RESEARCH ARTICLE OPEN ACCESS

Design and Implementation of Search Engine using Quick Sort Algorithm

Hemant A. Lad [1], Shubham V. Kamble [2], Piyush A. Wanjari [3], Priyanka Tekadpande [4]
Students [1], [2] & [3], Guide, Assistant Professor [4]
Department of Information Technology
RTMNU KITS Collage and Ramtek
India

ABSTRACT
We present a web search engine for indexing and searching html documents. A search engine is the software program that searches for sites based on the words that you designate as search terms. Search Engines appear through their own databases of information in order to discover what it is that you are looking for, we expect it will be useful for learning information retrieval techniques, particularly web search engine technology. We are using quick sort algorithm for both crawling and indexing the html web pages.
Keywords :—Word Wide Web, Information Retrieval, Search Functions Quick Sort Algorithm,

I. INTRODUCTION

Many papers written in the web information retrieval (IR) field develop their individual web crawlers to crawl, index, and explore contents (including hyperlink texts) of the pages and network structure of the web. Occasionally a search function to return related pages to the user's queries is also provided. Crawler and search function are considered to be the essential components of a search engine, and each has its own research challenges and problems [1].

Web crawler, also known as spider or robot, is in charge for fetching pages, parsing hyperlinks, managing crawl queue, and indexing contents of the pages. Some of the actual working crawlers like IRLbot, Mercator, Polybot, iRobot, UbiCrawler, and Googlebot offer good documentations on their designs and implementations [1,2].

Search function is an interface between a search engine and users. This function receives user's queries and returns related pages to the queries. The pages usually are sorted according to some criteria. The most basic criterion is Boolean match between contents of the pages and words in the queries. More advanced mechanisms use hyperlink structure of the web graph (e.g., PageRank, Quick Sort, and Partition), anchor text information, and user-click behaviour to determine ranking of the relevant pages [4].

II. SYSTEM OVERVIEW

The system is designed and implemented by relating the original architecture. In the top-level view, the system consists of crawler, searcher, and database components. Figure 1 shows overall design of the system, figure 1 shows more details of the system design, and figure 2 show structures of the crawler and searcher classes respectively. The following subcategories term the modifications to the original system.

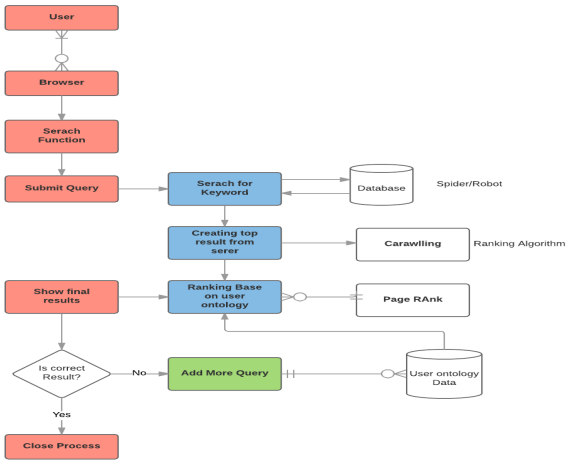

Fig 1. System overview in Top Level View

A. Web Crawler

Web crawler is a comparatively simple automated program, a script, that logically scans or crawls through internet pages to create an index of that data its looking for. Alternative names for a web crawler includes web spider, web robot, bot, crawler, and automatic indexer. Search engine use web crawler to collect information about what is available on public web page [5].

There main purpose is to collect data so that when internet surfers enter a search term on their search box, they can quickly deliver the surfer with relative websites. When a search engine web crawler visit a webpage, it reads the visible text, the hyperlink, and the content of the various text used in the sites, such a keyword rich metatag. Using the information gathered from the crawler, a search engine will then determine what the site is about an index the information. A vast number of webpages are continuously being added every day, and information is constantly changing. A web crawler is a way for the search engine and other user to regularly ensure that their databases are up to date [5,6].

B. Basic Web Crawler Structure

Given this picture, we would like to design a flexible system that can be accepted to different application and strategies with a reasonable amount of work. Note that there are significant differences between the scenarios. For example, a broad breadth-first crawler has to keep track of which pages has been crawl or ready; this is commonly done using a "URL seen" data structure that may have to decide on disk for large crawls [7].

C. Working of Crawler

We partition the system into two major component crawling system and crawling applications. The crawling system itself several specialized components, in particular a crawl manager, one or more downloaders, and one or more DNS resolver. All of these components, plus the crawling application, can run on different machine and can be replicated to increase the system performance. The crawler manager is responsible for receiving the URL input stream from the application and forwarding it to the available downloaders and DNS resolvers while enforcing rules about exclusion and crawl speed. A downloader is a high performance asynchronous HTTP client capable of downloading hundreds of web page ion parallel, while a DNS resolver is an optimize stub DNS resolver that forwards queries to local DNS server. The crawler manager is the central components of the system, and the first component is started up [7,8]

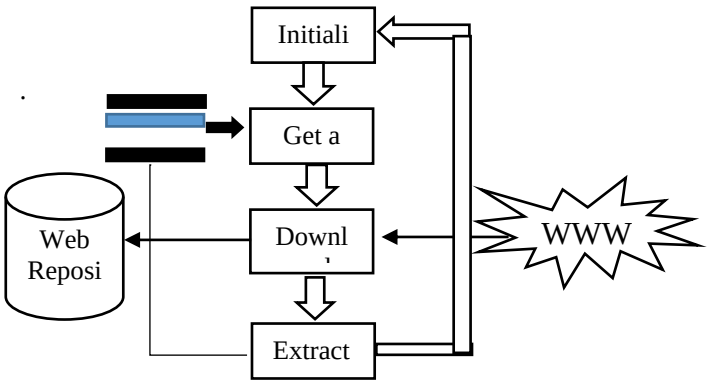

Fig 2. Working of Crawler

D. Page Rank

Page Rank is a method of measuring a page importance. When all the other factors such as title tag and keyword are taken into account, google uses page rank to adjust results so that sites that are more important will move up in the result page of a user's search accordingly [8].

This is the order of ranking in Google:

- Find all pages matching the keywords of the search.
- Rank accordingly using "on the page factor" such as keyword.
- Calculate in the inbound anchor text.
- Adjust the results by page rank scores.

E. Quick Sort Algorithm for Searching

Quick Sort is a Divide and Conquer algorithm, Quick sort is a highly efficient sorting algorithm and is based on partitioning of array of data into smaller arrays. A large array is partitioned into two arrays one of which holds values smaller than the specified value, say pivot, based on which the partition is made and another array holds values greater than the pivot value. Quick sort partitions an array and then calls itself recursively twice to sort the two resulting subarrays [10].

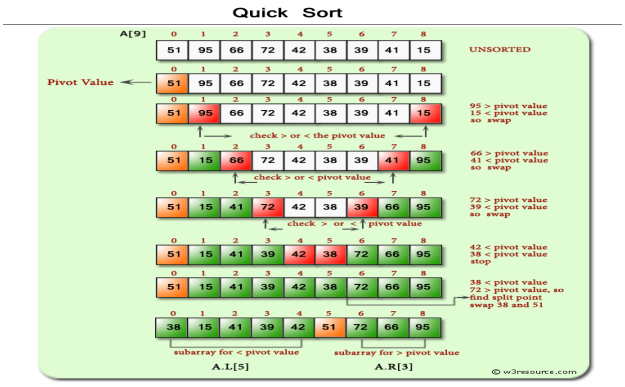

Fig.3 Quick Sort Algorithm for Searching

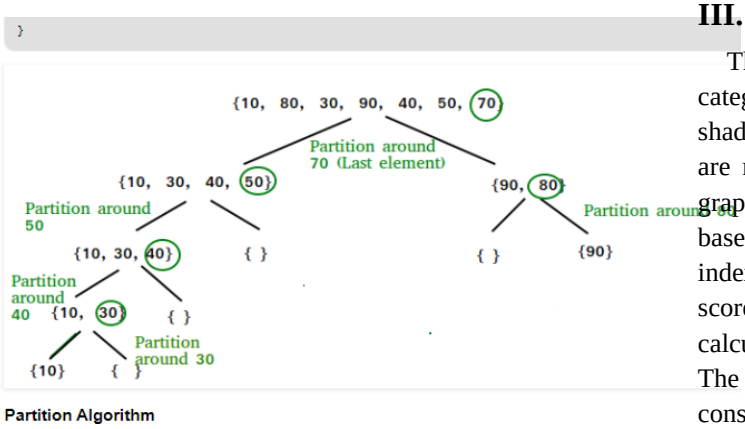

Fig.4 Partition Algorithm

F. Our search engine is responsive designed

Responsive web design (RWD) is an approach to web design which makes web pages render well on a variety of devices and window or screen sizes. Recent work also considers the viewer proximity as part of the viewing context as an extension for RWD[1]. Content, design and performance are necessary across all devices to ensure usability and satisfaction.

G. Onclick function

The onclick event handler captures a click event from the users' mouse button on the element to which the onclick attribute is applied. This action usually results in a call to a script method such as a JavaScript function, like this:

The **onclick event handler** is defined in an <input> tag (i.e. <input type="button" name="bt1" onclick = "JavascriptCode">). It executes some JavaScript code or function when the pointing device button (for example mouse) is clicked on **button object** [10,11,12].

H. like operation

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column [13]. There are two wildcards used in conjunction with the LIKE operator:

- % - The percent sign represents zero, one, or multiple characters
- _ - The underscore represents a single character

Note: MS Access uses a question mark (?) instead of the underscore (_). The percent sign and the underscore can also be used in combinations!

III. DATABASE DESIGN

There are 12 tables in the database which divided into three categories, content, web graph, and score indexes tables. The shaded tables are the added tables. The content indexes tables are mainly used in the content-based scores functions, web graph indexes tables are mainly used in the link structure-based and anchor text-based scores functions, and score indexes tables are the tables that store query-independent scores. Query-dependent scores, as noted earlier, cannot be calculated in advanced, so they cannot be stored in tables.

The url list table stores the crawled web addresses in consecutive numbers based on the time they were crawled. These numbers, row ids, become identity numbers for the corresponding web addresses. All other tables that have to utilize web addresses use the row ids instead of directly using URL names. The arrows from url list to entries in others tables indicate that those entries are pointers to urls by using row ids of url list table [18].

The same strategy is also applied in wordlist table, where its row ids become identity numbers for corresponding words. And then word location table utilizes both url list and wordlist to make a list of locations of all words in every page. The location itself is simply the order of corresponding word appearance on the pages [14].

The link table stores network structure of the crawled web pages. It is the table that the link structure ranking algorithms use to calculate pages scores. The link word table stores anchor texts of the corresponding links which are used in the anchor text analysis and scoring.

All tables in score indexes simply store the scores of each page in the collection. When users input queries, search function will call appropriate scores functions that utilize these tables. The third entries in auth_myhits, hub_myhits, and mypagerank are constants defined in our proposed

algorithms calculated based on number of inlinks and outlinks of each page.

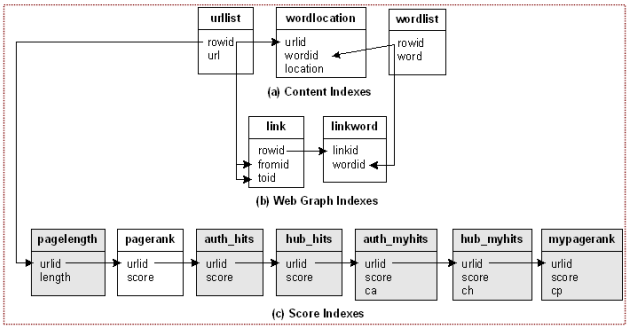

Figure 5. Database Structure

IV. CONCLUSIONS

We have presented web search engine software that is suitable for research and learning purposes because of its simplicity, portability, and modifiability. The strength of our program is in the search function component since we provided many scores functions to sort relevant pages to user queries; especially, the inclusion of anchor text examination makes our program can also find related pages that do not contain terms in the queries.

In the crawler component, only small alteration was made. However, this small alteration can improve the crawling reliability significantly. Readers who have read the documents would notice that the crawling method is breadth first search without politeness policies (e.g., obeying robot.txt and controlling access to the servers), spam pages detection, priority URLs queue, and memory management to divide the load of crawling process between disk and RAM. Without good memory organization, all of URLs seen tasks are conducted by searching in url list table in the disk which is very time consuming. We will address the crawler design problem in the future researches.

Because there is no single finest way to choose the combination between scores functions and their weights, we support users to experiment with their own databases. But maybe some guidelines here can be considered. For example if database contains a set of pages that linked by very descriptive anchor texts, like documents from online encyclopedia, then linktextscore() and anchorscore() can be given comparatively strong weights. And because authors can easily create pages with many occurrences of keywords, frequencyscore() and bm25score() will give misleading results in this situation, so their weights must be set less important compare to more reliable content-based metric, locationscore().

ACKNOWLEDGMENT

We express our special thanks to Mrs. Saroj A. Shambharkar, Head of Information Technology department, for her kind support and allowing us to use all the facilities that are available in the department during this project. Our sincere thanks to Dr. B Ram Ratan Lal, Principal, Kavikulguru Institute of Technology and Science Ramtek for extending all the possible help and allowing us to use all the resources that is available in the institute.

REFERENCES

[1] Andri Mirzal “Design and Implementation of a Simple Web Search Engine”, International Journal of Multimedia and Ubiquitous Engineering Vol. 7, No. 1, January, 2012

[2] S. Raghavan and H. Garcia-Molina,” Crawling the hidden web”, in Proc. Of 27th International Conference on very large Data Bases, sept 2001.

[3] A. Heydon and M. Najork, ―Mercator: A Scalable, Extensible Web Crawler, ‖ World Wide Web, Vol. 2, No. 4, (1999) pp. 219—229.

[4] V. Shkapenyuk and T. Suel, ―Design and Implementation of a High-Performance Distributed Web Crawler,‖ in Proc. IEEE ICDE (2002) pp. 357—368

[5] Z. Bar-Yossef, I. Keidar, and U. Schonfeld, ―Do Not Crawl in the DUST: different URLs with similar text, ‖ in Proc. 16th International WWW Conference (2007) pp. 111—120.

[6] R. Cai, J.M. Yang, W. Lai, Y. Wang, and L. Zhang, ―IRobot: An Intelligent Crawler for Web Forums, ‖ in Proc. 17th International WWW Conference (2008) pp. 447—456.

[7] P. Boldi, M. Santini, and S. Vigna, ―UbiCrawler: A Scalable Fully Distributed Web Crawler,‖ Software, Practices & Experience, Vol. 34, No. 8 (2004) pp. 711—726

[8] R. Lempel, S. Moran The stochastic approach for link-structure analysis (SALSA) and the TKC effect Computer Networks 33 (2000) 387–40

[9] Marcotte, Ethan (May 25, 2010). "Responsive Web design". A List Apart.

[10] "Ethan Marcotte's 20 favourite responsive sites". .net magazine. October 11, 2011.

[11] Gillenwater, Zoe Mickley (December 15, 2010). "Examples of flexible layouts with CSS3 media queries". Stunning CSS3. p. 320. ISBN 978-0-321-722133.

[12] Schade, Amy (2014-05-04). "Responsive Web Design (RWD) and User Experience". Nielsen Norman Group. Retrieved 2017-10-19.

[13] Thijs Westerveld, Wessel Kraaij, and Djoerd Hiemstra1 Retrieving Web Pages using Content, Links, URLs and Anchors

[14] Kiduk Yang Combining text- and link-based retrieval methods for Web IR School of Information and Library Science University of North Carolina

[15] Google's PageRank and Beyond: The Science of Search Engine Rankings