ISSN No. 0976-5697

Volume 2, No. 5, Sept-Oct 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Optimized Searching Algorithm Based On Page Ranking: (OSA – PR)

Anu Kundu* and Sona Malhotra
CSE Dept. UIET Kurukshetra Universitry,
Kurukshetra, India
anu.haryana@gmail.com

Abstract: Search engines have greatly influenced the way people access information on the Internet as such engines provide the preferred entry point to billions of pages on the Web. Therefore, highly ranked web pages generally have higher visibility to people and pushing the ranking higher has become the top priority for webmasters. As a matter of fact, search engine optimization (SEO) has became a sizeable business that attempts to improve their clients' ranking. Still, the natural reluctance of search engine companies to reveal their internal mechanisms and the lack of ways to validate SEO's methods have created numerous myths and fallacies associated with ranking algorithms. In this paper a new mechanism is proposed that improves page ranking system for search engine optimization and then provides optimized results for user.

Keyword: Search Engine, Crawler, Ranking Engine

I. INTRODUCTION

Search engine is a program that accesses the information on the Internet as search engine provides the preferred entry point to billion of pages on the Web. Search engines use automated software programs know as spiders or bots to survey the web and build their databases. Web documents are retrieved by these programs and analyzed. Data collected from each web page are then added to the search engine index. When a user enter a query at a search engine ***site***, then the user's input is checked against the search engine's index of all the web pages it has analyzed. The best urls are then returned to user as hits, ranked in order with the best results at the top. Some basic search engines are Google, Alta Vista, Yahoo! Etc.

a) Problem definition- The aim of this study is to understand the important factors that affect the ranking of a web page as viewed by popular search engines. In this work, we focus on search engine, and objective to study the relative importance of web page features that potentially affect the ranking of a web page. *Web spam gives description* to hyperlinked pages on the Worldwide Web that are created with the intention of misguiding search engines. With the search engines' increasing importance in the everybody's life, there are more and more attempts to mischievously affects the page rankings. This kind of action called web spamming .Web spamming is illegal, as it misleads both search engines and users . Web spamming is the practice of introducing artificial text and links into web pages to affect the results of searches. It is also a serious problem for users because they are not aware of it and they tend to confuse trusting the search engine with trusting the results of a search.[1]

For each web page (URL), we collect 17 ranking features. These ranking features can further be divided into 7 groups. The page group represents characteristics attached with the web page including page rank score (PR) and the age of the web page (AGE). The URL group represents features associated with the URL of the web page. Parameters HOST and PATH counts, the number of occurrences of the keyword that appear in the hostname and number of occurrences of the keyword in the page segment of the URL, respectively. The domain group consists of features related to the domain of a web site. D SIZE reports the number of web pages indexed by Google in the domain and D AGE reports the age of the first page index by archive.org in the domain. Group's header, body, heading and link are features extracted from the content of the web page. TITLE counts the number of occurrences of the keyword in the title tag. M KEY counts the number of occurrences of the keyword in the meta keyword tag and M DES counts the number of occurrences of the keyword in the meta description tag. DENS is the keyword density of a web page which is calculated as the number of occurrences of the keyword divided by the number of words in the web page. H1 through H5 is the number of occurrences of the keyword in all the headings H1 to H5, respectively. ANCH counts the number of occurrences of the keyword in the anchor text of an outgoing link and IMG counts the number of occurrences of the keyword in an image tag.

Table 1 Ranking features

Group	Feature	Detail
Page	PR	pagerank score
	AGE	age of the web page
URL	HOST	keyword appear in hostname
	PATH	keyword in the path segment of url
Domain	D_SIZE	size of the web site's domain
	D_AGE	age of the web site's domain
Header	TITLE	keyword in the title tag of html header
	M_KEY	keyword in meta-keyword tag
	M_DES	keyword in meta-description tag
Body	DENS	keyword density
Heading	H1	keyword in h1 tag
	H2	keyword in h2 tag
	H3	keyword in h3 tag
	H4	keyword in h4 tag
	H5	keyword in h5 tag
Link	ANCH	keyword in anchor text
	IMG	keyword in image tag

b) Methodology – In diagram one, we have shown the architecture of our system . Two major components are the crawler and the ranking engine. Data collection is

performed by the *crawler* which queries Google and receives the ranked search results. Additionally, it downloads HTML web pages from their original web sites and queries domain information. Next, since multiple features can affect the ranking of web pages in complicated ways, the ranking engine extracts features under study from raw web pages and performs learning to train several ranking models to approximate the ranking results by Google. Here, we are making many contributions:

i. We prove that Google's ranking function is not a simple linear function of all the features, by showing a nonlinear model can outperform, *i.e.*, approximates Google's ranking better than, a simple linear model. Never the less a nonlinear model is difficult for humans to digest.

ii. We introduce a simple Optimized Searching Algorithm Based On Page Ranking procedure based on a simple linear model and show that it might succeed comparable accuracies to the nonlinear model. The theoretical underpinning for such a procedure is that recursive application of a linear model (function) may effectively approximate a non-linear function. In addition, the linear model converges more efficiently and outputs more human readable results.

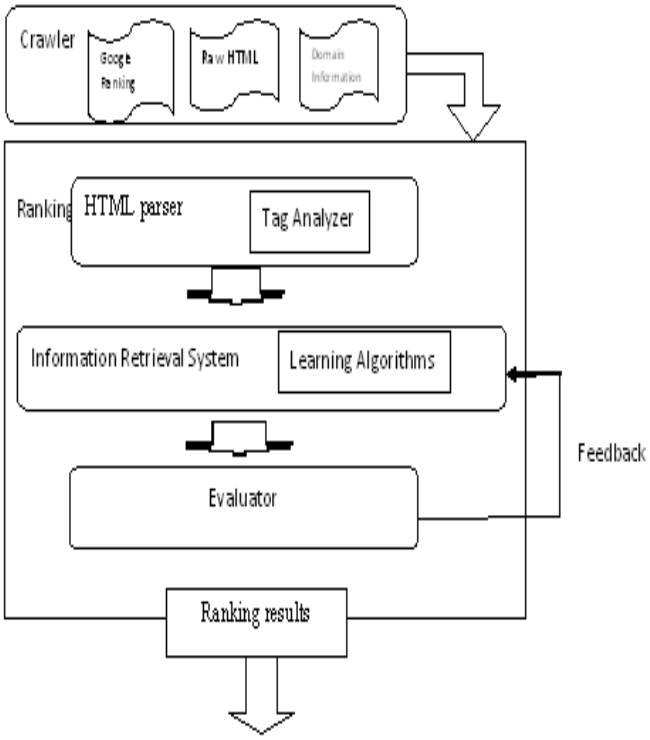


Figure1 System Architecture

A. The Crawler:

The crawler submits queries to search engine and gathers top 100 web pages (URL) for each keyword. We limited our queries to HTML files to avoid web pages generated dynamically by server side scripts such as CGI or PHP, without losing generality. In addition, we emphasis on web pages composed in English in our experiment. Finally, for each web page, the crawler does the following:

a. Downloads the web page from the original web site.
b. Queries the URL's page rank score by Google toolbar's API.
c. Gathers the age of a page (the date Google indexed the web page) by parsing the search result page.
d. Obtains the size (the total number of pages) of the domain by querying Google with site:[domain].
e. Queries archive.org and fetches the age of the web site (the date when the first web page was created on this web site) [5].

B. The Ranking Engine:

There are 3 components in the ranking engine. The HTML parser changes web pages into the document object model (DOM) for the tag analyzer to exam the number of keywords that appear in different HTML tags such as anchor text. The ranking engine trains the ranking model by combining features obtained from the web page contents, page rank scores, and domain information. After the model is created, the ranking engine judges the testing sets by applying the model. The evaluator then analyzes the results and provides feedback to the ranking engine which is used to adapt parameters in the learning algorithms such as error threshold. We describe the two ranking models– Linear programming and SVM. We always use ranking features to train our ranking models.

a) Linear Programming Ranking Model: We explain our linear programming ranking model. Given a set of documents $I = (i1, i2, ..., in)$, pre-defined Google ranking $G = (1, 2, ..., n)$, and a ranking algorithm A, the objective is to find a set of weights $W = (w1, w2, ...,wm)$ that makes the ranking algorithm re-produce Google ranking with minimum errors. The objective function of the linear programming algorithm attempts to minimize errors (the sum of penalties) of the ranking of a document set. Equation (1) defines the objective function which is a pair wise comparison between two documents in a given data set.

$$\Phi(W) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_i \cdot |i - j| \cdot D(i, j) \qquad (1)$$

In Equation (1), *ci* is a factor that weights the importance of the *ith* document (*e.g.*, a top 5th page is more important than a top 50th page). $|i - j|$ is the distance (ranking difference) between the *ith* and the *jth* page. Finally, $D(i, j)$ is a decision function we define as

$$D(i,j) = \begin{cases} 0 & \text{if } f(A,W,i) >= f(A,W,j), \\ 1 & \text{if } f(A,W,i) < f(A,W,j). \end{cases} \qquad (2)$$

where $f(A,W, i)$ is the score produced by algorithm A with a set of weights W for the *ith* page in the given data set. Page X is ranked higher than page Y if it receives a higher score than page Y. The decision function refers that if the ranking of the two pages maintains the order as Google's ranking, the penalty is zero. Otherwise, the penalty will be counted in the error function which is denoted in Equation (1). Since we cannot import conditional functions (*e.g.*, $D(i, j)$) into a linear programming solver, we transform the decision function into the following form:

$$f(A,W,i) + D_{ij}F_{\max} \geq f(A,W,j), \qquad (3)$$

where Fmax is the maximum value to which $f(A,W, \cdot)$ would evaluate, and $Dij \in \{0, 1\}$. When $Dij = 0$, the preceding inequality is satisfied only if

$$f(A,W,i) \geq f(A,W,j). \qquad (4)$$

When $Dij = 1$, the inequality is always satisifed. Therefore, we have effectively converted the original

minimization problem into the problem of minimizing the *Dij* . Hence, we can now replace *D*(*i, j*) in the objective function with *Dij* . Finally, the score function *f*(*A,W, i*) can be represented by a dot product of ranking parameters *X* = (*x*1, *x*2, ..., *xm*) and weights *W* = (*w*1, *w*2, ...,*wm*) denoted as *f*(*A,W, i*) = *fA*(*wi* · *xi*). For example, the parameter *xi* can be the number of keywords that occur in the title tag and *wi* is the weight associated with *xi*.

In addition to the objective function, we set constrains to our linear programming model. For each pair of pages (i, j) where i < j (i is ranked higher than j by Google), we have a constrain:

$$f(A,W,j) - f(A,W,i) \leq \tau \qquad (5)$$

where τ is the maximum allowed error which is set to a predefined constant τ . The constant τ is adjusted by the feedback from the evaluator to refine the ranking results. For example, when linear programming solver cannot find a feasible solution, we relax the maximum allowed error τ. In addition, we apply a Optimized Searching Algorithm Based On Page Ranking algorithm.

b) Support Vector Machines Ranking Model*:* Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and learning ranking functions. In a SVM, data points are viewed as n-dimensional vectors (n equals to the number of ranking features in our case). A SVM builds a hyperplane or a set of hyperplanes in a high-dimensional space, which is used to categorize to separate data points. We use the SVM-rank implementation with linear and polynomial kernels to train the ranking functions. The parameter c in SVM-rank controls the compromise between training error and margin. The ranking engine adjusts the value of parameter c according to the feedback provided by the evaluator in order to find the best value for prediction accuracy. Finally, we perform a Optimized Searching Algorithm Based On Page Ranking: (OSA – PR)[6,11,13]

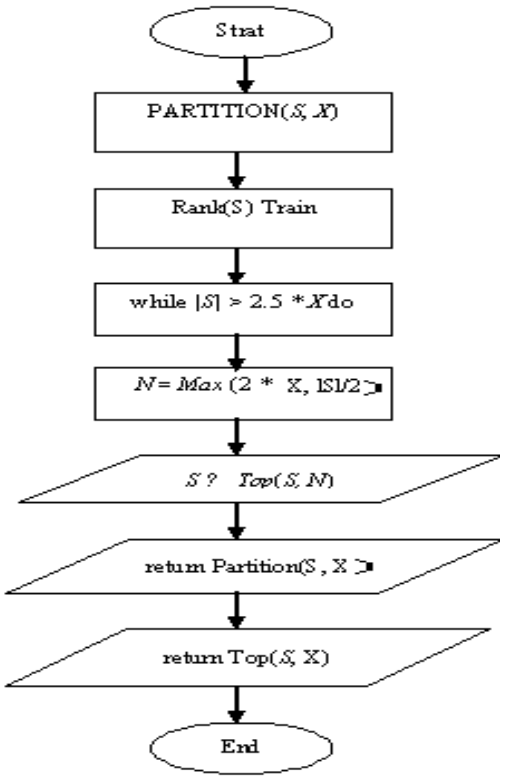
C. Optimized Searching Algorithm Based On Page Ranking: (OSA – PR):

It is common that a search engine keeps several layers of indices in practice. For example, the first layer of indices may serve as a cache and it is able to answer queries for top 20 pages. When the first layer query fails, it is then sent to subsequent indices. In Addition to this, the search engine's internal ranking algorithm may be non linear. To capture such a non-linear and/or non equational behavior for search engine's ranking function, we developed a Optimized Searching Algorithm Based On Page Ranking: (OSA – PR) to approximate this ranking behavior. We apply this algorithm to both our linear and SVM models. First, we describe our Optimized Searching Algorithm Based On Page Ranking: (OSA – PR) with pseudo code shown below.

D. Optimized Searching Algorithm Based On Page Ranking: (OSA – PR): Proposed Algorithm:

Step 1: procedure PARTITION(*S, X*) S: a set of pages, X: top X
Step 2: Rank(S) Train or apply ranking models
Step 3: while |*S*| > 2.5 * *X* do
Step 4: *N* = *Max* (2*X|*S*|/2)
Step 5: *S* ← *Top*(*S, N*) Return top N pages
Step 6: return Partition(S, X)
Step 7: end while
Step 8: return Top(*S*, X) Return top X pages
Step 9: end procedure

By using this algorithm proposed for page ranking can be used for search engine optimization and finding better results according to user requirement.



Flow Chart

II. WORKING OF ALGORITHM

In algorithm 1, *S* denotes a set of pages in a dataset and *X* denotes the target top X pages to be evaluated (*e.g.*, top 10 pages). Algorithm 1 can be explained by giving an example of how to train Optimized Searching Algorithm Based On Page Ranking ranking models for selecting top 10 pages (*X* = 10) out of 100 web pages (|*S*| = 100). In the first round of recursion the learning algorithm produces a set of weights by training all 100 pages (in line #2 of algorithm 1). Next, line #4 calculates *N* = 50. In line #5, the function returns the top 50 pages out of 100 using the model learned previously.

Next, the algorithm moves on to the second recursion with a set of 50 pages in *S*. Similarly, in the second recursion, a new set of weights for ranking is learned and the variable *N* in line #4 becomes 25. The partitioning algorithm further extracts top 25 pages from the 50 pages (in line #5) and proceeds to the third round. In the third round, an additional new set of weights is learned in line #2 of the algorithm 1. The condition statement in line #3 is not met in this round. Therefore, the algorithm escapes the recursive while loop. Algorithm 1 then proceeds to line #8 and return the top 10 pages by applying the ranking model learned in the third round. The process of evaluating the testing sets using our recursive partitioning ranking algorithm is similar to the steps we described above. The input of this evaluation process contains a set of pages to evaluate *S*, a variable *X*, and ranking models learned in the previous training procedure. For example, to evaluate top 10 pages out of 100 pages in a dataset, the Optimized Searching Algorithm

Based On Page Ranking algorithm first obtains top 50 pages using the weights learned in the first round of the training process. The top 50 pages are sent to the second round and the algorithm extracts top 25 pages using the set of weights 53 learned in the second round. Finally, the third round evaluates top 10 pages out of the 25 pages using the third set of weights learned in the third round of the training process.

III. RESULTS

In paper, this search engine is very beneficial for the students to finding reading materials and research work material of their thesis work. Using this search engine when user enters any keyword into textbox then it will go to different search engines like Google, yahoo etc. and filters related useful data links and shows in our search engines screen. When user enters any keyword into the text box then our search engine filters data from Google, yahoo etc. search engines and only useful data retrieve from these search engines and shows in our page. In following points we will present the various screenshots as well as their descriptions for them.

Figure: 1

In this screenshot we show when the page gets loaded, the drop down box gets loaded with the search engine names. The parameters also change with the search engine selection. These parameters are hard coded in a JavaScript object, and can be changed easily in code.

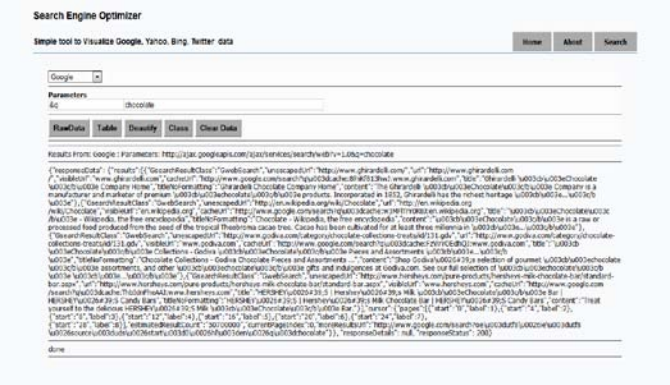


Figure:2

This screenshot shows raw data outputs.

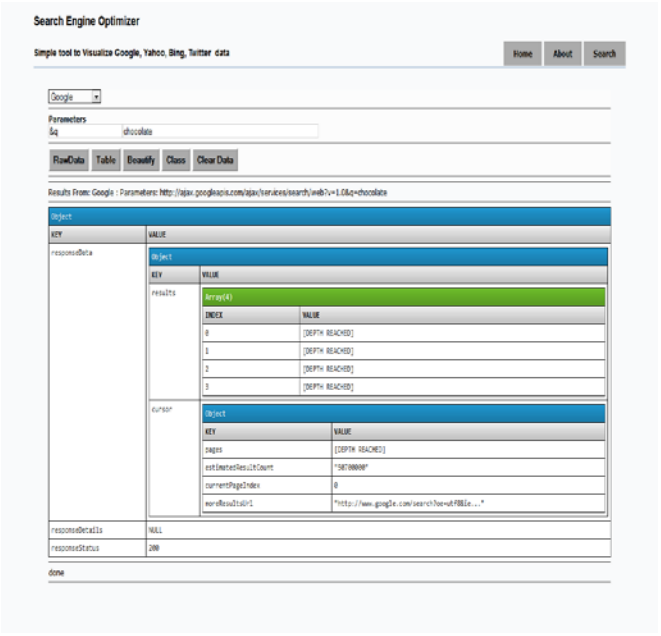


Figure: 3

This is useful when you want to look at the general structure and also when you want to print the results.

IV. CONCLUSION

Using this concept we will develop search engine for all type of data searching. Using this search engine user will get only useful data which is retrieves from different search engines. In this paper, we have proposed an Optimized Searching Algorithm Based On Page Ranking. Using this mechanism we improve page ranking system for search engine optimization.

V. REFERENCES

[1]. "A breakdown of Google's ranking factors," http://www. pronetadvertising.com/articles/a-breakdown-of-googles-ranking-factors.html.

[2]. "Google Ranking Factors," http://www.vaughns-1-pagers.com/internet/google-ranking-factors.htm.

[3]. "Google SEO Test Google Prefers Valid HTML & CSS," http://www.hobo-web.co.uk/seo-blog/index.php/official google-prefers-valid-html-css/.

[4]. "Google's Top Search Engine Ranking Factors," http://lornali.com/ online-marketing/seo/googles-top-search-engine-ranking-factors.

[5]. "PageRank on Google Toolbar," http://www.google.com/support/toolbar/bin/answer.py?hl=en &answer=79837.

[6]. "SVM-rank Support Vector Machine," http://www.cs.cornell.edu/People/tj/svm Flight/svm rank.html.

[7]. "Top 10 Most Important Google RankingFactors,"http: //blogs.myspace.com/index.cfm?fuseaction=blog.view&friendId=21196&blogId=493022330.

[8]. A. A. Bencz´ur, K. Csalog´any, T. Sarl´os, and M. Uher, "SpamRank–Fully Automatic Link Spam Detection," in Proc. AIRWeb, 2005, pp. 25–38.

[9]. J. Cho and S. Roy, "Impact of search engines on page popularity," in Proc. WWW '04: Proceedings of the 13th international conference on World Wide Web. New York, NY, USA: ACM, 2004, pp. 20–29.

[10]. Z. Gyongyi, P. Berkhin, H. Garcia-Molina, and J. Pedersen, "Link spam detection based on mass estimation," in Proc. VLDB '06: Proceedings of the 32nd international conference on Very large data bases. VLDB Endowment, 2006, pp. 439–450.

[11]. T. Joachims, "Making Large–Scale SVM Learning Practical," in Proc. Advances in Kernel Methods — Support Vector Learning, B. Sch¨olkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 169–184.

[12]. M. Moran and B. Hunt, Search Engine Marketing, Inc.: Driving Search Traffic to Your Company's Web Site. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005.

[13]. N. V. Vapnik, The Nature of Statistical Learning Theory. Springer- Verlag, New York., 2000.

[14]. B. Wu and B. D. Davison, "Identifying link farm spam pages," in Proc. WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web. New York, NY, USA: ACM, 2005,pp.820–829