Web Search Result Optimization by Mining the Search Engine Query Logs

[1]A.K. Sharma, [2]Neha Aggarwal, [1]Neelam Duhan and [2]Ranjna Gupta
[1]Department of Computer Engineering, YMCA University of Science & Technology, Faridabad, India
[2]Department of Computer Science & Engineering, B.S.A. Institute of Technology & Management, Faridabad, India

***Abstract*—Modern Information Retrieval Systems match the terms of a user query with available documents in their index and return a large number of Web pages generally in the form of a ranked list. It becomes almost impractical at the user end to examine every returned document, thus necessitating the need to look for some means of result optimization. In this paper, a novel result optimization technique based on learning from historical query logs is being proposed, which predicts users' information needs and reduces their navigation time within the result list. The method first performs query clustering in query logs based on a novel similarity function and then captures the 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 PageRank values of pages using the discovered sequential patterns. The proposed work results in reduced search space as user intended pages tend to move upwards in the result list.**

Keywords: Web, Query Logs, Query Clustering, Sequential Pattern Mining, PageRank Algorithm

I. INTRODUCTION

With the advancement in information technology, the Web [1] has become a huge information repository covering almost every topic, in which a human user could be interested. However, with the overwhelming volume of information on the Web, the task of finding relevant information related to a specific query/topic is becoming increasingly difficult. Many advanced Web searching techniques have been recently developed to tackle this problem and are being used in the commercial Web search engines such as Google and Yahoo. In spite of the recent advances in the Web search engine technologies; there are still many situations in which the user is presented with non-relevant search results. One of the major reasons for this problem is the lack of user knowledge in framing queries. The search engine users are not well trained in organizing and formulating their input queries, on which the search engine relies to find the desired search results. Moreover, search engines often have difficulties in forming a concise and precise representation of the user's information need. Nowadays, providing a set of web pages based on user query words is not a big problem in search engines [1]. Instead, the problem is that a search engine returns a large number of web pages in response to user queries and users have to spend much time in finding their desired information from this long list resulting in information overload problem [2]. Although many search engines provide a user friendly ranked list in response to user queries, there still remains a challenge in finding the desired information content within the result list due to the presence of user desired pages being scattered through out in the result list.

Almost all search engines store their user activities in the form of query logs. Query logs provide an excellent opportunity for gaining insight into how a search engine is used and what the users' interests are since they form a complete record of what users searched for in a given time frame. This log data is popularly known as click-through data and can be mined for understanding user information needs.

The work proposed in this paper aims to optimize the results of a search engine by returning the more relevant and user desired pages on the top of search result list. To perform the required task, the work takes into account the mined output of query logs. The paper has been organized as follows: Section II describes the related work and terminologies used in the work; section III illustrates the proposed optimization technique in detail along with the proposed similarity measure and algorithms. The practical evaluation of the work is given in section IV, while section V concludes the paper with some discussion on future research.

II. RELATED WORK

The notion of *Web Log Mining* has been a subject of interest since many years. The typical logs [3] of search engines include the following entries: (1) User (session) IDs, (2) Query *q* issued by the user, (3) URL *u* selected by the user (4) Rank *r* of the URL *u* clicked for the query q and (5) Time *t* at which the query has been submitted for search. A sample query log is shown in Table I to better understand this format.

TABLE I: A SAMPLE CLICKTHROUGH DATA FOR THE QUERY LOG

UserID	Query	Clicked URL	Rank	Time
270972	Data mining	www.datamining/typepad.com	6	2006-03-01 00:01:10
270972	Data mining	www.dataoutsourcingindia.com	5	2006-03-01 00:01:10
48785	Database	www.webopedia.com/database.html	5	2006-03-01 00:01:16
48785	Database	en.wikipedia.org/wiki/database	7	2006-03-01 00:01:16
48785	Database	http://www.cmie.com	6	2006-03-01 00:01:16
…..	……..	…………..	...	……..

A number of researchers have discussed the problem of analyzing the query logs [4, 5, 6, 7]. The information contained in query logs has been used in many different ways, for example to provide context during search, to classify queries, to infer search intent, to facilitate personalization etc. In various studies [8, 9], researchers and search engine operators have used information from query logs to learn about the search process and thus improve search engines.

Thorsten Joachims [5] proposed an approach to automatically optimizing the retrieval quality of search engines using click-through data stored in query logs and the log of links the users clicked on in the presented ranking. Taking a Support Vector Machine (SVM) approach, the approach presented a method for learning retrieval functions.

Besides learning about search engines or their users, query logs are also being used to infer semantic concepts [3, 6] or relations. Hao Ma et al. [6] suggested a query recommendation system by extracting the semantic relations between user submitted queries stored in query logs. They developed an effective and efficient two-level query suggestion model by mining clickthrough data, in the form of two bipartite graphs (user-query and query-URL bipartite graphs). Based on this, they proposed a joint matrix factorization method which utilizes two bipartite graphs to learn the low-rank query latent feature space, and then build a query similarity graph based on the features. After that, they designed an online ranking algorithm to propagate similarities on the query similarity graph, and finally recommended latent semantically relevant queries to users.

The search engines like Google generally return ranked documents in their search results. For ranking the Web pages, various algorithms have been introduced in the literature e.g. *PageRank* [10, 11] and *Hypertext Induced Topic Selection* (HITS) [12, 13], which are based on link-oriented approaches. Google uses *PageRank* Algorithm to calculate the rank score and combines the text matching scores at the query time to return the important and relevant pages. The PageRank formula can be defined as:

$$PR(u) = (1 - d) + d \sum_{v \in B(u)} \frac{PR(v)}{N_v} \quad (1)$$

where u represents a web page, $B(u)$ is the set of pages that point to u. $PR(u)$ and $PR(v)$ are rank scores of page u and v, respectively. N_v denotes the number of outgoing links of page v, d is a factor used for normalization that is usually set to 0.85. The ranking score PR is generally calculated independent of users' query words. Without considering the text matching score, relation between web pages and the requirement of a user could not be completely matched. Thus, the most relevant web pages to users' query words will not be shown at the top of the search result list. The work aims to improve the rank score of pages by considering the sequential order of pages accessed by the users.

The problem of mining sequential patterns of web pages has been an active area of research since last few years. It helps in finding the order in which the web pages get visited by the users. AprioriAll [14, 15] finds all patterns but it is a three-phase algorithm. It first finds all item-sets with minimum support, transforms the database so that each transaction is replaced by the set of all frequent itemsets contained in the transaction, and then finds sequential patterns. Empirical evaluation by the researchers indicates that Generalized Sequential Pattern (GSP) algorithm [15, 16] is much faster than the AprioriAll algorithm.

A critical look at the available literature indicates that from the very beginning, search engines are using some sort of optimization measures on their search results but the user is still posed to problems of finding the required information within the search results. The proposed method gives prime importance to the information needs of users and optimizes the rank values of returned web pages according to the sequential order of pages accessed by them.

III. THE PROPOSED OPTIMIZATION SYSTEM

A novel optimization method based on learning from historical query logs is proposed to predict user's information needs in a better way. The main aspect of the method is to perform query clustering based on user query keywords and their feedback. Then, the frequent sequential patterns of web pages visited by the users within each cluster are generated with the help of GSP (Generalized Sequential Patterns) algorithm [14]. The final approach is to re-rank the search result list by modifying the already assigned rank score of the web pages using the discovered sequential patterns. The rank updation improves the relevancy of the web pages which are already visited by some users in the past. By this way, the time user spends for seeking out the required information from search result list can be reduced and the more relevant Web pages can be presented.

The proposed architecture of search result optimization is shown in Fig. 1, which consists of following functional components:

1. Similarity Analyzer
2. Query Clustering Tool
3. Pattern Generator
4. Rank Updater

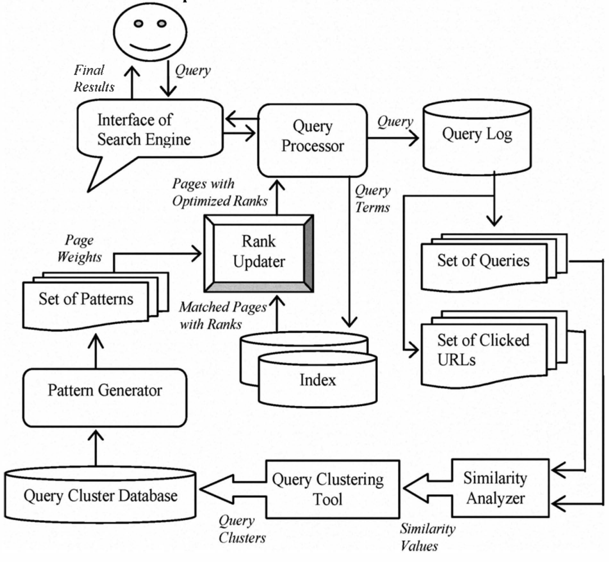

Fig. 1: Architecture of Result Optimization System

When user submits a query on the search engine interface, the query processor component matches the query terms with the index repository of the search engine and returns a list of matched documents in response. On the back end, result optimization system performs its task of gathering user intentions from the query logs. The user browsing behaviour including the submitted queries and clicked URLs get stored in the logs and are analyzed continuously by the *Similarity Analyzer* module, the output of which is forwarded to the *Query Clustering Tool* to generate potential groups of queries based on their similarities. The *Pattern Generator* module discovers sequential patterns of web pages in each cluster. The *Rank Updater* component works online and takes as input the matched documents retrieved by query processor. It improves the ranks of pages based on the weights assigned to each according to sequential patterns which were discovered offline.

The working and algorithms for different functional modules are explained below.

A. Query Similarity Analyzer

The approach taken by this module is based on two criteria: one is on the queries themselves, and the other on cross-references. These approaches are formulated below.

1. Similarity Based on Query Content Words

If two queries contain the same or similar terms, they denote the same or similar information needs. Following formula is used to measure the content similarity between two queries.

$$Sim_{keyword}(p,q) = \frac{KW(p,q)}{\max(kw(p),kw(q))} \tag{2}$$

where *kw(p)* and *kw(q)* are the number of keywords in the queries *p* and *q* respectively, *KW(p, q)* is the number of common keywords in two queries.

It is estimated that longer the query, the more reliable it is. However, as most of the user queries are short, this principle alone is not sufficient. Therefore, the second criterion is also used in combination as a complement.

2. Similarity Based on User Feedback

Two queries are considered similar if they lead to the selection of same or similar documents. Document selections are comparable to user relevance feedback in the traditional IR environment, except that document clicks denote implicit relevance and not always valid relevance judgments. The following formula dictates this similarity function.

$$Sim_{click}(p,q) = \frac{RD(p,q)}{\max(rd(p),rd(q))} \tag{3}$$

where *rd(p)* and *rd(q)* are the number of referred documents for two queries *p* and *q* respectively, *RD(p, q)* is the number of document clicks in common.

3) Combination of Multiple Measures

The two criteria have their own advantages. In using the first criterion, queries of similar compositions can be grouped together. In using the second criterion, benefit can be taken from user's judgements. Both query keywords and the corresponding document clicks can partially capture the users' interests when considered separately. Therefore, it is better to combine them in a single measure. A simple way to do it is to combine both measures linearly as follows:

$$Sim_{combined}(p,q) = \alpha{*}Sim_{keyword}(p,q) + \beta{*}Sim_{click}(p,q) \tag{4}$$

where α and ß are constants with 0≤α, ß≤1 and α+ß=1. There is an issue concerning the setting of these parameters and can be decided by the expert analysts of concerned domain. In the current implementation, these parameters are taken to be 0.5 each.

The notion of query similarity has several advantages. First, it is simple and easy to compute. On the other hand, it helps to find relating queries that happen to be worded differently but stem from the same information need. Therefore, semantic relationships of queries can be captured by (4).

B. *Query Clustering Tool*

For the clustering process, the *Query Clustering* module uses the algorithm shown in Fig. 2, where each run of the algorithm computes *k* clusters. As query logs are dynamic in nature, query clustering algorithm should be incremental in nature. Many clusters represent clearly defined intentions of search engine users.

The algorithm is based on the simple perspective: initially, all queries are considered to be unassigned to any cluster. Each unclassified query is examined against all other queries (whether classified or unclassified) by using (4). 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 same process is repeated until all queries get classified to any one of the clusters. The algorithm returns overlapped clusters i.e. a single query may span multiple clusters. Each returned cluster is stored in the *Query Cluster Database* along with the associated query keywords and the clicked URLs.

Let us take an example of particular query sessions from a query log shown in Table II. For calculating the query similarity, any one of the measures (2, 3) or the combined measure (4) can be used. The clusters obtained by using different measures are described below.

1. If the keyword-based measure (2) with threshold value 0.6 is applied, the queries are divided into 3 clusters C1, C2 and C3, where:

C1: Query 1
C2: Query 2
C3: Query 3 and query 4

TABLE II: AN EXAMPLE QUERY LOG FOR QUERY CLUSTERING

Sr. No.	Query	UserId	Clicked URL
1	Data Mining	1571911	http://www.A
		1571120	http://www.B
2	Data Warehousing	1572790	http://www.B
		1572300	http://www.A
3	Data Base System	1571100	http://www.P
		1571911	http://www.Q
		1573421	http://www.R
4	Data Base Management System	1571100	http://www.A
		1571202	http://www.Q
.....	....	....	....

Queries 1 and 2 are not clustered with any query.

2. If the clicked documents-based measure (3) is applied with threshold value set to 0.7, the clusters formed are:

C1: Query 1 and query 2
C2: Query 3
C3: Query 4

In this case, queries 3 and 4 are not judged to be similar.

3. Now let us use the combined measure (3), where α and ß are set to 0.5 and similarity threshold set to 0.54. The queries are now clustered in the desired way:

C1: Query 1 and query 2
C2: Query 3 and query 4

By analyzing the results obtained above through different approaches, it is determined that the combination of both keyword and clicked documents based approach is more appropriate for query clustering.

```
Algorithm: Query_Clustering(Q, α, β, τ)
Given: A set of n queries and corresponding
clicked URLs stored in an array $Q[q_i, URL_1,...,URL_m]$, $1 \le i \le n$
$\alpha = \beta = 0.5$
Minimum similarity threshold $\tau$
Output: A set C= {C1, C2,...Ck} 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 classified
For (each $p \in Q$ with ClusterId(p)=Null )
{
ClusterId(p)= Ck;
Ck={p};
For (each $q \in Q$ such that $p \neq q$ )
{
$Sim_{keyword}(p,q) = \frac{KW(p,q)}{\max(kw(p),kw(q))}$
$Sim_{click}(p,q) = \frac{RD(p,q)}{\max(rd(p),rd(q))}$
$sim_{combined}(p, q) = \alpha * Sim_{keyword}(p,q) + \beta * Sim_{click}(p,q)$

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

Fig. 2: Algorithm for Clustering the Queries

C. *Sequential Pattern Generator*

This component takes as input the query clusters and finds the sequential patterns in each cluster. To achieve this, modified version of GSP algorithm [10] is called. Modified_GSP shown in Fig. 3 makes multiple passes over the URL sequences stored in each query cluster. The set of queries in a particular cluster constitute the session set and there is a sequence of clicked URLs corresponding to each session (query). Given a set of frequent *n-1* length patterns of URLs, the candidate set for next generation are generated from input set according to minimum support thresholds.

Only frequent patterns in the current set are considered for generating the next candidate sequence.

For candidate generation, it merges pairs of frequent subsequences found in *(k-1)th* pass to generate candidate k-length sequences. A frequent *(k-1)*-sequence *w1* is merged with another frequent *(k-1)*-sequence *w2* to produce a candidate *k*-sequence if the subsequence obtained by removing the first event in *w1* is the same as the subsequence obtained by removing the last event in *w2* e.g. merging the page sequences w1=<{C} {D E} {F}> and w2 =<{D E} {F G}> will produce the candidate page sequence < {C} {D E} {F G}> because the last two events in w2 (F and G) belong to the same element.

```
Algorithm: Modified_GSP(QC, min_sup )
Given: A Query Cluster QC with a set of URLs/pages.
Threshold value min_sup // for pruning the candidates
Output: Set of frequent sequential Patterns
// Start of Algorithm
Result= ∅ // result set contains all sequential patterns
P1 = Set of Frequent 1-Length Page Sequence;
                    // Pages whose support ≥
        min_sup
k=2;
While (P(k-1) != Null)
{
Generate candidate sets Ck (set of candidate k-
sequences);
For (all page sequences S in the cluster QC)
{
        Increment count of all c ∈ Ck if S supports c;
        Pk = {c ∈ Ck | sup(c) ≥ min_sup}
k= k+1;
Result = Result ∪ Pk
}
```

Fig. 3: The Algorithm for Modified GSP

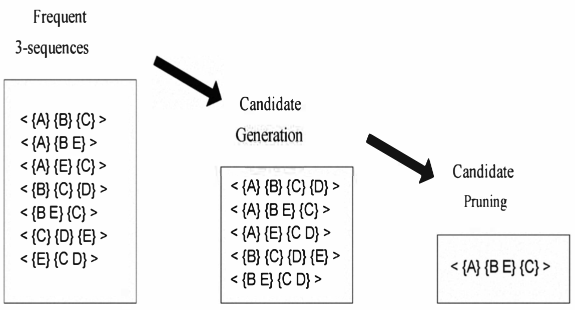

Fig. 4: An Example of Sequential Pattern Generation

A pruning phase eliminates subsets of infrequent patterns. For all patterns *P* in the candidate set with length *k*, all URL sequences are processed once and the count is incremented for each detected pattern in the candidate set. At each iteration, candidate *k*-sequences whose actual support is less than *min_sup* are eliminated by the module.

The example showing the pattern generation of web pages (say A, B, C, D and E) is given in Fig. 4 to better understand the algorithm. The figure shows the final iteration of candidate generation and pruning phase.

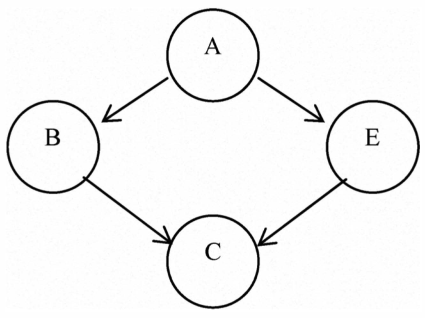

Fig. 5: Pictorial Representation of a Sequential Pattern

D. *Rank Updater*

This module takes its input from the query processor i.e. the matched documents of a user query and an update is applied to modify the rank score of the returned pages. The module operates online at the query time and applies the necessary updates on the concerned documents. The updated documents in question are those which are most frequently accessed by the users and are detected by the Sequential Pattern Generator. The updater works in the following steps:

Step 1: 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. This can be achieved using the formula given in (4).

Step 2: The sequential patterns of the concerned cluster are retrieved from the local repository maintained by the *Sequential Pattern Generator.*

Step 3: The level weights are calculated for every page *X* present in the sequential patterns.

Step 4: Final rank of a page *d* ∈ *D* 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.

By improving the ranks on the basis of user feedbacks, the results of a search engine can be optimized so as to better serve the user needs. The user can now find the popular and relevant pages upwards in the result list and need not to sift enough to find his desired ones.

1) *Weight Calculation*

The weight of the URL is estimated to be its popularity and order of access corresponding to the user query. Suppose a sequential pattern <{A}{B E}{C}> belongs to a query cluster matched with the user query. This pattern can be graphically presented as shown in Fig. 5.

The page (more precisely the URL) A lies at level 1, page B and E are on level 2, while C being at level 3. The weight of a page X is inversely proportional to its position in the sequential pattern and is calculated as:

$$Weight(X) = \frac{\ln(len_{pat(X)})}{level(X)} \quad (5)$$

where $len_{pat(X)}$ is the effective length/depth of the sequential pattern in which X occurs and $level(X)$ is the depth of X in the pattern. Considering the example pattern of Fig. 5, the weights of pages comes out to be:

Weight (A)= ln (3)/1= 1.099

Weight (B)= 0.549 etc.

2) *Rank Improvement*

The rank of a page can be improved with the help of its assigned weight. The new rank now becomes:

$$New\_Rank(X) = Rank(X) + Weight(X) \quad (6)$$

where *rank(X)* is the existing rank value (PageRank) of page X and *weight(X)* is the popularity given to X.

IV. Experimental Results

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

1. Keyword similarity ($Sim_{keyword}$),
2. Similarity using documents clicks (Sim_{click}),
3. Similarity using both keyword and document clicks ($Sim_{combined}$)
4. Query clustering
5. Modified GSP algorithm
6. Rank Updation

In the third and fifth function, α, β and τ are set to 0.5 and min_sup is taken to be 2.

1) *Similarity and Clustering Calculations*

Suppose we want to calculate the similarity between the first 2 queries. Let us say, q_1= *Maruti Swift Price* and q_2= *Maruti Swift Dzire*.

$Sim_{keyword}(q_1,q_2)=2/3$ and $Sim_{click}(q_1,q_2)=1/3$.

Finally, $Sim_{combined}(q_1,q_2)=(0.5)(2/3)+(0.5)(1/3)=1/2=0.5$

Table III: Sample Fragment of Log for experimental evaluation

UserId	Query	Clicked URL
1220051	Maruti Swift Price	www.marutiswift.com
		www.gaadi.com
		www.marutidzire.com
1220051	Maruti Swift Dzire	www.marutidzire.com
		www.cardekho.com
1220051	Maruti Swift	www.marutiswift.com
		www.gaadi.com
		www.carwale.com
1220051	Maruti Swift Dzire Price	www.marutiswift.com
		www.marutidzire.com
		www. carwale.com
		www.cardekho.com
1220051	Ray Ban sunglasses	www.ray-ban.com
		www.hisunglasses.com
1220051	Ray Ban sunglasses India	www.ray-ban.com
		www.emporiumonet.com
1220052	Maruti Swift Price	www.marutiswift.com
		www.gaadi.com
		www.carwale.com
1220052	Maruti Swift	auto.indiacar.com
1220052	Ray Ban sunglasses India price	www.eyeweartown.com
		www. ray-ban.com
		www.apparell.shop.ebay.in
1220053	Maruti Swift Price	www.marutiswift.com
		www.gaadi.com
		www.carwale.com
1220054	Maruti Swift Dzire Price	www.cardekho.com,
		www. marutisuzuki.com
		www.marutitruevalue.com
1220054	Maruti Swift	www.gaadi.com
		www.carwale.com
1220054	Maruti Swift Price	www.marutiswift.com
		www.marutisuzuki.com
		www.marutitruevalue.com
1220054	Ray Ban sunglasses	www.ray-ban.com
		www.apparell.shop.ebay.in
		www.emporiumonet.com
.....	....	....

Since the value of τ *is* 0.5, both are grouped in the same cluster C1 and stored in a Query Cluster Database with keywords {maruti, swift, dzire, price} and a set of URLs {www.marutidzire.com, marutiswift.com, gaadi.com, cardekho.com}. In the similar way, similarities for other queries can be determined and clustering can be performed. The final clusters obtained are: C1={q1,q2,q3,q4,q7,q8,q10, q11,q12,q13} and C2= {q5,q6,q9,q14}. By increasing the value of τ, more precise gruoping can be found.

2) *Pattern Generation*

For each cluster C1 and C2, corresponding URLs from the Query Cluster Database are extracted and sequential patterns are generated. Two columns (ClusterId and URL) need to be retrieved. To simplify the calculations, URLs are assigned to different variables. For example,

A=www.marutiswift.com,

B=www.gaadi.com,

C=www.marutidzire.com, and so on.

From the sample fragment, one pattern D→BE→A is found corresponding to C1.

3) *Weight Calculation and Rank Updation*

From the above result, a weight of each URL can be determined. Below are the weights of each URL in the pattern:

Weight(D)= 1.09, Weight(B)= 0.549

Weight(E)= 0.549, Weight(A)= 0.366

The Table IV shows the optimized rank values of the pages. The improved page rank is used only for the result presentation and does not affects the page rank stored in the search engine's repository.

It is evaluated from the above results that the ranking of search results has been modified to a great extent and the more relevant Web pages can be presented according to the above implementation.

TABLE IV: RANK MODIFICATION WITH WEIGHTS OF PAGES

Page /URL.	Previous Rank	Weight	New Rank
D	5	1.099	6.099
B	4	0.549	4.549
E	6	0.549	6.549
A	4	0.366	4.366
.....	....	....	....

V. CONCLUSION AND FUTURE SCOPE

A novel approach based on query log analysis is proposed for implementing effective web search. The most important feature is that the result optimization method is based on users' feedback, which determines the relevance between Web pages and user query words. The returned pages with improved page ranks are directly mapped to the user feedbacks and dictate higher relevance than pages that exist in the result list but are never accessed by the user. The results obtained from practical evaluation are quite promising in respect to reduced search space and improved effectiveness of interactive web search engines.

As the proposed approach demonstrates quite effective results, further investigation on mining log data deserves more of our attention. Further study may result in more advanced mining mechanism which can provide more comprehensive information about the relevancy of the query terms and allow identifying user's information need more effectively. Bipartite graph techniques can be employed on query logs to retrieve a better clustering of user queries and thus returning more efficient results.

REFERENCES

[1] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, and S. Raghavan, "Searching the Web," ACM Transactions on Internet Technology, Vol. 1, No. 1, pp. 97–101, 2001.

[2] A. Borchers, J. Herlocker, J. Konstanand, and J. Riedl, "Ganging up on information overload," Computer, Vol. 31, No. 4, pp. 106-108, 1998.

[3] Edgar Meij, Marc Bron, Bouke Huurnink, Laura Hollink, and Maarten de Rijke, "Learning semantic query suggestions". In 8th International Semantic Web Conference (ISWC 2009), Springer, October 2009.

[4] J. Wen, J. Mie, and H. Zhang, "Clustering user queries of a search engine". In Proc. of 10th International World Wide Web Conference. W3C, 2001.

[5] Thorsten Joachims, "Optimizing search engines using clickthrough data". Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining, 2002, pp: 133–142, New York.

[6] H. Ma, H. Yang, I. King, and M. R. Lyu, "Learning latent semantic relations from clickthrough data for query suggestion". In CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management, pages 709–718, New York, NY, USA, 2008, ACM.

[7] Isak Taksa, Sarah Zelikovitz, Amanda Spink, "Using Web Search Logs to Identify Query Classification Terms". Proceedings of Fourth International Conference on Information Technology (ITNG'07), pp: 469–474, 2007.

[8] K. Hofmann, M. de Rijke, B. Huurnink, E. Meij, "Semantic Perspective on Query Log Analysis". In Proceedings of CLEF '09 Workshop, 2009, Corfu, Greece.

[9] Z. Zhuang, S. Cucerzan, "Exploiting Semantic query Context to Improve Search Ranking". Proceedings of IEEE International Conference on Semantic Computing, pp: 50–57, 2008.

[10] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine". In Proc. of Computer N/ws and ISDN Systems, pp: 107–117, 1998.

[11] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the Web". Technical report, Stanford Digital Libraries SIDL-WP-1999–0120, 1999.

[12] B. Amento, L. Terveen, and W. Hill, "Does Authority Mean Quality? Predicting Expert Quality Ratings of Web Documents". In Proceedings of 23th International ACM SIGIR, pp. 296–303, 2000.

[13] S. Chakrabarti, B.E. Dom, S.R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson, and J. Kleinberg, "Mining the Web's link structure". *Computer*, 32(8):60–67, 1999.

[14] R. Agrawal and R. Srikant, "Mining sequential patterns". Proc. of the 11th International Conference on Data Engineering (ICDE'95), pp. 3–14, 1995.

[15] Murat Ali Bayir, Ismail H. Toroslu, Ahmet Cosar, "A Performance Comparison of Pattern Discovery Methods on Web Log Data". Proceedings of 4th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA-06)", pp. 445–451, 2006.

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