Wilfrid Laurier University
Department of Physics and Computer Science

CP423A | Text Retrieval & Search Engine | Winter 2023

1st Assignment

Overview: A web crawler, also known as a spider or robot, is a program or automated script that browses the World Wide Web in a methodical, automated manner. Web crawlers are typically used by search engines to index the content of websites, so that users can search for specific information. They typically start with a list of URLs to visit and then follow the links on those pages to other pages, and so on. The process is repeated until all of the pages on a website have been indexed.

This task consists of three questions that aim to enhance your practical understanding of web crawling. Completing this assignment will also prepare you for future assignments and the final project, where web crawling will be a key component.

What to submit? For the assignment, each question explains its specific deliverables. Also, you need to submit a **report.pdf** file that includes group members information, details of each group member contribution, and also your thorough explanation for each of questions. Finally, you are required to upload the **A1.zip** file into MyLearningspace drop box by the end of **Monday 13th February**. Only one of group members should submit group's deliverable. Therefore, arrange it internally. Please avoid redundant submissions for a group.

CP423A | Text Retrieval & Search Engine | Winter 2023

1st Assignment

Question-1(30 points). write webcrawler1.py that crawl through a web site. Your program should be callable from command prompt like so:

```
python webcrawler1.py [options] initialURL
```

a) Your program should:
   1. Download content of `initialURL`
   2. Write downloaded content in H.txt which H is calculated hash value for `initialURL`.
      i. You can use any hash library such as `hashlib`
   3. Extract all hyperlinks of the downloaded content.
      i. You can consider `<a href="URL" …>…</a>` as hyperlinks to extract URLs.
   4. Append a line at the end of file `crawler1.log` that includes `<H,URL,Download DateTime, HTTP Response Code>`
   5. Repeat step 1-4 for all extracted links.
      i. Consider parameter `--maxdepath` to stop your crawling. It is a compulsory parameter to input.

b) `[Options]` are as below:
   1. `--maxdepath`: Maximum number of depths to crawl from initialURL.
   2. `--rewrite`: if value is True and file H.txt exists for current URL, it re-download and re-write URL. Otherwise, skips step 2-5. Default value is False.
   3. `--verbose`: if Ture, print <URL,depth> on the screen. Default value is False.

Question-2(35 points). Google Scholar is a widely accessible platform that offers information related to academic papers and researchers. It serves as a database where researchers who have published papers can be located. Each researcher indexed in the repository is assigned a unique URL, which leads to their individual page. This page contains a wide range of information pertaining to the researcher, such as their co-authors, statistics, and a list of their published papers. It serves as a valuable tool for academic researchers, students, and educators looking for information on specific individuals or topics in the academic field. It allows them to easily access and discover relevant research materials, and also to find the researchers working on similar topics. Additionally, Google Scholar is a search engine that indexes the full text of scholarly literature across an array of publishing formats and disciplines. This enables users to search for articles, theses, books, and conference papers, from academic publishers, professional societies, preprint repositories, universities, and other scholarly organizations.

Write webcrawler2.py that crawl through a Google Scholar page. Your program should be callable from command prompt like so:

```
python webcrawler2.py researcherURL
```

a) Your program should:
   1. Download content of `researcherURL`
   2. Write downloaded content in H.txt which H is calculated hash value for `researcherURL`.
      i. You can use any hash library such as `hashlib`
   3. Extract all information from the pages as below and save it in H.json format as below:

```
{
"researcher_name" : '…",
"researcher_caption" : "…",
"researcher_institution" : "…",
"researcher_institution" : "…",
"researcher_keywords" : {all keywords (comma separated)},
"researcher_imgURL": "…", //link of user's image
"researcher_citations" : {"all":"…", "since2018" : "…"},
"researcher_hindex" : {"all":"…", "since2018" : "…"},
"researcher_i10index" : {"all":"…", "since2018" : "…"},
"researcher_coauthors" : { {"coauthor_name":"…", "coauthor_title":"…",
"coauthor_link":"…"},{…}},
"researcher_papers" : {{"paper_title":"…", "paper_authors":"…", "paper_journal":"…",
"paper_citedby":"…", "paper_year":"…"},{…}}
}
```

b) **Bonus (10 points):** A significant number of researchers who have a substantial number of publications, have a feature on their Google Scholar page where a button labeled "Show More" is located at the bottom of the page. This button needs to be clicked in order to load and view the additional papers. However, extracting all papers needs more advanced techniques to automate your crawler for simulating click. To make the process more efficient, add this feature to your code.

c) **Note:** It is important to note that when sending multiple requests in a short period of time, Google may identify you as a potential threat and block your access for a certain amount of time. In order to avoid this, for the purpose of this assignment, which is focused on extracting information, it is recommended to crawl and save a few pages that contain all of the necessary information to extract. This way, you can dedicate most of your time to experimenting and refining your code using the collected data. Once you feel confident in your data extraction abilities, you can then integrate all of the various parts of your code and test it as a whole.

Question-3(35 points). The process of extracting information from HTML pages is crucial for multiple reasons. One of its main benefits is that it enables the collection of data from the web in an automated and efficient manner. This can be applied for various purposes such as data mining, natural language processing and web scraping. This is particularly beneficial for companies that need to gather and examine large amounts of information from the web, for instance, market analysis, sentiment analysis, and price comparisons. The Section 3.8 of textbook delves into the proportion of HTML tags within and outside the primary content of a page. Additionally, the section presents a function to find the approximate location of the start and end of the content within an HTML page, thereby maximizing the results.

$$f(i,j) = \sum_{n=0}^{i-1} b_n + \sum_{n=i}^{j} (1 - b_n) + \sum_{n=j}^{N-1} b_n$$

Write webcrawler3.py that asks for a URL and find content and save content. Also, it should plot function $f(i,j)$. Your program should be callable from command prompt like so:

```
python webcrawler3.py initialURL
```

Your program should:

- Download the HTML content of `initialURL`
- Save downloaded content in H.html which H is hash value of URL
- Replace all HTML tags with 1
  - You can assume that HTML tag as "<?????>" where ???? is a combination of any characters having any length. For example, <H1>, <a htef="google.com">, </p>, <script>, etc.
  - You can use regular expressions to simplify this HTML tags' identification and replacement process.
- Replace words (tokens) with 0.
  - Note: you should replace a word with 0 not a character with 0.
- For all $0 \leq i < j \leq N-1$ calculates $f(i,j)$ and prints (i^*, j^*)which is the best combination of i,j that maximizes $f(i,j)$.
- Write the identified main content of downloaded page (which is located between i^* and j^*) in H.txt file. H is hash value of URL.
- For all calculated values of i,j plot function $f(i,j)$ similar to one of the following plots. X-axis and y-axis are for i and j, respectively. The color of histogram for 2D or z-axis for 3D represents $f(i,j)$.