International Journal of Computer Applications Technology and Research
Volume 5– Issue 8, 551 - 555, 2016, ISSN:- 2319–8656

An Optimized Search Engine for Academics

Abbas Fadhil Mohammed Ali AL-Juboori
Department of Computer Science
University of Kerbala
Kerbala, Iraq

Abstract: Search engines are among the most useful and high-profile resources on the Internet. The problem of finding information on the Internet has been replaced with the problem of knowing where search engines are, what they are designed to retrieve, and how to use them. The main function of An Optimized Academic Search Engine is to allow its users to search for academic files. It also allows the users to specify query for searching phrases. The ranking and optimization was achieved for the result by the most website visit. The system have been designed by using PHP, MYSQL, and WAMP server.

Keywords: Search engine; Optimization; Academic; Information retrieval; Ranking

1. INTRODUCTION

Search Engine technology was born almost at the same time as the World Wide

Web [1], and has certainly improved dramatically over the past decade and become an integral part of everybody's Web browsing experience, especially after the phenomenal success of Google.

At the first glance, it appears that Search Engines have been studied very well, and many articles and theories including the paper by the founders of Google [2] have been published to describe and analyze their internal mechanisms.

1.2 The Basic Components of a Search Engine

All search engines includes:

1. A Web crawler.

2. A parser.

3. A ranking system.

4. A repository system.

5. A front-end interface.

These components are discussed individually below.

The starting point is a Web Crawler (or spider) to retrieve all Web pages: it simply traverses the entire Web or a certain subset of it, to download the pages or files it encounters and save for other components to use. The actual traversal algorithm varies depends on the implementation; depth first, breadth first, or random traversal are all being used to meet different design goals. The parser takes all downloaded raw results, analyze and eventually try to make sense out of them. In the case of a text search engine, this is done by extracting keywords and checking the locations and/or frequencies of them. Hidden HTML tags, such as KEYWORDS and DESCRIPTION are also considered. Usually a scoring system is involved to give a final point for each keyword on each page. Simple or complicated, a search engine must have a way to determine which pages are more important than the others, and present

them to users in a particular order. This is called the Ranking System. The most famous one is the Page Rank Algorithm published by Google founders [2].

A reliable repository system is definitely critical for any application. Search engine also requires everything to be stored in the most efficient way to ensure maximum performance. The choice of database vendor and the schema design can make big difference on performance for metadata such a URL description, crawling date, keywords, etc. More challenging part is the huge volume of downloaded files to be saved before they are picked up by other modules. Finally, a front-end interface for users: This is the face and presentation of the search engine. When a user submits a query, usually in the form of a list of textual terms, an internal scoring function is applied to each Web page in the repository [3],and the list of result is presented, usually in the order or relevance and importance .Google has been known for its simple and straight forward interface, while some most recent competitors, such as Ask.com, provide much richer user experience by adding features like preview or hierarchy displaying.

1.3 Search Engines Available Today

Other than well-known commercial products, such as Google, Yahoo and MSN, there are many open source Search Engines, for example, ASPSeek, BBDBot,Datapark Search, and ht://Dig. Evaluating their advantages and disadvantages is not the purpose of this thesis, but based on reviews and feedbacks from other people [4], they are either specialized only in a particular area, or not adopting good ranking algorithms, or have not been maintained for quite a while. Another important fact is that while most current search engines are focused on text, there is an inevitable trend that they are being extended to the multi-media arena, including dynamic contents, images, sounds and others [5]. None of the open source engines listed above has multimedia searching modules, and none of them is flexible enough to add new ones without significant effort.

1.4 Issues in Search Engine Research

Design of Web crawlers: Web crawler, also known as robot, spider, worm, and wanderer, is no doubt the first part of any search engine and designing a web crawler is a complex endeavor. Due to the competitive nature of the search engine business, there are very few papers in the literature describing the challenges and tradeoffs inherent in web crawler design [6]. Page ranking system: Page Rank [2] .is a system of scoring nodes in a directed graph

based on the stationary distribution of a random walk on the directed graph. Conceptually, the score of a node corresponds to the frequency with which the node is visited as an individual strolls randomly through the graph. Motivated largely by the success and scale of Google's Page Rank ranking function, much research has emerged on efficiently computing the stationary distributions of Web-scale Markov chain, the mathematical mechanism underlying Page Rank. The main challenge is that the Web graph is so large that its edges typically only exist in external memory and an explicit representation of its stationary distribution just barely fits in to main memory[7]. Repository freshness: A search engine uses its local repository to assign scores to the Web pages in response to a query, with the implicit assumption that the repository closely mirrors the current Web [3]. However, it is infeasible to maintain an exact mirror of a large portion of the Web due to its considerable aggregate size and dynamic nature, combined with the autonomous nature of Web servers. If the repository is not closely synchronized with the Web, the search engine may not include the most useful pages, for a query at the top of the result list. The repository has to be updated so as to maximize the overall quality of the user experience. Evaluating the feedback from users: Two mechanisms have been commonly used to accomplish this purpose: Click Popularity and Stickiness [8]. Click Popularity calculates how often a record in the returned list is actually clicked by the user, and promote/demote its rank accordingly. Stickiness assumes the longer an end user stays on a particular page, the more important it must be. While being straightforward, the implementation of these two algorithms can be quite error prone. The data collecting the most difficult part, as the server has to uniquely identify each user. This has been further complicated by the fact that many people want to manually or programmatically promote their own Web sites by exploiting the weaknesses of certain implementations [9]. Two graduate students at UCCS [10][11] have been working on an Image search engine and a text search engine, respectively. Part of their work is to adopt the published Page Rank algorithm [2], and the results are quite promising. However, giving the experimental nature of these two projects, they are not suitable for scaling up and not mature enough to serve as a stable platform for future research. A complete redesign and overhaul is needed.

1.5 The Original Page Rank algorithm

Google is known for its famous Page Rank algorithm, a way to measure the importance of a Web page by counting how many other pages link to it, as well as how important those page themselves are. The published Page Rank algorithm can be described in a very simple manner:

PR(A) = (1-d) + d (PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))

In the equation above: PR(Tn): Each page has a notion of its own self-importance. That's "PR(T1)" for the first page in the web all the way up to PR(Tn) for the last page. C(Tn): Each page spreads its vote out evenly amongst all of its outgoing links. The count, or number, of outgoing links for page 1 is C(T1), C(Tn) for page n, and so onfor all pages.PR(Tn)/C(Tn): if a page (page A) has a back link from page N, the share ofthe vote page A gets is PR(Tn)/C(Tn). d: All these fractions of votes are added together but, to stop the other pages having too much influence, this total vote is "damped down" by multiplying it by 0.85(the factor d). The definition of d also came from an intuitive basis in random walks on graphs. The idea is that a random surfer keeps clicking on successive links at random, but the surfer periodically "gets bored" and jumps to a random page. The probability that the surfer gets bored is the dampening factor.(1 - d): The (1 – d) bit at the beginning is a probability math magic so the "sum of all Web pages" Page Rank is 1, achieved by adding the part lost by the d(....) calculation. It also means that if a page has no links to it, it still gets a small PR of 0.15 (i.e. 1 – 0.85). At the first glance, there is a paradox. In order to calculate the PR of page A, one must first have the PR of all other pages, whose Page Rank is calculated in the same way. The algorithm solves it by first assuming all pages to have the same PR of 1, and at each iteration PR is propagated to other pages until all PR stabilize to within some threshold. Because the large dataset PR algorithm deals with, measuring the stabilization of the PRs can be a difficult job itself. Research indicates that in some cases PR can be calculated in as few as 10 iterations [12], or it may take more than 100 iterations [13]. Another important fact is that when a page does not have outgoing links, the C(Tn), this page becomes a dangling URL, and must be removed from the whole picture. If such "pruning" was not done, the dangling may have critical implications in terms of computation. First, Page Rank values are likely to be smaller than they should be, and might become all zero in the worst case. Second, the iteration process might not converge to a fixed point [14] .

1.6 Crawler

A primitive implementation was written at very early stage of the project to retrieve some data for other modules to work with. While functioning correctly, this version rather is plain in terms of features: it is single threaded and does not have retrying, repository refreshing, URL hashing, smart checking on dynamic URLs, smart recognizing on file types, and avoiding crawler traps, etc. Its speed is also quite questionable and can only retrieve about 2000 URLs per hour on a fast network in the UCCS lab. Improvements can be made to add the features above and improve its speed. Fortunately two UCCS graduate students are already working on this area [14] .

1.7 Parsers

Same as the crawler, a simple functional text parser was written to glue the whole system together. It only parses certain selected areas of a document such as Meta data, title, anchor text, three levels of headers, and a short part at the beginning of each paragraph. A complete full text parser with satisfactory performance is in immediate need. Image processing is not currently implemented [14] .

2. INFORMATION RETRIEVAL AND RANKING

Web search engines return lists of web pages sorted by the page's relevance to the user query. The problem with web search relevance ranking is to estimate relevance of a page to a query. Nowadays, commercial web-page search engines combine hundreds of features to estimate relevance. The specific features and their mode of combination are kept secret to fight spammers and competitors. Nevertheless, the main types of features at use, as well as the methods for their combination, are publicly known and are the subject of scientific investigation.

Information Retrieval (IR) Systems are the predecessors of Web and search engines. These systems were designed to retrieve documents in curated digital collections such as library abstracts, corporate documents, news, etc. Traditionally, IR relevance ranking algorithms were designed to obtain high recall on medium-sized document collections using long detailed queries. Furthermore, textual documents in these collections had little or no structure or hyperlinks. Web search engines incorporated many of the principles and algorithms of Information Retrieval Systems, but had to adapt and extend them to fit their needs. Early Web Search engines such as Lycos and AltaVista concentrated on the scalability issues of running web search engines using traditional relevance ranking algorithms. Newer search engines, such as Google, exploited web-specific relevance features such as hyperlinks to obtain significant gains in quality. These measures were partly motivated by research in citation analysis carried out in the biblio metrics field. For most queries, there exist thousands of documents containing some or all of the terms in the query. A search engine needs to rank them in some appropriate way so that the first few results shown to the user will be the ones that are most pertinent to the user's need. The interest of a document with respect to the user query is referred to as "document relevance." this quantity is usually unknown and must be estimated from features of the document, the query, the user history or the web in general. Relevance ranking loosely refers to the different features and algorithms used to estimate the relevance of documents and to sort them appropriately. The most basic retrieval function would be a Boolean query on the presence or absence of terms in documents. Given a query "word1 word2" the Boolean AND query would return all documents containing the terms word1 and word2 at least once. These documents are referred to as the query's "AND result set" and represent the set of potentially relevant documents; all documents not in this set could be considered irrelevant and ignored. This is usually the first step in web search relevance ranking. It greatly reduces the number of documents to be considered for ranking, but it does not rank the documents in the result set. For this, each document needs to be "scored", that is, the document's relevance needs to be estimated as a function of its relevance features. Contemporary search engines use hundreds of features. These features and their combination are kept secret to fight spam and competitors. Nevertheless, the general classes of employed features are Publicly known and are the subject of scientific investigation. The main types of relevance features are described in the remainder of this section, roughly in order of importance. Note that some features are query-dependent and some are not. This is an important distinction because query-independent features are constant with respect to the user query and can be pre-computed off-line. Query-dependent features, on the other hand, need to be computed at search time or cached [15].

3. SYSTEM DESIGN

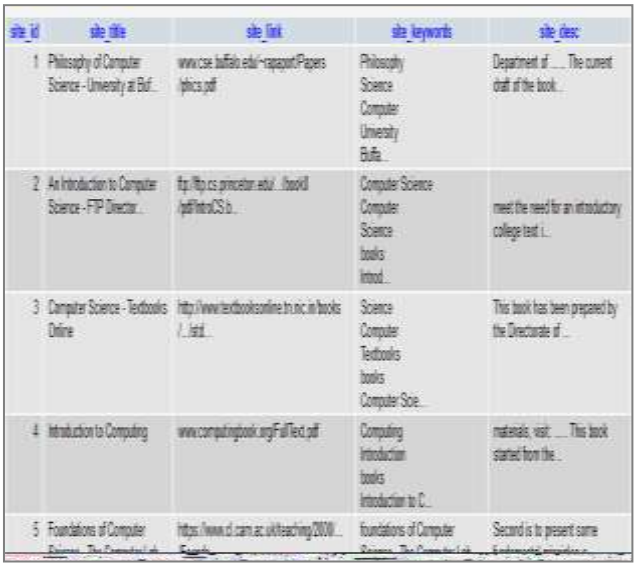
The displayed search results based on the number of visits .The system designed by using HTML, PHP and MYSQL. And WampServer.

Our system divided into two sides, client side and server side which contain the database of the system.

3.1. Database

Our of System consists of Database which is built in MYSQL. The type of data entered is (PDF, DOC, and PPT).it contains six fields which are explained in the table (1) blow:-

Table (1): Database of the system



site_id	site_title	site_link	site_keywords	site_desc
1	Philosophy of Computer Science - University at Buf...	www.cse.buffalo.edu/~rapaport/Papers /phics.pdf	Philosophy Science Computer University Buffa...	Department of .... The current draft of the book...
2	An Introduction to Computer Science - FTP Director...	ftp://ftp.cs.princeton.edu/.../bookB /pdf/IntroCS.b...	Computer Science Computer Science books Introd...	meet the need for an introductory college text i...
3	Computer Science - Textbooks Online	http://www.textbooksonline.tn.nic.in/books /...text...	Science Computer Textbooks books Computer Scie...	This book has been prepared by the Directorate of ...
4	Introduction to Computing	www.computingbook.org/FullText.pdf	Computing Introduction books Introduction to C...	materials, visit: .... This book started from the...
5	Foundations of Computer	https://www.cl.cam.ac.uk/teaching/2009/...	foundations of Computer	Second is to present some

The description of the table above explained as follows:

1-site_id: it is the primary key of database.
2-site_title: contain Web addresses.
3- Site_link: contain URLs.
4- site_keywords : It contains reserved words that are on the basis of which Search.
5- Site_desc: It has a simple description of the sites.
6- site_counter : A dynamic where it calculates the number of visits to the site.

4. SYSTEM IMPLEMENTATION

The following three steps in the process are:

1. Entering the word, or phrase of the file to be searched.
2. Getting the search results, or receiving the list of found documents back to terminal.
3. Finding the right file, or the information you were looking for and downloading to our own terminal.

This system can be implement by opening the first page of the system site as shown in figure(1) :

Figure (1):the Home page

We can write the keyword we need in search bar (any words or phrase) to search about, for example (Computer Science) and after that click on (Search) , the search results for that keyword will appear as shown in figure(2):

Figure (2): (computer science)Search results

Another example (Thesis) as shown in figure (3):-

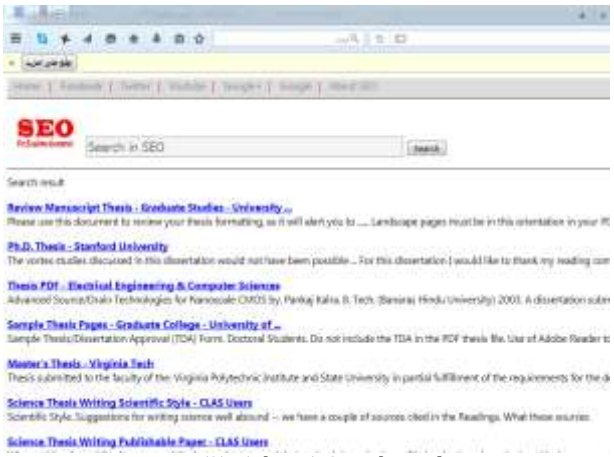


Figure (3): (Thesis) Search results

Ranking used in this system depending on the most Visit of any website included in the database. For example when we write (Articles) in the search bar, after that the results appear. If we enter the link time (Read the 5 Most Download Articles in 2011 for Free!) as shown in figure below (4):

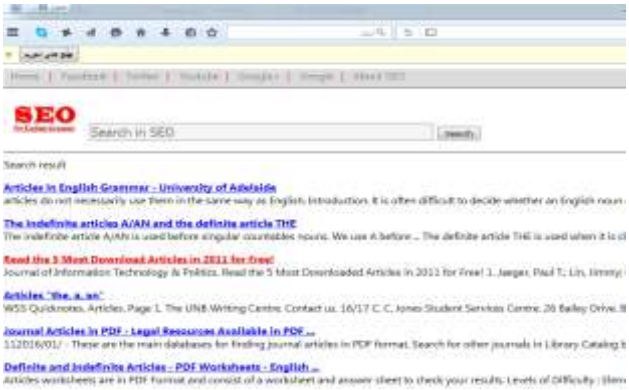


Figure (4): Search results before ranking

for the first time, and then visit this link many times more than the other links, the ranking of the search result will be changed when we write the same keyword in the search bar as shown in figure (5)below:

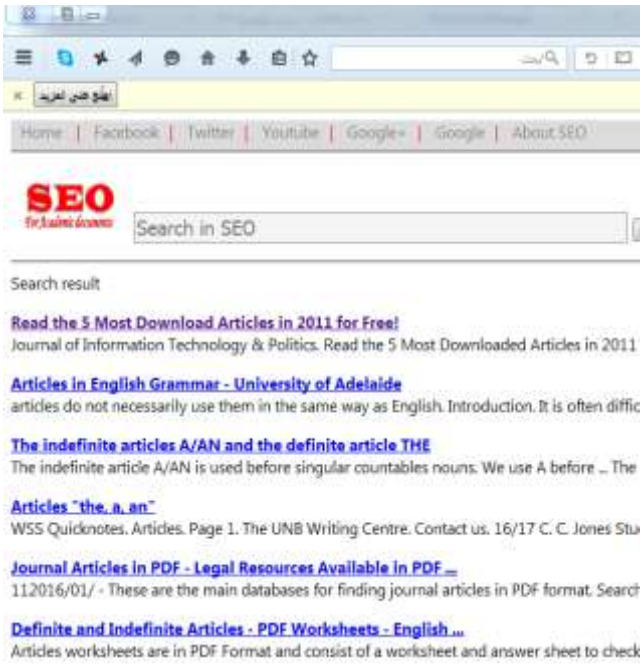


Figure (5): Search results after ranking

5. CONCLUSIONS

1-In this paper we conclude that there is an ability to search for information has already been entered into the database. The ranking in our search engine was achieved by using the most visit of any website included in the database. The suggestions we recommend to be achieved in the future works is to add Boolean operators to help in the search and increase the size of the database, also we can recommend to choose other Ranking algorithms to include the system.

6. REFERENCES

[1] Wall, Aaron. History of Search Engines & Web History. Visited November, 2005.

[2] Brin, Sergey and Lawrence Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. Proceedings of the Seventh International Conference on World Wide Web 7, Brisbane, Australia, Pages 107 – 117, 1998

[3] Pandey, Sandeep and Christopher Olston, User-Centric Web Crawling, International World Wide Web Conference, Chiba, Japan, May 10-14, 2005.

[4] Morgan, Eric. Comparing Open Source Indexers. O'Reilly Open Source Software Conference, San Diego, CA, July 23-27, 2001.

[5] Wall, Aaron. Future of Search Engines. Visited November, 200**5.**

[6] Allan Heydon, Marc Najork, Mercator: A scalable, extensible Web Crawler, World Wide Web 2, Pages 219-229, 1999

[7] McSherry, Frank, A Uniform Approach to Accelerated Page Rank Computation, International World Wide Web Conference, Chiba, Japan, May 10-14, 2005.

[8] Nowack, Craig. Using Topological Constructs To Model Interactive Information Retrieval Dialogue In The Context Of Belief, Desire, and Intention Theory. Dissertation of Ph.D. Pennsylvania State University, Pennsylvania, 2005.

[9] Harpf, Lauri. Free website promotion tutorial. Visited Nov, 2005.

http://www.apromotionguide.com/

[10] Jacobs, Jing. CatsSearch An Improved Search Engine Design For web pages in the UCCS Domain. University Of Colorado at Colorado Springs, December, 2005.

[11] Kodavanti, Apparao. Implementation of an Image Search Engine. University Of Colorado at Colorado Springs, December, 2005.

[12] Haveliwala, Taher H. Effcient Computation of Page Rank. Technical Report. Stanford University, California, 1999

[13] Kamvar Sepandar D, Taher H. Haveliwala, Christopher D. Manning, and Gene H. Golub. Extrapolation methods for accelerating Page Rank computations. The 12th International. Conference on the World Wide Web, pages 261–270, 2003.

[14] Kim, Sung Jin and Sang Ho Lee. An improved computation of the Page Rank algorithm. The European Conference on Information Retrieval (ECIR), pages 73–85, 2002.

[15] Hugo Zaragoza and Marc Najork Web Search Relevance Ranking. Yahoo Research Group, Barcelona, Spain. 2009.