CS5406 Performance Engineering of Computer Systems

Lab 3 – Search Engine

Learning Outcomes

In this lab we will develop a simple search engine that supports exact match queries. At the end of the lab you will be able to:

- extract keywords from a given set of web pages
- design a suitable file format to store the inverted index
- respond to a set of queries with 1 or more search terms
- measure and optimize the performance of the search engine

Search Engine

A simple search engine runs through a set of steps which can be broadly categorized into 2 parts as index generation and searching.

Index Generation:

1. Given a set of web pages, the index generator extracts the keywords in each web page.
    - While extracting keywords stop words are ignored. Stop words are words which are filtered out before or after processing of natural language text. While there is no universally accepted list of stop words, we will consider the following words as stop words for our search engine:
        - the, is, at, which, and, i.e., in, a, that, to, or, of
    - We can also ignore symbols like &, ., >, <, ?, #, @, ~, !, “, $, %, *
2. After reading all the web pages, an inverted index is then generated for each extracted keyword (from at least one web page).
    - An inverted index typically has the following format:

      ```
      {list of <keyword, {list of URLs of web pages with that keyword}>}
      ```

    - For example:

```
{(Performance,  {www.url1.com/index.html,url21.org/news.html,
                www.url13.com/homework/hw1.html}),
(Computer,      {www.url123.com/computers.html, www.url13.com/homework/hw1.html}),
(Engineering,   {www.url123.com/computers.html, ur321.net/news.html,
                www.ieee.org/eng.html, www.url13.com/homework/hw1.html})}
```

- The index generator then saves this inverted index as a file with a suitable file structure. This file is to be later read by the search engine.

Searching:

3. When the search engine loads, it first loads the inverted index from the file.
    - The inverted index may be loaded to the memory, if it is not very large. We will assume this to be the case for our search engine.
4. The search engine then waits for a search query from a user. Once a search query is received, it searches the inverted index for matching web pages.
    - For example, if the search query contains the term `Computer` it returns the following list of URLs:

```
www.url123.com/computers.html
www.url13.com/homework/hw1.html
```

- If search query contains the 2 terms `Computer Engineering` it returns URLs in the following order:

  ```
  www.url123.com/computers.html
  www.url13.com/homework/hw1.html
  ur321.net/news.html
  www.ieee.org/eng.html
  ```

  - Note that first 2 URLs have both the search terms while last 2 have only the term `Engineering`
  - Same set of URLs will appear even if the search query is "`Engineering Computer`". However, order may be different.
- For our simple search engine, it is ok to consider only the exactly matched keywords, e.g., the word `Engineering` does not need to match to `Engineer`.

5. The list of ranked (based on no of key words in URL) URLs are then send to the user.
6. The search engine then waits for another search query from users and the process continuous from Step 4.

Tasks

Step 1: Download Lab3.zip files from LMS and extract it.

It has a set of file named as `urlx.txt`, where `x` indicates the file no. First line of each file contains the URL of the web page where the text was extracted. Then the following lines contain the extracted text. To simplify your task all the HTML tags are removed.

A list of queries to be issued to the search engine is given in `queries.txt`. Each line is a separate query and has 1 or more keywords. It is ok for some queries not return an answer, if no web page contains that keyword.

Step 2: Design how you want to store your inverted index in a file. Think about the storage efficiency and read efficiency. [3 marks]

Step 3: Develop the index generator (you may use a common programming language like C++, Java, or Python). [4 marks]

The index generator should be able to scan through all the `urlx.txt` files and save the inverted index file.

Step 4: Design how you want to store your inverted index once it is loaded into the memory of the search engine. [3 marks]

Step 5: Develop the search engine. [8 marks]

The search engine should be able to read the inverted index from the file (generated by the index generator, which is a separate program) and load it into memory.

It should then take each query from the `query.txt` and attempt to resolve one query at a time. Response to each query need be ranked based on the number of matching keywords.

Once the query is resolved answer should be appended to another file named `answers.txt`. `answer.txt` should have the following format:

```
Search term(s)
URL1
URL2
...
```

```
URLn
<blank line>
<blank line>
Search term(s)
URL1
URL2
...
URLn
<blank line>
<blank line>
...
no_searched
total_search_latency
average__search_latency
```

total_search_latency is the time gap between issuing the first search and completion of last search. The search engine can terminate once all the queries are resolved.

Add necessary code to find the time between starting the search engine (including loading of inverted index) and resolving all the queries.

Step 6: Run the search engine a couple of times until your performance results are within an accuracy of ±5% and 95% confidence level. [2 marks]

Step 7: Optimize your code to reduce the total execution time of the search engine. Remaining marks will be given depending on how fast your solution is compared to the rest of the class. [5 marks]

What to Submit

Submit the following:

- Designs of the inverted index file format and the memory layout of the search engine. Justify the efficiency and performance of your design. Also describe other optimizations you have used in the search engine. Submit as a PDF file.
- Source code of index generator
- Source code of search engine
- Performance results and `answer.txt`
- Specification of the machine you used for the performance analysis

Note that the instructor should be able to execute your code and check its performance in a Linux environment.