UACEE International Journal of Computer Science and its Applications - Volume 2: Issue 3 [ISSN 2250 - 3765]

Optimize Rank of Search Engine Query Results Using Log Mining

Kajal Y. Vyas[1], Kiran R. Amin[2]
Computer Engineering Department,
U. V. Patel College of Engineering, Ganpat University,
Kherva-Mehsana, India
E-mail: kajalvyas55@gmail.com[1]
kiran.amin@ganpatuniversity.ac.in[2]

***Abstract*— Search engine transaction logs have been investigated through large number of studies. Here mining of search engine query log is done. By mining user's feedback, ranks of search engine result pages are optimized, so that related pages come earlier in the list. In this way user can easily and quickly find desired pages. Here web mining techniques are used to order the documents. This method first mines query logs using a novel similarity function to perform query clustering of similar queries. Then it discovers sequential pattern of clicked URLs in each cluster using existing Sequential Pattern mining algorithm PLWAP. In the end, search result list is re-ranked by updating existing rank values of pages using discovered sequential patterns. By this method, user desired relevant pages move upwards in result list and reduce search space for users.**

Keywords— **Web mining, Search engine, Query logs, Ranking algorithm, Sequential pattern mining, Clustering.**

I. Introduction

To find relevant information from large amount of data on Internet, search engine is an effective tool. Queries are given through a combination of keywords on the interface of search engine. Previously, search engines were worked on keyword based similarity function between the query and the documents, which was likely to poor quality of search results. Then ranking methods were proposed to use linkage structure of the web (web structure mining), instead of using the content, to improve the search result quality. Because the ambiguity and the short lengths of query terms that do not describe the user's personal requirements enough, Search engine will return a lot of web pages which contain many useless pages and some web pages only related to query terms but they are not necessarily the correct Web pages. One of the major reasons for this problem is the lack of user knowledge in giving queries. Moreover, search engines often have difficulties in forming a concise and precise representation of the response pages corresponding to a user query.

Although many search engines apply ranking, clustering and other web mining methodologies to optimize their search results, there still remains a challenge in providing the user required content with less efforts. Search engines must have a mechanism to optimize the results of queries by finding the users' interest. For that, search engine can use query log files maintain by it [1-3] from which we can get user's feedback. Here, query log mining is used to improve performance of the search engine by utilizing the mined knowledge.

II. Basic Definitions And Related Work

A. *Query Logs of Search Engine*

Web log/structure mining works on web log data. When users access search engine, every time an entry corresponding to query is recorded in the log file by search engine servers. Search engine query logs contain following fields:

1. User ID/Session ID
2. Query entered by user
3. URL clicked by user for query
4. Rank of URL clicked for query
5. Time of query submission in search engine.

TABLE I. Search Engine Query Logs

User ID	Query	Clicked URL	Rank	Time
8041	plus projects in chemistry	http://www.super-science-fair-projects.com	1	2006-05-12 21:09:34
8041	plus projects in chemistry	http://www.super-science-fair-projects.com	2	2006-05-12 21:09:34
8041	plus projects in chemistry	http://www3.sdstate.edu	6	2006-05-12 21:09:34
12694	jobs san diego	http://www.sandiegojobs.com	1	2006-03-15 15:58:16
12694	jobs san diego	http://sandiego.employmentguide.com	7	2006-03-15 15:58:16
12694	jobs san diego	http://careers.signonsandiego.com	3	2006-03-15 15:58:16
12694	jobs san diego	http://www.sandiegocareers.com	2	2006-03-15 15:58:16

The problem of analyzing query logs has been discussed by many researchers [1, 5-7]. The information within the query logs have been used in many areas like query classification, personalization, to infer search intent, etc. Information from query logs is used for search process and to improve search engine by various researchers in many studies.[1, 3, 8]

B. Ranking

Ranking algorithms are used by Search Engines to extract relevant documents and pages from the web database and provide the necessary information to the users. There are many different ranking algorithms used by different search engines like HITS(Hypertext Induced Topic Search), PageRank, DistanceRank, Weighted PageRank, etc. [9-11]. Some ranking algorithms depend only on the link structure of the documents i.e. their popularity scores (web structure mining), whereas others look for the actual content in the documents (web content mining), while some use a combination of both i.e. they use content of the document as well as the link structure to assign a rank value for a given document.

Google and AOL uses PageRank algorithm to calculate the rank score. PageRank can be calculated by following formula:

$$PR(A) = (1 - d) + d(PR(T_1)/C(T_1)+\ldots+PR(T_n)/C(T_n))$$

Here T_1 to T_n are incoming links in Page A. A page obtains a high rank if the sum of the ranks of its back links (incoming links) is high. The parameter d is a damping factor, usually sets it to 0.85 (to stop the other pages having too much influence, this total vote is “damped down” by multiplying it by 0.85). C(A) is defined as the number of links going out of page A. PR(A) is PageRank of A. PR is not calculated by text matching. So, the most relevant web pages to users' query words may not be shown at the top of the search result list.

C. Sequential Pattern Mining

Here, the work aims to improve the rank score of pages by considering the sequential order of pages accessed by the users. Mining sequential pattern will help to find web pages, in which order they get visited by users. AprioriAll finds frequent pattern, but it is a slow algorithm. It first finds all frequent item-sets. Then it will filter item-set with minimum support, transforms the database so that each transaction is replaced by the set of all frequent item sets contained in the transaction, and then finds sequential patterns. Generalized Sequential Pattern (GSP) algorithm is much faster than the AprioriAll algorithm[12]. But it also uses same method of candidate set generation as Apriori and AprioriAll. It has some drawbacks. So in this paper better algorithm than GSP, PLWAP[13] is used.

III. An Optimization System – Proposed Solution

The architecture of search result optimization is shown in Figure 1.

Architecture proposed in [2, 3] consists of following functional components: A. Similarity Analyzer, B. Query Clustering Tool, C. Pattern Generator, and D. Rank Updater

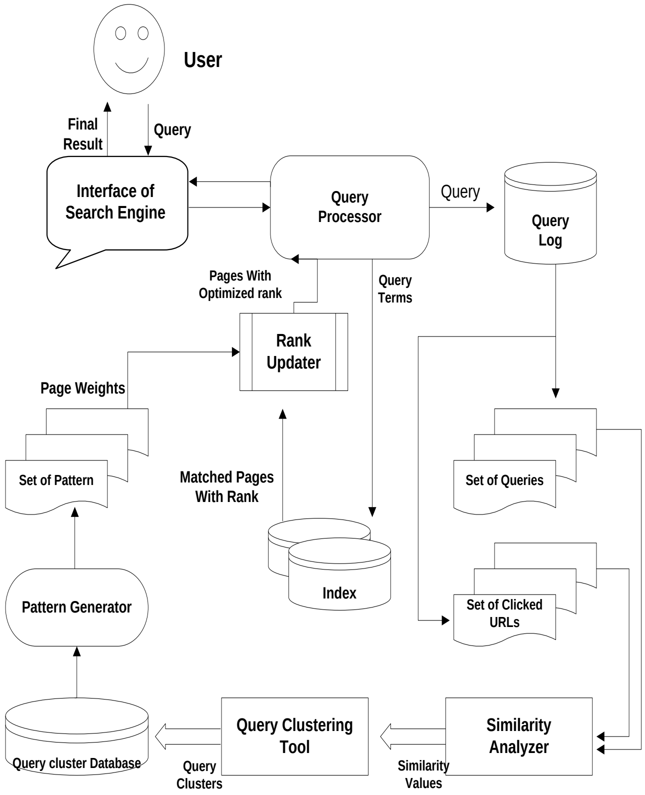

Figure 1 Architecture of Rank Optimization System [2, 3]

Query terms of user given query are matched with index repository of search engine by Query processor and it returns a list of matched documents. The result optimization system first performs similarity function between queries. User browsing behavior is stored in query log file. Using similarity analyzer, similarity between user queries are found and based on its output query clustering is done. So query clustering tool generates groups of similar queries. Then Pattern generator captures the frequent sequential patterns of clicked web pages in each cluster using a sequential pattern mining algorithm. Finally, search result list is re-ranked by updating the existing Rank values of pages using the discovered sequential patterns in rank updater module. The proposed work results in reduced search space as user wished-for pages tend to move upwards in the result list. According to architecture, Rank updater works online, while sequential patterns are discovered offline.

A. Similarity Analyzer

This function analyzes queries for similarity. It continuously checks submitted queries and clicked URLs get stored in the logs, and. Similarity can be checked by following equations:

<u>Similarity based on words of queries</u>:-

Two queries can fulfill similar information needs if they contain same or similar words. To check query content similarity between 2 queries, following formula of cosine similarity can be used [6].

$$Sim_{keyword}(p,q) = \frac{\sum_{i=1}^{k} cw_i(p) \times cw_i(q)}{\sqrt{\sum_{i=1}^{m} w_i^{2}(p)} \times \sqrt{\sum_{i=1}^{n} w_i^{2}(q)}} \text{----- (1)}$$

where $cw_i(p)$ and $cw_i(q)$ are the weights of the i-th common keyword in the query p, and q respectively and $w_i(p)$ and $w_i(q)$ are the weights of the i-th keywords in the query p and q respectively. In this paper, for query weighting tf (term frequency) is used.

Similarity based on User Feedback:-

If two queries lead to the selection of same or similar documents/pages/URLs, they are considered similar.

For any query, top k clicked URLs are taken. Each URL is then broken into tokens. First, we split the URL by pre-defined separators, such as slashes (/), dots (.), etc. Second, numbers (except for those in website domains) and URL stop words (e.g. www, .com and .index) are removed. Finally, the URL is represented by tokens. [7]

Each query q_i is then represented by a vector of tokens as follows:

$$\vec{q_i} = < w_{i1}, w_{i2}, \ldots, w_{in} >$$

where w_{ij} is the weight of token t_j in $q_{i.}$ Here weight is calculated by taking term frequency. Then, the similarity between two queries p and q- $Sim_{URL}(p,q)$ is determined using cosine similarity as per equation (1). Here weights of tokens are taken instead of weights of keywords.

Combination of both measures:-[6]

Both criteria can be combined and advantages of both can be used. Both query keywords and the corresponding document clicks can partially capture the users' interests when considered separately. Therefore, it is better t combine them in a single measure. A simple way to do it is to combine both measures linearly as follows:[5]

$$Sim_{combined}(p,q) = \alpha * Sim_{keyword}(p,q) + \beta * Sim_{URL}(p,q) \text{----- (2)}$$

Where α and β are constants where 0<=α and $\beta \leq 1$ and $\alpha + \beta = 1$. In the current implementation, these parameters are taken to be 0.5 each.

B. *Query Clustering*

Clusters of similar queries are made in this module. Query Clusters show what user wants actually from search engine. For obtaining these clusters, the Query Clustering module uses the algorithm, where each run of the algorithm computes k clusters. As query logs are not static, so query clustering algorithm should be incremental. Initially all queries are considered to be unassigned to any cluster. Each query is examined against all other queries. If the similarity value turns out to be above the pre-specified threshold value (τ), then the queries are grouped into the same cluster. The process is repeated until all queries get classified to any one of the clusters.

The clustering algorithm takes O (n^2) worst case time to find all the query clusters, where n is the total number of queries. The following clustering algorithm uses similarity functions. [2,3, 5]

```
Algorithm: Query_Clustering(Q, α, β, τ)

Given: A set of n queries and corresponding clicked URLs stored in an array Q[ $q_i$ , $URL_1$, ...,$URL_m$], 1≤i≤n
α = β = 0.5
Similarity threshold τ
Output: A set C= {$C_1$, $C_2$,…$C_k$} of k query clusters

// Start of Algorithm
k = 0; //k is the number of clusters.
For (each query p in Q)
Set ClusterId(p)= Null; // initially no query is clustered
For (each p $\in$ Q )
{
    ClusterId(p)= $C_k$;
    $C_k$={p};
    For (each q $\in$ Q such that p $\neq$ q)
    {
        $Sim_{keyword}(p,q) = \frac{\sum_{i=1}^{k} cw_i(p) \times cw_i(q)}{\sqrt{\sum_{i=1}^{m} w_i^{2}(p)} \times \sqrt{\sum_{i=1}^{n} w_i^{2}(q)}}$
        Divide URLs of query i into n tokens and calculate weights of tokens.
        $\vec{q_i} = < w_{i1}, w_{i2}, \ldots, w_{in} >$
        $Sim_{clickedURL}(p,q) = \frac{\sum_{i=1}^{k} cw_i(p) \times cw_i(q)}{\sqrt{\sum_{i=1}^{m} w_i^{2}(p)} \times \sqrt{\sum_{i=1}^{n} w_i^{2}(q)}}$

        $Sim_{combined}(p,q) = \alpha * Sim_{keyword}(p,q) + \beta * Sim_{clickedURL}(p,q)$

        If ($sim_{combined}(p,q) \geq \tau$ ) then
                set ClusterId(q)= $C_k$;
                $C_k$= $C_k \cup$ {q};
        else
                continue;
    }// end for
    k= k+1;
} //end outer for
Return Query cluster set C
```

C. Sequential Pattern Generator

Here PLWAP (position coded pre-order linked wap-tree) algorithm[13] is used for sequential pattern mining.

To find pages which are frequently accessed by users, this module is used. This component finds sequential pattern in each cluster of queries. It takes as input the query clusters. To find sequential pattern, many algorithms are used by many researchers. Apriori is widely known algorithm. AprioriAll is next to Apriori. GSP is also based on same concept as Apriori and AprioriAll but it is faster than both these algorithms. But GSP has some drawbacks. GSP uses candidate generate-and-test approach in which huge set of candidate sequences generated, more memory is required. When min_support drops, the number of frequent sequences grows up exponentially and it costs an exponential growth amount of time to process a pretty small database. So it is inefficient for mining long sequential patterns.

WAP-tree mining is a non-Apriori method which stores the web access patterns in a compact prefix tree, called WAP-tree, and avoids generating long lists of candidate sets to scan support for. However, WAP-tree algorithm has the drawback of recursively re-constructing numerous intermediate WAP-trees during mining in order to compute the frequent prefix subsequences of every suffix subsequence in a frequent sequence. This process is very time-consuming.

The PLWAP algorithm is a version of WAP tree algorithm. It assigns a unique position code to each node of the WAP-tree, builds the WAP-tree head links in a pre-order fashion (root, left, right) rather than in the order the nodes arrive as done by the WAP-tree algorithm. To assign position codes to a PLWAP node, the root has null code, and the leftmost child of any parent node has a code that appends ’1’ to the position code of its parent, while the position code of any other node has ’0’ appended to the position code of its nearest left sibling. With the pre-order linked, position coded WAP-tree, the PLWAP algorithm is able to mine frequent sequences, starting with the prefix sequence without the need to recursively re-construct any intermediate WAP-trees. This algorithm gives better performance than GSP and WAP-tree.[14]

D. Rank Updater

This module takes input from query processor means the matched documents of a user query and an update is applied to modify the rank score of the returned pages. It works online at query time and returns updated ranks of documents. Steps of this rank updating are as follow:

-Given an input user query q and matched documents D collected from the query processor, the cluster C is found to which the query q belongs.

-The sequential patterns of the concerned cluster are retrieved from the local repository maintained by the Sequential Pattern Generator.

-The level weights are calculated for every page X present in the sequential patterns.

-Final rank of a page is computed if it happens to be present in the patterns of cluster C. The improved rank is calculated as the summation of previous rank and assigned weight value.

(a) Weight Calculation

This can be calculated by the formula

$$\text{Weight}(X) = \frac{\ln(len_{pattern(X)})}{level(X)}$$

Where $len_{pattern(X)}$ is the effective length/depth of the sequential pattern in which X occurs and level(X) is the depth of X in the pattern.

(b) Rank improvement

Actual Rank used in a search engine can be improved by adding weight value to actual rank. Formula is as below:

New_Rank = Rank(X) + Weight(X)

IV. CALCULATION AND RESULTS

To show the practical evaluation of the proposed architecture, a fragment of sample query Log is considered (given in Table II). Because the actual number of queries is too large to conduct detailed evaluation, only 14 query sessions are chosen at present.

TABLE II. SAMPLE FRAGMENT OF LOG FOR EXPERIMENT EVALUATION [3]

	UserID	Query	Clicked URL
q1	1220051	Price Maruti Swift	www.marutiswift.com www.gaadi.com www.marutidzire.com
q2	1220051	Maruti Swift Dzire	www.marutidzire.com www.cardekho.com
……	…………	…………..	………………..
q5	1220051	Ray Ban sunglasses	www.ray-ban.com www.hisunglasses.com
……	………….	………….	………………..
q14	1220054	Ray Ban Sunglasses	www.ray-ban.com www.apparell.shop.ebay.in www.emporiumonet.com

For the given sample log, we want to show practical evaluation of proposed architecture.

Here α=0.5, β=0.5, τ=0.5 and min_sup=2.[3]

Similarity Function and Clustering

Suppose we want to calculate the similarity between the first 2 queries. Let us say,

q_l= Maruti Swift Price so q_1={Maruti, Swift, Price} and

q_2= Maruti Swift Dzire so q_2={Maruti, Swift, Dzire}

Here Maruti and Swift are 2 common words and frequency of them is 1 in both queries. Weights of both words are 1.

According to equation (1) $Sim_{keyword}(q_1,q_2)$= $\frac{1+1}{\sqrt{1+1+1}\sqrt{1+1+1}}$ =0.66. Here Length of both queries is 3. So in denominator 3 weights are taken 2 times.

For same queries, if find similarity between clicked URLs then URLs can be divided into tokens like q_1=<marutiswift, gaadi, marutidzire>,

q_2 =<marutidzire,cardekho>. Both have only 1 similar token and its weight is 1. So $Sim_{clickedURL}(q_1,q_2)$=

$$\frac{1}{\sqrt{1+1+1}\sqrt{1+1}} = 0.41$$

By replacing these values to equation (2), $Sim(q_1,q_2)$ is 0.53 > τ. So both queries are similar.

If similarity between query q_1 and q_5 are found, $Sim(q_1,q_5)$,no common click or no common keywords. So $Sim(q_1,q_5)=0$.

Similarly, clustering between other queries can be find. Final 2 clusters obtained are:

C1={q1,q2,q3,q4,q7,q8,q10, q11,q12,q13}

C2= {q5, q6, q9, q14}

Pattern generation

If URLs are assigned to different variables for easy calculation, let

A=www.marutiswift.com

B=www.gaadi.com

C=www.marutidzire.com

D=www.cardekho.com

E=www.carwale.com,   and so on.

one pattern D→BE→A is found corresponding to C1.

Weight Calculation and Rank updation

For the pattern D→BE→A, D is at level 1, B and E at level 2 and A at level 3.

Length of pattern=3
So, Weight (D) = ln(3)/1 = 1.099
Weight (B) =Weight (E) = ln(3)/2 = 0.549
Weight (A) = ln(3)/3=0.366
Suppose, Actual ranks of D,E,B,A were found 5,4,6,4.

New_Rank=Actual Rank + Weight value
New_Rank(D)=5+1.099=6.099
New_Rank(E)=4+0.549=4.549
New_Rank(B)=6+0.549=6.549
New_Rank(A)=4+0.366=4.366

From the above results, it is evaluated that the ranking of search results can be modified to a great extent and the more relevant Web pages can be presented according to the suggested implementation. The user can now find the popular and relevant pages upwards in the result list and need not to work much to find his desired ones.

V. CONCLUSION

Search Engine returns results based on various ranking algorithm. In response to user's query, search engine returns very long result list. It is very time consuming process to find related pages from it. Here rank optimization technique is suggested based on query log mining, which optimize the results of a search engine by returning the more relevant and user desired pages upward in search result list. It reduces user's efforts and seek time.

VI. SCOPE OF FUTURE WORK

Equation of rank improvement can be changed. For weight calculation, equation can be replaced by better one. Algorithm of sequential pattern mining can be improved.

REFERENCES

[1] Thorsten Joachirns, "Optimizing search engines using c1ickthrough data," Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 133-142, 2002.

[2] Neelam Duhan, A. K. Sharma, "“Rank Optimization and Query Recommendation in Search Engines using Web Log Mining Techniques”," Journal of computing, vol. 2, December 2010.

[3] A.K. Sharma, Neelam Duhan, Neha Aggarwal, Ranjna Gupta, "Web Search Result Optimization by Mining the Search Engine Query Logs," IEEE, 2010.

[4] Rajni Pamnani, Pramila Chawan, "Web Usage Mining: A Research Area in Web Mining."

[5] J. Wen, J. Mie, H. Zhang, "Clustering user queries of a search engine," In Proc. at 10th International World Wide Web Conference, 2001.

[6] Ji-Rong Wen, Jian-Yunnie, Hong-Jiang Zhang, "Query Clustering Using User Logs," ACM Transactions on Information Systems, vol. 20, pp. 59-81, January 2002.

[7] Yi Liu, Liangjie Zhang, Ruihua Song, Jian-Yun Nie, Ji-Rong Wen, "Clustering Queries for Better Document Ranking," CIKM'09, Hong Kong, China, 2009.

[8] K. Hofmann, M. de Rijke, B. Huurnink, E. Meij, "A Semantic Perspective on Query Log Analysis," In Working notes for the CLEF 2009 Workshop, Cortu, Greece, 2009.

[9] Ashutosh Kumar Singh, Ravi Kumar P, "A Comparative Study of Page Ranking Algorithms for Information Retrieval," International Journal of Electrical and Computer Engineering, vol. 4, 2009.

[10] Jon M. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, 1998.

[11] A.M. Zareh Bidoki, N. Yazdani, "DistanceRank: An intelligent ranking algorithm for web pages," Information Processing and Management, vol. 44, 2008.

[12] Ramakrishnan Srikant, Rakesh Agrawal, "Mining Sequential Patterns:Generalizations and performance improvements," Proc. of 5th International Conference Extending Database Technology (EDBT), France, March 1996.

[13] C. Ezeife and Y. Lu. Mining web log sequential patterns with position coded pre-order linked wap-tree. International Journal of Data Mining and Knowledge Discovery (DMKD) Kluwer Publishers,10(1):5–38, 2005.

[14] C.I.Ezeife, Yi Lu, Yi Liu. PLWAP Sequential Mining: Open Source Code *WOODSTOCK* '97 El Paso, Texas USA