Documentation for Restaurant Scraper

Overview

This documentation provides a detailed description of the Restaurant Scraper script, its purpose, functionality, and how to use it. The script is designed to extract restaurant information from Google search results and save the data into a CSV file.

Features

- **Automated Scraping**: Extracts restaurant names, ratings, addresses, and phone numbers.
- **Multi-Page Support**: Can scrape multiple pages of Google search results.
- **Data Export**: Saves the scraped data to a structured CSV file.
- **Error Handling**: Manages exceptions and ensures proper WebDriver cleanup.

Dependencies

The script relies on the following Python libraries:

1. **selenium**: Automates interaction with web pages.
2. **pandas**: Handles data manipulation and CSV creation.
3. **time**: Introduces delays to prevent detection as a bot.

Install dependencies using pip:

pip install selenium pandas

Script Components

1. Imports

```
import time
import pandas as pd
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.action_chains import ActionChains
```

2. Selenium WebDriver Setup

Configures Chrome WebDriver to run in headless mode:

chrome_options = Options()

chrome_options.add_argument("--headless")

driver = webdriver.Chrome(options=chrome_options)

3. scrape_restaurant_info Function

Purpose:

Scrapes restaurant details from Google search results.

Parameters:

- query (str): Search query (e.g., "restaurants in New York").
- num_pages (int): Number of search result pages to scrape (default is 5).

Functionality:

- Constructs a Google search URL.
- Iterates through pages, extracting:
    - Restaurant names.
    - Ratings.
    - Addresses.
    - Phone numbers.
- Handles missing data gracefully.

Returns:

A list of restaurant details.

Example:

restaurant_info = scrape_restaurant_info("restaurants in San Francisco", num_pages=3)

4. save_to_csv Function

Purpose:

Saves scraped data to a CSV file.

Parameters:

- data (list): List of restaurant details.

- filename (str): Name of the output CSV file (default is restaurants.csv).

Functionality:

- Converts the data into a pandas DataFrame.
- Optionally sorts data by restaurant name.
- Exports the DataFrame to a CSV file.

Example:

save_to_csv(restaurant_info, filename="output.csv")

5. Main Execution Block

Executes the scraping process and saves the data:

```
if __name__ == "__main__":
    search_query = "restaurants in New York"
    num_pages = 5
    restaurant_info = scrape_restaurant_info(search_query, num_pages)

    if restaurant_info:
        save_to_csv(restaurant_info)
    else:
        print("No data to save.")
```

Usage

1. **Update the Search Query** Modify the search_query variable in the script to your desired location or keyword.
2. search_query = "restaurants in Los Angeles"
3. **Set the Number of Pages** Adjust the num_pages variable to control how many pages of results to scrape.
4. num_pages = 3
5. **Run the Script** Execute the script in a Python environment:
6. python main.py
7. **Check the Output** The scraped data will be saved in a CSV file (default: restaurants.csv).

Output Format

The CSV file contains the following columns:

1. **Restaurant Name**
2. **Rating**
3. **Phone Number**
4. **Address**

Error Handling

- Ensures that the WebDriver quits even if an error occurs.
- Logs errors to the console for debugging.

Potential Improvements

1. **Dynamic CSS Selectors**: Enhance flexibility to adapt to changes in Google's DOM structure.
2. **Proxy Support**: Add proxy configuration to avoid IP blocking.
3. **User-Agent Rotation**: Rotate user-agents to mimic real users.
4. **Parallel Execution**: Scrape multiple pages simultaneously for faster execution.

Conclusion

This script provides an automated solution for extracting restaurant information from Google search results. With minor enhancements, it can be extended to scrape data from other sources or handle more complex requirements.