CS 104 Project - Web Crawler

Saksham Rathi

March 30, 2024

Contents

1 Objective

The python file `web_crawler.py` implement a Web Crawler using Python and its libraries. Another python file `customized_web_crawler.py` has some added customizations.

2 Introduction to Web Crawler

Web crawler is a computer program that is used to search and automatically index website content and other information over the internet. These programs, or bots, are most commonly used to create entries for a search engine index.

3 Applications

Some of the applications of a web crawler have been described in brief below:

- **Search Engine Indexing**: They are used extensively by search engines like Google, Bing and Yahoo to discover web pages, index their content and update their search engine databases.
- **Security Assessments**: They can be utilized for security purposes, such as detecting vulnerabilities and security flaws on websites. They can help identify broken authentication mechanisms and insecure configurations.
- **Data Extraction**: They can be used to extract data from websites at scale. Examples include gathering pricing data, extracting news articles etc.

4 Basic Idea of the code

The base website link will be provided as a command line argument by the user. (Refer 6) Now the python file will recursively find out all the unique referenced web links eith "href", "src" or some commonly used attributes with different tags.

Recursion means, from the first input link given, it will be parsed. Let us suppose that the parsed HTML will give 10 links. Now, the code will visit these 10 links individually and then from each of these get more links.

These all links found will be written in a file. The recursion will apply only to internal links i.e. the links that belong to the same domain. Although the external links will be listed, but they will not be crawled on. The basic idea of the code has been taken from [2]

5 Working of the code

The Python code uses different libraries for performing different tasks. The applications of these libraries are explained in the code using comments and also become clear as they get used later in the code. The use of different variables including lists, dictionaries etc also has been explained, moreover desciptive names have been used for the ease of undestanding. There is a main recursive function called scrape. It takes three parameters and then later recursively calls itself. It checks for different attributes in the HTML source file and then updates the dictionaries accordingly. If it finds that the link is internal and has not been called before, then it recursively calls the function on that link. The code also breaks the terminal arguments and then performs its work. If the output file is given as an argument then the result is printed in that file with a legible syntax to allow the users to understand the output. If an output file is not present, then the output is printed on the terminal. The file `customized_web_crawler.py` also produces two graphs at the end of the code.

6 Usage

The usage of the files is simple. Open a terminal on your computer. Change the directory to the location where the python file is present. Ensure that you have the following libraries installed: argparse, requests, urllib, sys, os, re, matplotlib and collections. Now run the following command on the terminal.

python web_crawler.py -u <URL> -t <threshold of recursion> -o <output file>

- **python**: This is a message to the terminal, to compile the python file. This may vary according to the operating system used by the user. Some people have to write python3 instead of python.
- **web_crawler.py**: This is the file name. The user has to ensure that the present working directory has the required file present. For the customized version of the code, we should write `customized_web_crawler.py` instead.
- **-u <URL>**: -u denotes the start of the URL. In place of `<URL>`, provide the link which the program is supposed to crawl upon. If this is not provided, the program will raise an error message.
- **-t <threshold of recursion>**: -t denotes the start of the maximum recursion level to be specified by the user. In place of `<threshold of recursion>`, the user is supposed to write a valid depth until which the recursion has to go. The value should be greater than zero. The default value is set to 10, if not specifed.
- **-o <output file>**: -o denotes the start of the name of the output file. In place of `<output file>`, specify the file name in which the output has to be printed. If not specified, the output will be written on terminal.

Now open the output file (if provided) to see the links which the program came across. The usage of terminal arguments in python has been taken from [1]

7 Customizations Included

In the file `customized_web_crawler.py`, there are various customizations included which are as follows:

1. **Attributes**: This python web crawler not only crawls over html or src attributes. It will also print the links which are present in `<style>, <audio> and <video>` tags. Therefore, it provides a more detailed description of the website considered.
2. **Type of Files**: On running the code, a pdf file named "`Type_of_Files.pdf`" will be automatically saved to the location where the code is present. This file contains a bar chart showing the number of files/links of each type. Some of the prominent types include image, document, pdf, php etc. Based on the file type chart, a user can decide whether the website is image-based or text-based, whether it is interactive or not, etc.
3. **Count of files at each recursive level**: Another pdf file named "`Count_of_files_level.pdf`" will also be saved just after the complete execution of the code. This file will be a line chart showing the number of files/links found at each recursive depth. This file also shows the power of recursion, as generally the number of files will increase exponentially after each level.

The above customizations are based on the fact that pictures speak more than words. These graphs produce great visual clarity for the user. The user can collect important facts and statistics from these graphs and can get impressed by the power of my favourite library: "Matplotlib". The useful functions of matplotlib have been taken from [3]

8 Makefile

I have also prepared two Makefiles for the running of code and compilation of latex report.

One of the `Makefile` is present in the same directory as the python files. For running the Makefile, open the terminal for that directory. Write `make basic URL=<url> THRESHOLD=<threshold of recursion> and OUTPUT=<output file>` on the terminal window. Instead of the `<variables>`, write your own input.

For running `customized_web_crawler.py`, write `make advanced URL=<url> THRESHOLD=<threshold of recursion> and OUTPUT=<output file>` on the terminal window. I have also defined a `clean target` in the Makefile, which will remove all the output files, so produced.

The other `Makefile` is present in the directory same as that of the latex file. (`report.tex`) For running this makefile, just open the terminal for that location and write `make` on the terminal window. The Makefile will automatically compile latex and bibtex files and will produce the `report.pdf` file.

This extensive use of Make eases our work. We have to write short commands on the terminal. Moreover, this is also quite user-friendly.

9 Sample

The code has already been tested on numerous websites. A sample output file has been attached (output.txt and two graphs) in the folder same as that of the code files. In this sample output, the code has been run on https://www.google.co.in/ for recursion level 3. Being a heavy website, google has large number of links and thus the process took around 10 seconds. The user can see the sample file for the validation of the working of the code. Certain screenshots of the command line arguments and the output files have also been attached in the same folder as that of the code files.

10 Precautions

The execution of the code requests the HTML source file using certain python libraries. This will require the use of internet at the user's end. Some of the websites require the use of certain certificates for accessing their HTML source file. The user is expected to avoid the use of such websites without proper SSH (Secure Socket Shell) certificates. Sometimes, the website may allow the code to be extracted for the first recursion, but the further process may get blocked. It also happens that often due to improper internet connection at the user's end, the final output gets terminated. Certain warning messages may get printed on the terminal during the execution of the code when used for certain websites. However, if the output produced seems complete, such messages may be ignored safely. For higher recursion levels, the time it takes to crawl the website increases exponentially. So, the user may have to wait for the complete execution of the code. However, with proper internet and free access websites, the program proves to be quite useful.

11 LaTeX Report

The file `report.tex` contains the report typed in LaTeXlanguage. On compilation of the file either directly using terminal or using Makefile, a pdf `report.pdf` will be generated. There will also be some auxiliary files generated which include: report.toc (for table of contents), report.synctex.gz (Synchronization between source latex file and pdf generated), report.out (Output messages and warning generated during compilation), report.fls (records the files accessed during compilation), report.fdb_latexmk (for interactive preview), report.blg (log of bibliographic references using BibTeX), report.bbl (formatted references) and report.aux(labels, cross-references, citations).

Bibliography

[1] Command Line Arguments in Python. https://www.geeksforgeeks.org/command-line-arguments-in-python/.

[2] Python Program to Recursively Scrape all the URLs of the Website. https://www.geeksforgeeks.org/python-program-to-recursively-scrape-all-the-urls-of-the-website/.

[3] Kameswari Chebrolu. CS 104 Course Slides - Python, Matplotlib, 2023.