Table of Contents

Table of Contents
Objectives
Search Engine Modules
- Web Crawler [20%]
- Indexer [30%]
- Query Processor [10%]
- Phrase Searching[5%]
- Ranker [20%]
- Web Interface [15%]

Implementation and Deliverables
- Deadlines
- Teams
- Implementation

Evaluation and Grading Criteria

Objectives

The aim of this project is to develop a simple Crawler- based search engine that demonstrates the main features of a search engine (web crawling, indexing and ranking) and the interaction between them. Also it is intended to enhance your Java programming skills.

Search Engine Modules

Web Crawler [20%]

The web crawler is a software agent that collects documents from the web. The crawler starts with a list of URL addresses (seed set). It downloads the documents identified by these URLs and extracts hyper-links from them. The extracted URLs are added to the list of URLs to be downloaded. Thus, web crawling is a recursive process.

Care should be taken when implementing web crawlers. At minimum, you have to take care of the following issues:

- The crawler must not visit the same URL more than once.
- The crawler can only crawl documents of specific types (HTML is sufficient for the project).
- The crawler must maintain its state so that it can, if interrupted, be started again to crawl the documents on the list without revisiting documents that have been previously downloaded.
- Some web administrators choose to exclude some pages from the search such as their web pages check for Robot.txt.
- Provide a multithreaded crawler implementation where the user can control the number of threads before starting the crawler.
- Take Care of the choice of your seeds.
- When Crawler finishes one iteration by reaching stopping criteria, it restarts again, Frequency of crawling is an important part of a web crawler. Some sites will be visited more often than others. You have to set some criteria to the sites. In another words, during recrawl, you don't have to repeat all the sites again.
- No of Crawled pages is 5000 page (for the sake of the project).
- The crawler is independent program or process than the Indexer.

Indexer [30%]

The output of web crawling process is a set of downloaded HTML documents. To respond to user queries fast enough, the contents of these documents have to be indexed in a data structure that stores the words contained in each document and their importance (e.g. whether they are in the title, in a header or in plain text). This data structure has to satisfy the following properties:

- Persistence: The index has to be maintained in secondary storage. You can implement your own file structure or use a database.
- Fast Retrieval: The index must be optimized for responding to queries like:
    - The set of documents containing a specific word (or set of words)
    - The set of words contained in a specific document.
- Incremental Update: It must be possible to update an existing index with a set of newly crawled HTML documents.

- When designing the Indexer, consider how you will store your result by looking ahead on Ranker and Searching.

Query Processor [10%]

This module receives search queries, performs necessary preprocessing and searches the index for relevant documents. Retrieve documents containing words that share the same stem with those in the search query. For example, the search query “travel” should match (with lower degree) the words “traveler”, “traveling” … etc.

Phrase Searching[5%]

Search engines will generally search for words as phrases when quotation marks are placed around the phrase.

Ranker [20%]

The ranker module sorts documents based on their popularity and relevance to the search query.

1. Relevance
   Relevance is a relation between the query words and the result page and could be calculated in several ways such as tf-idf of the query word in the result page or simply whether the query word appeared in the title, heading, or body. And then you aggregate the scores from all query words to produce the final page relevance score.

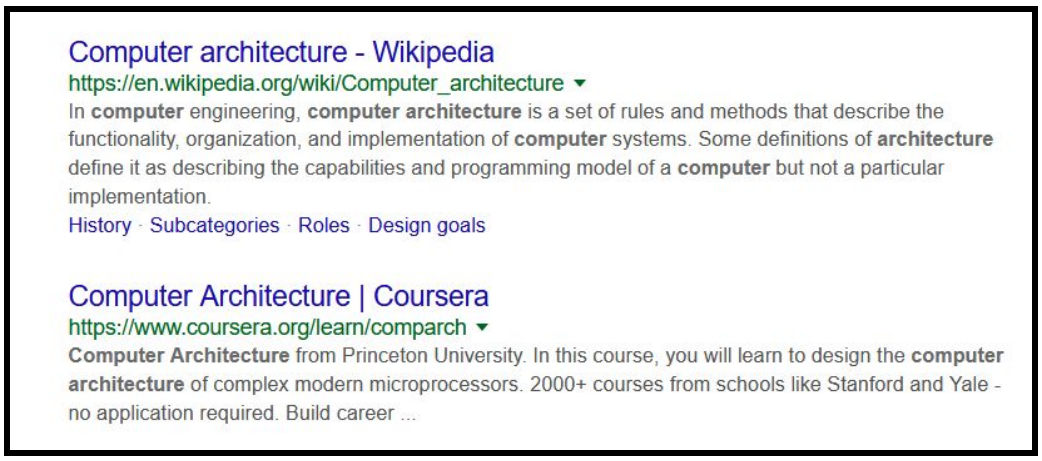
2. Popularity
   Popularity is a measure for the importance of any web page regardless the requested query. You can use pagerank algorithm (as explained in the lecture) or other ranking algorithms to calculate each page popularity .

Grading criteria (20%): 5% for efficiency, 10% for correctness/understanding, 5% for implementation

Web Interface [15%]

You have to implement a web interface for your search engine.

- This interface receives user queries and displays the resulting pages returned by the engine
- The result appears with **snippets** of the text containing queries words. The output should look like google/bing's results page



- Pagination of results (i.e. if you got 200 results, they should appear on 20 pages, each page with 10 results)
- Add suggestion mechanism that stores queries submitted by all users. As the user types a new query, your web application should suggest popular completions to that query using some interactive mechanism such as AJAX.

If your web interface is having any problem showing results, you should make sure that the query retrieval is working correctly by displaying the results in a file/console to make sure that everything is working except for the interface.

Grading criteria(15%): 5% for neatness, 5% for correctness/implementation, 5% for suggestion mechanism

Bonus

There are additional features that you can add to your project for bonus grades

- We are very open to ideas but you should submit a proposal for it before Feb 22nd for approval.
- Efficient Innovation on indexing & ranking.

Implementation and Deliverables

Deadlines

	Phases	Deadline	Discussion
Phase 0	Group formation, Credit, Semester	Thursday, February 22nd, 2018	...

Phase 1	Webcrawler and Indexer	Saturday, March 28th, 2018(week9)	Same week sections
Phase 2	The rest of modules	Saturday, April 27th, 2018 (week13)	Same week Sections

In each phase, A link to your github or gitless source project :

- Your code files
- A readme.txt, explaining how to run your code
- A members.txt containing the names and IDs of each student in the group and whether you're semester or credit
- A PDF file containing any algorithms you've used

Also provide a zipped folder containing the same material, Name the zipped folder Phase<1>_<team number>_<semester>.zip (replace 1 with 2 for the second phase, and replace semester with credit, check your team number in this)

You will upload the zipped folder and write your git links on elearning.

Teams

Work in groups of 3

- If you have any idea different from the project you might submit a proposal with it.

Implementation

- Implementation is done mainly in Java.
- It is your responsibility to select the best technique and tool that enhances the performance of your project.

Evaluation and Grading Criteria

- The project is graded as a whole and the discussion decides what's the grade of each student (there's no piggybacking)
    - 50% of the project grade will be on requirement completeness
    - 40% of the project grade will be on understanding how everything works in your system
    - 10% of the project grade will be on code organization, neatness, using source control and naming convention.
- Any delay in any phase will be penalized by losing 10% of the grade for each late day
- Code must be original, and may not be copied or shared from any other source, except as provided by the class instructor
- Plagiarism will be very strictly punished

~~~~~ Good Luck ~~~~~
~~~~~