QUERY RECOMMENDATION APPROACH FOR SEARCHING DATABASE USING SEARCH ENGINE

Geetanjli Gaur*

Ashish Oberoi*

Mamta Oberoi**

ABSTRACT

Search Engines generally return long lists of ranked pages, finding the desired information content from which is typical on the user end and therefore, Search Result Optimization techniques come into play. The proposed system based on learning from query logs predicts user information needs and reduces the seek time of the user within the search result list.

To achieve this, the method first mines the logs using a similarity function to perform query clustering and then discovers the sequential order of clicked URLs in each cluster . Finally, search result list is optimized by re-ranking the pages. The proposed system proves to be efficient as the user desired relevant pages occupy their places earlier in the result list and thus reducing the search space. This thesis also presents a query recommendation scheme towards better information retrieval.

Keywords: *World Wide Web (WWW), Information Retrieval, Search Engine, Query processing*

*Department of Computer Engineering, M.M. University, Mullana, Ambala, Haryana, India

**Department of Statistics, MLN College, Yamuna Nagar, Haryana, India

1. INTRODUCTION / LITERATURE SURVEY

The Query Recommendation provides an excellent opportunity for gaining insight into how a search engine is used and what the users' interests are. Query Logs prove to be important information repositories to keep track of user activities through the search results, knowledge about which can improve the performance of a search engine. In spite of the recent advances in the Web search engine technologies; there are still many situations, in which user is presented with undesired and non-relevant pages in the top most results of the ranked list. One of the major reasons for this problem is the lack of user knowledge in framing queries. Moreover, search engines often have difficulties in forming a concise and precise representation of the response pages corresponding to a user query. Nowadays, providing a set of web pages based on user query words is not a big problem in search engines. Instead, the problem arises at the user end as he has to sift through the long result list, to find his desired content. This problem is referred to as the Information Overkill problem. Search engines must have a mechanism to find the users' interests with respect to their queries and then optimize the results correspondingly. To achieve this, query log files maintained by the search engines play an important role. The logs provide an excellent opportunity for gaining insight into how a search engine is used and what the users' interests are.

The goal of this approach describes the various terms & approaches that are used in optimization of web search and provide an overview of framework used in the field of web mining

1.1 TERMS USED IN WEB MINING

Various terms that are commonly used in web mining are:

QUERY LOGS

The log keeps users' queries and their clicks, as well as their browsing activities. In the context of search engines, servers record an entry in the log for every single access they get corresponding to a query. The typical logs search engines include the following entries:

1) User (session) IDs,
2) Query q issued by the user
3) URL u accessed/clicked 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.

2. Proposed Model

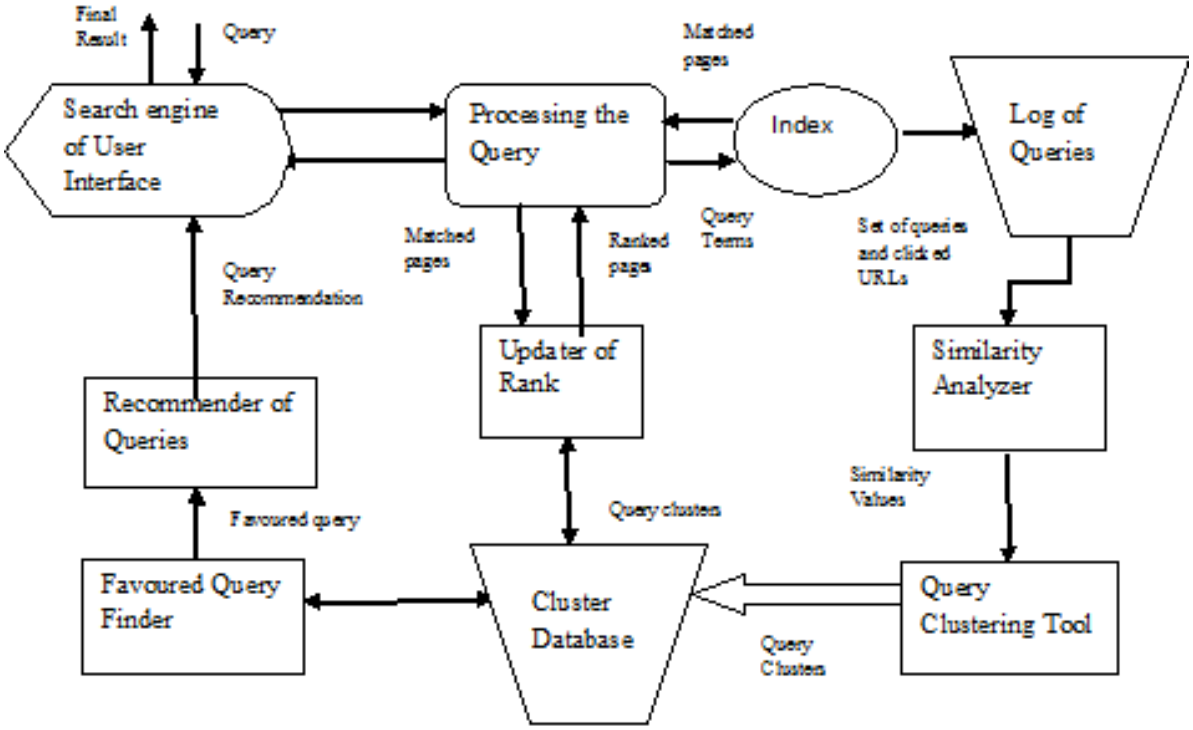

Fig.1 Proposed model

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. User browsing behavior 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 groups of queries based on their similarities.

Favored Query Finder extracts most popular queries from each cluster and stores them for future reference. The Rank Updater component works online and takes as input the matched documents retrieved by query processor. The Query Recommender guides the user with similar queries with the most famous query.

The proposed system works in the following steps

1. Similarity Analyzer
2. Query Clustering Tool
3. Favored Query Finder
4. Updater of Rank
5. Recommender of Query

3. EXPERIMENTAL RESULTS

To show the validity of the proposed architecture, a fragment of sample query log is considered (given in Table 1). Because the actual number of queries is too large to conduct detailed evaluation, only 7 query sessions are chosen in present illustration. The following functions are tested on the 7 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. Updater of Rank

Table 1. Simple Query Log

s.no	Id	User_id	Query	Clicked_id
1	2	admin	Data mining	http://www.A
2	3	admin	Data ware housing	http://www.B
3	4	admin	Data mining	http://www.B
4	5	admin	Data warehousing	http://www.A
5	6	admin	Search engine	http://www.B
6	14	admin	Database	http://www.B
7	15	admin	Data base	http://www.A

3.1 SIMILARITY AND CLUSTERING CALCULATIONS

Query Similarity Analyzer

The approach taken by this module is based on two principles:

1) Similarity based on the queries themselves and

2) Based on cross-references.

3.1.1 Similarity based on query keywords

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

$$Sim(p,q) = \frac{|KW(p,q)|}{|kw(p) \cup kw(q)|}$$

Where kw (p) and kw (q) are the sets of keywords in the queries p and q respectively, KW (p, q) is the set 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 used in combination as a complement.

3.1.2 Similarity Based On Clicked URLs

A query vertex is joined with a document vertex if document has been accessed by a user corresponding to the said query. The numerical integer on each edge dictates the number of accesses to the document by distinct users for a particular query. For example a value 10 between Q1 and D1 says that 10 users have clicked on D1 corresponding to Q1. In the figure above: D1, D2, D4 are accessed with respect to Q1, thus are relevant to Q1 and D2, D3, D4 are relevant to Q2 and so on. As Q1 and Q2 share two documents D2 and D4, they can be considered similar but similarity is decided on the basis of number of document clicks.

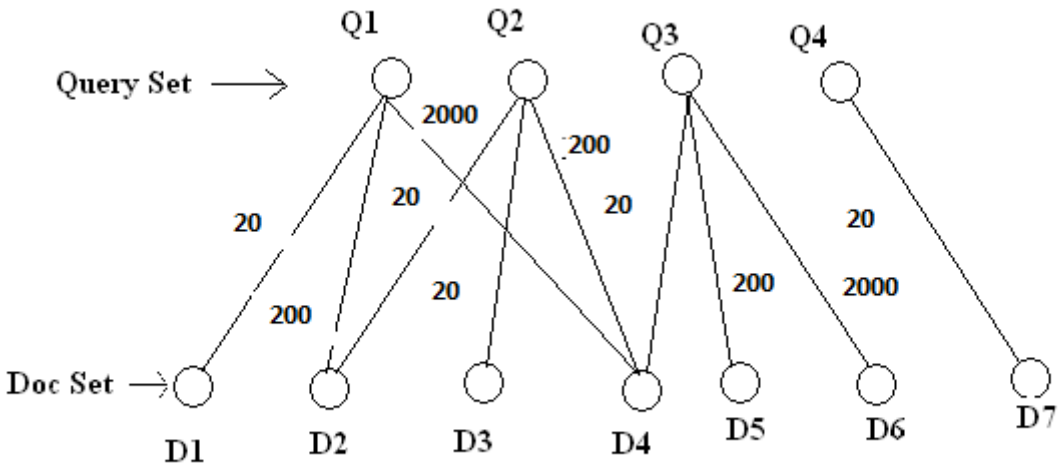

Fig. 2 Similarity Based on Clicked URLs

3.1.3 Biparite Graph of Query Log

If two queries p and q share a common document d, then similarity value is ratio of the total number of distinct clicks on d with respect to both queries and the total number of distinct clicks on all the documents accessed for both queries. If more than one document is shared, then numerator is obtained by summing up the document clicks of all common documents.

The following formula dictates the similarity function based on documents clicks.

$$Sim_{clickURL(p,q)} = \frac{\sum C(p,di) + C(q,di)}{\sum C(p,xi) + C(q,xi)}$$

Where C (p, d) and C (q, d) are the number of clicks on document d corresponding to queries p and q respectively. C (p) and C (q) are the sets of clicked documents corresponding to queries p and q respectively.

As an example illustration, Q1 and Q2 share two common documents D2 and D4, while D1, D2, D3 and D4 are accessed either by Q1 or Q2 or both. The similarity between two queries is

$$Sim_{clicked\ url}\ (Q1, Q2) = \frac{(200+20) + (2000+200)}{(20+0) + (200+20) + (0+20) + (2000+200)}$$

= 0.984

Similarly, the similarity between Q1 and Q3 is:

$$Sim_{clicked\ url}\ (Q1, Q3) = \frac{2000}{4440}$$

= 0.455

The similarity values always lie between 0 and 1. The measure given in declares two queries similar by imposing a threshold value on the similarity.

3.1.4 Combined Similarity Measure

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 judgments. 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:

$$Sim_{combines}(p,q) = \alpha . Sim_{Keyword}(p,q) + \beta . Sim_{clickURL}(p,q)$$

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

The values of constants can be decided by the expert analysts depending on the importance being given to two similarity measures. In the current implementation, these parameters are taken to be 0.5 each.

3.1.5 CLUSTERING ALGORITHM

Initially, all queries are considered to be unassigned to any cluster. Each 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 (□□), 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 queries, query keywords and the clicked URLs.

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=1; // k is the number of clusters

For (each query p in Q)

Set Cluster_Id(p) - Null; //Initially No Cluster is clustered

For (each p € Q)

{

Cluster_Id(p) = Ck;

Ck –{ p };

For each q € Q such that p ≠ q

{

$$Sim(p,q) = \frac{|KW(p,q)|}{|kw(p) \cup kw(q)|}$$

$$Sim_{clickURL}(p,q) = \frac{\sum LC(p,di)+LC(q,di)}{\sum LC(p,xi)+LC(q,xi)}$$

$$Sim_{combines}(p,q) = \alpha.Sim_{Keyword}(p,q) + \beta.Sim_{clickURL}(p,q)$$

If($Sim_{combines}(p,q) > \tau$)

Set Cluster_Id(q) = Ck;

Ck = Ck U {k};

Else

Continue;

} // End For

K=K+1;

} //End Outer For

Return Query Cluster Set C;

Snap shots:

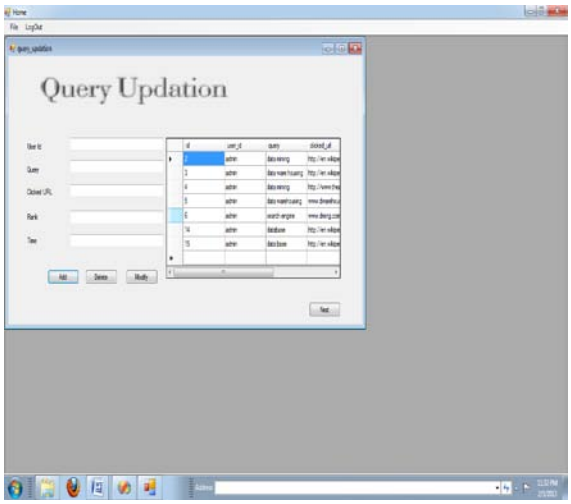

Fig 3. Query Updation

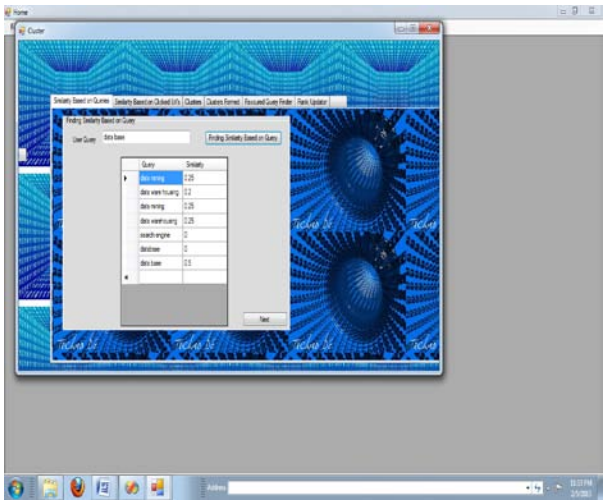

Fig. 4 Similarity form (based on keywords)

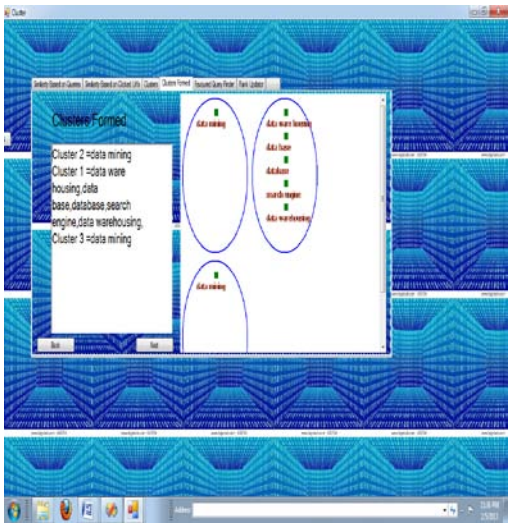

Fig 5. Cluster Formation

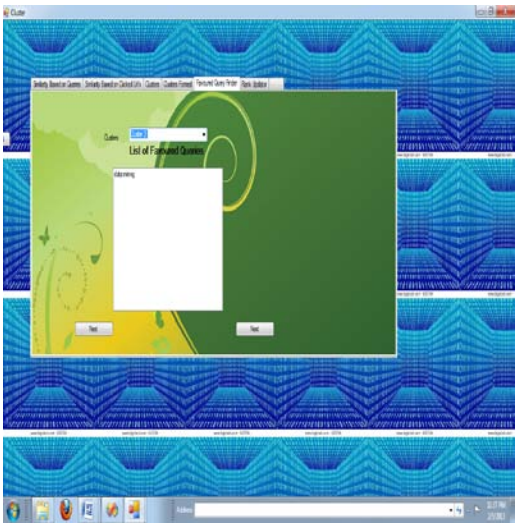

Fig. 6 Favoured Query Finder

4. CONCLUSION & FUTURE SCOPE

This 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. Since result improvement is based on the analysis of query logs, the recommendations and the returned pages are mapped to the user feedbacks and dictate higher relevance than the pages, which exist in the result list but are never accessed by the user. 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 results obtained from practical evaluation are quite promising in respect to improving the effectiveness of interactive web search engines. 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 relevancy of the query terms and allow identifying user's information need more effectively.

REFERENCES

[1] A. K. Sharma, Neelam Duhan, Neha Aggarwal, Rajang Gupta. Web Search Result Optimization by Mining the Search Engine logs. Proceedings of International Conference on Methods and Models in Computer Science (ICM2CS-2010), JNU, Delhi, India, Dec. 13-14, 2010.

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

[3] A. Birchers, J. Her locker, J. Konstantin, and J. Riel, "Ganging up on information overload," Computer, Vol. 31, No. 4, pp. 106-108, 1998.

[4] B. Amen to, L. Tureen, 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

[5] Beeferman and Berger A., 2000. Agglomerative clustering of a search engine query log. In Proceedings of the 6th ACMSIGKDD International Conference on Knowledge Discovery and Data Mining, (August). Acme Press, New York, NY, 407–416.

[6] D. Zhang and Y. Dong, "An Efficient Algorithm to Rank Web Resources," In Proceedings of 9th International World Wide Web Conference, pp. 449-455, 2000.

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

[8] Bernard J. Jansen and Undo Pooch. A review of web searching studies and a framework for future research. J. Am. Soc. Inf. Sci. Technol., 52(3):235–246, 2001.

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

[10] M.R. Her zinger, "Hyperlink Analysis for the Web," IEEE Internet Computing, Vol. 5, No.1, pp. 45-50, 2001.

[11] K. Bharat and G.A. Michaela, "When Experts Agree: Using Non- Affiliated Experts to Rank Popular Topics," ACM Transactions on Information Systems, Vol. 20, No. 1, pp. 47-58, 2002.