Project 3 – Search Engine

Due dates: 5/24 and 6/2

This assignment is to be done in groups of 3, preferably the same groups that were in place for Project 2. Although this is presented as one single project that will take until the end of the quarter to complete, internally it is organized in 2 separate milestones, each with a specific deadline, deliverables, and score. In doing milestone #1 make sure to look at the evaluation criteria not just of that milestone but also of milestone #2 –part of the milestones' evaluation will be delayed until the final meeting with the TA.

You can use code that you or any classmate wrote for the *previous* projects. You cannot use code written for *this* project by non-group-member classmates. Use code found over the Internet at your own peril -- it may not do exactly what the assignment requests. If you do end up using code you find on the Internet, you must disclose the origin of the code. **As stated in the course policy document, concealing the origin of a piece of code is plagiarism**.

Use the Discussion Board on Canvas to post your questions about this assignment so that the answers can help you and other students as well.

Goal: Implement a complete search engine.

Milestones Overview

	Deadline	**Goal**	**Deliverables**	**Score** (out of 50)
#1	5/24	Produce an initial index for the corpus and a basic retrieval component	Short report (no demo)	20% (12)
#2	6/02	Complete Search System	Code or artifacts + Demonstration	80% (48)

PROJECT: SEARCH ENGINE

Corpus: all ICS web pages

We will provide you with the crawled data as a zip file. This data contains the downloaded content of the web pages that were crawled in assignment 2. You are expected to build your search engine for this data.

Main challenges: full HTML parsing, File/DB handling, handling user input (either using command line or desktop GUI application or web interface)

COMPONENT 1 - INDEX:

Create an inverted index for all the corpus given to you. You can either use a database to store your index (e.g. MongoDB) or you can store the index in a file. Either choice is acceptable.

The index should store more than just a simple list of documents where the token occurs. At the very least, your index should store the tf-idf of every term/document.

Sample Index:

> Note: This is only for your understanding. Please do not consider this as an expected index format. There are much better ways to structure your index.
>
> Index Structure: token – docId1, tf-idf1 ; docId2, tf-idf2
>
> > Example: informatics – doc_1, 5 ; doc_2, 10 ; doc_3, 7

Words in bold and in heading (h1, h2, h3) should be treated as more important than the other words. You can handle this as you want: create separate indexes, or add metadata about the words to the single index.

Optional:

Extra credit will be given for ideas that improve the quality of the retrieval, so you may add more metadata to your index, if you think it will help improve the quality of the retrieval. For this, instead of storing a simple tf-idf count for every page, you can store more information related to the page (e.g. position of the words in the page). To store this information, you need to design your index in such a way that it can store and retrieve all this metadata efficiently. Your index lookup during search should not be horribly slow, so pay attention to the structure of your index

COMPONENT 2 – SEARCH AND RETRIEVE:

Your program should prompt the user for a query. This doesn't need to be a Web interface, it can be a console prompt. At the time of the query, your program will look up your index, perform some calculations (see ranking below) and give out the ranked list of pages that are relevant for the query.

Optional:

Extra credit will be given if your search interface has a GUI.

COMPONENT 3 - RANKING:

At the very least, your ranking formula should include tf-idf scoring, but you should feel free to add additional components to this formula if you think they improve the retrieval.

Optional:
Extra credit will be given if your ranking formula includes parameters other than tf-idf

---

Milestone #1

Goal: Build an index and a basic retrieval component

By basic retrieval component; we mean that at this point you just need to be able to query your index for links (The query can be as simple as single word at this point).

These links do not need to be accurate/ranked. We will cover ranking in the next milestone.

At least the following queries should be used to test your retrieval:

1 – Informatics

2 – Mondego

3 – Irvine

4 – artificial intelligence

5 – computer science

Note: query 4 and 5 are for milestone #2

Deliverables: Submit a report (pdf) in Canvas with the following content:

1. A table with assorted numbers pertaining to your index. It should have, at least the number of documents, the number of [unique] words, and the total size (in KB) of your index on disk.
2. URLs retrieved for each of the queries above

Evaluation criteria:

- Was the report submitted on time?
- Are the reported numbers plausible?
- Are the reported URLs plausible?

Milestone #2

Goal: complete search engine

Deliverables:

- Submit a zip file containing all the artifacts/programs you wrote for your search
- A live demonstration of your search engine

Evaluation criteria:

- Does your program work as expected of search engines?
- How general are the heuristics that you employed to improve the retrieval?
- How complete is the UI? (e.g. links to the actual pages, snippets, etc.)
- Do you demonstrate in-depth knowledge of how your search engine works? Are you able to answer detailed questions pertaining to any aspect of its implementation?