Mini Search Engine Project

CS 2420, Spring 2014

100 points total [30% of your final grade]

Due Date for Presentation Slides: April 21, 2014 by 11:59pm
Due Date for Project Report and Program: May 2, 2014 by 11:59pm
[No late submissions accepted!]

Delivery: Submit via canvas

Introduction

In this project, you will be designing and implementing a mini search engine. You are probably familiar with Google, Yahoo or Bing, which are some of the most popular search engines that you can find on the Web. The task performed by a search engine is, as the name says, to search through a collection of documents. Given a set of texts and a query, the search engine will locate all documents that contain the keywords in the query. The problem may be therefore reduced to a search problem, which can be efficiently solved with the data structures we have studied in this class.

Your Task

Your task is to design and implement an algorithm that searches a collection of documents. You will be provided with a set of 50 documents and a set of sample queries. You have the freedom to select the data structures and algorithms that you consider to be more efficient for this task. Of course, you will have to justify your decisions.

First, you will process the documents and store their content (i.e. words / tokens) in the data structures that you selected (in information retrieval, this phase is called ***indexing***). Next, for every input query, you will process the query and search its keywords in the documents, using the previously implemented data structures and an algorithm of your choice. (this phase is called ***retrieval***). For each such query, you will have to display the documents that satisfy the query.

The queries may contain simple Boolean operators, that is AND and OR, which act in a similar manner with the well known analogous logical operators. For instance, a query: "*Keyword1 AND Keyword2*" should retrieve all documents that contain both these keywords (elements). "*Keyword 1 OR Keyword 2*" instead will retrieve documents that contain either one of the two keywords.

Example

Consider the following sample documents.
Doc1: I like the class on data structures and algorithms.
Doc2: I hate the class on data structures and algorithms.
Doc3: Interesting statistical data may result from this survey.

Here are the answers to some queries:
Query 1: data
Doc1, Doc2, Doc3
Query 2: data AND structures
Doc1, Doc2

Query 3: like OR survey
Doc1, Doc3

Hints
Take a look first at the format of the documents. (They will be available from the class webpage and Canvas). You will have to parse the input. You may ignore all lines starting with "<", these are all SGML tags that are useful for certain tasks, but you will probably not find them very useful in this project. The punctuation is already separated from the words, so you do not have to worry about that. You will have to read one word at a time and add it to your data structure.

As data structures, you may consider using dictionaries / hashtables, trees – such as binary search trees and AVL trees. Any other data structures are admissible as well – although, again, you have to be able to justify your selection. For every word, you should store a list of documents where it occurs, in order to allow for efficient searches and Boolean operators later on.

What to turn in
There are three main parts in this project, all of them contributing to the final project grade.
1. You will have to write a project report (about 5-8 typed pages – single space – 12pt font) that includes:

- design issues, what are the problems and how you solve them
- data structures you use, the decision behind selecting them
- algorithms you employ, again with a justification of your decision
- particular emphasis should be placed on the running time of your algorithm
- optimization issues: what could you do to further optimize your algorithm
- you need to specifically address the problem of scalability: would you implementation be efficient in the case of very large text collections?
- the report will also include results you obtained for the given sample queries, and for at least five additional queries of your choice
- any other remarks about your design and implementation

The due date for your final written report is 05/02/2014.

2. You will have to send in a fully working program, written in C++, that you can provide with a query, and obtain the list of documents. It is mandatory that you include a README file, as detailed as possible, including compilation and running instructions. It is also mandatory that your programs are fully documented, that is they should include detailed comments on what is included in each file and what each method does. Submit your programs via Canvas. Make sure you submit your programs by 05/02/2014.

3. In class presentation, you will have to prepare a short presentation, to last about 5-7 minutes, where you will present the design and implementation decisions you made in this project. Make use of examples, compare with other possible approaches, and use any other means you wish to make your point (that your design is the best). You may show us the demo! You should prepare PowerPoint slides. Project presentations will take place in class on 04/22/2014 and 04/24/2014.

Grading

• Design issues, data structures efficiency, algorithms, other issues addressed in the written report	- 30 points
• Program (should compile and run correctly, have an associated README file, should be fully documented)	- 20 points
• In class presentation (presentation materials, speaking ability, presentation polish, audience contact, presentation time, questions/answers, etc)	- 50 points
• Extra	- 15 points

Queries

1. flow
2. flow AND stream
3. flow OR stream
4. supersonic AND speeds
5. the AND boundary AND layer
6. boundary-layer
7. velocity OR speed
8. (optional) reynolds AND number OR reynolds AND numbers

Extra Points

Up to 15 points extra credit may be given if you handle some of these issues:

- Stop words elimination. Given a list of very frequent words, like the, a, of, etc., you may consider avoiding the storage of these elements.
- Incomplete matches. That is, allow for a search for number*, which will match all words starting with number, i.e. number, numbers, numbering, etc.
- Fancy interface (Web based, Mobile based, other) - feel free to use any programming language for the fancy interface!
- Document scoring with respect to the query (i.e. keep track of the number of occurrences of words in documents, and calculate a score based on that). If you plan to implement this part, please see me for additional details.