Homework #3

CS 547/DS 547, Fall 2024

---

100 points total [6% of your final grade]

Due: October 21, 2024 by 11:59pm
[no submission will be accepted after October 24, 2024 at 11:59pm]

Delivery: Submit via Canvas

Part 1. Ranked retrieval using PageRank (80 points)
In this assignment, you will crawl a collection of web pages, calculate their PageRank scores, and then use these PageRank scores in tandem with a boolean query operator to rank the results. Start by downloading hw3.zip and decompressing it; you should find two python files.

a) cs547.py - Just like HW1, this helper class will be used to represent a student's identifying information. Any assignments without an instantiated student object of type Student will not be graded. You do NOT need to modify this file.

b) hw3.py: This is where you will fill out three functions: index_dir(...), tokenize(...) and ranked_search(...).

- index_url(self, url): This function crawls through a web directory of html files and generates an index of the contents. It should also extract hyperlink references for use in computing PageRank scores.

- tokenize(self, text): This function converts a string of terms into a list of valid tokens. For the purposes of this assignment, a valid token consists of only English alphabet characters (a-z, A-Z) or numbers (0-9). All other characters are considered as token delimiters. Convert all letters to lower case. Treat the html source as your input document for purposes of this assignment.

- ranked_search(self, text): This function searches for the terms in "text" in our index and returns **at most** the 10 highest-ranked results based on their PageRank scores. Return **a list of tuples containing (url, the PageRank score)** of relevant search results. Note that returned url documents should include **all terms** in the query/text. The returned tuples should be in descending order by the PageRank score.

Use a teleportation factor of 0.1 in your PageRank calculation (meaning that 90% of the time the random surfer follows links on a page and that 10% of the time, the random walker gets bored and randomly teleports to any page with equal probability).

Feel free to create member variables and inner functions in the PageRankIndex class, but your code should satisfy the basic requirements of index_dir(...), tokenize(...) and ranked_search(...).

We have provided a web based corpus and an index page listing all documents in this corpus in here (http://web.cs.wpi.edu/~kmlee/cs547/new10/index.html). Treat this index as the root node to our webgraph. Pull out anchor tags based on <a> and not based on http:// since URLs may exist in the document but not as an anchor tag. For example, an anchor tag is Wikipedia. Extract http://en.wikipedia.org/wiki/Main_Page. If there is http://en.wikipedia.org/wiki/page in the html without <a> tag, you will ignore it and do not consider it as a valid hyperlink reference. Each valid anchor tag in the html corpus should be treated as a node in the webgraph. If that node doesn't exist in our web corpus, treat it is a leaf node.

Beautiful Soup is a Python library for pulling data out of HTML and XML files. Install the Beautiful Soup library to parse the html documents.

You may install and use the Numpy library in your solution in order to deal with a large matrix, as that will be available in the grading environment. It is not necessary, but may be useful and can be found at http://numpy.scipy.org/.

You should be able to manually check whether your code is doing what you think it should. Once you submit your final code, we will evaluate it by constructing an index and issuing several queries (single term and multiple terms). Note that multiple terms mean more than one term and will be ANDed together for purposes of document retrieval.

Part 2. Traditional Search Engines vs. Generative AI-based Search Engines (20 points)
In this assignment, you will compare traditional search engines (e.g., Google, Bing) with generative AI-based search engines (e.g., perplexity.ai and you.com). Specifically, you'll need to try at least three different search queries, report which queries you used, and assess the pros and cons of both types of search engines. Finally, provide a conclusion on when it is most appropriate to use traditional search engines vs. generative AI-based search engines.

What to turn in:

- Rename hw3.py to hw3_firstname_lastname.py (e.g., hw3_elon_musk.py) and submit to Canvas your **hw3.py file and a pdf file for part 2**.
- This is an individual assignment, but you may discuss general strategies and approaches with other members of the class (refer to the syllabus for details of the homework collaboration policy). At the top of hw3.py you will see a list of COLLABORATORS. Please fill this out with the names of classmates you consulted and the nature of your discussion.