Project 3 – Search Engine

Due dates: 5/19 and 5/27

In this assignment you will be building a simple search engine based off of the concepts you have learned in the lectures so far. When working on this project, you will be working on achieving two milestones each with its deliverables and deadline. Bear in mind that when planning for milestone #1, you are also cognizant of the requirements of milestone #2. This is to ensure you are on the right track to completing this project successfully. When you will have completed milestone #2, you will be required to demonstrate the functioning of your search engines to the TAs (F2Fs).

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

Milestone	**Deadline**	**Goal**	**Deliverables**	**Score** (out of 50)
#1	5/19	Produce an initial index for the corpus and a basic retrieval component	Short report (no demo)	50% (25)
#2	5/27	Complete Search System with good search results	Code or artifacts + Demonstration	50% (25)

PROJECT: SEARCH ENGINE

Corpus: all ICS web pages

We will provide you with the crawled data as a zip file (webpages.zip). This contains the downloaded content of the ICS web pages that were crawled by us. You are expected to build your search engine index off of 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 – INVERTED INDEX:

Create an inverted index for all the corpus given to you. You can either use a database to store your index (MongoDB, Redis, memcached are some examples) or you can store the index in a file. You are free to choose an approach here.

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 pair.

Sample Index:

Note: This is a simplistic example provided for your understanding. A good inverted index will store more information than this in order to obtain better search results.

Index Structure:	token – docId1, tf-idf1 ; docId2, tf-idf2
Example:	informatics – doc_1, 2.3 ; doc_2, 3.1 ; doc_3, 1.3

You are encouraged to come up with heuristics that make sense and will help in retrieving relevant search results. For e.g. - words in bold and in heading tags (h1, h2, h3) could be treated as more important than the other words. These are useful metadata that could be added to your inverted index data.

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. Such as, for instance, the positions of the words in the page, HTML tag weights, etc. 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 and retrieval algorithms.

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 good GUI. The GUI has to be an intuitive search engine interface and should be able to run on the web browser.

COMPONENT 3 - RANKING:

At the very least, your ranking formula should include TF-IDF (term frequency-inverse document frequency) scoring, but you should feel free to add additional components to your ranking formula to improve the retrieval.

Optional:
Extra credit will be given if your ranking formula includes heuristics more than just TF-IDF. You can apply the ranking schemes that were taught in class.

NOTE: the TA will ask detailed questions regarding your search engine's ranking heuristics. So ensure you thoroughly understand the ranking methods that you'll implement.

Use of libraries:

To extract the content from the HTML tags, you will be using an HTML parser. There are many libraries available to achieve this task and we encourage you to compare the available options before selecting a library to perform HTML parsing for you (Suggestions: Beautifulsoup, HTMLParser)

It is strictly not allowed to use libraries that perform the **entire task of index creation or ranking** for you. Hence, libraries such as **Lucene or Elastic Search or TF-IDF calculating libraries** are not allowed.

You may use libraries that help you achieve specific tasks such as parsing. For example, you can use a tokenizer such as NLTK to tokenize your content, as text processing is not the primary focus of this assignment.

Milestone #1 – Due on May 19th

Goal: Build an index and a basic retrieval component. (Component 1 as described above)

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 details pertaining to your index. It should have the following details regarding your inverted index:
   a. Number of documents of the corpus
   b. Number of [unique] tokens present in the index
   c. The total size (in KB) of your index on disk.
2. URLs retrieved for each of the queries above. You can submit the first 10 results for each of the search queries requested above.
   a. Please do not submit all the URL's that you obtain by running the query.
   b. The quality of the search results does not matter for this milestone. You will work on improving the results in the second milestone.

Evaluation criteria:

- Were the reported details regarding your inverted index plausible?
- Are the reported URLs plausible?
- Was the report submitted on time?

Milestone #2 – Due on May 27th

Goal: Complete Search Engine. Components 2 and 3 as described above.

For this milestone, you will complete your search engine's search and ranking components. You are free to implement more than TF-IDF for your ranking scheme. If you feel there is a better ranking scheme that does not use TF-IDF altogether, you're free to choose that as well, but ensure that you do get better results than TF-IDF.

Deliverables:

- Submit a zip file containing all the artifacts/programs you wrote for your search engine project.

Following this milestone, your project group will meet with a TA to show a live demo of your search engine.

Evaluation criteria:

- Does your program work as expected of search engines?
- How good are the heuristics that you employed to improve the retrieval? Was there an actual improvement in the search from the results obtained in Milestone 1?
- 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?

Extra Credit:

- How good is the web GUI? (e.g. links to the actual pages, snippets, etc.)
- Does the ranking formula incorporate more than just TF-IDF?