Web Search Engine (Group 31)

Li Zhou
Master of Applied Computing
University of Windsor
Windsor, ON, Canada
zhou18j@uwindsor.ca

Yehan Li
Master of Applied Computing
University of Windsor
Windsor, ON, Canada
li1as@uwindsor.ca

Yidi Deng
Master of Applied Computing
University of Windsor
Windsor, ON, Canada
deng12h@uwindsor.ca

Agreement: "I confirm that I will keep the content of this assignment confidential. I confirm that I have not received any unauthorized assistance in preparing for or writing this assignment. I acknowledge that a mark of 0 may be assigned for copied work."

***Abstract*—CC Engine is a web search engine, based on algorithms and JAVA. And our topic is about Coronavirus. We crawled 12000+web pages on Coronavirus news. It includes four modules: crawling URL, parsing filtering web pages and saving them to local computer, the third part is creating indexes on keywords, and the last part is searching.**

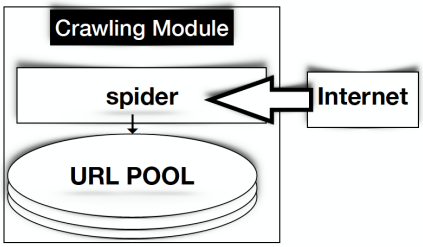
***Index Terms*—Crawling, Web Page processing, Tokenizers, Searching**

I. Crawling

A. Explanation:

- *In this section, we do crawling URLs. At first we parse the URLs using regular expression. And use Regex.compiler the compile the pattern, and then use matcher to find the URLS we need. Once we find the ideal URLs we store them in the Hash-Set, also can been seemed as URL pool, which ensures the uniqueness of the URL. And we also set the limitation to 2000 on the size of the URL pool to parse and save pages, and to 10000 to exit crawling. The regular expression is (), which means we will find the URLs with the topic on Corona-virus.*

B. Crawling Figure:



II. Web Page Processing

A. Back-end Mechanism:

In this module, we parse the URLs in the URL pool and save the content into a local folder named Corona-virus. At first, we read the URL index file on local computer into a list, including the version, published time, and MD5.

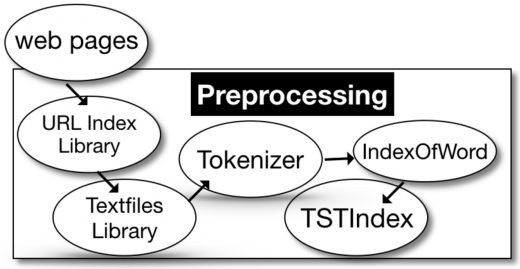
Second, we use Document.hastext() to filter out URLs in non-plain file format, like images, videos. Then we use MD5 algorithm to encode the content of every URL to filter out the URLs with overlap content.

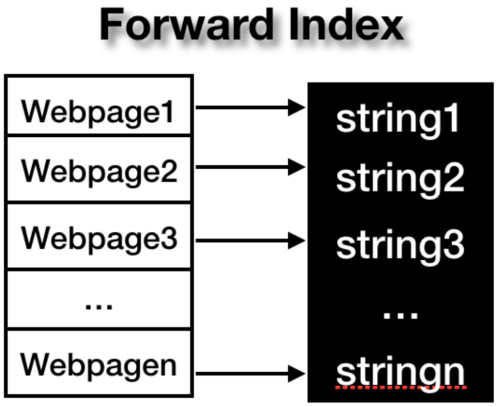
If MD5 are different while the URLs are same, then we will check the published time, if the URLs in the URL pool has a newer published time, then it will be retained and update its original version(the initial version is version1.0).

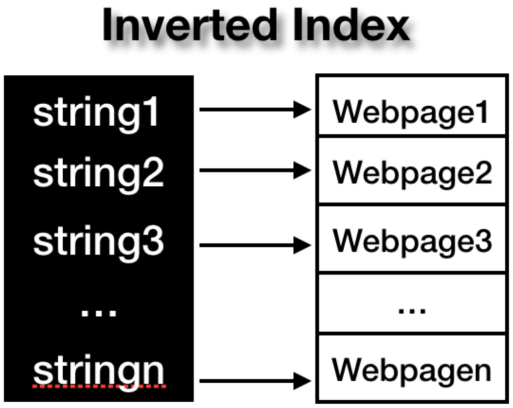
After all of these preparations are done, we begin to create index on these URL, the elements will be stored in the URL index file are version, published time, MD5, filename, file path and URL.

Then, we use MD5 as the filename to ensure the uniqueness of the filename, and also we add the URL as a header followed by the content of the URL

B. Flow Chart:







III. Tokenizer:

A. In this module, we use TST to segment the strings and TF-IDF algorithm to create index on every keyword. At first, we read Filter.csv into a binary search tree to filter the meaningless strings and then we store all the keywords in a TST tree and count the number of total keywords. For every keyword, we count its total number in all of these 10000 web pages, and the total number this keyword appearing in a file. Then we can get its weight by dividing its total number in all of these 10000 web pages by the number of all the keywords.

B. Methods Used:

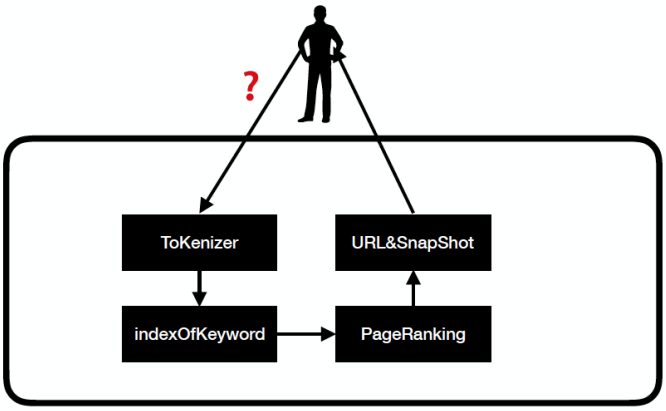
1) *keywords matching*
2) *TST algorithm:*
3) *data dictionary*

IV. Searching/Use Cases

A. Explanation:

In this module, we read the inputs from the end, and analyze the keywords and then we read the word frequency and weight of the keywords from the index file and multiple these two numbers and then add them together as a basis for web-page ranking.

B. Flow Chart:



V. Github Repository

https://github.com/Lily-Wenxi/CC-Engine.git

VI. Conclusion:

To sum things up, the mechanism behind our project mainly focuses on the 4 sections that we have explained. The crawler module, html processing module, tokenizer and a searching module. They fit closely together to make our search engine function properly.

Acknowledgment

At the every end, we'd like to take this chance to thank all tutors, professors and team members. We won't be able to achieve all of these things without your help ! Especially in this special period of time when the coronavirus is spreading all overall the world.

References

1) *DataStructure and Algorithms in JAVA. Michael T.Goodrich/Roberto Tanassia 4Edition*
2) *Code Sources: Comp 8117 Lab and Lecture Resources Java documentation: http://docs.oracle.com/javase/8/docs/*
3) *Java tutorials: http://docs.oracle.com/javase/tutorial/*
4) *The Eclipse Foundation: http://www.eclipse.org/*
5) *Java by Oracle: http://www.oracle.com/technetwork/java/index.html*
6) *Java Standard Edition (Java SE 8): http://www.oracle.com/technetwork/java/javase/overview/index.html*
7) *Java Enterprise Edition (Java EE 8): http://www.oracle.com/technetwork/java/javaee/overview/index.html*
8) *Dual-pivot Quicksort: https://web.archive.org/web/20151002230717/ http://iaroslavski.narod.ru/quicksort/DualPivotQuicksort.pdf*
9) *Internet movie database: http://www.imdb.com/*
10) *Algorithms by Sedgewick, examples and source code: http://algs4.cs.princeton.edu/home/, http://algs4.cs.princeton.edu/code/*
11) *NCBI'S Sequence read archive (SRA): http://www.ncbi.nlm.nih.gov/sra*
12) *Java Regex documentation: http://docs.oracle.com/javase/8/docs/api/ java/util/regex/package-summary.html*
13) *Editor Kit of Java: http://docs.oracle.com/javase/8/docs/api/ javax/swing/text/EditorKit.html*
14) *Tools for parsers and generators: http://catalog.compilertools.net/lexparse.html*
15) *JFLAP: http://www.jflap.org/*