Romil V. Shah

20008692

CS / CPE 600

Prof. Reza Peyrovian

Search Engine Project

Submission Date: 12 / 15 / 2022

ReadMe File

Project: Search Engine

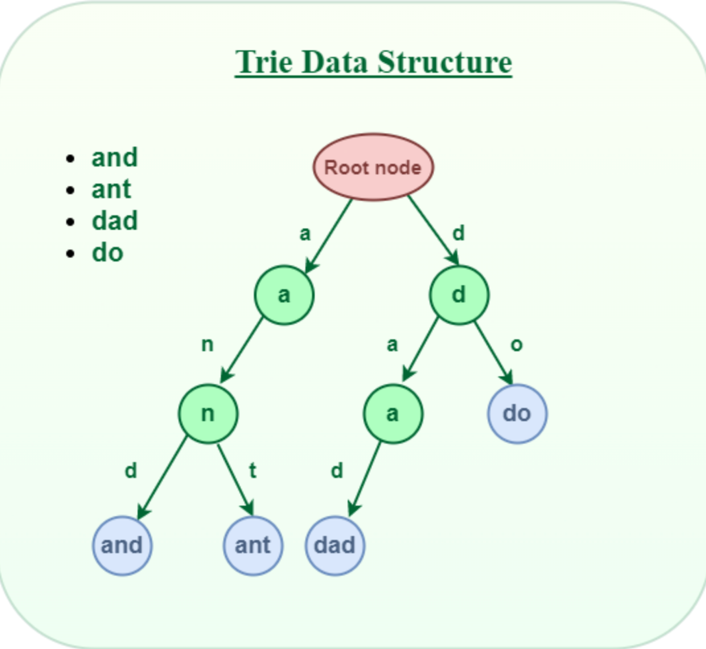
Implement the simplified Search Engine described in Section 23.6 (Subsection; Search Engine) for the pages of a small Web site. Use all the words in the pages of the site as index terms, excluding stop words such as articles, prepositions, and pronouns.

Submit the following files:

1. A read me file that contains details of your approach to the problem, including description of Algorithms and Data Structure used.
2. Your coded, well-commented code file in your favorite language, such as Python, Java, C++, etc.
3. The input file that contains the few pages you have used as input, including some links to your other pages.
4. Output file that has samples of your run. Make sure you have tested the boundary conditions.
5. Provide a short video demonstrating the execution of your project. (Include the testing of boundary conditions as well).

My project has been coded in Java. I have used the ANI News Website for input using jsoup. I have taken 13 different defense news links as input. The paragraphs from the webpages are read and the words are extracted and stored in a trie data structure.

The search for a word in a Trie Data Structure is implemented as shown in the figure below:



We use Hashmap to keep count of occurrences of each word from each webpage. The hashmap displays each word count respectively and if the word is not present in one of the documents the count is shown as zero.

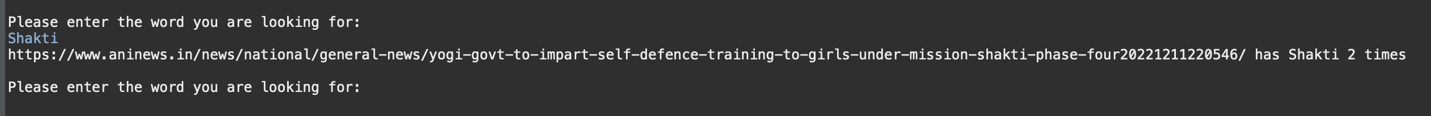
For removing the stop words such as articles, prepositions and pronouns, we read each word from the webpage’s paragraph, pass that word to the ‘stopword’ function which calls upon a file containing list of all stop words (stopwords.txt). If a match is found that word is not inserted into the trie data structure and the hashmap.

To display total occurrences of all words from all documents, a separate hashmap has been created which displays the word and its total occurrences.

For ranking, we have ranked webpages based on number of occurrences of the word on a page.   
We also have ranked webpages and sorted them using merge sort so, it can be efficient and fast.

Here we have some examples of just one keyword entry. For this, use Main\_Single.java

1. The word “Shakti” is only available in one of the links.

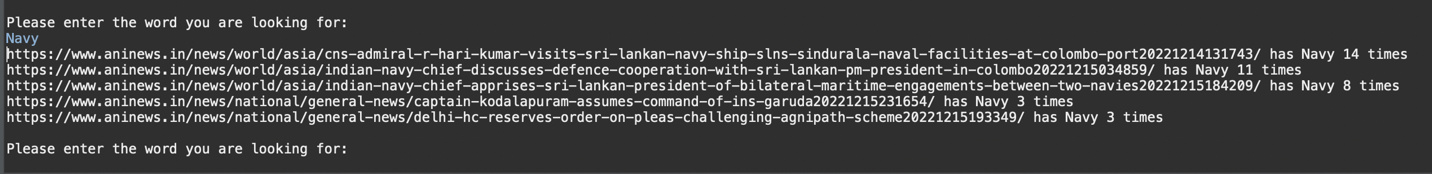


1. The word “Indian”

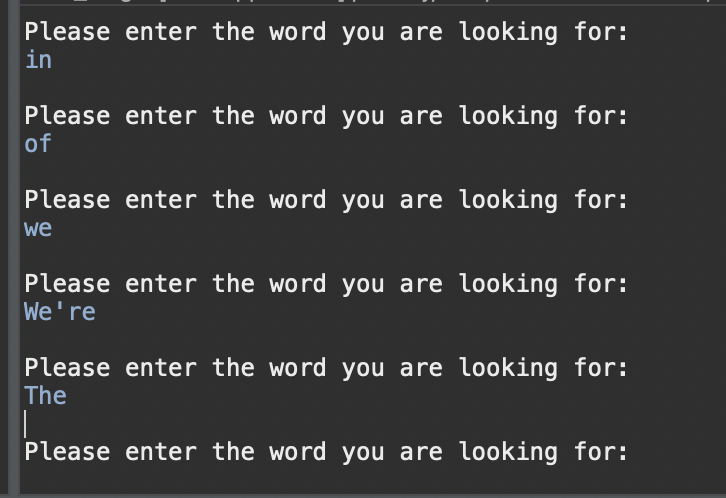
Text

Description automatically generated

1. The Word “Navy”



1. Stop Words being eradicated in output results (Stored in stopwords.txt):

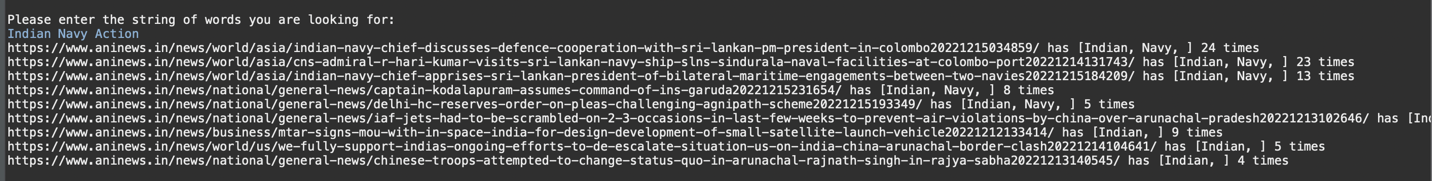


FOR MULTIPLE KEYWORDS

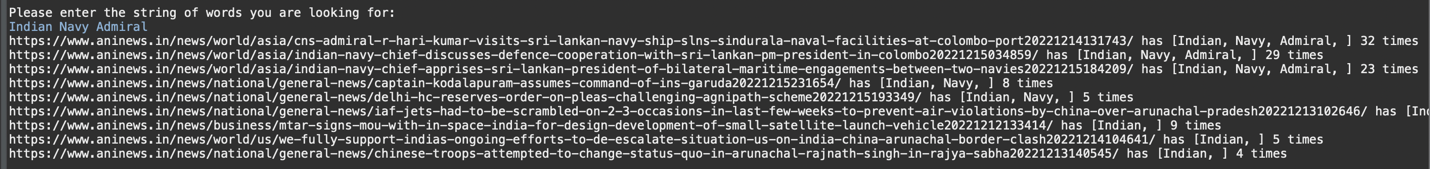
For more than one keywords, we search the pages containing the word by trie and sort them first based on number of keywords out of search query and then in buckets by number of their individual occurrences.

For this use, Main\_Multi.java

1. Words: ‘Indian’, ‘Navy’, ‘Action’



1. Words: ‘Indian’, ‘Navy’, ‘Admiral’



1. Words: ‘Admiral’, ‘in’, ‘Indian’, ‘Navy’

Note: Here, the result is produced same as the 2nd query as in is a stop word and is ignored by the program.

