Assignment 3

Information Retrieval and Web Search
Winter 2024
Total points: 90 points
Issued: 02/20/2024 Due: 03/11/2024

All the code has to be your own (exceptions to this rule are specifically noted below). The code must run on the CAEN environment. For the purpose of this assignment, you can use the following libraries: requests, bs4, urlib, numpy. If you need other libraries, please check with the instructors.

You can discuss the assignment with others, but the code is to be written individually. You are to abide by the University of Michigan/Engineering honor code; violations will be reported to the Honor Council.

<u>General Canvas submission instructions:</u>

- Include all the files for this assignment in a folder called *[your-uniqname].Assignment3/* **<u>Do not</u>** include the content of the pages you crawled.
  For instance, mihalcea.Assignment3/ will contain crawler.py, crawler.output, links.output, pagerank.py, pagerank.output, crawler.pagerank.answers
- Archive the folder using tgz or zip and submit on Canvas by the due date.
- Make sure you include your name and uniqname in each program and in the *crawler.pagerank.answers* file.
- Make sure all your programs run correctly on the CAEN machines.

[45 points] Web crawler.

Write a Web crawler that collects the URL of webpages from the umich domain. Your crawler will have to perform the following tasks:
a. Start with https://eecs.engin.umich.edu/
b. Perform a Web traversal using a breadth-first strategy. You should only crawl html webpages.
c. Keep track of the traversed URLs, making sure:
 - they are part of the eecs.engin.umich domain (or the old eecs.umich domain). This will include any pages under eecs.umich, eecs.engin.umich, ece.engin.umich, cse.engin.umich
Hence, your algorithm should only store links that are within the list above.
 - they were not already traversed (i.e., avoid duplicates, avoid cycles)
d. Stop after you *<u>identified</u>* 2500 URLs.

<u>Programming guidelines:</u>
Write a program called *crawler.py* that implements the Web crawler. The program will receive two arguments on the command line:

- the first one consists of the name of a file containing all the seed URLs (for the purpose of this assignment, this file will only contain one seed, namely http://eecs.engin.umich.edu);

- the second one consists of the maximum number of URLs to be identified (for the purpose of this assignment, the value of this parameter will be 2500)

The *crawler.py* program should be run using a command like this:
% python crawler.py myseedURLs.txt 2500

It should produce two files:

- A file called *crawler.output* including a list of all the URLs being identified by the crawler, in the order in which they are identified. E.g.:
  http://eecs.engin.umich.edu
  http://eecs.engin.umich.edu/eecs/academics/academics.html
  http://eecs.engin.umich.edu/eecs/about/why-eecs.html
  Etc.
- A file called *links.output*, including all the links (in the form of *(source_URL, URL))* that you identify in your crawl, which will serve as the directed edges in the graph representation for the next problem. E.g.,
  http://eecs.engin.umich.edu http://cse.engin.umich.edu
  http://eecs.engin.umich.edu http://ece.engin.umich.edu
  http://eecs.engin.umich.edu https://cse.engin.umich.edu/news/
  Etc.

Notes:

- The URLs in the crawler.output file will include both URLs corresponding to pages you downloaded, as well as URLs corresponding to pages you identified on crawled pages (but you did not download). In other words, when you hit the max of 2500 URLs identified, some of the URLs will be for pages that have not been downloaded yet.
- You can use the request library to download the webpages and Beautiful Soup to find the links.
- Consider using a try-except block to prevent network issues that may crash your code.
- Many of the URLs in the links.output file will likely appear only once (i.e., they are only connected to one other URL in your dataset)
- For the purpose of this assignment, checking the robots.txt is not required as we are staying within the eecs/engr domain.
- URLs with the same domain but different protocols should be treated as the same (i.e. https://eecs.engin.umich.edu & http://eecs.engin.umich.edu).
- Make sure you only crawl html files, non-html files should be ignored.

[45 points] PageRank.

Implement the PageRank algorithm and apply it to determine the PageRank score for each of the 2500 URLs you identified. The PageRank implementation should assume:

- An initial value of 0.25 for all the URLs.

- A value of 0.85 for *d.*
- Convergence when the difference between the scores obtained with two iterations of PageRank for each of the URLs falls below 0.001.

<u>Programming guidelines:</u>
Write a program called *pagerank.py* that implements the PageRank algorithm. The program will receive three arguments on the command line:

- the first one consists of the name of the file containing all the URLs (for the purpose of this assignment, this file will contain all the URLs you identified in the previous problem, i.e., crawler.output);
- the second argument is the name of the file containing all the links in the form of *(source_URL, URL)* pairs (for the purpose of this assignment, this file will contain all the URLs you identified in the previous problem, i.e., links.output);
- the third one consists of the convergence threshold for PageRank (for the purpose of this assignment, the value of this parameter will be 0.001).

The *pagerank.py* program should be run using a command like this:
% python pagerank.py crawler.output links.output 0.001

It should produce a file called *pagerank.output* including a list of all the URLs in the *crawler.output* file, along with their PageRank score. The list should be sorted in reverse order of their PageRank score. E.g.:
http://www.eecs.umich.edu 0.9
http://eecs.umich.edu/eecs/academics/academics.html 0.6
http://eecs.umich.edu/eecs/about/why-eecs.html 0.2
Etc.

Notes:

- The number of iterations is not important as long as your implementation converges.
- If you identify links not in the domain, you can disregard them.
- Identifying means that during your crawling, you will identify 2500 URLs, not all of which will be downloaded. In other words, as you keep downloading URLs and identifying new URLs on these downloaded pages, stop when you reach 2500 URLs.
- You will need to handle redirects when identifying URLs and ensure that they are valid.
- Crawler.output and links.output do not necessarily have to have the same number of lines.

<u>Write-up guidelines:</u>

Create a file called *crawler.pagerank.answers*. Include in this file the following information:
1. Amount of time (in seconds) your crawler needed to identify the 2500 URLs.
2. Number of iterations your PageRank implementation performed to convergence.