CSC322: A Modern Programming Language
Project2: Design and Development of a Search Engine

Deadline
Final Submission — On or before 12noon on Monday, July 15, 2024

Problem definition

Search engines are part of our everyday life and a component of almost all software solutions especially information systems. Searching for information reduces the time spent in navigating through several pages or documents found in an information system. The University has approached the Department of Computer Science to help them out with a search tool that could be integrated on the University's website and intranet. The department is throwing the challenge to its students because it believes in the creativity of its students. The department is willing to reward a unique, efficient and excellent search solution that could be easily integrated into the University's website and intranet. CSC322 has exposed you to the design and implementation of abstract data type. You have been taught to always see a software design project as an opportunity to create user-defined type. Your solution should thus be in the form of an API which could be deployed and used by any client interested in implementing a search feature.

Design specifications

- Input document types: pdf, doc, docx, ppt, ppts, xls, xlsx, txt, html and xml
- Query response time of 0.01s
- Query auto complete
- A maximum of 2hrs delay before a new page added to the website or a new document added to the intranet is indexed.

Purpose

The main aim of this project is to enable you to reinforce your knowledge and gain experience in:

- designing and implementing Abstract Data Type (ADT);
- using common design patterns;
- using a parser generator.

Other objectives of this project include:

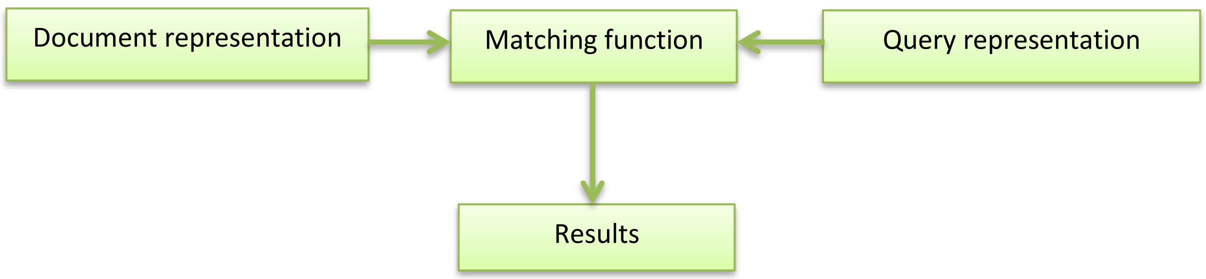
- illustrating the benefits of software engineering techniques;
- strengthening your ability to define, clarify and decompose problems,
- encouraging collaboration and team work.

Design process

Search engines are part of our everyday life. They are subsets of general information retrieval systems. Search engines are made up of crawlers, indexers and ranking algorithms. The crawlers locate web pages and other documents, and pass them to an indexer which automatically generates a representation of those pages in terms of indexes (keywords). When a user submits a query to the search engine, it also generates a representation of the query. The ranking algorithm matches the representation of the query to that of the documents to determine how relevant the documents are to the query. It then ranks the documents from the most relevant to the least relevant.

In this project, we are not going to develop a crawler to look for webpages but we will assume that all our documents are already in a repository. Our concern will be to develop an indexer and to implement a matching/ranking algorithm.

Our work will be based on simple abstraction of an information retrieval system given below:



- Document representation: you need to design an indexer that can extract the keywords present in your documents. It must also compute the frequency of occurrence of each keyword. When indexing a document, there are some words that are considered as stop words. They are words that do not reflect the content of a document e.g. prepositions (to, from, in, on etc.), articles (the, a, an). Such words should not form part of your indexes.
- Query representation: every query submitted to a search engine is passed through a query parser. A query parser generates a representation of your query in a manner understandable to the matching function and should also capture the semantic of the query.
- Matching function: this is one the most important aspect of the design. Matching function takes the query representation as input and maps it to the document representation. It retrieves all the documents it judges relevant to the query and ranked them based on their degree of relevance. The response time of the matching function depends largely on the data structure used to store the indexes of your documents.
- Results: the documents retrieved in response to a query are considered as the search results. Even though the major requirement in this project is to come up with a search API, you will however need to demonstrate that your API is efficient by creating a graphical user interface that allows the user to interact with the API. The interface should contain at least a search page with a text field and a button. Search results should be displayed in a new page and the interface should afford the users the opportunity to view the content of the retrieved documents.

Design approach

The general architecture of an information retrieval system has been given above. For each of the modules, you are expected to do a proper design using class diagrams to show the components of the module and the relationship between them. This is very important and it should be done before you start coding. You are free to discuss your design with your course lecturer before you start the implementation. Remember to always state clearly the specification for your classes and methods. Test first programming style should be adopted in the implementation of your design.

Background reading

You might need to read the following documents to familiarize yourself with automatic documenting indexing, and page ranking techniques:

- Inverted index
  - http://en.wikipedia.org/wiki/Inverted_index
  - http://nlp.stanford.edu/IR-book/html/htmledition/a-first-take-at-building-an-inverted-index-1.html
- Ranking
  - http://sentiment.christopherpotts.net/vectors.html
  - http://orion.lcg.ufrj.br/Dr.Dobbs/books/book5/chap14.htm