Web Scraping and Job Recommender System

Koustubh Sinha
Computer Science Engineering
Jaypee Institute of Information Technology
Delhi, India
koustubh13sep@gmail.com

Priyansh Sharma
Computer Science Engineering
Jaypee Institute of Information Technology
Delhi, India
15priyansh13.2001@gmail.com

Harshit Sharma
Computer Science Engineering
Jaypee Institute of Information Technology
Noida, India
sharma.harshit0059@gmail.com

Krishna Asawa
Computer Science Engineering
Jaypee Institute of Information Technology
Noida, India
krishna.asawa@mail.jiit.ac.in

2023 Second International Conference on Informatics (ICI) | 979-8-3503-4383-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICI60088.2023.10420941

***Abstract*—Web scraping is the process of autonomously obtaining data from webpages using computer software or tools. It has grown in popularity in recent years because of the volume of data available on the internet and the necessity for organizations and individuals to analyze and utilize this data to get the necessary and important information out of it for further analysis and application. Web scraping, together with automation tools, can create wonders. In this paper, we have focused on the latest web scraping techniques and their applications. One of the ideas is to assist those looking for employment on different firm job portals, which is discussed in this paper. Job tracking and scraping collect data from multiple platforms and are web-based tools that help job searchers locate the ideal job for them based on their categories and organize their job search more effectively. People are always on the lookout for a place where they can get their desired information so that they can easily apply for a job. After the collection of data based on user profile attributes, we recommend their success rate based on previous records and an appropriate machine learning model, and we also suggest what skills they can improve to land that job. Web scraping tools allow us to extract the necessary data from different websites in real time and present it to the user. We have discussed various ways to implement scraping and propose a job algorithm to process the extracted data and make it into meaningful information to help users land the best job.**

Keywords—Web Scraping, Web automation, Job recommendation system, Job Helper, Job domain, Data analysis, Web Scraping Applications, Artificial Intelligence, Machine Learning, Content based filtering.

I. INTRODUCTION

The automated extraction of data from websites using software or tools is known as web scraping. This technology has risen in popularity in recent years due to the abundance of information available on the internet and the need for organizations and individuals to study and use this data for a variety of purposes, including research [2]. Web scraping can be used to quickly and efficiently collect large amounts of data from many sources, which is extremely useful for research projects requiring data from multiple websites. It is crucial to highlight, however, that online scraping can pose ethical and legal concerns, particularly when data is extracted without consent or the terms of service of a website are breached. To ensure that web scraping is done responsibly and ethically, we propose a solution to extract data in a manner that is available to be seen publicly and follows the same steps as humans do manually. Also, regarding data that is not allowed to be scraped and stored, the permission of the website owners must be obtained, and their terms of service must be followed. There are various tools available for automation and scraping, with selenium being one of the most powerful of them. These tools help us get the data that we can use to extract meaningful information out of it for various advantages in machine learning, big data, artificial general intelligence, automating human tasks, and many more [2].

Web scraping consists of two components: a crawler and a scraper. Selenium is needed for web automation and a parser for scraping the data. Automated data scraping is way faster than what humans can do manually. We don't have to go to each and every webpage and get the data from there. Here we propose "Job Helper," which is one of the applications of web scraping, but web scraping has a vast field of applications. The largest hub of engineers in the world, India, produces a pool of nearly 1.5 million engineering graduates every year, as stated in a 2017 report by the National Institution for Transforming India (NITI Aayog) [6].With the rising number of engineering graduates every year and the increase in automation in nearly every field, competition for jobs in India has also increased. The struggle exists not only for freshmen but also for experienced professionals. Moreover, the increasing

979-8-3503-4383-0/23/$31.00 ©2023 IEEE

number of engineering graduates has not been met with an increasing number of engineering jobs, leaving a heavy toll on the graduates. Often, good engineers are not able to secure a good job, forcing them to work at much lower salaries than expected. The COVID-19 pandemic further added to the woes of job seekers, as many companies had put a hold on hiring or had reduced the number of vacancies. According to a report by the Centre for Monitoring the Indian Economy (CMIE), around 1.5 million salaried jobs were lost in the first quarter of the financial year 2020–21. While there is no shortage of engineering graduates, they mostly lack the skills required by the industry. A survey found that only 4.77% of engineering graduates in India have the required coding skills for software-related jobs. This skills gap has led to a significant mismatch between the industry's requirements and the skills possessed by engineering graduates. What if we could also tell the candidates the best job roles based on their skills and what their chances are of getting placed in them? The problem we are trying to solve through our job-help system is a multi-faceted one where several factors contribute to the job market. The development of a job helper is a promising solution to the problem of bringing together all the jobs currently available in the market into a single destination. This platform can help job seekers keep track of job openings available at leading tech companies, which can enable them to apply for these opportunities in an efficient manner and also help them not miss out on opportunities. On the other hand, with the collected information, we select the best match based on the skills of the user and the skills required in the job description. Data from previous years helped us determine the success rate.

II. LITERATURE SURVEY

Web scraping, also known as web extraction or harvesting, is a technique to extract data from the World Wide Web [1]. It is one of the most used and effective ways to collect data from different websites in a very effective and speedy manner. We can use many tools, like Beautiful Soup, Puppeteer, etc., to execute web scraping. But web scraping comes with its own challenges, which are hard to tackle. Some of these major reasons are:

- Legal Issues: Many websites do not allow web scraping or any infiltration of their websites by bots. This may be against their terms of service. Some websites have explicit policies against web scraping, and they may take legal action against you too if you are caught cheating.
- Ethical Reasons: A small fraternity of people may consider this against ethical reasoning, as the data that is being scraped might be intended for other purposes.
- Data Quality Issues: Sometimes scraping the data can lead to incomplete data, leading to the generation of illogical data. Also, maintaining data quality during the process can be hectic.
- Changing Website Structure: The websites tend to change in nature over time; thus, a web scraper that is not evolving over time may face challenges. Thus, one needs to constantly keep changing the web scraper to meet the current requirements.

Web scraping may now be done manually or automatically, and there are several tools and modules available for initiating HTTP queries, parsing HTML code, and extracting desired information. A Python package known as Beautiful Soup extracts data from HTML and XML files. It works in addition to your preferred parser to give idealistic methods of exploring, searching, and altering the parsed tree. It frequently saves programmers hours or even days of effort. There are various parser libraries available with lovely soup. We chose LXML's [3] HTML parser because of its benefits. Let's see in detail about LXML's [13] and algorithms in further sections. Below is a detailed comparison between different types of scrapers.

TABLE I
Different types of parser comparison [3]

Parser	Usage	Advantage/ unique feature	Disadvantage
Python's html.parser	BeautifulSoup(markup, "html.parser")	Batteries involved, Good speed, Lenient	Not as fast as lxml and less lenient than html5lib.
lxml's HTML parser	BeautifulSoup(markup, "lxml")	Super-fast, Lenient as well	External C dependency.
lxml's XML parser	BeautifulSoup(markup, "lxml-xml") BeautifulSoup(markup, "xml")	Very fast	External C dependency Currently supported XML parser only.
html5lib	BeautifulSoup(markup, "html5lib")	Extremely lenient. Parses web pages the same way a web browser does.	Very slow External Python dependency.

[a]This table compares different types of parsers based on various criteria, such as speed, accuracy, and memory usage. The specific criteria and details are discussed in the corresponding section of this paper.

Job recommendation systems have risen in popularity in recent years as firms seek to efficiently match job seekers with appropriate job openings. These systems use a variety of technologies, including machine learning, to analyze job seeker profiles and job advertisements and make recommendations based on the match between the two. Content-based filtering [8] is a machine learning technique that proposes things based on their properties. In the context of job recommendations, content-based filtering can be used to suggest openings based on job posting attributes such as job title, job description, required skills, and region. Finally, job recommendation systems match job seekers with appropriate job prospects using a range of machine learning techniques, such as collaborative filtering, content-based filtering, and hybrid recommendation.

III. COMPARISON WITH EXISTING APPROACH

With already existing job recommendation platforms like GFG [11], Unstop [12], and LinkedIn, which have been very successful and have benefitted many students in their journey of job hunting and also securing those jobs, these platforms either collaborate with different companies or the companies come on these platforms to post job opportunities and conduct job fairs, hackathons, etc. But the major disadvantages of this platform are:

- These platforms are specially designed to connect companies who come there to use these platforms to hire, but most of these firms have their own websites to post these job opportunities.
- Most of these platforms are in collaboration with specific companies and show jobs related to those unsought companies more than the companies sought.
- These platforms work as a social platform for connecting candidates and companies, which means both parties need to be involved. But it's seen that there are many opportunities that are on one platform but not the other, which makes job-finding difficult and, again, time-consuming. These things may result in missing out on opportunities that are not present there or not being able to do them at the right time.

Our new approach looks forward to countering all these problems by producing a new platform that can cater to everyone's needs. The platform mostly focuses on the use of web scraping using Beautiful Soup [3] and Selenium to improve with the latest techniques. Jobs are displayed on a real-time basis, so the extracted information is current and relevant to the openings. This approach does not give the idea of a social platform where both companies are involved in posting jobs; instead, it is a platform where candidates can locate the best opportunities available in the market in real time. Since the companies are not involved, whatever they put on their websites and postings can be extracted and shown to the candidates. There are three most important categories to choose from: internships, new graduates, and experienced professionals. This helps us get only jobs that fit our roles. The final result is to sort out the most relevant job based on your skills. A job success rate predictor based on user skills and the requirements of that job posting. When it comes to job recommendation algorithms, there are various methods. We choose content-based filtering because of its certain advantages. We can compare it with resume parsing, a famous algorithm where a resume goes through the resume parser, and based on the resume, a skill score is given as to how much impact your resume can make. Content-based filtering holds several advantages over resume parsing in the realm of job matching and recommendation systems. Content-based filtering primarily serves the purpose of matching users with content or job listings based on their preferences and the similarity of content attributes. Users benefit from reduced data entry requirements since it relies on their behavior and interaction history, alleviating the need for detailed resume submissions. Unlike resume parsing, content-based filtering also excels at adapting to dynamic user preferences and evolving content without the constant need for structured data updates. Additionally, it sidesteps data quality concerns often associated with resume parsing and can provide diverse recommendations for a more comprehensive user experience. This approach is particularly valuable for new users entering the job market, as it can offer recommendations even without a formal resume in the system. In essence, content-based filtering is about user-content matching, while resume parsing deals with organizing and assessing candidate data extracted from resumes.

IV. JOB HELPER: Design and algorithms

We propose our solution to one of the most useful use cases of web scraping. The 'Job Helper' intends to fill the void and solve the problem of people not being able to apply to their dream jobs just because they didn't happen to know about it. It enables students and professionals across domains, irrespective of their background, to access job opportunities across several top-tier companies. Our solution consists of mostly three modules, which are discussed below:

- *Web Scraping Module*

Web scraping consists of two components: a crawler and a scraper. Selenium is needed for web automation and a parser for scraping the data. There are various scraper parsers available, with their benefits and disadvantages. To apply web scraping, we can think of a human doing the same task. For example, if I want to find a job on a website, what are the steps we need to take? We go to sites to select the proper categories. Click on the job that appeared and read the description to see if we are eligible. What are the skills demanded in the job description, together with the job type, location, and title? We are going to mimic this same human behavior step by step, as it is being done by us, at a much faster rate [2]. Considering legal issues, we looked at the data, which is only viewable publicly and is important for our project, meaning it is not causing any legal action. We extract the website's inner HTML, which is the frontend that we view, locate the elements that we are targeting or searching for, and use appropriate locators like XPATHS, CLASS, ID, or NAME [1] to extract the desired information. We chose XPATHS here, which is the most reliable method out of the others. We extract the above data from multiple different websites at the same time.

- *Job Success Rate Predictor Module*

No data is useful if we can't extract meaning from it. The candidates must share their details with us, where they tell us the skills they have, like programming languages, software frameworks, etc., along with the level of confidence they have in their skills. We process the collected data for the job description, where the requirements and important information about the job are mentioned. Find out what skills are needed for the jobs and how their importance is rated. Now that user skills ratings and job description ratings are compared, if our ratings are higher than the job description demand, that clearly means it is an advantage. Otherwise, it is added to the list of missing skills and recommends users improve them. Based on this, we sort the jobs according to user preference. For the success rate, we have used previous years data with appropriate machine learning algorithms, user skills, and other relevant information to support their skills. This helped us to know which is currently our best match and what to work on.

- *Graphical User Interface (GUI) Module*

Finally, a graphic user interface for the consumer to interact with, which connects both the above modules and presents the user with the most relevant and valuable information. The workflow of solutions is here as follows:

Step 1: The candidates must share their details with us, where they tell us the skills they have, like programming languages, software frameworks, etc., along with the level of confidence they have in their skills.

Step 2: Users select a choice from the three categories: internship, new graduates, and experienced.

Step 3: We extract the job data from multiple sites at the same time and bring it to one place.

Step 4: We find the key skills required based on the job description and compare them with the skills that the user input to determine the success rate based on previous data using machine learning.

Step 5: Further suggestions to improve your credibility for getting into your dream job.

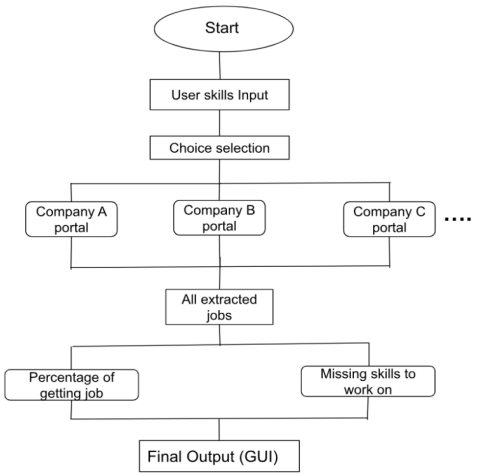

Fig. 1. Workflow

We are using Web Scraping to scrape the data related to several jobs available at the company's spread across various locations. We are storing the scraped data in our database for further processing. Our idea is to mimic human-like behavior to automate tasks, which are related to artificial general intelligence as well. We are using Xpaths and headless browsers for our use case. In the case of multiple websites to be scrapped, when we run our processes sequentially, it increases the run time and ultimately takes more time to extract our job openings. To overcome this problem, we used parallel processing using threading to simultaneously scrape the data together. We applied parallel processing to scrape multiple websites at the same time, resulting in greatly reduced runtime and faster results. The scraping algorithm goes as follows: It operates in a headless browser mode, which means it doesn't display a user interface. Once the webpage is loaded, you employ XPath expressions to locate specific elements on the

page, likely including job listings. The automation process involves filtering job results based on certain criteria, presumably related to the job category, location, or other attributes. Then, looping through each job instance in the list, you retrieve the page's HTML content and use libraries like BeautifulSoup and lxml to parse and scrape the data. This scraping process likely involves extracting information such as job titles, descriptions, and any other relevant details. The extracted job information is then stored in a database for later use or analysis.

- *Job success rate predictor*

Based on user skills and skills demanded in the job description, we give users their chance at a particular job. For sorting all the available results according to relevance, see if the user rating is higher than the demanded rating, add it to the advantage and otherwise to the missing skill. In this scenario, content-based filtering works best. It is a machine learning technique that proposes things based on their properties. In the context of job success rate, content-based filtering can be used to suggest openings based on job posting attributes such as job title, job description, required skills, and regions. We used the past-case data to predict the chance, and together with the missing skills, we recommend users improve their skills, which can increase their chances of securing that job. In later phases, we can look at hybrid recommendation systems as well. It combines the strengths of collaborative filtering and content-based filtering techniques. These systems can be used in job recommendations to recommend jobs based on the job seeker's past behavior as well as the characteristics of job listings. Consider the skills as data points, and if the user data points are close to the demanded data points, it increases their success. Content-based filtering Using Cosine Similarity is an algorithm designed for feature extraction and skill matching in the context of job descriptions. It begins by targeting a specific feature in the job description column. It aims to extract and add key features from these descriptions into a new feature column, such as skills like C++, Python, and others. This process is implemented by iterating through a list of skills and using regular expressions to identify their presence in the job descriptions. When a skill is found, it is concatenated into a new feature column. To enhance the matching capabilities, the algorithm considers the user's skills, which we asked for from them earlier and which are specified as input. These user skills are then combined with the previously extracted features column. Subsequently, the algorithm employs a count vectorizer to transform the features into a matrix, considering the frequency of skills. This matrix is then used to compute cosine similarity, allowing for the measurement of the similarity between different job descriptions based on the skills identified. To facilitate user interaction, a function is included to identify the index of the user's skill within the dataset, which is stored in the variable. This index becomes instrumental in generating a best match matrix, which is a list of pairs comprising the index and the corresponding cosine similarity score. The cos_sim matrix is a numpy array that contains the estimated cosine similarity of each job description together with the user skill.

V. RESULT AND DISCUSSION

The final result is a graphic user interface to showcase all the extracted jobs based on category (internship, new graduates, experienced) choice, together with the most valuable information obtained after the processing of the stored data. The information includes our success rate at a particular job, the missing skills that we can improve to increase our success rate, and the stored data according to our relevance. Our job success rate predictor results show that machine learning algorithms can effectively connect job seekers with relevant job prospects. Users received accurate job recommendations from the system based on their job search history and interests.If we talk about the runtime of scraping since we are applying parallel threads, it reduces our extraction runtime to the time taken in scraping only one website. Below is the table for runtime comparison if we talk about scraping 20 different companies at the same time.

TABLE II
Runtime comparison for 20 different companies - series and parallel

Series Processing	Parallel Processing
Runtime - 20 minutes	Runtime - 1 minute 20 sec

[a]Runtime values are in seconds. 'Series' denotes sequential execution, and 'Parallel' signifies concurrent execution for 20 different companies.

We found that content-based filtering was an effective technique for job recommendations, as it considered the attributes of job postings to provide relevant recommendations. If we compare, collaborative filtering can be useful too, as it leverages the behavior of similar users to make personalized recommendations. With the previous placement data, user honest skill input, and above filtering, we can provide them with the most valuable information they can have. And lastly, we are giving them insight into their own missing skills, which they can work on to gain better results. Now here is the resultant matrix of job results derived from content-based filtering using cosine similarity. Talking about resume parsing results, it rates resumes

of individuals, job descriptions of posts have no role in it. Now, comparing with content-based filtering results in terms of job recommendation, content-based filtering works better because it is highly personalized, as it matches users with content of job listings based on their unique preferences and attributes. Below is a table with results obtained from the discussed algorithm.

TABLE III
Best match - Content based filtering using cosine similarity matrix

Index	Type	Size	Value
0	tuple	2	(0, 0.8461538461538461)
1	tuple	2	(1, 0.9828161281822770)
2	tuple	2	(2, 0.0)
3	tuple	2	(3, 0.4652488264328260)
4	tuple	2	(4, 0.5407380870435670)

[a]This table presents the results of best match recommendations obtained through content-based filtering techniques and a cosine similarity matrix.

We can also incorporate a hybrid approach, which combines both collaborative and content-based filtering techniques, resulting in improved job recommendations for users. The need for job-searching-friendly websites is a necessity and highly in demand among students, new graduates, and experienced professionals. The percentage can help a user analyze himself, get the dream job he is searching for, and truly know the chance he or she has of cracking that job. The missing skills and our shortcomings are not easy to find, but using the methodologies discussed can help you increase your probability of cracking those jobs.

Overall, our study demonstrated the potential of machine learning algorithms for improving job matching and providing more personalized job recommendations to job seekers. Future research can continue to explore the effectiveness of these techniques and develop more sophisticated algorithms to improve job recommendation systems even further. This was one of many applications of web scraping combined with machine learning and AI. Our idea is to create a system to help humans make their lives easier with the help of the latest technologies available.

ACKNOWLEDGEMENT

We would also like to thank the participants in our study who generously gave their time and shared their insights with us. We are grateful to the various job board websites and online resources that provided us with the data and information necessary for our analysis. We also thank the data science and machine learning communities for their invaluable contributions to the development and advancement of data scraping techniques. We extend our appreciation to our colleagues and peers who provided feedback, constructive criticism, and valuable insights during the development of this system.

REFERENCES

[1] Bo Zhao, "Web Scraping," Springer International Publishing AG (outside the USA), 2017.
[2] Vidhi Singrodia, Anirban Mitra, and Subrata Paul, "A Review on Web Scraping and its Applications," January 2019.
[3] Leonard Richardson, "Beautiful Soup Documentation," 2004-2015.
[4] L. Ganagadhara Reddy and DR. P. Viswanath, "A study on web scraping of selected job portals" September 2022.
[5] Suganya R, R.S. Krupasree, S. Gokulraj, and B. Abinesh, "Product Review Analysis by Web Scraping Using NLP," in Smart Data Intelligence. Algorithms for Intelligent Systems, R. Asokan, D.P. Ruiz, Z.A. Baig, and S. Piramuthu, Eds. Singapore: Springer, 2022.
[6] National Institution for Transforming India, "Annual Report 2017-2018".
[7] S. Caldwell, "A Framework for Identifying Malware Threat Distribution on the Dark Web," 2023, Theses and Dissertations, no. 141.
[8] A. Stefanos, G. Demirtsoglou, X. Zisopoulos, and S. Karagiannidis, "Content-Based Recommendation Systems," Computer Science Department, Aristotle University of Thessaloniki, 2008.
[9] G. Barcaroli et al., "Use of Web Scraping and Text Mining Techniques in the Istat Survey on Information and Communication Technology in Enterprises," in European Conference on Quality in Official Statistics (Q2014), Vienna, Austria, June 2014
[10] D. Krijnen, "Automated Web Scraping APIs", 2014.
[11] GFG Job Portal. https://practice.geeksforgeeks.org/jobs
[12] Unstop. https://unstop.com/
[13] Lxml. Ian Bicking. https://lxml.de/lxmlhtml.html
[14] S. Philip, P.B. Shola, and O. John, "Application of Content-Based Approach in Research Paper Recommendation System for a Digital Library," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 5, no. 10, 2014.
[15] P. Resnick and H. Varian, "Recommender Systems," Communications of the ACM, pp. 56-58, 1997.