WEB SCRAPING

Summer Internship Report Submitted in partial fulfilment

of the requirement for undergraduate degree of

Bachelor of Technology

In

COMPUTER SCIENCE AND ENGINEERING

By

JANGAM ASHMITHA MARIA

HU21CSEN0400232

Under the Guidance of

Mr. SANTHOSHACHANDRA RAO K

Department Of Computer Science and Engineering

GITAM School of Technology

GITAM (Deemed to be University)

Hyderabad-502329

December 2023

DECLARATION

I submit this internship work entitled **“Web Scraping”** to GITAM (Deemed to Be University), Hyderabad in partial fulfilment of the requirements for the award of the degree of “**Bachelor of Technology**” in “**Computer Science and Engineering**”. I declare that it was carried out independently by me under the guidance of **Mr. Santhoshachandra Rao K,** Asst. Professor, GITAM (Deemed To Be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD	Name: Jangam Ashmitha Maria
Date: 26-12-23	Roll no: HU21CSEN0400232

ACKNOWLEDGEMENT

Apart from my effort, the success of this Internship-1 largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful competition of this Internship-1.

I would like to thank respected **Prof. D. Sambasiva Rao**, Pro Vice Chancellor, GITAM Hyderabad, and **Dr. N. Seetharamaiah**, Associate Director, GITAM Hyderabad.

I would like to thank respected **Dr. Mahaboob Shaik Basha,** Head of the Computer Science and Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present an internship-1 report. It helped me a lot to realize what we study for.

I would like to thank the respected faculties **Dr. Santhoshachandra Rao K,** who helped me to make this Internship-1 a successful accomplishment.

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end.

Jangam Ashmitha Maria

HU21CSEN0400232

ABSTRACT

In an era dominated by digital information, this web scraping project emerges as a crucial tool for navigating the complexities of the internet and extracting meaningful insights. Employing Python alongside the `requests`, `BeautifulSoup`, and `spacy` libraries, the project embodies a versatile and efficient solution for data extraction from diverse websites.

A distinctive feature of the project lies in its user-centric approach, allowing individuals to interactively search for sentences containing specific words and filter words by character length. The incorporation of spaCy's natural language processing capabilities enhances the depth of analysis, contributing to a more nuanced understanding of the extracted content. Ethical considerations are paramount, with the project emphasizing compliance with website terms of service and legal standards. By adopting a sequential execution approach and leveraging `matplotlib` for visualizations, the project not only simplifies the data extraction process but also empowers users to make informed decisions through insightful analysis of the web-based information landscape.

Table of Contents:-

1. INTRODUCTION	5-12
2. PROBLEM STATEMENT	13
3. OBJECTIVE	13
4. KEY FEATURES	13-14
5. REQUIREMENTS	15
6. PROCESS	17-20
7. CODE	21-22
8. KEY FUNCTIONS	22
9. OUTPUT	24
10. FUTURE ENHANCEMENTS	25
11. CONCLUSION	25

INTRODUCTION:-

Web Scraping for Intelligent Data Extraction:-

In today's information-driven age, the ability to harness valuable insights from the vast expanse of the internet is paramount for businesses, researchers, and decision-makers. This web scraping project emerges as a crucial enabler, offering a sophisticated solution for intelligent data extraction. Leveraging the power of Python and prominent libraries such as `requests`, `BeautifulSoup`, and `spacy`, the project is tailored to navigate the complexities of diverse websites. Its user-centric design allows for interactive searches, enabling users to extract targeted information based on specific criteria. The integration of natural language processing adds a layer of sophistication, enhancing the precision of data analysis. Ethical considerations remain a cornerstone, with a commitment to compliance with website terms of service and legal standards. By adopting a sequential execution approach and incorporating visualizations through tools like `matplotlib`, this project not only simplifies the data extraction process but also empowers users to derive meaningful insights, thus contributing to informed decision-making in the dynamic landscape of digital information.

Legal and Ethical Considerations

Web scraping, a potent data extraction tool, necessitates careful consideration of legal and ethical dimensions. To ensure responsible usage, practitioners must navigate the following aspects:

1. Respecting Terms of Service:

- Importance: Violating a website's terms of service can lead to legal consequences.
- Best Practices: Thoroughly review and comply with the terms of service of the target website before initiating scraping activities.

2. Compliance with Laws:

- Legal Landscape: Web scraping legality varies; comply with data protection and intellectual property laws.
- Best Practices: Stay informed about regional legal requirements and adhere to applicable regulations.

3. Robots.txt and Crawl-Delay:

- Guidelines: Respect rules in the `robots.txt` file, and implement crawl delays when specified.
- Best Practices: Check `robots.txt` for guidelines and avoid overloading servers by respecting crawl delays.

4.Avoiding Unauthorized Access:

- Unauthorized Access: Illegitimate access to restricted areas is both unethical and illegal
- Best Practices: Limit scraping to publicly accessible information, and avoid attempting unauthorized access.

5. Data Privacy and Consent:

- Privacy: Safeguard personal data and obtain proper consent for its collection.
- Best Practices: Prioritize data anonymization, adhere to privacy best practices, and obtain required user consent.

6. Monitoring and Adaptation:

- Website Changes: Regularly monitor websites for changes in terms of service or structure.
- Best Practices: Adapt scraping strategies to accommodate any changes on target websites.

7. Openness and Transparency:

- Transparency: Communicate the nature of scraping activities, data sources, and methodology.
- Best Practices: Ensure transparency regarding data sources, methodology, and potential biases associated with the extracted information.

In summary, a conscientious and ethical approach to web scraping involves understanding legal requirements, adhering to website terms of service, and committing to privacy and transparency. By navigating these considerations, practitioners can harness the benefits of web scraping responsibly.

TOOLS AND TECHNOLOGIES

The provided code for web scraping project is written in Python and utilizes several libraries for different tasks. The key tools and technologies used in the given project are:

1. Python:

-Purpose: Python serves as the primary programming language for implementing the web scraping project. Python is widely used for its simplicity, readability, and a rich ecosystem of libraries, making it an ideal choice for web scraping projects.

2. Requests Library:

- Purpose: The `requests` library is employed for making HTTP requests to the specified URL, fetching the HTML content of the web page. The `requests` simplifies the process of interacting with web servers, allowing for seamless retrieval of web page content.

3. BeautifulSoup:

- Purpose: BeautifulSoup is used for parsing HTML and extracting data from the website's HTML structure. BeautifulSoup provides a convenient way to navigate and search the HTML document, making it easier to extract relevant information.

4. Spacy:

-Purpose: Spacy, a natural language processing (NLP) library, is used for sentence tokenization to identify and extract sentences from the HTML content. Spacy enhances the project's capabilities by providing accurate and efficient sentence tokenization, aiding in the extraction of meaningful content.

5. Concurrent Futures:

- Purpose: The `concurrent futures` module is used for parallelizing the execution of web scraping functions. Parallelization improves the efficiency of the scraping process by allowing multiple URLs to be processed concurrently.

6. Pandas:

Pandas is used for creating a Data Frame to organize and analyse the frequency of words extracted from the web page. Pandas simplifies data manipulation and analysis, providing a structured format for presenting and visualizing the scraped data.

7. Matplotlib:

Matplotlib is employed for creating bar charts to visualize the frequency of words. Matplotlib enables the generation of clear and informative visualizations for better understanding and interpretation of the data.

8. Counter:

The `Counter` class from the `collections` module is used for counting the frequency of words in the scraped data. Counter simplifies the process of creating a frequency distribution, a key step in analysing the extracted textual content.

These tools and technologies collectively contribute to the effectiveness and functionality of the web scraping project, allowing for efficient data extraction, analysis, and visualization.

ADVANTAGES

1.Data Accessibility:

Web scraping provides access to a vast array of data, overcoming limitations of traditional data collection methods. It allows for the extraction of data from various websites, enabling a comprehensive and diverse dataset for analysis.

2. Automation and Efficiency:

Automation is a key strength of web scraping, significantly improving efficiency. Repetitive tasks, such as gathering data from multiple pages or websites, can be automated, saving time and resources.

3. Real-Time Data Collection:

Web scraping allows for the extraction of real-time data, making it an invaluable tool for staying abreast of dynamic information. This is particularly advantageous for industries where timely updates are crucial.

4. Competitive Intelligence:

Businesses leverage web scraping to gain competitive intelligence. By monitoring competitors' websites, pricing strategies, customer
reviews, and product offerings, companies can make informed decisions to stay competitive in the market.

5. Research and Analysis:

Researchers benefit from web scraping in academic studies and market research. It provides a wealth of data for analysis, enabling the identification of patterns, trends, and correlations that contribute to valuable insights.

LIMITATIONS

1. Legal and Ethical Challenges:

Legal implications arise when web scraping violates a website's terms of service. Ethical concerns emerge, particularly when scraping involves sensitive or personal information without proper consent, requiring practitioners to navigate these challenges responsibly.

2. Website Structure Changes:

Websites undergo structural changes over time, impacting the functionality of existing web scraping scripts. Regular maintenance is necessary to adapt scripts to evolving website structures.

3. Dynamic Content and JavaScript:

Web scraping tools may struggle with websites that heavily rely on dynamic content generated by JavaScript. In such cases, additional tools like Selenium, capable of interacting with dynamic elements, may be required.

4. Data Quality and Accuracy:

Extracted data may contain inaccuracies or inconsistencies, particularly when dealing with complex HTML structures or variations in data presentation. Validation processes are needed to ensure data quality.

5. Server Load and IP Blocking:

Intensive web scraping activities can increase server load, leading to IP blocking by websites as a defensive measure. Practitioners must implement strategies such as rate limiting to avoid disruptions.

6. Dependency on Website Stability:

The reliability of web scraping is contingent on the stability and uptime of the target website. Downtime or changes in website structure can disrupt scraping processes, emphasizing the need for robust error handling.

7. Limited Interaction with Interactive Content:

Traditional web scraping methods may struggle to interact with highly interactive content or forms on websites. Solutions like Selenium, which allows for browser automation, become essential when user interaction is necessary.

Understanding these nuances helps practitioners navigate the complexities of web scraping, enabling them to capitalize on its advantages while mitigating potential limitations and challenges responsibly.

PROBLEM STATEMENT

The objective is to develop a robust web scraping tool that facilitates seamless data extraction, enabling users to obtain timely and structured information for analysis, research, and decision-making across diverse domains.

The solution should overcome challenges such as dynamic website structures, anti-scraping measures, and frequent website updates. By addressing these issues, the project seeks to empower businesses and users with up-to-date and relevant pricing intelligence, enabling them to adjust their pricing strategies dynamically and enhance their competitiveness in the online marketplace.

OBJECTIVES

1. Data Extraction:

The primary goal of web scraping is to extract specific, relevant data from websites. This could include text, images, prices, or any other information valuable for analysis.

2. Data Aggregation:

Web scraping is used to collect and aggregate information from multiple sources, allowing for the creation of comprehensive datasets. This is particularly valuable for market research, trend analysis, and decision-making.

4. Automation:

Web scraping automates the process of data collection, saving time and resources. Automation is especially important when dealing with large volumes of data or when updates need to be monitored regularly.

5. Market Intelligence:

Web scraping provides insights into market trends, consumer behavior, and other relevant factors. This information is crucial for businesses to make informed decisions and stay competitive in their respective industries.

These objectives collectively contribute to the efficiency, competitiveness, and informed decision-making of businesses and researchers using web scraping as a tool for data extraction and analysis.

KEY FEATURES

1. Versatile Web Scraping Functionality:

The project offers versatile web scraping functionality, allowing users to extract valuable information from diverse websites. The use of Python, along with libraries such as `requests`, `BeautifulSoup`, and `spacy`, ensures adaptability to various website structures and content types.

2.**Parallel Processing:**

The code employs concurrent.futures.ThreadPoolExecutor for parallel processing, enhancing efficiency by concurrently handling multiple URLs

3. User-Friendly Interface:

Users are prompted to input the target website URL, and interactive features allow customization of the scraping process based on individual needs.

4. Tailored Data Retrieval:

The project includes a second function (`scrape_words_by_length`) that enables users to filter and display words based on their desired length. This feature enhances the flexibility of information retrieval.

5.Data Analysis:

The project performs data analysis on the extracted words, generating a frequency distribution. The top 10 words by frequency are visualized using a bar char (example).

6. Ethical Web Scraping Practices:

A fundamental aspect of the project is its commitment to ethical web scraping practices. It adheres to legal and ethical standards, respecting the terms of service of scraped websites and prioritizing user privacy.

.

Scraping Data from Websites

Process of Scraping Data

Scraping data from websites involves the following steps:

1.**Sending HTTP requests:**

Scrapers send requests to the target website's server to retrieve the HTML content of the desired web page.The requests library in Python is commonly used for making HTTP requests. It simplifies the process of sending GET or POST requests and handling the server's response.

2.Parsing HTML:

The HTML content is parsed to extract the relevant data using techniques such as XPath or CSS selectors.The BeautifulSoup library in Python is a popular choice for parsing HTML. It provides a simple interface for navigating and searching the HTML tree.

3.Extracting relevant data:

The scraped data is extracted and stored in a structured format such as CSV or JSON for further analysis.

With BeautifulSoup, you can navigate the HTML tree by selecting tags, attributes, or searching for specific patterns. Methods like find, find_all, and CSS selectors help locate elements on the page.

4.Data Extraction and Cleanup:

Extract the desired data from the selected HTML elements and perform any necessary data cleaning or formatting.

5.**Data Analysis:**

The code utilizes the Counter class from the collections module to count the frequency of each word. The results are then structured into a Pandas DataFrame for analysis.

6.Data Visualization:

The top 10 words by frequency are visualized using Matplotlib in the form of a bar chart. This chart is displayed to the user (example).

7.Parallel Processing:

The code employs concurrent.futures.ThreadPoolExecutor to parallelize the execution of the functions for eachURL, enhancing the efficiency of the scraping process.

8.ResultPresentation:

The extracted sentences containing the specified word and the words of thespecified length are printed to the console.

9.**Iterate or Navigate toOther Pages (if needed):**

For scraping multiple pages, iterate through alist of URLs or navigate to different pages by following links.

>***The program utilizes parallel processing to improve the efficiency of web scraping by concurrently executing tasks related to different URLs. It also incorporates data analysis techniques to process and visualize the collected data.***

Parallel Processing

1. **ThreadPoolExecutor:**

   The program utilizes concurrent.futures.ThreadPoolExecutor to implement parallel processing. ThreadPoolExecutor creates a pool of worker threads, allowing multiple tasks to be executed concurrently.

   ○

2. **Execution of Functions:**

   The executor.map method is used to parallelize the execution of the web scraping functions (scrape_sentences_with_word and scrape_words_by_length) for each URL. Each function is assigned to a separate thread, enabling parallel execution for multiple URLs.

   ○

3. **Improved Performance:**
4. Parallel processing improves the overall performance by allowing the program to make simultaneous requests and process data from different URLs concurrently.Particularly beneficial when dealing with multiple independent tasks, as in this case where web scraping is performed on different URLs.

Data Analysis Techniques

Counting Word Frequency:

1. The Counter class from the collections module is used to count the frequency of each word in the collected data. It creates a dictionary where keys are unique words, and values are their respective frequencies.
   -

Pandas DataFrame:

The program uses the pd.DataFrame class from the Pandas library to organize the word frequencies into a tabular structure. The DataFrame provides a convenient way to manipulate and analyze the data.

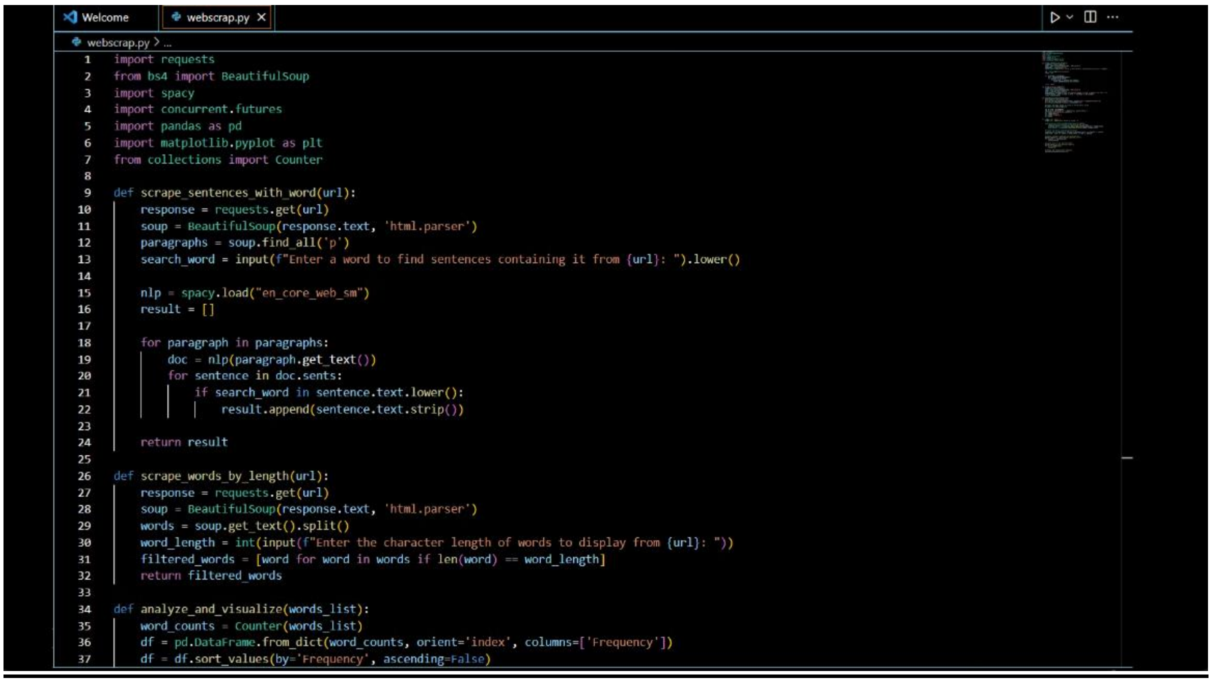
Sorting and Filtering Data:

The word frequencies are sorted in descending order using the sort_values method, allowing identification of the most frequently occurring words. Filtering is applied to extract the top 10 words for visualization.

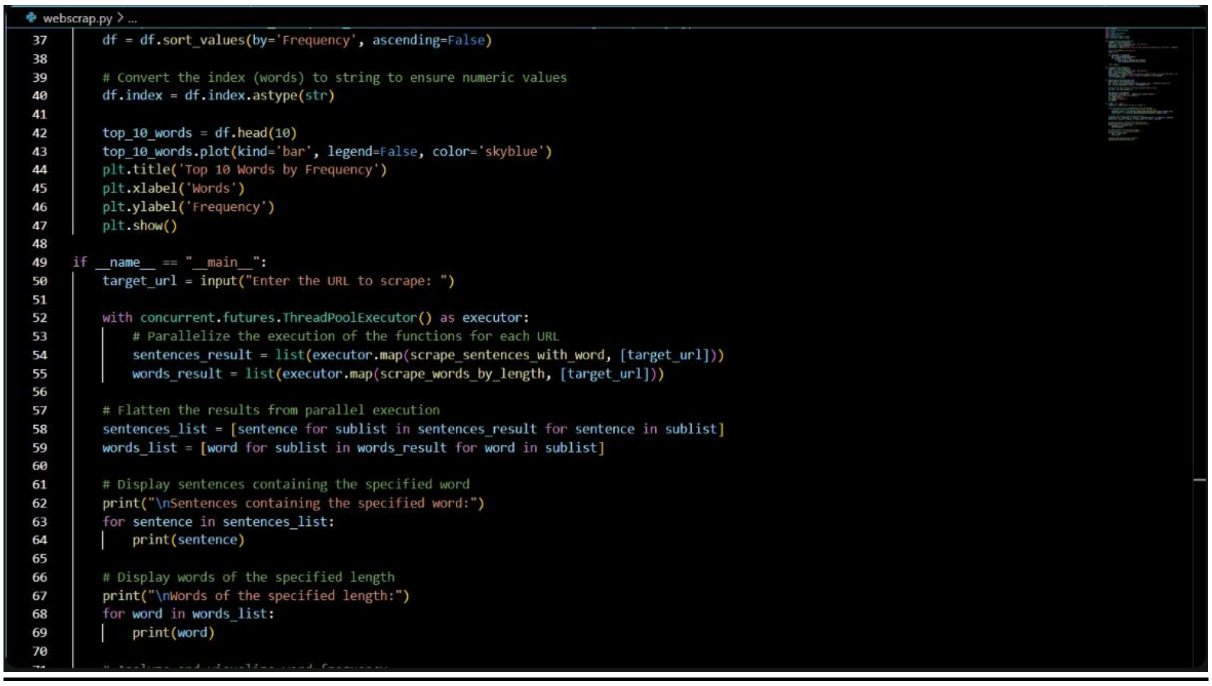
Matplotlib for Visualization:

The Matplotlib library is used for data visualization. The program creates a bar chart to visualize the top 10 words by frequency, providing a clear representation of the most common words.

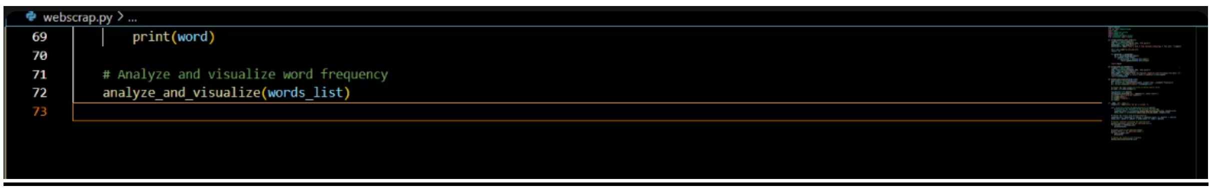
CODE



```python
import requests
from bs4 import BeautifulSoup
import spacy
import concurrent.futures
import pandas as pd
import matplotlib.pyplot as plt
from collections import Counter

def scrape_sentences_with_word(url):
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    paragraphs = soup.find_all('p')
    search_word = input(f"Enter a word to find sentences containing it from {url}: ").lower()

    nlp = spacy.load("en_core_web_sm")
    result = []

    for paragraph in paragraphs:
        doc = nlp(paragraph.get_text())
        for sentence in doc.sents:
            if search_word in sentence.text.lower():
                result.append(sentence.text.strip())

    return result

def scrape_words_by_length(url):
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    words = soup.get_text().split()
    word_length = int(input(f"Enter the character length of words to display from {url}: "))
    filtered_words = [word for word in words if len(word) == word_length]
    return filtered_words

def analyze_and_visualize(words_list):
    word_counts = Counter(words_list)
    df = pd.DataFrame.from_dict(word_counts, orient='index', columns=['Frequency'])
    df = df.sort_values(by='Frequency', ascending=False)
```



```python
    df = df.sort_values(by='Frequency', ascending=False)

    # Convert the index (words) to string to ensure numeric values
    df.index = df.index.astype(str)

    top_10_words = df.head(10)
    top_10_words.plot(kind='bar', legend=False, color='skyblue')
    plt.title('Top 10 Words by Frequency')
    plt.xlabel('Words')
    plt.ylabel('Frequency')
    plt.show()

if __name__ == "__main__":
    target_url = input("Enter the URL to scrape: ")

    with concurrent.futures.ThreadPoolExecutor() as executor:
        # Parallelize the execution of the functions for each URL
        sentences_result = list(executor.map(scrape_sentences_with_word, [target_url]))
        words_result = list(executor.map(scrape_words_by_length, [target_url]))

    # Flatten the results from parallel execution
    sentences_list = [sentence for sublist in sentences_result for sentence in sublist]
    words_list = [word for sublist in words_result for word in sublist]

    # Display sentences containing the specified word
    print("\nSentences containing the specified word:")
    for sentence in sentences_list:
        print(sentence)

    # Display words of the specified length
    print("\nWords of the specified length:")
    for word in words_list:
        print(word)

```



```python
        print(word)

    # Analyze and visualize word frequency
    analyze_and_visualize(words_list)

```

Key functions in the program and their purposes

scrape_sentences_with_word(url):

Purpose: Scrape sentences containing a specified word from a given URL.

scrape_words_by_length(url):

Purpose: Scrape words of a specified character length from a given URL.

analyze_and_visualize(words_list):

Purpose: Analyze and visualize word frequency data.

__main__ block:

Purpose: Main execution block.

Key Steps:

Take user input for the target URL. Utilize parallel processing for efficient scraping. Display results and visualize word frequency. External Libraries Used:

requests Library:

Used for making HTTP requests to fetch the HTML content of web pages.

BeautifulSoup Library:

Used for parsing HTML content and navigating the HTML tree structure.

spacy Library:

Used for natural language processing tasks, specifically sentence boundary detection (sentencizer).

concurrent.futures Library:

Used for parallelizing the execution of web scraping functions.

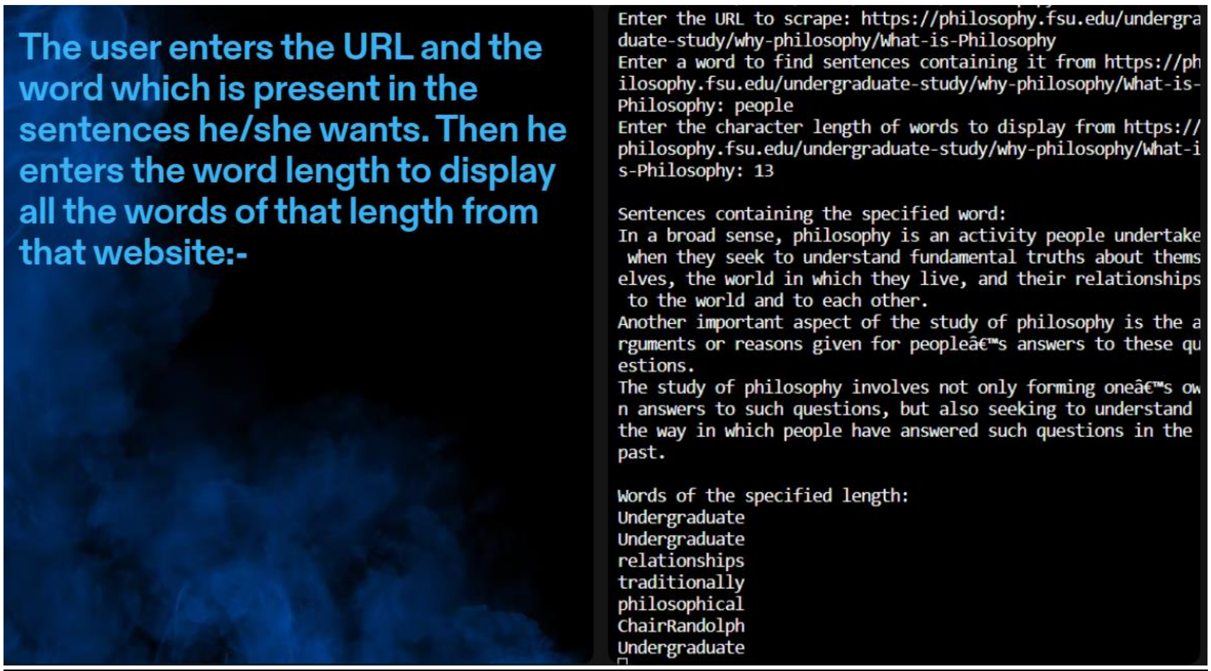
pandas Library:

Used for creating and manipulating DataFrames for efficient data organization.

matplotlib Library:

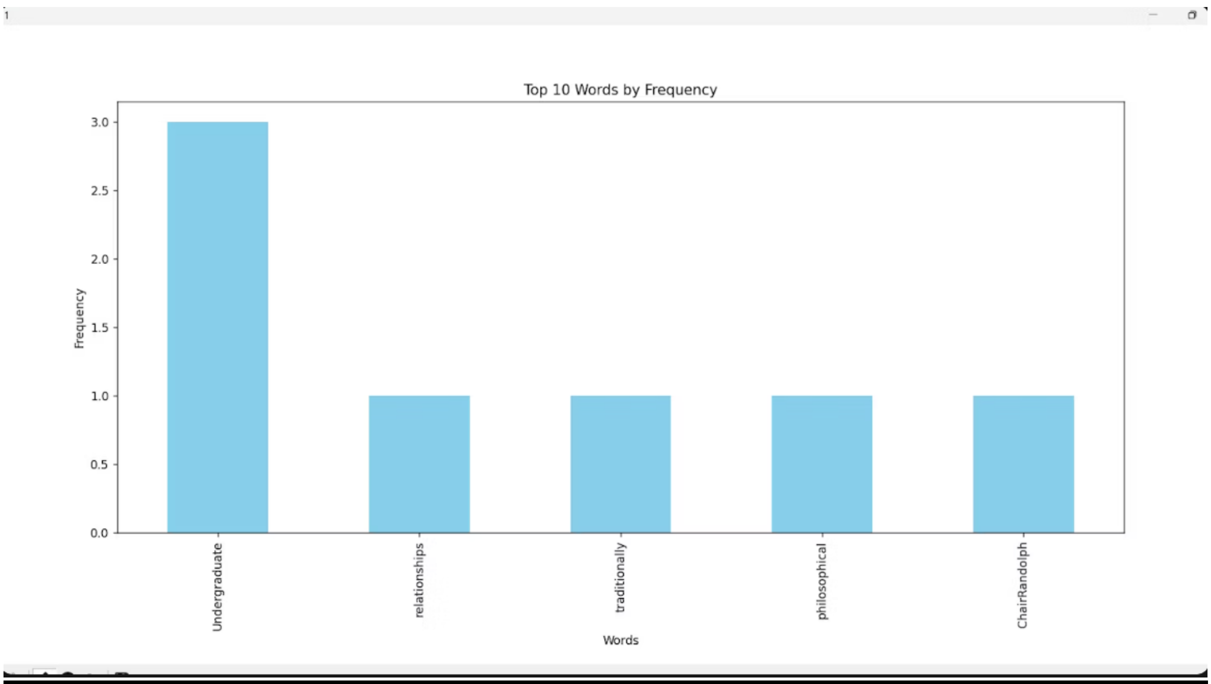
Used for creating visualizations, specifically a bar chart to visualize word frequency.

OUTPUT



The user enters the URL and the word which is present in the sentences he/she wants. Then he enters the word length to display all the words of that length from that website:-

```
Enter the URL to scrape: https://philosophy.fsu.edu/undergra
duate-study/why-philosophy/What-is-Philosophy
Enter a word to find sentences containing it from https://ph
ilosophy.fsu.edu/undergraduate-study/why-philosophy/What-is-
Philosophy: people
Enter the character length of words to display from https://
philosophy.fsu.edu/undergraduate-study/why-philosophy/What-i
s-Philosophy: 13

Sentences containing the specified word:
In a broad sense, philosophy is an activity people undertake
 when they seek to understand fundamental truths about thems
elves, the world in which they live, and their relationships
 to the world and to each other.
Another important aspect of the study of philosophy is the a
rguments or reasons given for peopleâ€™s answers to these qu
estions.
The study of philosophy involves not only forming oneâ€™s ow
n answers to such questions, but also seeking to understand
the way in which people have answered such questions in the
past.

Words of the specified length:
Undergraduate
Undergraduate
relationships
traditionally
philosophical
ChairRandolph
Undergraduate
```



Considerations for future improvements in the web scraping tool include:

>Configurability

>Data Storage and Retrieval

>Support for Multiple Languages

>Interactive Data Exploration

>Legal and Ethical Considerations

>USER INTERFACE

>Configurability

>Error Handling and Logging

>Data Storage and Retrieval

>Support for Multiple Languages

CONCLUSION

In summary, the web scraping project successfully showcased the extraction of targeted information from web pages. The efficient utilization of the `requests` and `BeautifulSoup` libraries for web data retrieval and parsing, along with the integration of spacy for effective natural language processing would be among the key points to note.

The implementation of parallel processing through `concurrent.futures.ThreadPoolExecutor` significantly enhanced the project's ability to handle multiple URLs concurrently, improving overall efficiency. The analysis and visualization of word frequency, utilizing the `Counter` class and Matplotlib, provided valuable insights into the content extracted from the web.As future considerations, potential enhancements were identified, encompassing user interface improvements, configurability options, and adherence to ethical scraping practices. These insights lay the groundwork for refining the tool's functionality and extending its capabilities.

In conclusion, the project establishes a robust foundation for web scraping endeavors, emphasizing adaptability, scalability, and a commitment to best practices. The outcomes pave the way for further development and refinement, ensuring the tool's effectiveness in diverse web scraping scenarios.

REFERENCES

Taken help from wikipidea and websites:-

GeeksforGeeks

https://www.geeksforgeeks.org/introduction-to-web-scraping/

CareerFoundry

https://careerfoundry.com/en/blog/data-analytics/web-scraping-guide/