Research Output," *Serials Review*, vol. 46, no. 3, pp. 184–189, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[57] Johannes I. Single, Jürgen Schmidt, and Jens Denecke, "Knowledge Acquisition from Chemical Accident Databases Using an Ontology- Based Method and Natural Language Processing," *Safety Science*, vol. 129, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[58] Clément Nicolas, Jinwoo Kim, and Seokho Chi, "Natural Language Processing-Based Characterization of Top-Down Communication in Smart Cities for Enhancing Citizen Alignment," *Sustainable Cities and Society*, vol. 66, pp. 1-15, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[59] Geoff Boeing, and Paul Waddell, "New Insights into Rental Housing Markets across the United States: Web Scraping and Analyzing Craigslist Rental Listings," *Journal of Planning Education and Research*, vol. 37, no. 4, pp. 457–476, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[60] Stephanie Lunn, Jia Zhu, and Monique Ross, "Utilizing Web Scraping and Natural Language Processing to Better Inform Pedagogical Practice," *Proceedings of IEEE Frontiers in Education Conference*, pp. 1-9, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[61] Nichapat Sangkaew, and Hongrui Zhu, "Understanding Tourists' Experiences at Local Markets in Phuket: An Analysis of TripAdvisor Reviews," *Journal of Quality Assurance in Hospitality & Tourism*, vol. 23, no. 1, pp. 89–114, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[62] Haiyang Ai, and Xiaoye You, "The Grammatical Features of English in a Chinese Internet Discussion Forum," *World Englishes*, vol.

34, no. 2, pp. 211–230, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[63] Elif Ensari, and Bilge Kobaş, "Web Scraping and Mapping Urban Data to Support Urban Design Decisions," *A/Z ITU Journal of Faculty Architect*, vol. 15, no. 1, pp. 5–21, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[64] Kylie L. Anglin, "Gather-Narrow-Extract: A Framework for Studying Local Policy Variation Using Web-Scraping and Natural Language Processing," *Journal of Research on Educational Effectiveness*, vol. 12, no. 4, pp. 685–706, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[65] Cristin M. Hall, Nicole C. Breeden, and Nicklaus Giacobe, "Gone Viral: Content Characteristics and Relative Quality of Highly Shared School Psychology-Related Content on Pinterest," *Psychology in the Schools*, vol. 56, no. 6, pp. 959–976, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[66] Peter J. Franz et al., "Using Topic Modeling to Detect and Describe Self-Injurious and Related Content on a Large-Scale Digital Platform," *Suicide Life-Threatening Behavior*, vol. 50, no. 1, pp. 5–18, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[67] Igor Mintz et al., "Individuals with Back and Neck Pain on Medical Forums: What do they Mention? What do they Fear?," *European Journal of Pain*, vol. 24, no. 10, pp. 1915–1922, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[68] Laurence Sophie Jouaville, Tulika Paul, and Mariana Ferreira Almas, "A Review of the Sampling Methodology Used in Studies Evaluating the Effectiveness of Risk Minimisation Measures in Europe," *Pharmacoepidemiology and Drug Safety*, vol. 30, no. 9, pp. 1143–1152, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[69] Burcu B. Keskin et al., "Cracking Sex Trafficking: Data Analysis, Pattern Recognition, and Path Prediction," *Production Operations Management*, vol. 30, no. 4, pp. 1110–1135, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[70] Maria D. Molina, and S. Shyam Sundar, "Can Mobile Apps Motivate Fitness Tracking? A Study of Technological Affordances and Workout Behaviors," *Health Communication*, vol. 35, no. 1, pp. 65–74, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[71] Dryscrape 1.0, Dryscrape. [Online]. Availabe: https://pypi.org/project/dryscrape

[72] plash - A Javascript Rendering service, Splash. [Online]. Availabe: https://splash.readthedocs.io/en/stable/

[73] T. McCabe, "Software Quality Metrics to Identify Risk," 2008. [Google Scholar]