Sensitive Content retrieval and analysis for python security

Chris D Wise Jr

Honor Project

s01095843@acad.tri-c.edu

How to Install and Run the Application

After downloading the zip file from GitHub, the first step is to extract its contents. You can do this by locating the zip file on your system and using an extraction tool or command. Once the extraction is complete, open your terminal and navigate to the extracted folder. While running the application, you may encounter an error regarding missing modules. If this happens, you will need to install two specific libraries: “requests” and “beautifulsoup4”. You can easily install these packages using pip, the Python package manager, by executing the following commands in your terminal:

-bash

pip install requests

pip install beautifulsoup4

-

Once the necessary libraries are installed, make sure you are in the “SCRAPS-mainz” directory, as this is crucial for the application to run correctly. To confirm your location in the terminal, you can use the “pwd” command (on macOS/Linux) or `cd` command (on Windows) to verify that you are in the right folder.

Finally, to execute the application, run the following command:

-bash

python Honors.py

-

It's important not to forget the file extension “.py”, as it specifies that you are trying to run a Python script. Following these steps will help you successfully install and run the application.

A guide to the application’s functionality and user interface.

The scrape application is designed to analyze a user's input URL and search for sensitive information such as email addresses, phone numbers, Social Security numbers, and more. It allows users to extract this data into a CSV file. When the application is launched, the user starts at the main menu, which displays the heading and provides two options: "Scrape Website" or "Exit." If the user selects "Scrape Website," they will be prompted to enter a URL. If data is successfully found, a message will appear stating "Data successfully scraped," followed by another set of choices:

1. Display Scraped Data

2. Export Data to CSV File

3. Exit

If the user selects option one, the scraped data will be displayed on the screen in the terminal. If option two is chosen, a CSV file containing the data will be created in a list format.

Known limitations and future improvement suggestions.

The program currently provides a basic search functionality to help identify the sensitive data we are looking for. However, there are limitations; since we rely on specific patterns to detect this data, it may struggle to find non-standard data formats. Additionally, this tool does not validate website terms of service or data privacy regulations. There are also performance constraints, particularly when scraping large or complex web pages, as this may result in slowdowns due to its dependence on requests. Furthermore, scraper works only on static HTML content, so if a website uses JavaScript to load data, you may encounter issues.

For future improvements, we can integrate new libraries, such as Selenium, to help manage content loading from websites. Additionally, we can implement a system to validate the terms of service of these websites before we start scraping, making the process more ethical. We could also explore using machine learning modules to detect patterns that are difficult to identify, especially those in different languages or international formats. This approach would enable the system to learn how to recognize patterns in various ways, potentially identifying insights that the naked eye might overlook.

Explanation of the code structure for future developers.

The program includes six different functions. One of them is the "center text" function, which allows the heading, my name, my school, and my school email to stand out by aligning them in the center at the beginning.

The next function is "display data," which will present the information in a structured and readable format. It organizes the data in dictionary form, with keys representing the patterns being searched for and the values containing the matches found. The function then displays this information clearly. If no data is found, it will provide the message, “No data to display.”

The following is the "web scraper" function, which retrieves data from a website based on the user's selection. The program prompts the user to enter a URL, and if the URL is in a valid format, it fetches the web page content using a GET request and returns it as a formatted HTML string. If an error occurs during this process, the function will provide an appropriate error message along with the cause of the error.

The "regex" function searches the scraped HTML content for sensitive information using predefined patterns. The types of sensitive data we are looking for include Social Security numbers, phone numbers, email addresses, credit card numbers, and birth dates. Any matches found are stored in a dictionary, where the key represents the data type and the values contain the corresponding matches.

This function returns any found data in a directory format. The export to CSV function is designed to take a directory of the scraped data and save it to a CSV file, allowing you to store the extracted data.

The main menu function serves as the user interface, providing all the necessary prompts for users to navigate through the program. It has a loop that allows users to continue, to perform multiple scrapes.