**Web\_finder Documentation**

**1. Overview**

The Web\_finder script automates the process of gathering textual information and relevant images for a given topic from the internet. It utilizes Google search capabilities to find relevant web pages, parses their content to extract important points, and attempts to locate and display a direct image related to the topic. The extracted information is then saved to a text file, and a Google Custom Search Engine (CSE) block is displayed for manual searching.

**2. Features**

* **Topic-based Information Retrieval**: Searches the web for general information about a specified topic.
* **Important Points Extraction**: Extracts key paragraphs from search results to summarize the topic.
* **Image Search and Display**: Attempts to find a direct image URL related to the topic and opens it in the default web browser.
* **Information Storage**: Saves the extracted text and image URL (if found) into a .txt file.
* **Manual Search Option**: Provides an embedded Google Custom Search Engine (CSE) for users to conduct further manual searches.

**3. Dependencies**

The script relies on the following Python libraries:

* googlesearch-python: For performing Google searches.
* requests: For making HTTP requests to retrieve web page content.
* beautifulsoup4: For parsing HTML content and extracting data from web pages.
* webbrowser: For opening URLs in the default web browser.
* IPython.display: For displaying HTML content (specifically for the Google CSE).

These dependencies can be installed using pip:

Bash

pip install googlesearch-python requests beautifulsoup4

**4. Functions**

**4.1** get\_topic\_info(topic)

This function searches the web for general information related to the given topic and extracts important textual points.

* **Parameters**:
  + topic (str): The subject for which to retrieve information.
* **Returns**:
  + list: A list of strings, where each string represents an important point or paragraph extracted from the search results.
* **Process**:
  + Performs a Google search for the topic.
  + Iterates through the top 5 search results.
  + For each URL, it makes an HTTP GET request to fetch the page content.
  + Parses the HTML content using BeautifulSoup.
  + Extracts the first 3 paragraphs from the page.
  + Adds paragraphs longer than 50 characters to the important\_points list.
  + Includes a 2-second delay between searches to avoid being blocked.

**4.2** find\_and\_display\_image(topic)

This function searches Google Images for an image related to the topic and attempts to open a direct image URL in the user's default web browser.

* **Parameters**:
  + topic (str): The subject for which to find an image.
* **Returns**:
  + str or None: The direct URL of the found image if successful, otherwise None.
* **Process**:
  + Constructs an image search query (e.g., "topic image").
  + Performs a Google search for the image query, limiting results to 5 and enabling safe search.
  + Iterates through the search results, specifically looking for Google Images URLs.
  + Attempts to find a direct image URL from og:image meta tags or <img> tags.
  + Filters out thumbnail URLs.
  + If a direct image URL is found, it attempts to open it in the default web browser.

**4.3** create\_info\_file(topic, points, image\_url=None)

This function creates a text file containing the extracted important points and the URL of the displayed image.

* **Parameters**:
  + topic (str): The original search topic.
  + points (list): A list of important textual points.
  + image\_url (str, optional): The URL of the image that was opened in the browser. Defaults to None.
* **Process**:
  + Constructs a filename based on the topic (e.g., your\_topic\_info.txt).
  + Writes a header with the topic.
  + Writes the extracted important points, numbering each one.
  + Adds a section indicating whether an image was opened in the browser and provides its URL if available.

**4.4** get\_info\_from\_web(query, num\_results=3)

This function performs a general web search for a given query and extracts content from the top search results.

* **Parameters**:
  + query (str): The search query.
  + num\_results (int, optional): The number of search results to process. Defaults to 3.
* **Returns**:
  + list: A list of dictionaries, where each dictionary contains the 'url' and 'content' extracted from a web page.
* **Process**:
  + Performs a Google search for the query.
  + Iterates through the specified number of search results.
  + For each URL, it fetches the page content and parses it.
  + Extracts the first 5 paragraphs from the page.
  + Adds paragraphs longer than 100 characters to the extracted content.
  + Stores the URL and extracted content in a dictionary and appends it to the extracted\_info list.
  + Includes a 1-second delay between processing URLs.

**4.5** display\_google\_cse(cx\_code="23780fbb727f84965")

This function displays an embedded Google Custom Search Engine (CSE) block, allowing users to perform manual searches directly within the environment where the script is run (e.g., a Jupyter notebook or Google Colab).

* **Parameters**:
  + cx\_code (str, optional): The Google Custom Search Engine ID. Defaults to "23780fbb727f84965".
* **Process**:
  + Generates HTML and JavaScript code for the Google CSE.
  + Uses IPython.display.HTML to render this code, making the search bar interactive.

**5. How to Run**

1. **Save the script**: Save the provided code as a Python file (e.g., web\_finder.py).
2. **Install dependencies**: Open a terminal or command prompt and navigate to the directory where you saved the script. Run the following command:

Bash

pip install googlesearch-python requests beautifulsoup4

1. **Execute the script**: Run the script from your terminal:

Bash

python web\_finder.py

1. **Enter topic**: The script will prompt you to "Enter the topic you want to search for:". Type your desired topic and press Enter.

**6. Output**

Upon execution, the script will:

1. Print messages indicating its search progress for general information and images.
2. Open a new tab in your default web browser displaying an image related to your topic (if found).
3. Create a text file named {your\_topic}\_info.txt in the same directory as the script. This file will contain:
   * A header with the topic.
   * A list of important points extracted from web pages.
   * The URL of the image that was opened in the browser.
4. Print the extracted information directly to the console, showing content from different URLs.
5. Display an interactive Google search block in the console or your environment (if running in an IPython-compatible environment like Colab or Jupyter).

**7. Error Handling**

The script includes basic error handling for common issues such as:

* requests.exceptions.RequestException: Catches errors related to network issues or inability to access a URL.
* **General** Exception: Catches other unexpected errors during web parsing or search operations.
* IOError: Catches errors when writing to the information file.

These errors are printed to the console, allowing the user to understand if a specific URL could not be accessed or processed.