Survey: Web Data Scrapers

Suhas Khandiga Suresh
Arizona State University
12118393761

Abhee Parekh
Arizona State University
1217116095

Rajashekar Reddy Aluka
Arizona State University
1217211645

Subha Pattanayak
Arizona State University
1218503208

Vivek Kumar Maskara
Arizona State University
1218403147

Vishal Singh Deepak
Arizona State University
1217145267

Sravani Gandla
Arizona State University
1218471345

***Abstract*—Web scraping is a process of automating the extraction of data in an efficient and fast way. The primary aim of Web Scraping is to extract data from one or more websites and process it into simple human-understandable structures like spreadsheets, databases, CSV files, etc. Apart from being a very complicated task, Web Scraping is also resource and time consuming, especially if it is carried out manually. The purpose of this project is to compare and contrast the different Web Scraping tools in the market and their features like ease of scraping, total time taken to scrape the data, and identifies the capabilities and limitations of each of them. This project should give the reader an insight into various scraping frameworks and how to go about selecting one of them for use in their application.**

***Index Terms*—Web Scraping, Scrapy, Selenium, Puppeteer, Facebook, Reddit, Twitter**

I. Introduction

The world wide web is one of the major source of information for many professionals in various sectors. It contains useful and useless, structured and non-structured information, in different formats, and from various sources. The ease of accessibility of the Internet and the ever-increasing popularity of Social Media has fueled the generation of an enormous amount of data. Below are some of astonishing facts based on the digital global overview report published by datareportal [4], that show how the Internet is changing the world:

- **53.6%** of the world's population uses social media. The average daily usage is **2 hours and 25 minutes**.
- **4.66 billion** people around the world now use the internet, of those users, **316 million** new users have come online within the last 12 months.
- **5.22 billion** unique mobile users.
- Facebook is the most used Social media platform with around **2.740 billion** users.
- Twitter has **353 million** users and **192 million** daily active users.

Variety of techniques exists to retrieve data from a web page: Human copy-and-paste, Text pattern matching, HTTP programming, HTML parsing, DOM parsing, Vertical aggregation, Semantic annotation recognizing, Computer vision web-page analysis and Web-Scraping. Among these Application Programming Interface given by the Social media sites and Web Scraping is considered to be advanced techniques to collect data from the web. A web scraper is, a software that simulates human browsing on the web to collect detailed information from different websites. The advantage of a scraper resides on its speed and its capacity to be automated and programmed. However, no matter what technique is used, the goal still remains the same: collect the information from the Web and present it in a structured human readable format. This project reviews existing scraping frameworks for their ease of scraping, time taken to scrape the data, and identifies the capabilities and limitations of each of them.

II. Problem Statement

This project aims to study various scraping tools and perform an extensive comparison between them based on various metrics such as - performance, ease of use, time is taken to scrape the data, languages, and platform supported, etc. To be able to compare the results generated by different tools, we would be scrapping similar data using different scraping tools from social media platforms.

Web Scraping is a process of automating the extraction of data from the Web in an efficient and fast way. It has applications in various domains like data mining, sentiment analysis over social media websites, etc.

To conduct our research and perform the experiments, we have selected three social media platforms - **Twitter**, **Reddit**, and **Facebook**. We would be scrapping data from these platforms using three major scrapping tools. After doing extensive research and survey about different options available for scraping tools, we decided upon using **Scrapy**, **Selenium**, and **Pupetter**. The objective of this study is to scrape similar data using the different scraping tools, so that we can compare the results. This

would help us perform an extensive comparison among various scrapping tools based on multiple data points.

III. Related Works

Ramírez et al. [6] discuss a data collection process by merging data collection tools (Selenium and Scrapy) to overcome the limitations of each of these tools. The work transformed the data collection process from collecting structured data to collecting semantic data. Shi et al. [7] discuss an incremental web crawling approach using bloom filter to crawl popular news pages using Scrapy framework. This avoided repeated crawling of web page links. Wang et al. [8] analyzed online marketing transactions in e-commerce. The authors further discussed the challenges faced to crawl Taoboa using the Scrapy framework and studied the relationship between buyers and sellers connected by items. Meschenmoser et al. [5] discuss challenges and solutions for scraping scientific documents. The authors provide a way to make the performance metrics of scientific publications transparent and verifiable by providing the data in scientific Web Repositories. Emilio et al. [3] provide a survey about various techniques used in Web Data Extraction. The authors' groups the applications of web data scraping as either Enterprise Level or Social Web Level. The survey discusses the possibility of cross-fertilization, that is, re-using a given Web Data Scraping technique in multiple domains. Balduzzi et al. [1] provide an approach to take advantage of social media sites such as Facebook to scrap personal user information. This method can provide a user profile based on an email address. This would allow attackers to launch targeted attacks or improve the efficiency of spam campaigns. Baumgartner et al. [2] discuss wrapper technologies that create unified services by integrating data from Web Applications and Web Services. The authors describe the Lixto approach and present Lixto Suite as an example of Web Process Integration.

In this project, we provide a survey of data collection tools namely: Selenium, Scrapy, and Puppeteer for extracting data from Social Media websites. The tools are selected based on the popularity of these tools found from the literature survey. In this survey, we will discuss the advantages and limitations of each of these tools in scraping various social media websites. Popular social media websites such as Facebook, Twitter, and Reddit are used to study the tools. These websites are selected due to a large number of applications involving scraping these websites. The goal of the survey is to help users select appropriate web scraping based on the application. The survey will benefit users to select appropriate scraping tools in their application.

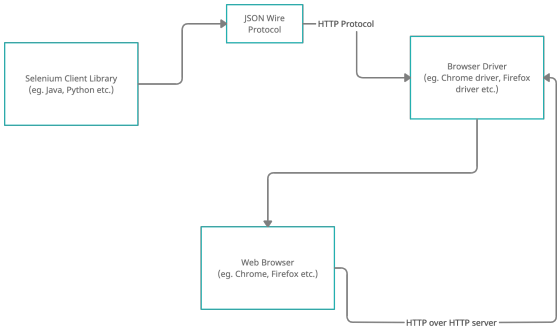


Fig. 1: Selenium architecture

IV. System Architecture & Algorithms

A. *Selenium*

Selenium provides a robust web automation framework with open open source APIs available for multiple platforms and programming languages. Selenium WebDriver's APIs provides capabilities to automate user actions using most of the popular web browsers. WebDrivers allows mimicking of real-world scenarios and makes the tests agnostic to the technology used for developing the website. Moreover, Selenium supports multiple programming languages which allows scraping using user's preferred language.

Selenium Web Driver is comprised of multiple components which are required for web scraping. Fig. 1 shows the different components of Selenium. It contains 4 main components:

- **Selenium Client Library**: Selenium supports multiple libraries such as Java, Ruby, Python, etc. Selenium Developers have developed language bindings to allow Selenium to support multiple languages.
- **JSON Wire Protocol Over HTTP**: JSON stands for JavaScript Object Notation. It is used to transfer data between a server and a client on the web. JSON Wire Protocol is a REST API that transfers the information between HTTP servers. Each BrowserDriver (such as FirefoxDriver, ChromeDriver, etc.) has its own HTTP server.
- **Client Browser Drivers**: Each browser contains a separate browser driver. Browser drivers communicate with the respective browser without revealing the internal logic of the browser's functionality. When a browser driver has received any command then that command will be executed on the respective browser and the response will go back in the form of an HTTP response.
- **Browsers**: Selenium supports multiple web browsers like Firefox, Chrome, IE, Safari, etc.

When a Web Driver code is executed, an HTTP request is generated and it is sent to the respective

web browser. The browser driver receivers the request through a HTTP server and decides on the actions which needs to be executed on the browser. The browser in turn executes these instructions and returns the execution status to the HTTP server via the driver. The status is then relayed to the scraping script which displays the result.

B. Scrapy

Scrapy is an application framework written in Python for collecting website data and extracting structured data [9]. Scrapy is commonly used in many applications, such as data mining, information processing or storage. Scrapy implements many complex operations, including concurrent requests, no login, and URL deduplication. Scrapy has advanced and simple design concepts on the main operating system platforms, high efficiency, scalability, high portability and good performance (Gong Baosheng, 2016). Scrapy contains 5 main components as shown in Fig 2.

- **Scrapy Engine**:The engine manages the data flow between all components of the system
- **Scheduler**: The Scheduler accepts requests from the engine and requests them to be fed later when the engine requests them.
- **Downloader**: It retrieves the web pages and feeds them to the Scrapy engine.
- **Spiders**: Spiders are custom classes written by the end users to parse extract objects and responses.
- **Item Pipeline**: It processes the items once the items are being extracted by the spider.

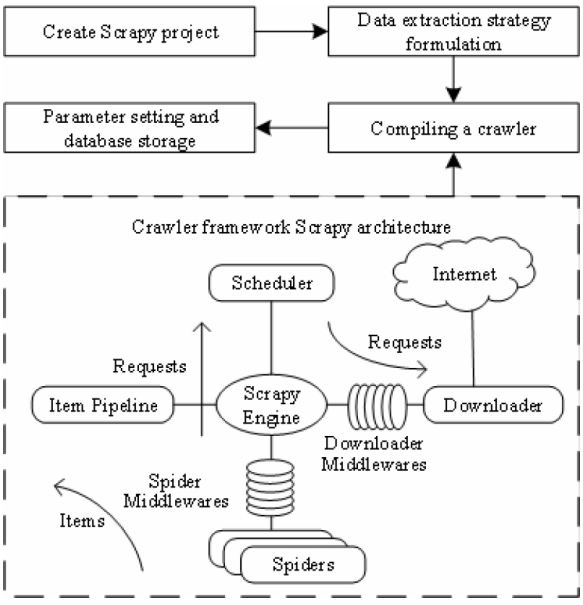


Fig. 2: Scrapy Architecture

For example to extract tweet text from twitter for HASHTAG (elonmusk) we first load the page , identify the corresponding HTML tag id and location (Fig 3) and finally extract the field using xpath, CSS and regular expression (Fig 4).

Fig. 3: HTML response for a given twitter HASHTAG

Fig. 4: Scraping Tweeter text using Scrapy

C. Puppeteer

Puppeteer is a **Node.js** library which controls the browser using **chrome developer protocols**. By default Puppeteer installs chromium browser to do the browser automation work. It can also be configured to run on a different set of browsers like Chrome and Firefox. Puppeteer runs by default in headless mode but it also can be run in non-headless mode.

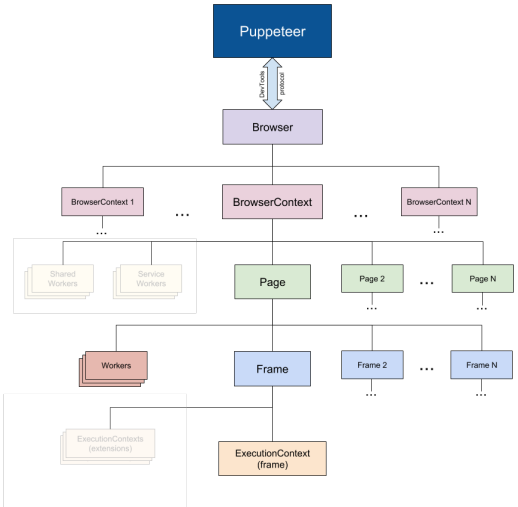


Fig. 5: Puppeteer Architecture

Different components of Puppeteer architecture:

- **Puppeteer API** communicates with the browser with the **DevTools Protocol.**
- A **Browser** creates a **BrowserContext** instance and has a browsing session. **Browser** can have multiple **BrowserContexts** and each has its own individual session where it saves the session details and can own multiple pages.
- **Page** has at least one **frame**: main frame and other frames. Each frame has at least one execution context where the **frame's JavaScript** is executed.
- **Worker** has a single execution context and facilitates interacting with **Web Workers**.

Lets take an example to scrape the data from wikipedia using puppeteer. In Fig.6, to scrape the highlighted title, we gonna need its HTML tag id or the class to find its location.

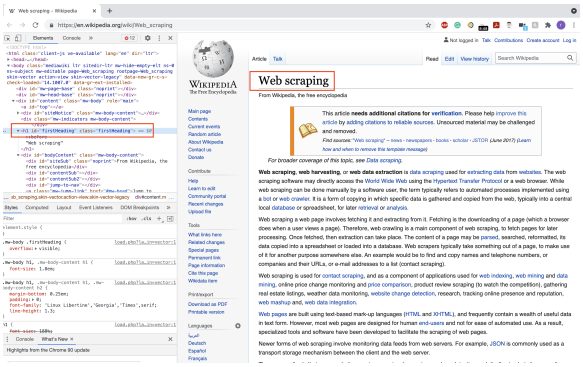
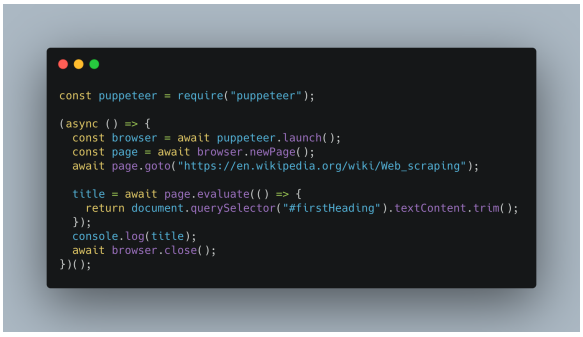


Fig. 6: Wikipedia Page

In Fig. 7, to scrape the data first we launch the browser, then create a page and go to the desired URL i.e. wikipedia page. Once the page is loaded we will search for the **#firstHeading** as highlighted in Fig. 6 to get the title. Finally we print/store the extracted details and close the browser.



```
const puppeteer = require("puppeteer");

(async () => {
  const browser = await puppeteer.launch();
  const page = await browser.newPage();
  await page.goto("https://en.wikipedia.org/wiki/Web_scraping");

  title = await page.evaluate(() => {
    return document.querySelector("#firstHeading").textContent.trim();
  });
  console.log(title);
  await browser.close();
})();
```

Fig. 7: Scrapping data from wikipedia using Puppeteer

V. Datasets (Descriptions, Sizes and Preprocessing Steps)

Based on our survey of social media sites, we have decided to scrape the following platforms for our project:

- Twitter
- Reddit
- Facebook

Data Types: User ID's, Hyperlink (@, #), post contents, comments/replies, popularity metrics such as upvotes, likes and shares

Data Collection Process:

- Firstly, we have prepare the execution of the scraping agent.
- Secondly , **Data extraction strategy** was formulated according to the defined target platform. The data extraction strategy selected using **Regular expression** , **CSS selectors** and **xpath**(Fig 8).
- Once it is ready, we have started the process of data extraction process from the chosen platform.
- We have collected the raw data from the platform until the extraction process ends or stops.
- Finally, we have clean the extracted data and structure it in a tabular format.

```
username = tweet.xpath('.//span[@class="username u-dir u-textTruncate"]/b/text()').extract()[0]
ID = tweet.xpath('.//@data-tweet-id').extract()
tweet_text = ''.join(tweet.xpath('.//div[@class="js-tweet-text-
container"]/p//text()').extract()).replace('\' ','\'').replace(' @ ', '@')
no_retweet = tweet.css('span.ProfileTweet-action--retweet > span.ProfileTweet-
actionCount').xpath('@data-tweet-stat-count').extract()
```

Fig. 8: Extracting twitter username, ID and twitter text using CSS , xpath and regex

The extracted data for twitter platform using scrapy is shown in Fig 9.

tweet_id	username	tweet_url	tweet_text	tweet_datetime	number_of_likes	no_of_retweets	no_of_replies
1382421897636368	WealthAlways	/WealthAlways/status	Elon Musk believes	2021-04-14 12:53:58	240	63	9
1381194821293907	defeat_will	/defeat_will/status/13	What we are witness	2021-04-11 3:38:01	145	45	5
1381968671837462	_atanas_	/_atanas_/status/138	Monkey "MindPong"	2021-04-13 6:53:01	31	13	0
1382014106434203	pat5725	/pat5725/status/1382	Go visit @MattWallo	2021-04-13 9:53:34	181	69	3
1381244866571681	mvizcaino1108	/mvizcaino1108/stati	Our #dogecoin grew	2021-04-11 6:56:53	57	13	3
1380734010213683	JustInScerini	/JustInScerini/status/	# ElonMusk Puttin #	2021-04-09 21:06:55	38	12	1
1381200326771011	MustafaBurakBU2	/MustafaBurakBU2/st	# dogearmy #dogec	2021-04-11 3:59:54	14	2	0
1381220400625901	NjcarrIngton1	/NjcarrIngton1/status/	HODL!!!! Let's get #c	2021-04-11 5:19:40	63	18	1
1382467351627587	1FaithInTorunu	/1FaithInTorunu/statu	When I look at Doge	2021-04-14 15:54:36	14	4	1

Fig. 9: Twitter CSV output

VI. Evaluations (Metrics, Experiments, Findings)

A. Design of Experiments and Evaluation

- Survey open source and collaborative frameworks such as Scrapy, Selenium and Puppeteer for evaluation
- Extract data from social media platforms Facebook, Twitter and Reddit

- Verification Phase - Verify the correctness of the results obtained from different scrapers
- Evaluation Phase - (a) Time taken to scrape a given website (b) Ease of scraping the website with a particular scraper (c) Capabilities and Limitations of the scraper

B. Findings

Below were the pros and cons found for each scraping framework after evaluation

1) Selenium

Pros:

- Open source and free to use, active contributions is made to the repo.
- Provides client libraries for many different languages(Java, C, Python etc).
- Multi browser support, platform independent and functionality can be extended with wide variety of plugins.
- Easy third party integration with the likes of Maven/TestNG/Jenkins etc.
- Multi OS support (including android).
- Lightweight in terms of resources, compared to some alternative tools.

Cons:

- Can scrape for web based applications only, not windows based applications.
- The scraping process is slower.
- No built-in image comparison. generates a high amount of network traffic.

2) Puppeteer

Pros:

- Easy to set up and good documentation.
- Bi-Directional (events) – automating things like console logs is easy.
- Supported by Google.
- Javascript helps in reading and manipulating the HTML pages.

Cons:

- Limited cross-browser support—only Chrome and Firefox.
- Often needs re-implementation of testing-related tools.
- Grids (running concurrently) in production are difficult to manage and monitor.
- By default uses Chromium and not Chrome.

3) Scrapy

Pros:

- Works good on big data set. Performance is good as Scrapy can use concurrent requests.
- Extensibility is good. We can easily develop custom middleware or pipeline to add custom function, easy to maintain.

Feature	Selenium	Scrapy	Puppeteer
Javascript Support	Yes	Limited	Yes
Data Size	Ideal for smaller websites	Supports big data set	Supports any source to scrape the data
Extensibility (Code reusability)	Easily extensible	Easily extensible	Easily extensible
Ecosystem	Supports all major platforms	Supports all major platforms	Supports all major platforms
Languages	Java, C#, Python	Python	Javascript, Python, Java
Learning Curve	Well documented and easy to use	Well documented and easy to use	Well documented and easy to use
Performance	Dependant on page load duration	Works best for Static web pages	Very fast (100x) With headless browser

TABLE I: Evaluation summary of different scraping frameworks.

- Large ecosystem. Many related projects, plugins on open source websites such as Github, and many discussions on StackOverflow can help you fix the potential issue.

Cons:

- It has limited Javascript Support. It is time consuming to inspect and develop spider to simulate ajax/pjax requests.
- It has some limitations like scrolling of dynamic web pages. (Ex Twitter , Reddit etc)
- Very limited browser support.

VII. Division of Work and Team Members' Contributions

Table II shows the details about the division of task among the team. Individual contribution details of the team members are mentioned below:

Vivek Maskara: He contributed in setting up the initial Github repo and in implementing the scrapper for Reddit using Selenium. He used Selenium's Java APIs to write an application that scraps all information from posts for the specified subreddit. He also contributed in the preparation of project proposal and demo presentations.

Abhee Parekh: He studied the Puppeteer framework and was responsible for implementing a scrapper to extract data from Reddit using Puppeteer framework. For a specified subreddit topic, the scraper can extract all the post information by crawling the subreddit page. He was responsible for presenting the Project Proposal. He contributed in the preparation of the project proposal, demo presentation and the project report.

Task No.	Task Name	Assigned to
1	Project Proposal	All members
2	Survey of Scrapy Framework	Sravani and Subha
3	Survey of Selenium Framework	Vivek and Suhas
4	Survey of Puppeteer Framework and Implementation	Abhee, Raja and Vishal
5	Summary and Evaluation	All members
6	Demo	All members
7	Final submission	All members
8	Final Report	All members

TABLE II: Division of work.

Rajashekar Reddy Aluka: He studied the Puppeteer framework and was responsible for setting up the starter code for puppeteer. The code is setup in away where we can run the code either as a rest API or through command line. In Addition I also implemented the scrapper to extract data from Facebook and Twitter using Puppeteer. For a specified keyword, the scraper can extract all the posts/tweets information by crawling the Facebook/Twitter pages. He was responsible for presenting the Project Proposal. He contributed in the preparation of project proposal and demo presentations.

Suhas Khandiga Suresh: He studied the Selenium framework and was responsible for setting up starter code for the Selenium Web driver, which was used across all the Selenium-based scrapers in this project. He implemented a web scraper to extract data from Twitter using Selenium's Java APIs. He contributed to the preparation of project proposals and demo presentations.

Subha Pattanayak: He studied the Scrapy framework and was responsible for setting up starter code for the Scrapy project, which was used across all the Scrapy-based scrapers in this project. He implemented a web scrapper to extract Tweet information for a given hashtag or user using Scrapy framework. He contributed to the preparation of project proposals and demo presentations.

Sravani Gandla: She studied the Scrapy framework and was responsible for implementing a web scrapper to extract data from Reddit. For a specified subreddit topic, the scraper can extract all the post information by crawling the subreddit page. She contributed to the preparation of project proposals and presented Scrapy demo for Twitter and Reddit.

Vishal Singh: He helped in doing extensive research about various scraping tools and the social media website we chose for this project. He further helped Raja and Sravani in completing the tasks pertinent to Puppeteer. Moreover, he presented the final project and contributed to preparing the final report.

VIII. Conclusions

We were able to scrap data using all the three tools without any major challenges and found that all these tools allow for code re-usability and support all major platforms. In terms of programming language support, Scrapy supports only python but both Selenium and Puppeteer support all popular programming languages.

Scrapy offers limited Javascript support whereas other tools fully support javascript. Since Selenium mimics user actions using a real web browser, it is more suited for smaller datasets since the scraping is dependent on the page load times and the size of the website. On the other hand Scrapy and Puppeteer can easily be used for large datasets as well. Selenium's performance is dependent on the website where as Scrapy works best for static web pages. On the other hand, Puppeteer which offers a headless mode, can ideally perform 100x faster than the other tools.

We conclude that all these tools provide general capabilities for scraping but the choice of the tool should be made based on the use-case. Our evaluation metrics provides insights on the pros and cons of all the tools and the users can make a decision based on what metric concerns them the most.

References

[1] Marco Balduzzi, Christian Platzer, Thorsten Holz, Engin Kirda, Davide Balzarotti, and Christopher Kruegel. Abusing social networks for automated user profiling. In Somesh Jha, Robin Sommer, and Christian Kreibich, editors, *Recent Advances in Intrusion Detection*, pages 422–441, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[2] Robert Baumgartner, Alessandro Campi, Georg Gottlob, and Marcus Herzog. *Chapter 6: Web Data Extraction for Service Creation*, pages 94–113. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[3] Emilio Ferrara, Pasquale De Meo, Giacomo Fiumara, and Robert Baumgartner. Web data extraction, applications and techniques: A survey. *CoRR*, abs/1207.0246, 2012.

[4] Simon Kemp. Digital 2021 global overview report. https://datareportal.com/reports/digital-2021-global-overview-report, 2021. Accessed: 2021-01-27.

[5] Philipp Meschenmoser, Norman Meuschke, Manuel Hotz, and Bela Gipp. Scraping scientific web repositories: Challenges and solutions for automated content extraction. *D-Lib Magazine*, 22, 09 2016.

[6] Ramiro Ramírez-Coronel, Ana Cárdenas, Maria Belen Mora Arciniegas, and Gladys Alicia Tenesaca Luna. *Semantic Architecture for the Extraction, Storage, Processing and Visualization of Internet Sources Through the Use of Scrapy and Crawler Techniques*, pages 301–313. 01 2019.

[7] Zejian Shi, Minyong Shi, and Weiguo Lin. The implementation of crawling news page based on incremental web crawler. In *2016 4th Intl Conf on Applied Computing and Information Technology/3rd Intl Conf on Computational Science/Intelligence and Applied Informatics/1st Intl Conf on Big Data, Cloud Computing, Data Science Engineering (ACIT-CSII-BCD)*, pages 348–351, 2016.

[8] Jing Wang and Yuchun Guo. Scrapy-based crawling and user-behavior characteristics analysis on taobao. pages 44–52, 10 2012.

[9] Hongxia Yang. Design and implementation of data acquisition system based on scrapy technology. In *2019 2nd International Conference on Safety Produce Informatization (IICSPI)*, pages 417–420, 2019.