Query Recommendation for Optimizing the Search Engine Results

Nikita Taneja
Assistant Professor
C.S.E Department
Manav Rachna College of engineering, Faridabad

Rachna Chaudhary
M.Tech (Pursuing)
Manav Rachna College of Engineering, Faridabad

ABSTRACT

Search engine query have been paying awareness in current days. Since web contents develop, the importance of search engines became more essential and at the same instance user performance reduces. Query recommendation is a method to improve search results in web. This paper presents a method for mining search engine query logs to obtain fast query recommendation on a large scale. Search engines generally return long list of ranked pages, finding the important information related to a particular topic is becoming increasingly difficult and therefore, optimized search engines become one of the most popular solution available. In this work, an algorithm has been applied to recommend related queries to a query submitted by user. For this, the technology used for allowing query recommendations is query log which contains attributes like query name, clicked URL, rank, time. Then, the similarity based on keywords as well as clicked URL's is calculated. Additionally, clusters have been obtained by combining the similarities of both keywords and clicked URL's to perform query clustering and further the sequential order of clicked URLs in each cluster has been discovered using the modified version of an existing sequential pattern mining technique. The final by product is further optimized by re-ranking the pages using the discovered sequential patterns. The proposed system here, is based on learning from query logs that predicts user information requirements and reduces the seek time of the user within the search result list.

General Terms

Algorithms, Performance

Keywords

World Wide Web, Search Engine, Query Log, Query Clustering, Rank Updater.

1. INTRODUCTION

With the development in information technology, the Web [8] has turn out to be a vast information repository covering almost every area, in which a human user could be involved. On the other hand, with the overwhelming volume of information on the Web, the task of finding important information related to a particular topic is becoming increasingly difficult. Many advanced Web searching techniques have been recently developed to deal with this problem and are being used in the commercial Web search engines such as Google and Yahoo. In Spite [2] of the current advances in the Web search engine technologies; there are still various situations during which the user is presented with non relevant search results. One of the major reasons for this difficulty is the need of user awareness in framing the queries. The search engine users are not well skilled in organizing and formulating their input queries, on which the search engine relies to find the required search results. Furthermore, search engines often have difficulties in forming a concise and precise representation of the user's information need. At the current time, providing a set of web pages based on user query words is not a big deal in search engines [8]. Instead, the difficulty is that a search engine returns a huge number of web pages back to user queries and users have to waste a lot of time in finding their preferred content. This difficulty is referred to as the Information Overkill problem [12].

While various search engines apply ranking, clustering and other web mining methodologies to optimize their search results, there still remains a challenge in providing the user essential content with less navigation overhead. Search engines are the necessity to locate the users' interests by means of their queries and then optimized the results in the form of query logs. Query logs offer a tremendous opportunity in support of gaining insight into how a search engine is used and what the users' interests are. While they form a complete record of what users searched for in a particular time frame. Nowadays, several Web applications are applying Web usage mining [2] techniques to predict users' navigational activities by automatically discovering the access patterns while one or more log files, but none have used them for search engine's result optimization.

The work proposed in this paper aims to optimized the results of a search engine by returning the more relevant and user most wanted pages on the top of search result list. To achieve the required task, the approach pre-mines the query logs to retrieve the potential clusters of queries followed by finding the most popular queries in all clusters. Every cluster entries are again mined to take out sequential patterns of pages accessed by the users. The outputs of both mining processes are utilized to return relevant pages to the user while recommending him with popular historical queries.

The paper has been organized as follows: Section 2 we describes the approaches that are already in use on the basis of proposed work and literature work that had been done. Section 3 explains a novel architecture of proposed optimization system in detail along with algorithms. Section 4 concludes the paper with some discussion on future research.

2. RELATED WORK

A good number of existing works have suggested methods in utilizing Web search engine query logs used for mining related queries. Baeza-Yates [5] has done a study to make the use of Web logs to develop different aspects of search engines. Wen et al. [9] suggested a technique for clustering the similar queries to recommend URLs to frequently asked queries of a search engine. They use four ideas of query distance: (1) based on keywords (2) based on string matching of keywords; (3) based on common clicked URLs; and (4) based on the distance of the clicked documents. Beeferman and Berger [11] also propose a query clustering technique based on distance ideas (3). Fonseca

[6] offered a new way to determine the related queries based on association rules .The queries represent items in tradition association rules. The query log file is viewed as set transactions that represent a session at which the user submit all related queries in a particular time. Their idea of query session is similar than the idea we use in this paper. The method showed fine results, however arising of two problems.

The primary problem is the difficulty in determining which sessions of these queries belongs to the similar search process. The next problem the most interesting related queries which are submitted by different users can't be presently discovered. For the reason that the support of a rule increases merely if its queries are in the similar session and they must be submitted by the similar user. M.Hosseini and H.Abolhassni have described a method used for recommending associated queries according to clustering process over web queries from search engines query log. Zaiane and Strilets [7] presented a method for recommending queries according to seven aspects of query similarity, three of them are moderated variations of the first and second ideas. In addition our method recommends the related queries in the direction of the input query although my search for different issues similar to the previous information from query log file.

There is one more approach to recommend related queries by means of query expansion. The researchers demonstrate that average query terms are near two [13]. As a result most of the time, queries are uncertain. One promising solution for this difficulty is to expand a query with latest terms. Query clustering helps to find appropriate terms for this expansion.

A significant look at the available literature indicates that, search engines are using some sort of optimization procedures on their search results but the user is still posed to problems of finding the vital information within the search results. The proposed method gives main importance to the information needs of users and optimizes the rank value of returned web pages according to the sequential order of pages accessed by them.

3. PROPOSED WORK

The proposed optimization system (Fig. 1) lying on learning from historical query logs is proposed to calculate user's information requirements in a better way. The proposed system works as follow. The prime feature of the system is to perform query clustering by finding the similarities between the two queries, which is based on user query keywords and clicked URLs. After that, the frequent sequential patterns of web pages visited by the users in each cluster are generated with the help of Generalized Sequential Patterns algorithm [14]. The final approach is to re-rank the search result list by modifying the previously assigned rank score of the web pages using the discovered sequential patterns. The rank updation enhances the relevancy of the web pages based on its access history. By this method, the time user spends looking for the required information from search result list can be reduced and the more relevant Web pages can be obtained.

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

1. Query Log

2. Query Similarity

3. Query Clustering Tool

4. Sequential Pattern Generator

5. Rank Updater

When user submits a query (Fig. 1) on the interface of search engine, the query processor component matches the query terms with the index repository of the search engine and taking a list of matched documents in reply. On the reverse order, result optimization system performs its task of gathering user intentions from the query logs. The user browsing behavior as well as the submitted queries and clicked URLs get stored in the logs and are analyzed continuously by the Query Similarity module, the output of which is forwarded to the Query Clustering Tool to create potential groups of queries based on their similarities. The Pattern Generator module discovers sequential patterns of web pages in every cluster. The Rank Updater component takes as input the matched documents retrieved by query processor. It improves the ranks of pages to all according to sequential patterns with optimized rank get stored in the interface of search engine which produces final results to user..

The working and algorithms for different functional modules are explained below in the next subsections.

3.1 Query Logs

Query log has been a popular data source for query recommendation. The view of Web Log Mining has been an area of interest as many years. In the context of search engine the typical logs [3] of search engines include the following entries: (A) User IDs, (B) Query q issued by the user, (C) URL u selected by the user (D) Rank r of the URL u clicked for the query q and (E) 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 1: Example Illustration of Query Log

ID	Query	Clicked URL	Rank	Time
Admin	Data Mining	www.dming.com	6	00:01:10
Admin	Data Ware housing	www.dming.com	5	00:01:10
Admin	Data Mining	www.google.com	5	00:01:16
Admin	Data Wareho using	www.datawarehousing.co	7	00:01: 16
Admin	Search Engine	www.dming.com	6	00:01: 16
Admin	Web Crawler	www.crawler.com	5	00:01:16

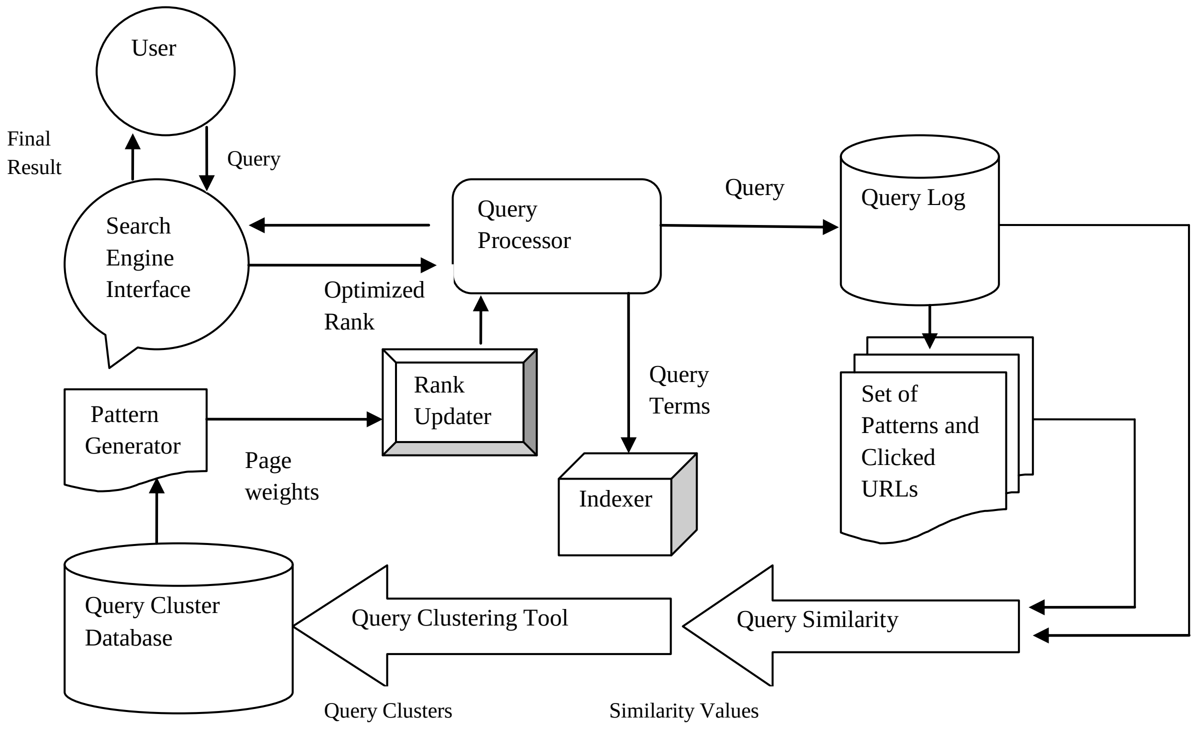

Fig 1: Architecture of Proposed Optimization System

3.2 Query Similarity

The next step in proposed system is computing the query similarity. It is an important crisis and has a wide range of applications in Information Retrieval in query recommendation. Traditional approaches [16] make the use of keywords extracted from documents. If two documents share some keywords, then they are thought to be similar to some extent. The more they share common keywords, and the more these common keywords are important, the higher their similarity is. This similar approach may also apply to query clustering, when a query may also be represented as a set of keywords in the same way as a document. On the other hand, it is well known that clustering using keywords has some drawbacks, due to the fact that keywords and meanings do not strictly correspond. The same keyword does not always represent the same information need and different keywords may refer to the same concept. As a result, the calculated similarity between two similar queries may be small, while two unrelated queries may be considered similar.

The approach taken by this module is based on two criteria: one is on the queries keywords, and the other on clicked URLs. These approaches are formulated below:

3.2.1 Similarity based on Query Keywords

If two queries contain the similar terms, they denote the same information needs. The below mentioned formula is used to measure the similarity between two queries.

$$\mathrm{Sim}_{\mathrm{keyword}}(x,y) = \frac{KW(x,y)}{\max\,(kW(x),kW(y))} \quad (1)$$

Where kW(x) and kW(y) are the number of keywords in the queries x and y respectively, KW(x, y) is the number of common keywords in two queries.

It is noted that longer the query, the more reliable it be. Though most of the user queries are short, this principle alone is not enough. Consequently, the second criterion is also used in combination as a complement.

3.2.2 Similarity based on Clicked URLs

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

$$\mathrm{Sim}_{\mathrm{clickURL}}(x,y) = \frac{RD(x,y)}{\max\,(rd(x),rd(y))} \quad (2)$$

Where rd(x) and rd(y) are the number of referred documents for two queries x and y respectively, RD(x, y) is the number of document clicks in common.

Now, previous work on query similarity aims to give a single similarity measure without knowing the information that queries are indefinite and generally have several search intents. By introducing search intents into the calculation of query similarity, we can get more exact and also useful Query similarity based on keywords as well as Clicked URLs as shown in fig 2.

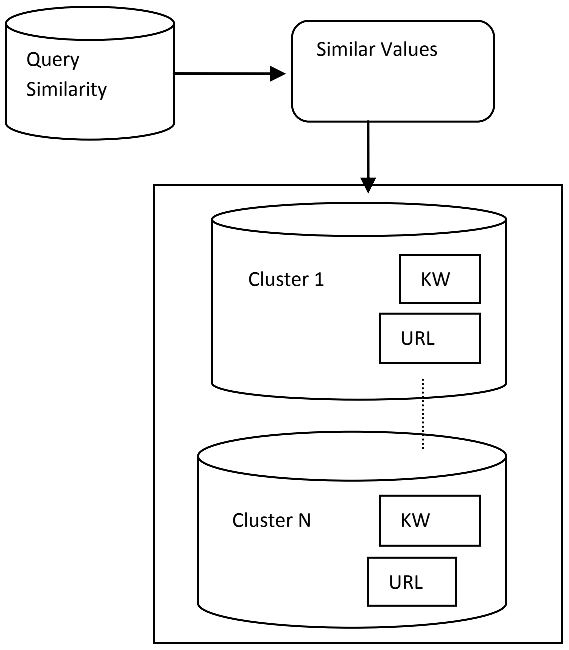

Fig 2. Query Similarity (KW: Keyword)

3.2.3 Combined Similarity Measure

The two criteria [1] have their own advantages. In using the first criterion, queries of similar compositions can be grouped together. In using the second criterion, advantage can be taken from user's judgments. Both query keywords and the corresponding clicks URLs can partially capture the users' interests when considered individually. For that reason, it is better to combine them in a particular measure. An easy way to perform is to combine both measures linearly as follows:

$$Sim_{combines(x,y)} = \alpha.\ Sim_{Keyword(x,y)} + \beta.\ Sim_{clickURL(x,y)} \quad (3)$$

Where α and β are constants with 0<=α (and β) <=1 and α + β=1

There is a question concerning the setting of these parameters and that can be decided by the specialist of concerned domain. In the present implementation, [1] these parameters are taken to be 0.5 each.

Similarity Calculations (Example)

Suppose we want to calculate the similarity based on queries keywords between the first 2 queries (Table 1). Let us say, ql= Data Mining and q2= Data Ware housing

Sim (q1, q2) =1/5=0.2

Similarly, the similarity between first and third queries are Q1=Data Mining and Q3=Data Mining

Sim (q1, q3) =2/4=0.5

Now, suppose we want to calculate the similarity based on clicked URLs between the first 2 queries.

ql= Data Mining and q2= Data Ware housing

$$Sim_{clickURL(q1,q2)} = \frac{6+5}{11+16} = 0.4$$

$$Sim_{combined} = (0.5)(0.2) + (0.5)(0.4) = 0.3$$

3.3 Query Clustering Tool

In support of the clustering process, the Query Clustering module uses the algorithm, where each run of the algorithm computes k clusters. As query logs are dynamic in nature, query clustering [2] algorithm should be incremental in nature.

The algorithm is based on the easy [2] perspective: firstly, all queries are considered to be unassigned to any cluster. Each query is examined against all other queries whether classified or unclassified by using (3). If the similarity value turns out to be greater than the pre-specified threshold value (T), then the queries are grouped into the same cluster. The similar 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. Every returned cluster is stored in the query cluster database along with the related query keywords and the clicked URLs. The clustering algorithm takes O (n^2) worst [2] case time to find all the query clusters, where n is the total number of queries.

Algorithm for clustering queries

```
Algorithm: Query Clustering (Q, α, β, τ)
Given: A set of n queries and corresponding clicked URL's stored in an array
Q [q1, URL1…..URL m] 1<=I <=n
α=β=0.5
Similarity Threshold τ
Output: A set C= {C1, C2….Ck} of k query clusters
//Start Algorithm
K=0;                              // k is the number of clusters
For (each query x in Q)
Set Cluster Id (x) = Null;        //Initially No query is clustered
For (each x € Q)
{
Cluster Id (x) = Ck;
Ck = {x};
For each b € Q such that x ≠ y
{
$Sim(x, y) = \frac{KW(x,y)}{max\ (kW(x),kW(y))}$

$Sim_{clickURL(a,b)} = \frac{RD\ (x,y)}{max\ (rd(x),rd(y))}$
$Sim_{combines(x,y)} = \alpha.\ Sim_{Keyword(x,y)} + \beta.\ Sim_{clickURL(x,y)}$

If ($Sim_{combines(b,a)} > \tau$)

Set Cluster Id (y) = Ck;
Ck = Ck U {y};
Else
Continue;
} // End For
K=K+1;
} //End Outer For
Return Query Cluster Set C;
```

Let us take an example of particular query sessions from a query log shown in Table 2. For calculating the query similarity, any one of the measures (1, 2) or the combined measure (3) can be used. The clusters obtained by using different measures are described below.

1. If the keyword-based similarity measure (1) between first and third queries with threshold value 0.5 is applied, the queries is divided into 1 cluster only

C1: Query 1 (Data Mining)

2. If the clicked URLs based similarity measure (2) between first and third queries is applied with threshold value set to 0.5, the clusters is formed:

C 1: Query 1(Data Mining)

3. Now let us make use of the combined similarity measure (3) between first and third queries where α and β are set to 0.5 and similarity threshold set to 0.5. The queries are now clustered in the desired way:

C I: Query 1 and Query 3(Data Mining)

In this case, queries 1 and 3 are judged to be similar but only 1 common cluster is formed from different queries.

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 as shown in table 2.

Table. 2 an Example of Query Log for Query Clustering

S.No	Query	User ID	Clicked URLs
1	Data Mining	Admin	www.dming.com
			www.google.com
2	Data Ware Housing	Admin	www.dming.com
3	Data Warehousing.com	Admin	www.datawarehousing.com

3.4 Sequential Pattern Generator

This component takes as input the query clusters and finds the sequential patterns in each cluster. Sequential pattern generator which discovers frequent [4] subsequences as patterns in a sequence database is an important data mining problem with broad applications.

It is based on their study of customer purchase sequences as follows: Consider a database of sequences, where each sequence is a list of transactions ordered by transaction-time, and every transaction is a set of items. The difficulty is to discover all sequential patterns with a user-specified minimum support, where the support of a pattern is the number of data-sequences that contain the pattern. Generalized Sequential Patterns is an algorithm that discovers all sequential patterns.

Generalized Sequential Pattern algorithm [4] makes multiple passes over the data. The first pass determines the support of each item, i.e. the number of data-sequences that include the item. At the end of the first pass, the algorithm knows which items are frequent, that is, have minimum support. Each such item yields a 1-element frequent sequence consisting of that item. Each subsequent pass starts with a seed set, the frequent sequences found in the earlier pass.

The seed set is used to generate new potentially frequent sequences, called candidate sequences. Each candidate sequence has one more item than a seed sequence so all the candidate sequences in a pass will have the same number of items. The support for these candidate sequences is found during the pass over the data. At the end of the pass, the algorithm determines which of the candidate sequences are really frequent. These frequent candidates become the seed for the next pass. The algorithm terminates when there are no frequent sequences at the end of a pass or when there are no candidate sequences generated.

Algorithm for Generalized Sequential Pattern

```
Pattern list 1: = Frequent atoms;
For k=2;
Pattern list k! = empty; k = k + 1
Ck = set of candidate
For all sessions in S do
Increment count of X ∈ Ck contained in S
End For
Pattern list k = {X ∈ Ck / support(X) > threshold}
End For
Set of all frequent Patterns: = ∪k Pattern list k
```

Generalized Sequential Pattern [14] finds every item sets with minimum support, transforms the database therefore each transaction is replaced by the set of all frequent item sets contained in the transaction and then finds the sequential patterns.

There are two problems with this approach. First, it is computationally costly to do the data transformation on the fly throughout pass while finding sequential patterns. Another is to transform the database once and store the transformed database, will be unrealistic for many applications. Second, while it is possible to expand this algorithm to handle time constraints and taxonomies. For the cases that the modified Generalized Sequential Pattern can handle, our experimental estimation finds that modified [14] Generalized Sequential Pattern is up to 20 times quicker.

Modified version of Generalized Sequential Pattern algorithm called as Modified_GSP [14] 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 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) the 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=<{A} {B} {C}> and w2 =< {A B} {C}> will produce the candidate page sequence <{A B} {C} {D}> because the first two events in w2 (A and B) belong to the same element.

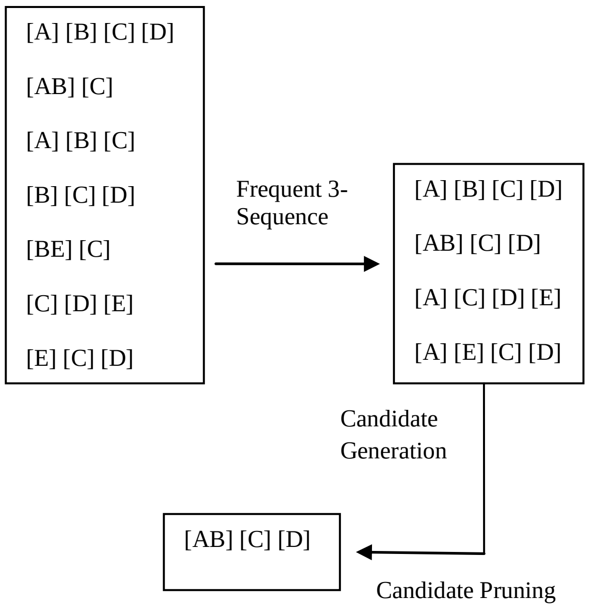

Fig. 3: Pictorial Representation of Sequential Pattern

Candidate Generation-It defines that [15] candidate's sequences are generated before the pass begins. We want to generate as few candidates as possible while maintaining completeness.

Candidate Pruning- A pruning phase [15] delete candidate sequences that have a contiguous k-1subsequence whose support counts is less than the minimum support. If there is no max-gap constraint, we also delete candidate sequences that have any subsequence without minimum support.

Algorithm for Modified Generate Sequential Pattern

```
Algorithm: Modified_GSP (QC, min_sup)
Given: A Query Cluster with a set of URLs
Threshold Value min sups          //for pruning the candidates
Output: Set of frequent Sequential Patterns
//Start of Algorithm
Result = ф        // 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 if S supports c;
K=k+1;
Result = Result U Pk
}//end While
Return Result;
```

3.5 Rank Updater

Rank Updater module [1] 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 needed updates on the concerned documents. The updated documents are those which are most often 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.
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 is computed if it happens to be present in the patterns of cluster C. The improved rank is calculated as the summing 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 requirements. At this time user can find the popular and relevant pages upwards in the result list.

The methods to find the weight and improved rank of a page are as follows:

A. Weight Calculation

The weight of the URL determined its popularity and order of access related to the user query. Suppose a sequential pattern

< {AB} {C} {E}> belongs to a query cluster matched with the user query. This pattern can be obtainable as shown in

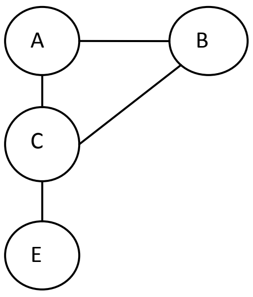

Fig 4: Pictorial Representation of Sequential Pattern

The page A as well as B lies at level 1, page C is at level 2, while D being at level 3. The weight of a page X is inversely proportional to its position in the sequential pattern and is calculated as:

$$\text{Weight}(X) = \frac{\ln(\text{len}_{\text{path}(X)})}{\text{level}(X)}$$

Where Len pat(X) is the effective depth of the sequential pattern in which X occurs and level(X) is the depth of X in the pattern.

B. Rank Improvement

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

New Rank(X) =Previous Rank(X) +Weight(X)

Where rank(X) is the existing rank value (Page Rank) of page X and weight(X) is the popularity given to X.

Weight Calculation and Rank Updation
From the above result (as shown in Fig. 3), a Sequential Pattern is generated as AB$\rightarrow$ C $\rightarrow$D weight of each URL can be determined as:
Weight (A) =1.09
Weight (B) =1.09
Weight (C) =0.54
Weight (D) =0.36

Table.3 shows the optimized the rank values and the new rank is used for the result presentation.

Table 3: Rank Optimization with weight of pages

Page	Previous Rank	Weight	New Rank
A	1.09	6	7.09
B	1.09	5	6.09
C	0.54	5	5.54
D	0.36	7	7.36

4. CONCLUSION AND FUTURE WORK

In this paper, a novel approach of proposed optimization system which is based on query log is proposed for implementing useful web search. The most vital part is that the proposed optimization method is based on user's feedback that determines the significance between Web pages and user query words. As result improvement is based on study of query logs, the recommendations and the returned pages with better page ranks are directly mapped to the user feedbacks and dictate higher relevance than pages which exist in the result list but are never accessed by the user. Hence, the time user spends for seeking out the necessary information from search result list can be reduced and the more important Web pages can be presented. The outcome obtained from practical estimation is fairly capable in respect to reduced search space and enhanced efficiency of interactive web search engines.
At the same time the proposed approach demonstrates fairly efficient results. In additional investigation on mining log data deserves need more attention. More study may result in further advanced mining mechanism which can give more comprehensive information about the relevancy of the query terms and allow identifying user's information require more effectively. Future Work includes applying a technique to overcome this problem.

5. ACKNOWLEDGMENT

This paper would not have been possible without my professor **Mrs. Nikita Taneja**. My deepest gratitude goes to her for the support and guidance. Her patience and encouragement throughout the research paper has motivated me to enthusiastically work and constantly improve. I am honored and thankful for the opportunity to be her student.

It is a great pleasure to thank my guide for being there for me always. Family values that I have gained directed me to the path of constant personal, professional and academic progress. My family has been supporting me in all mentioned aspects. I am very glad that I can share my success with them.

6. REFERENCES

[1] A.K Sharma, Neelam Duhan, Neha Aggarwal, Ranjana Gupta. Web Search Result optimization by mining the Search Engine Query Logs. Proc. of International Conference on methods and models in Computer Science, Delhi, India, Dec.13-14, 2010.

[2] Neelam Duhan, A.K Sharma."Rank Optimization and Query Recommendation in Search Engine using Web Log Mining Technique. Journal of computing. Vol 2, Issue 12, Dec. 2010.

[3] Edgar Meij, Marc Bron, Bouke Huurnink, Laura Hollink, and Maarten de Rijke. Learning Semantic query Suggestions. In 8th International semantic Web Conference, Springer, Oct. 2009.

[4] Murat Ali Bayer, Ismail H. Toroslu, Ahmet Cosar. A Performance comparison of Pattern discovery methods on web log data. Proceedings of AICCSA, pp 445-451. 2006.

[5] R. Baeza-Yates, Web Usage Mining in Search Engines." Web mining: applications and techniques, Anthony scime, Editor, Idea Group. 2004.

[6] B.M Fonseca, P.B Golger, E.S. De. Moura and N.Ziviani. " Using Association rules to discover search engines relating queries". In first Latin American web Congress, November, 2003.

[7] O.R Zaine and A. Strilets. "Finding similar queries to satisfy searches based on query traces. In proceedings of the international workshops on efficient web based information system, France Sept. 2002.

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

[9] J.Wen Nie and H.Zhang, Clustering user queries of a Search Engine. In Proceedings at 10th international World Wide Web Conference, pp 162-168, W3C, 2001.

[10] J.XY and W.B. Croft. "improving the effectiveness of information retrieval with the local context analysis. ACM Transaction of information system, 79-112, 2000.

[11] D.Beeferman and A. Berger. Agglomerative Clustering of a Search Engine Query log. In KDD, pages 407-416, Boston, MA USA, 2000.

[12] A. Borchers, J. Herlocker, J. Konstand, and J.Riedl,"Ganging up on Information Overload", Computer, Vol.31, No.4, pp.106-108, 1998.

[13] D.Jansen, A. Spink, J.Bateman and T.Saracivic. Real Life Information Retrieval: a study of user queries on the web".ACM SIGIR Forum, PP 5-17, 1998.

[14] Srikant R.and Aggarwal R. Mining Sequential Pattern: Generalizations and Performance Improvements .Proc of 5th International extending database technology, France March, 1996.

[15] R.Agrawal and R.Srikant, Mining Sequential Pattern. Proc of 11th International Conference data engineering, pp 3-14, 1995.

[16] Salton, G. and McGill, M.J. Introduction to modern information retrieval. McGraw hill-book Company, 1983.