International Journal of Database Theory and Application
Vol.9, No.6 (2016), pp.131-144
http://dx.doi.org/10.14257/ijdta.2016.9.6.13

Query Recommendation Based on User Browsing History

[1]Shilpa Sethi*, [2]Ashutosh Dixit

[1] Department of Computer Science, YMCA University of Science and Technology, Faridabad
[2] Department of Computer Science, YMCA University of Science and Technology, Faridabad
[1]munjal.shilpa@gmail.com
[2]dixit_ashutosh@rediffmail.com

Abstract

The most commercial search engines return the search list by matching the user query terms with the documents available in its database. The relative effectiveness of search list is highly affected by the extent to which the query keywords map to the actual need of user. User generally forms the short, ambiguous and instant queries which lead to inclusion of irrelevant documents in the search list. One well known solution to this problem is query suggestion also known as query recommendation. For query recommendations, the search systems maintain the query logs at server sites to better understand user's information need. But till now, the current search systems have partially solved this problem as they roughly offer the similar queries to all the users regardless of their actual interests. In this paper, A novel query recommendation technique based on user browsing patterns is proposed where user interest factor in different domains are computed and used to recommend personalised queries to each individual. The experimental evaluation shows that system is able to assist user in query formation phase and efficiently reduces the search space and time required to get the desired information.

***Keywords**: search engine, query recommendation, browsing behaviour, web usages mining*

1. Introduction

With the increase usage of the internet, the information on web is growing day by day. User relies on search engines to fulfil its information need. It expresses its need in the form of string of keywords also called as query. In order to efficiently cater user's information need, search engine retrieves the documents from its local database by applying keyword based similarity function between user query and web documents. It then sorts the matched documents according to some sophisticated ranking algorithm and presents back the sorted list to the user. But still there are many situations when undesired and irrelevant documents are placed higher in the sorted list. This problem arises due to either use of wrong or insufficient keywords in the user's query. Because search engine is retrieving the documents based on query keywords only. So, problem occurred at user's end during query formation phase. In recent years, many researches have been conducted and implemented in the area of query recommendation, query expansion and query filtering to help user in query formation phase. Google has been offering "auto complete facility" since 2008 (as an experiment feature back since 2004). It stores the information about the user browsing history such as queries, clicked URLs, time etc in query log [6]. The main focus is to identify the alternate queries by matching the user query keywords with the queries already stored in query logs. The matched queries are filtered on the basis

ISSN: 2005-4270 IJDTA

of popularity and/or location. For example, consider two different scenarios as listed below:

Scenario 1: "A user U1 is from the computer field and wants to search about the term ***Java****".*

It submitted the query ***java*** at Google interface. The alternate queries offered by the Google are *java learning online*, *java games*, *java programming* and *java verify* shown in Figure 1. The suggestions that Goggle offered all come from how people actually searched in past.

Scenario 2: "Another user U2 who has interest in coffee submitted the same query ***java*** *at Google interface".*

Google will offer the same suggestions as previous, regardless of user's actual information need. It means the problem of query recommendation is only partially solved. Thus more personalised query recommendation system is required which can infer each user need correctly. The goal of the proposed query recommendation system is to produce personalised queries that map correctly to individual user's information need. To achieve this, two steps filtering process is used. The first step aims to identify all those queries which are contextually related to user query instead of only considering the query keywords. The second step extracts user specific queries based on degree of user interest in specific domain thereby offering personalised queries to the each user.

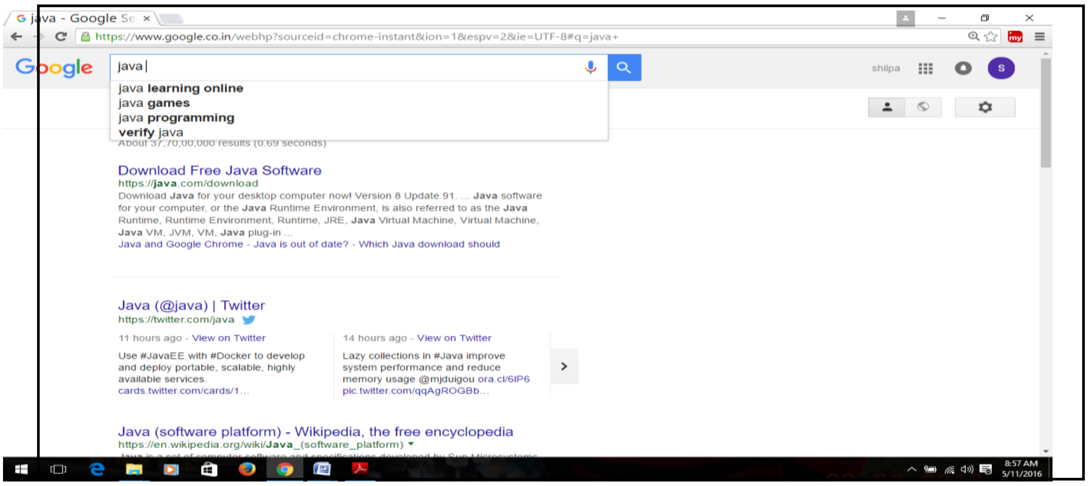

Figure 1. Example of Query Suggestion

The paper is organised as follows: In section 2 the query log concepts and different query similarity methods are discussed. The section also describes a popular query recommendation algorithm called BB's algorithm that forms the basis of proposed work. Section 3 describes the proposed query recommendation method along with example illustration. In section 4, the analysis of sampled web log files has been conducted to validate the proposed mechanism. Finally, the conclusion and future scope are given in section 5.

2. Related ork

This section provides a brief description of query log and most popular BB's algorithm for query recommendation.

2.1 Query Log

A query log can be defined as the electronic record that stores information about the interaction occurred between the search engine and its user. The modern search engine records the entry for every single access made by the user corresponding to a query in to the log files. The standard format of log files is shown in table1 [5].

Table 1. Query Log Instance- An Example of Click through Data

User ID / Session ID	Query	Clicked URL	PageRank	Time
861543	Core java tutorial	www.javabeginnerstutorial.com	2	2015-10-09 00:02:23
861543	Core java tutorial	www.tutorialspoint.com	4	2015-10-09 00:02:23
902341	Core java interview question	www.theserverside.com	4	2015-10-09 00:02:50
902341	Core java interview question	www.javaworld.com	5	2015-10-09 00:02:52
902341	Core java interview question	www.dzone.com	1	2015-10-09 00:02:52
....	....	....	....	

Many techniques had been proposed in past to mine the similar query from query logs such as similarity based on query keywords [2][3][4][9] , similarity based on click through data [6][8] , similarity based on web snippets [1][7] etc. In keyword based similarity measure each query is represented as keyword vector [9]. The cosine or Jaccard similarity function is used to measure the distance between the two queries as given in eqn (1)

W $$Sim_{keyword}(P,Q) = \frac{\vec{P}\cdot\vec{Q}}{(P).(Q)} \quad (1)$$

Where: P and Q are two queries. The relatedness of P and Q is the cosine of the keyword vector $\vec{P}$ and $\vec{Q}$. The method is simple and easy to implement, but fail to identify the relatedness between the queries which contains uncommon word belonging to same concept. For example the queries: "movie" and "film". Although they do not contain common keyword but they refer to same concept. To overcome the limitation of keyword based similarity, Beeferman and Burger proposed agglomerative clustering algorithm also known as BB's algorithm to cluster all the similar queries in to one group. The BB's algorithm is discussed in detail in following subsection.

2.2 BB's Algorithm

Given a search query log, the algorithm first constructs the bipartite graph with one set of nodes corresponding to user queries depicted by empty circle and other set of nodes corresponding to click URLs depicted by solid circles in Figure 2(a), 2(b) and 2(c). An edge is created between query node Q and URL node L, whenever the user clicks on L with respect to Q.

*Shilpa Sethi

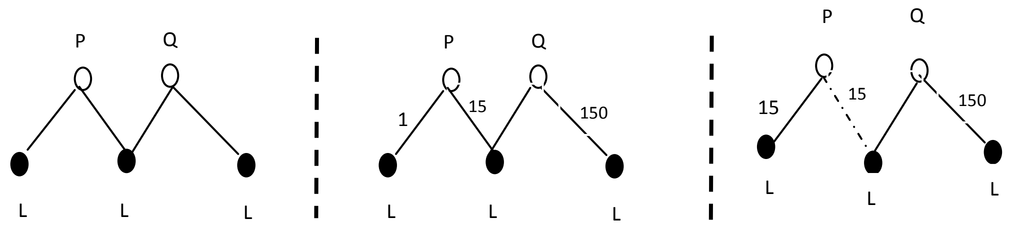

Figure 2(a) Bipartite Graph without No. of Clicks (b) With No. of Clicks (c) With a Noise Link Represented by Dotted Line.

According to this algorithm, two queries are said to be similar if their neighbouring nodes overlap i.e. share the common clicks .The similarity is evaluated by formula (2).

$$Sim_{URL}(P,Q) = \frac{|\,N(P) \cap N(Q)\,|}{|\,N(P) \cup N(Q)\,|} \quad (2)$$

Where: N(P) is set of neighbouring nodes of P and N(Q) is neighbouring nodes of Q. It means more the no. of common URLs between two queries, more similar the queries are. Although this method removes the limitation of keyword based measure, but still there exist a problem. It did not consider the relative clicks on common URL and sometimes cluster the less similar queries in same group

To explain this let us consider two different scenario shown in Fig 2(b) and 2(c).The number attached to each edge represents the total no. of clicks on a URL 'L' with respect to any arbitrary query (*say Q*) . In fig 2(b), The URL L1 and L2 earn 15 clicks with respect to query P, which imply that both links are equally relevant to P. Similarly, L2 and L3 are equally relevant to Q. whereas in fig 2(c), L2 receives only 15 clicks as compared to 1500 clicks of L1 with respect to query P. It implies that L1 is more relevant to P as compared to L2. Ideally P and Q cannot have the same similarity score as that of previous. But BB;s algorithm assigns same similarity score in all the three cases depicted in Figure 2(a) , 2(b) and 2(c) i.e 0.33.

So, a critical look at the available literature indicates the following shortcomings:

1) The Keyword based similarity function assigns the similarity score by comparing each keyword of query Q1 with query Q2 whereas it is not necessary that common keywords correspond to similar information need and vice versa.

2) The URL based similarity function erroneously groups less similar queries in same cluster.

3) All the query suggestion algorithms follow either keyword based approach or URL based approach or combination of both, but none of them considered the user browsing behaviour that may provide important clues while constructing the alternate queries to the user.

The proposed recommendation system discussed in the next section overcomes the aforementioned limitations by considering the no. of clicks on each link w.r.t a query and preparing the personalised queries for each user by considering the degree of interest of user in different domains.

3. Proposed Work

An efficient query recommendation technique based on user browsing history is being proposed here to assist the user in query formation phase .The primary goal of the system is to group similar queries in one cluster based on query terms and user click through data . When the user enters a query, the clusters that best matches with user query are identified. These identified clusters are mined on the basis of degree of interest of user in different domains to generate the personalised queries with respect to specific user .In this way, the search space is considerably reduced by recommending the personalised queries at the early stage of search process thereby serving the unambiguous relevant results to the user. The proposed query recommendation system is shown in Figure 3. It consists of four major components.

1) User interface
2) Profile generation module
3) Query clustering module
4) Query recommendation module

The detail description of each component is given in following subsections.

3.1 User interface

It is an interface where the user specifies its information need in the form of query. It first creates the account for a novice user or verifies the existing user with the help of special module named as profile generation module [10]. After creation/verification, it offers the set of personalised queries to the user with the help of query recommendation module. The user is expected to select one query out to offered queries. The selected query is then passed to query processing module to obtain the sorted list of URLs. At last, the sorted list is presented back to the user.

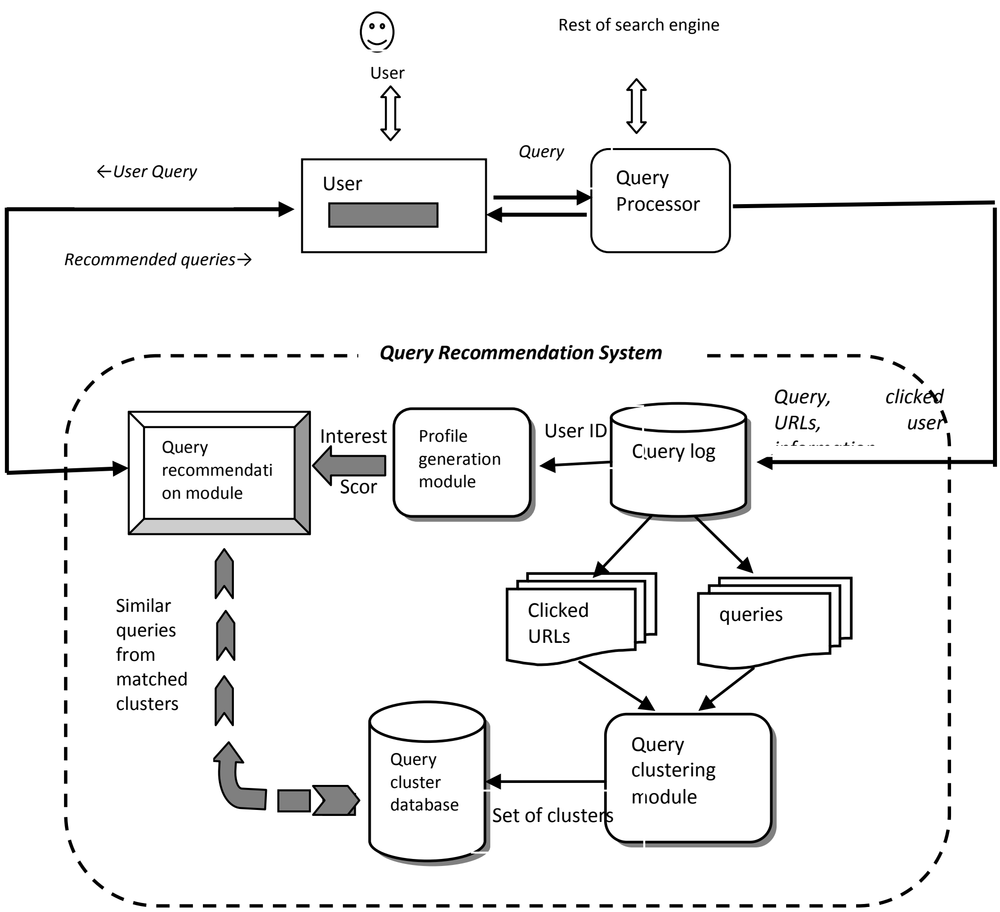

Figure 3. Proposed Query Recommendation System

*Shilpa Sethi

3.2 Profile Generation Module

This module maintains the user's information (such as user id, password, and degree of user's interest) in profile database. In the proposed system, the search engine database is partitioned in different classes C= $\{C_1, C_2 ....C_m\}$. For instance, in the current implementation the database is partitioned in five classes namely: education, travelling and tourism, entertainment, food & beverages and fashion & shopping. These classes are further extendible). The degree of user's interest in a specific class is denoted by $\Gamma_{(ua\ ,CK)}$. It is defined as follows:

Definition: the degree of user interest in specific domain, $\Gamma_{(ua\ ,CK)}$ can be defined as the ratio of no. of pages accessed by user ua in class C_k to the total no. of pages accessed by ua in all the classes.

Mathematically , it can be computed by the eqn (3) as given below:

$$\Gamma(ua\,, CK) = \frac{NC\,(ua, Ck)}{\sum_{i=1}^{m} NC\,(ua, Ci)} \qquad (3)$$

Where: $NC\ (u_a\ , C_k)$ denotes the no. of pages clicked by user u_a in class C_k, m is the no. of classes in search engine database. The working of profile generation module is depicted in Figure 4.

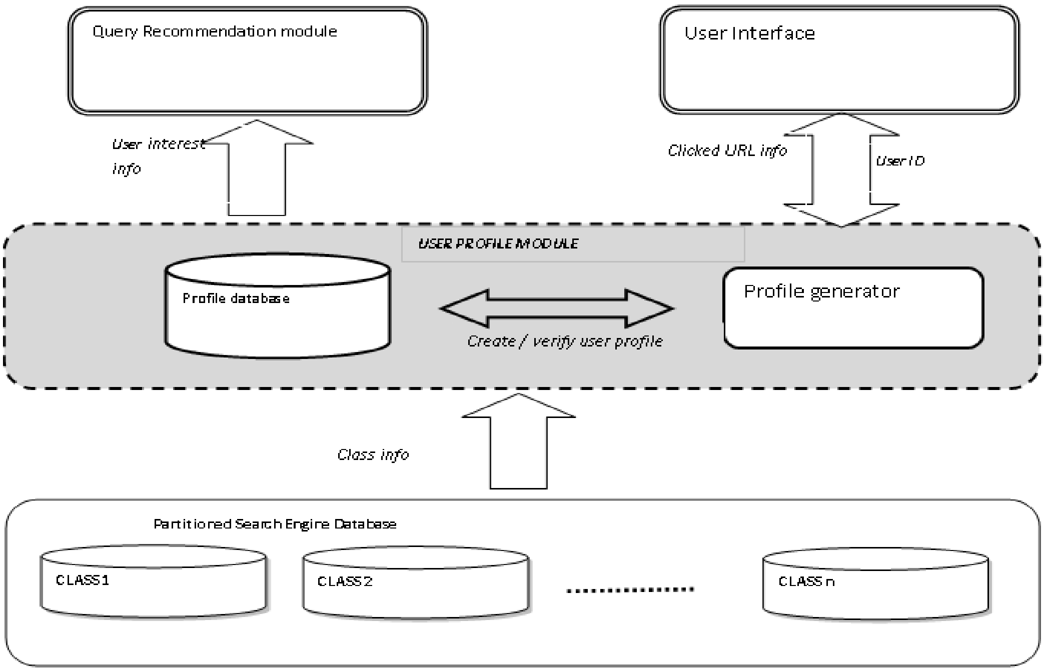

Figure 4. Working of Profile Generation Module

The algorithm for profile generation module is shown in figure 5.

Algorithm: Profile Generation module ( )
Given: partitioned database containing Set of classes C= {C1, C2......Cm} , user id, password.
Output: A table containing degree of interest of each user interest[n][m] ; where n is no. of users and m is no. of classes in search engine database.
Method: n=0; // initially there is no registered user
wait(user interface);
If (info(user interface)){
Check uid in profile database;
If (uid ε profile database){
Return(valid user) to user interface;
wait(click_uid); //wait for user click on some page
For each user click on any page P ε Ck{
NC(uid, Ck) ←NC(uid,Ck)+1;
$\Gamma(uid, CK) = \frac{NC(uid,Ck)}{\sum_{i=1}^{m} NC(uid,Ci)}$ }
Update the entry in profile table by $\Gamma(uid, CK)$; }
else{ Return (invalid user) to user interface;}
else{
n=n+1;
Create a new uid in the profile database;
for each Ck ε C{
NC[uid,Ck] ←0; // initially no. of user clicks in each class is 0
$\Gamma(uid, CK)$←0; //initially user interest in each class is 0
store $\Gamma(uid, CK)$ profile database;}}
}

Figure 5. Algorithm for Profile Generation Module

A small fragment of user profile database at arbitrary time ***t*** is shown in table 2.

Table 2. A Small Fragment of Profile Database at any Time t

User id	Password	Classes				
		C1	C2	C3	C4	C5
U1	xxx...	0.4	025	0	0.2	0.05
U2	Yyy...	0.3	0.1	0.45	.0.04	0.11
U3	Zzz....	0.5	0.1	0.35	0.15	0
U4	www....	0	0.7	0.2	0.09	0.01
U5	Vvv...	0.63	0	0.1	0.07	0.2

It may be analysed from the above table that each user posses different level of interest in different classes. This information is very useful in filtering out the alternate queries to be offered to the user.

3.3 Query Clustering Module

This module is responsible to group the similar queries under a common cluster based on two main concepts as discussed below:

3.3.1 Evaluating similarity based on context of query terms: Two queries are said to be similar if query terms or synonym of query terms matches above a threshold value $T_{context}$. To compute the context similarity between two queries P and Q, the eq^n (4) is used.

$$Sim_{context(P,Q)} = \max\begin{bmatrix} \frac{|QT(P)\cap QT(Q)|}{max\{|QT(P)|,|QT(Q)|\}}, \frac{|QT(P)\cap QT(SQ)|}{max\{|QT(P)|,|QT(SQ)|\}}, \\ \frac{|QT(SP)\cap QT(Q)|}{max\{|QT(SP)|,|QT(Q)|\}}, \frac{|QT(SP)\cap QT(SQ)}{max\{|QT(SP)|,|QT(SQ)|\}} \end{bmatrix} \quad (4)$$

*Shilpa Sethi

Where: QT (P), QT (SP) represents the terms in query P and synonym of query P respectively. |QT(P)| measures the no. of terms in query P. To explain this, let us measure the context similarity among the four queries q1, q2, q3 and q4 given in table 3. Initially the queries do not belong to any cluster i.e. set of cluster C=φ.

Table 3.Query Examples

Sr. No	Queries
1	apple jams recipes
2	jam recopies
3	apple os
4	fruit jam recipes

By applying eq[n] (4), the context similarity between the queries can be stored in a matrix represented by context similarity (Qi , Qi+1) as given below:

$$context\ similarity(Qi, Qi+1) = \begin{pmatrix} 1 & 0.66 & 0.33 & 1 \\ 0.66 & 1 & 0.5 & 0.66 \\ 0.33 & 0.5 & 1 & 0.33 \\ 1 & 0.66 & 0.33 & 1 \end{pmatrix}$$

Taking $T_{context}$ = 0.65. The four queries can be grouped in to two clusters i.e. C={C1,C2} such that C1={Q1, Q2,Q4} and C2={Q3}.

3.3.2 Evaluating similarity based on common clicked URLs: If two queries lead to the selection of same URL, then they may be considered as similar [6] [11]. In order to find the extent to which they are similar , the concept of no. of clicks on common URLs is introduced here. Formula for measuring the similarity between two queries based on no. of clicks on common URLs is given in eqn (5):

$$Sim_{ClickedURL}(P,Q) = \sum_{i=1}^{n} \frac{min\left(NC(P,Li), NC(Q,Li)\right)}{\max\left(NC(P,Li), NC(Q,Li)\right)} ; \forall Li \in CL(P) \cap CL(Q) \qquad (5)$$

Where CL(P) and CL(Q) are the sets containing the clicked URLs corresponding to query P and Q respectively. NC(P,Li) and NC(Q,Li) are no. of clicks on URL 'Li' with respect to query P and Q respectively.

For measuring the similarity based on above formula, the query clustering module first constructs the bipartite graph in which one set of nodes corresponds to queries and other set of nodes corresponds to clicked URLS as shown in Figure 6. The numeric value mentioned on an edge e_i joining Qi and Li represents the number of times the Li gets selected w.r.t. Qi. For example, in Figure 6 the value 40 mentioned on edge joining Q1 and L4 implies that 40 clicks are received on URL L4 w.r.t query Q1. Further, it is considered that the user click on any URL w.r.t a query can be taken as a

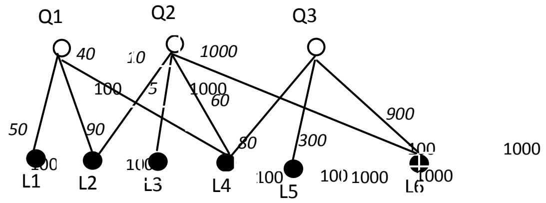

Figure 6. Sampled Bipartite Graph

good source of user feedback. In Figure 6, L1, L2 and L4 are selected with respect to query Q1, which implies that they are relevant to query Q1. Similarly, L2, L4 and L6 are considered relevant to query Q2 and L5, L4 and L6 are relevant to query Q3. As Q2 share common URLs with Q1 and Q2. So they may be considered as similar. The extent to which Q2 is similar to Q1 and Q3 can be measured using eqn (5) as follows:

$$Sim_{clickedURL(Q1,Q2)} = \frac{\min(90,10) + min(40,60)}{max(90,10) + \max(40,60)} = \frac{50}{150} = 0.33$$

$$Sim_{clickedURL(Q2,Q3)} = \frac{\min(60,80) + min(1000,900)}{max(60,80) + \max(1000,900)} = \frac{960}{1080} = 0.88$$

So, Query Q2 is considered more similar to Q3 as compared to Q1.

3.3.3 Combined similarity measure: The two similarity concepts described above have their own benefits. On the one hand, the contextual similarity groups all those queries which share the similar composition of query terms or synonyms of query terms in to one cluster. On the other hand, the common click based similarity takes the advantage of user feedback in identification of similar queries. But alone each of them can partially capture the similarity among the queries. So, it's better to combine both the measures in a single measure as shown in eqn (6)

$$Sim_{combined(P,Q)} = (1-\mu)Sim_{context(P,Q)} + \mu\, Sim_{clickedURL(P,Q)} \quad (6)$$

Where μ is similarity constant such that $\mu \in [0,1]$ .In the current implementation its value is taken as 0.5. If the $Sim_{combined(P,Q)}$ is greater than the pre defined threshold value $T_{combined}$,they are grouped under the same cluster. The algorithm for query clustering module is given in Figure7.

Algorithm: Query clustering module ()
Given: Similarity constantμ, similarity threshold $T_{combine}$, Query log containing the following fields:
1. User ID of each user
2. Query ID assigned to each query
3. Query
4. URLs selected by user corresponding to Query ID
5. No. of clicks on selected URLs
6. Class ID of selected URL
Output: Set of clusters denoted as Clust ={clust1,clust2........clustn}; each cluster contains the following information.
1. Cluster ID
2. A collection of similar queries with clicked URL and class ID.
3. Keyword set of each cluster
Method:

```
Set n=0;                    //count of no. of clusters
Set i=o;                    //counter for query
for each query q_i € Query log{
Flag [qi]=0;
Clust(qi)=∅;}
for each  query q_i € Query log{
If(Flag[qi]==0){
n=n+1;                  //create new cluster
Cluster(qi)=Clust_n;
Clust_n= {(qi,clicked URLs,class ID)};
Keyword_clustn=stem(qi);  // keyword set of new cluster
for each qi_{+1}€  Query log such that qi≠qi_{+1}{
Find Sim_combined(q_i,q_i+1) using eqn (6) ;
If(Sim_combined(q_i, q_i + 1) ≥ τ_combined)
{    Clust(q_{i+1})=Clust_n;
Clust_n= Clust_n U {(qi_{+1},clicked URLs,class ID)};
Keyword_clustn= Keyword_clustn U stem(q_{i+1});}}
else i=i+1; }
```

Figure 7. Algorithm for Query Clustering Module

[*]Shilpa Sethi

3.4 Query Recommendation Module

It receives the user query from search engine interface and returns the set of alternate queries to be presented to the user. It applies two level of filtering process to construct the set of alternate queries. First, all those clusters whose keywords matches with the query keywords are retrieved from query cluster database. Then the four most popular queries are extracted from each matched cluster. It is assumed that the query which is submitted /selected by many users is more popular.Second, the set of popular queries are filtered on the basis of user domain of interest. The profile generation module provides the interest score of each user in different domains. So, more personalised queries are returned to search engine interface to offer them to the user. The algorithm for query recommendation module is given in figure (8) as follows:

Algorithm: Query recommendation module ()
Given: user query Q, Query cluster database containing the following fields:
1. Cluster ID assigned to each cluster
3. Set of keywords associated with each cluster.
4. Set of queries with query weight
4. Set of clicked URLs for each query
5. Class ID of each URL
Output: Set of personalized queries $Q_{personalized}$={q1,q2,q3,q4}
Method:
1) Set $Clust_{matched}$=NULL for Q;
For i=1 to n { if (Keyword(Q)∩Keyword($clust_i$)t> Threshold)
$Clust_{matched}$=$Clust_{matched}$ ∪ {$Clust_i$}; }
2) Fetch the four most popular queries from each matched cluster and store them in $Q_{personalized.}$ along with their class ID.
3) Call the profile generation module to give Interest weight of UID in each class.
4) Sort the $Q_{personalized}$ on the basis of interest weight;
5) Pop the first four queries from $Q_{personalized}$ and return it to search engine interface,

Figure 8. Algorithm for Query Recommendation Module

4. Result Analysis

To evaluate the performance of proposed system, a dataset of 2000 web pages is partitioned in five different web categories as shown in table 4.The consideration is to check the performance of system on small set of classes, which can be easily extendible in future. User study is conducted with volunteer group of post graduate students. The system creates the profile for every user and stores their browsing history in query log. In order to show the validity of proposed query recommendation system, a fragment of query log, containing 16 queries (as whole data is too large to present here completely) is depicted in table 5. For the sake of simplicity, the nomenclature of the different domains is shown in table 4.

Table 4. Nomenclature of Domains

Class ID	Domain
A	Education
B	Sports
C	travelling & tourism
D	food & beverages
E	digital products

Table 5. Fragment of User Query Log

Sr No	Session ID	Query ID	Query	Clicked URL	Click count	URL Class ID
1	1520021	Q1	Best iphone price	www. Apple.com www. mysmartprice.com www.bestappleprice,com	10 14 20	E E E
2	1520021	Q2	Apple iphone	www. Apple.com www.techradar.com www.bestappleprice.com	25 17 23	E E E
3	1520021	Q3	Feature apple iphone	www. gadgets.ndtv.com www.apple.com www.tps.apple.com	12 19 20	E E E
4	1520021	Q4	Apple price India	www.apple.com www.mysmartprice.com www.techradar.com	22 10 16	E E E
5	1520021	Q5	Apple store	www.locateapple.com www.apple.com www.bestappleprice.com	26 17 13	E E E
6	1520022	Q6	Fruit cake store	www.foodnetwork.com www.vermentcountry.con www.allrecipes.com	10 16 5	D D D
7	1520022	Q7	Apple cake store	www.vermentcountry.con www.allrecipes.com www.foodnetwork.com	13 5 11	D D D
8	1520022	Q8	Blackberry juice	www.livestrong.com www.thejuicenut.com www.myrecipes.com	19 12 6	D D D
9	1520022	Q9	Best iphone	www. samsungindiastore.com www.apple.com www.mysmartprice.com	14 12 9	E E E
10	1520023	Q10	Compare iphone	www.bestappleprice.com www.moneysupermarket.com www.gadgets.ndtv.com	19 22 17	E E E
11	1520023	Q11	Blackberry phone	www.geekaphone.com www.gedgets.ndtv.com www.tradeupblackberry.com	20 16 25	E E E
12	1520023	Q12	Easy jam recipe	www.tasteofhone.com www.allrecipes.co.uk www.food.com www.bbcgoodfood.com	29 20 12 8	D D D D
13	1520024	Q13	Benefits blackberry juice	www.livestrong.com www.thejuicenut.com www.stylecraze.com	16 10 9	D D D
14	1520024	Q14	Java beans	www.javatpoint.com www.tripadvisor.com www.javabeanplus.com	10 16 11	C C C
15	1520024	Q15	Best mobile price	www.91mobiles.com www.mysmartprice.com www.samsung.com	22 18 14	E E E
16	1520024	Q16	Apple jam recipe	www.tasteofhone.com www.allrecipes.co.uk www.freshpreserving.com	25 26 10	D D D

The aim of the analysis is to group the similar queries under one cluster and generates the personalised queries for each user. The experiment evaluates the working of following similarity measures.

1) Context similarity measure; $Sim_{context}$
2) Common clicked URL similarity measure; $Sim_{clicked\ URL}$
3) Combined similarity measure; $Sim_{combined}$
4) Degree of interest; ┌ of each user

*Shilpa Sethi

4.1 Generation of query cluster based on Sim combined

Let us consider top two queries from table 5. The context similarity $Sim_{context,}$ and common URL click similarity, $Sim_{clickedURL}$ can be evaluated by eqn (4) and (5) as given below:
To simplify the calculation, Let Q1= Best iphone price and Q2= Apple iphone , The value of similarity constant, μ is taken as 0.5 and similarity threshold , $T_{combine}$ =0.65 .
Applying eqn (4), $Sim_{context}$(Q1,Q2) =2/3=0.33
Applying eqn (5) , $Sim_{clickedURL}$(Q1,Q2) =((10/25+20/23)= 1.27
Applying eqn (6) , $Sim_{combined}$ (Q1,Q2) = 0.5 x o.66 +0.5x1.27=0.8
Since $Sim_{combined}$(Q1,Q2) > $T_{combine}$, So queries Q1 and Q2 are grouped in same cluster, named $Clust_1$ along with clicked URL and class id information. The keyword set of clust1 named $Keyword_{clust1}$ = {Best, iphone, price, apple}. The same steps are repeated for other queries and finally three clusters are obtained for table 5 i.e $Clust_1$={Q1,Q2,Q3,Q4,Q5,Q9,Q10,Q11,Q12,Q15,Q16} and $Clust_2$= { Q6,Q7,Q8,Q13},$Clust_3$={Q14}

4.2. Personalised query generation

When the user submits a query, its keywords are matched with each cluster‘s keywords. Top four popular queries from matched clusters are extracted along with the following parameters:
1) Clust ID
2) Set of clicked URLs
3) Class ID of clicked URL.

These queries are further filtered by query recommendation module by applying the degree of interest of user. Table 6 shows list of recommended personalised queries presented to two different users on the basis of their interest areas.

Table 6. List of Query Recommendations by Proposed System for Query "Java Beans"

	USER ID	
	152001	152002
QUERY RECOMMENDATIONS	Java apps	Java beans coffee
	Java games	Java beans coffee shop
	Java script	Java beans coffee Jakarta
	Java coffee	Java beans coffee prejaken village

The result analysis is carried out with a group of post graduate students. They are asked to vote for queries recommended by proposed system and popular search system based on their satisfaction level. A fragment of student's satisfaction level for both the systems is shown in fig 9(As actual data is too large to show here). It may be observed from Figure 9 that more no. of students is satisfied with the queries recommended by proposed system. Thus more personalised queries can be offered to help user in query formation phase resulting in better search experience.

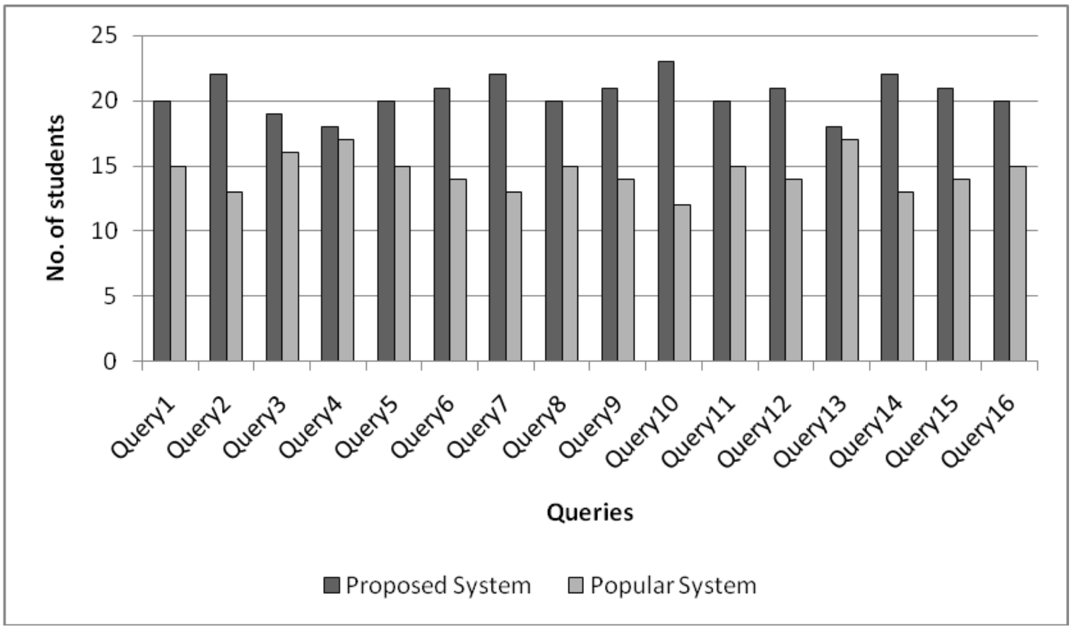

Figure9: Comparison of Proposed System with Existing Recommendation System

5. Conclusion and Future Scope

The paper proposed a novel query recommendation technique for implementing the efficient search engine .It suggests the personalised queries to individual user so that their diversified need can be fulfilled. The technique makes use of context similarity and click through data similarity among the queries to group them in relevant cluster. The user query is matched with query cluster to retrieve the relevant alternate queries from cluster database. The promising part of proposed system is that the alternate queries are further refined based on degree of interest of each user in different classes. By refining the user search need at early stage results in reduction of time user spent for seeking out the desired information from search list. The result obtained from the experimental evaluation shows the increase in user satisfaction level with respect to query suggested by proposed Further more personalised techniques may be embedded in ranking phase which can provide more comprehensive ranked list to each individual user.

References

[1] Kenneth Wai-Ting Leung, Wilfred Ng, and Dik Lun Lee, "Personalised Concept based Clustering of Search Engine Queries", IEEE Transactions on Knowledge and Data Engineering,, vol. 20, no. 11, (2008).

[2] **S.** Sukanya. Gawade, J.Gyankamal, Chhajed, "Using Feedback Sessions for Inferring User Search Goals", International Journal of Advanced Research in Computer and Communication Engineering ,Vol. 3, no. 6, (2014) .

[3] A.K.Sharma and Neelam Duhan, "Web Search Results Optimization by Mining the Search Engine query log".

[4] Saeedeh Shekarpour, Konrad Höffner, Jens Lehmann and Sören Auer, "Keyword Query Expansion on Linked Data using Linguistic and Semantic Feature", IEEE Seventh International Conference on Semantic Computing (2013).

[5] Sami Uddin and Amit kumar Nandandwar, "Recent Trends in Query Clustering", International Journal of Computer Science and Information Technology, vol. 5, no.5, (2014) 6744-6749.

[6] Lyes Limam, David Coquil and Harald Kosch, "Extracting User Interests from Search Query Logs: A Clustering Approach".

[7] Kenneth Wai-Ting Leung and Dik Lun Lee, " Deriving Concept based User Profile from Search Engine Logs", IEEE Transactions on Knowledge and Data Engineering Journal of Latex Class Files, vol. 6, no. 1, (2007).

*Shilpa Sethi

[8] Neelam Duhan and A.K.Sharma " Rank Optimization and Query Recommendation in Search Engine using Web Log Mining Techniques", Jour,nal of Computing, vol. 2, no. 12, (2010).
[9] Liu Ying, T. Bridget and McInnes, "Semantic Relatedness Study Using Second Order Co-occurrence Vectors Computed from Biomedical Corpora, UMLS and WordNet".
[10] Shilpa Sethi andAshutosh Dixit ," An Adaptive Web Search Engine for Retrieving Quality Data", International Journal of Computer Engineering and Applications, Vol 1 no. 10, (2016).
[11] Shilpa Sethi and Ashutosh Dixit, " An Efficient Personalised Query Suggestion Technique for Providing Relevant Results" In the Proceeding if 10th IEEE International Conference on Computing for Sustainable Global Development , March 2016 .

Authors

Shilpa Sethi received her M. Tech. in Computer Engineering from MD University Rohtak, in the years 2009 and Master of Computer Application from Kurukshetra University , Kurukshetrain 2005 . She is presently working as Astt Professor in the department of computer engineering at YMCA University of Science & Technology, Faridabad Haryana. She has published 10 research papers in various International journals and conferences. Her research interests include Internet Technologies and Data mining.

Ashutosh Dixit received his PhD and M. Tech. in Computer Engineering from MD University Rohtak, in the years 2010 and 2004 respectively. He is presently serving as Associate Professor in the department of computer engineering at YMCA University of Science & Technology, Faridabad Haryana. He has published around 80 research papers in various International journals and conferences. His research interests include Internet Technologies, Data Structures and Mobile and Wireless networks.

