Volume 3, Issue 12, December 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Enhanced Personalized Search Query Log Analysis on User Logs

Praveena.Karri*
Research Scholar
Department of CSE
MIC COLLEGE OF TECHNOLOGY
JNTUK, ANDHRA PRADESH, India

JanakiRamaiah.Bonam
Assoc.Professor
Department of CSE
MIC COLLEGE OF TECHNOLOGY
JNTUK, ANDHRA PRADESH, India

Abstract: User profiles can be used by search engines to provide personalized search results. Users are increasingly pursuing complex task oriented goals on the web, like making travel arrangement, planning purchase or managing finance. Organizing the user search logs is rapidly increasing in the field of data mining for finding the user interestingness and organizing the user search requirements in a proper way. Searchers create and use external records of their actions and the corresponding results by writing/typing notes using copy and paste functions. Daily billions of queries can be passed to the server for relevant information most of the search engines retrieves the information based on the query similarity score or related links with respect to given query. To better support users in their long-term information quests on the Web, the search engines keep track of their queries and clicks while searching online. This paper proposes to enhance search query log analysis by taking into account the semantic properties of query terms. User profiles were created by classifying the collected information into concepts in a reference concept hierarchy. We study the problem of organizing a user's historical queries into groups in a dynamic and automated fashion.

Keywords: Query terms Taxonomy, clustering, log analysis, query clustering and task identification

I. INTRODUCTION

Personalization is the process of presenting the right information to the right user at the right moment. Systems must collect personal information and store the results of the analysis in a user profile. The information can be collected from users in two ways:

- explicitly
- implicitly

Commercials systems tend to focus on personalized search using an explicitly defined profile. Implicitly created user profiles do not place any burden on the user and they provide an unbiased way to collect information. With the increasing number of published electronic materials, the World Wide Web (WWW) has become a vast resource for individuals to acquire knowledge. As the size and richness of information on the Web grows, so does the variety and the complexity of tasks that users try to accomplish in online. We use our memory to bridge across different information sources and activities but human memory is limited and selective. Users are usually reluctant to explicitly provide their preferences due to the extra manual effort involved. One of information-seeking tasks often performed by students is Information Gathering that is evaluating, extracting and organizing relevant information for a given topic.

Recently, some of the major search engines have introduced a new "Search History" feature, which allows users to track their online searches by recording their queries and clicks. Implicit indication of document relevance we can predict his/her reaction to the current retrieved documents. To achieve effective personalization, these profiles should be able to distinguish between long-term and short-term interests.

Several systems have attempted to provide personalized search based upon user profiles that capture one or more of these aspects [1]. A classification of interest-based user profiles is provided; they consider the different ways of creating and maintaining the user profile.
In general they split the definition of user profiles into 3 classes:

a. content-based profiles
b. collaborative profiles
c. rule-based profiles

Recommendations for search history displays and two search history based user interface tools are described here. In fact, identifying groups of related queries has applications beyond helping the users to make sense and keep track of queries and clicks in their search history. Query grouping allows the search engine to better understand a user's session and potentially tailor that user's search experience according to her needs.

Once query groups have been identified, search engines can have a good representation of the search context behind the current query using queries and clicks in the corresponding query group. Consider an example that the search engine knows that a current query “financial statement” belongs to a {“bank of America”, “financial statement”} query group. We study the problem of organizing a user’s search history into a set of query groups in an automated and dynamic fashion [2]. These query groups are dynamically updated as the user issues new queries, and new query groups may be created over time. Search history can be divided into two terms:

- *Short term*

  Short-term search history is limited to a single search session that contains a (normally consecutive) sequence of searches with a coherent information need and usually spans a short period of time.
- *Long term*

  Long-term search history is unlimited in time scope and may include all search activities in the past. Compared with short-term search history, it has several advantages.

Organizing the query groups within a user’s history is challenging for a number of reasons. Related queries may not appear close to one another as a search task may span days or even weeks. The interleaving of queries and clicks from different search tasks due to users’ multitasking, opening multiple browser tabs and frequently changing search topics.

Users’ document preferences are first extracted from the click through data, used to learn the user behavior model which is usually represented as a set of weighted features. User profiles are created based on the users’ preferences on the extracted topical categories. Web learners begin this process with recognizing an anomalous state of knowledge related to a topic. This state is the interest or concern mental state that triggers the information gathering process. This make an initial search plan based on their prior knowledge with each piece of new and useful information encountered giving them new ideas on their topic.

II. BACKGROUND WORK

This is the literature survey which is used for the work

1. *Heasoo Hwang, Hady W. Lauw, Lise Getoor and AlexandrosNtoulas[2].*

In this paper, organizing user’s historical queries in to groups in a dynamic and automated fashion by going beyond the textual similarity or time thresholds.

2. *E. Sadikov, J. Madhavan, L. wang and A. Halevy,[4].*

This paper proposes how the markov chain model can be used to improve the selection and placement of the query suggestions proposed by the search engine.

3. *J. Yi and F.Maghoul, [5]*

This paper deals with the problem of discovering query clusters from a click through graph by using scalable algorithm.

4. *J. Teevan, E. Adar, R. Jones, and M. A. S. Potts, [7].*

From this paper the repeat queries can be automatically detected by changing the rank of already searched query.

III. EXISTING SYSTEM

A user queries a search engine. The search engine displays results based page ranking algorithms. Users are no longer content with issuing simple navigational queries. The primary means of accessing information online is still through keyword queries to a search engine. Make an initial search plan based on their prior knowledge with each piece of new and useful information encountered giving them new ideas on their topic. Information gathering is a very complex information seeking task and it can be completed not by a specific answer but by a series of extractions, comparisons and syntheses of a broad range of information related to these topics/subtopics. Learners are frequently required to maintain many extracted results for later use and reference. To keep a huge amount of information in a human’s mind is difficult because the limitation of working memory. To support the limitation of memory capacity, learners have to employ external memory aids.

Many early commercial systems had a history feature that allowed users to recall past search commands and reuse them. Importance of search histories in user interfaces has remained clear in the decades that passed. She highlighted the need for search system user interfaces to show what steps had been taken in the past and what short- and long-term strategies had been followed. We concluded that user observations suggest the need for search histories in the user interface of information retrieval and visualization systems. She pointed out that these functions are not well supported in current systems.

The need for search histories in search interfaces is clear not many innovative solutions are available to present and manipulate them. The Aria dne system was proposed to support collaboration among users by visualizing search session histories. System captures “query–result set pairs and displays them to the user as thumbnails of screen shots. This article reports on the results of a thorough examination of the use of interaction histories in one specific application domain area. The frequently changed on attention make students easily disoriented. The structures of information organized in the three memory aids are inconsistent. To find and recall a piece of information that is previously kept in these memory aids becomes difficult.

A query group is denoted as s ={q1, clk1}, . . . , {qk, clkk}. The specific formulation of our problem is as follows:

Consider a set of existing query groups of a user i.e. S = {s1, s2. . . Sn} and her current query and click i.e. {QC, clkc}. Find the query group for {qc, clkc}, which is either one of the existing query groups in S that is most related to or a new query group sc = {qc, clkc} if there does not exist a query group in S that is not sufficiently related.

```
SelectBestQueryGroup
Input:
  1) the current singleton query group $s_c$ containing the
  current query $q_c$ and set of clicks $clk_c$
  2) a set of existing query groups $S = \{s_1, \ldots, s_m\}$
  3) a similarity threshold $\tau_{sim}$, $0 \le \tau_{sim} \le 1$
Output: The query group $s$ that best matches $s_c$, or a
new one if necessary
( 0) $s = \emptyset$
( 1) $\tau_{max} = \tau_{sim}$
( 2) for $i = 1$ to $m$
( 3)   if $sim(s_c, s_i) > \tau_{max}$
( 4)       $s = s_i$
( 5)       $\tau_{max} = sim(s_c, s_i)$
( 6)   if $s = \emptyset$
( 7)       $S = S \cup s_c$
( 8)       $s = s_c$
( 9) return s
```

Fig.1 **Algorithm for selecting the query group that is similar to the given query and clicked URLs.**

The core of the solution is a measure of relevance between two queries. We will further motivate the need to go beyond baseline relevance measures that rely on time or text and instead propose a relevance measure based on signals from search logs. A complex task such as travel arrangement has to be broken down into a number of co-dependent steps over a period of time. For instance, a user may first search on possible destinations, timeline, events, etc. After deciding when and where to go, the user may then search for the most suitable arrangements for air tickets, rental cars, lodging, meals, etc. Each step requires one or more queries, and each query results in one or more clicks on relevant pages.

```
Relevance (q)

Input:

1) The query fusion graph, QFG

2) The jump vector, g

3) The damping factor, d

4) The total number of random walks, numRWs

5) The size of neighborhood, maxHops

6) The given query, q
```

Fig.2 Input sequence algorithm

Keyword based search engines cannot address this kind of complicated tasks. So a better system is required that can enable the user to pursue complex search quests online. Search Engine tries to construct user profile based on his ipaddress/login credentials from its user search history repositories. If the user already exists, the search engine checks from its user search history repositories up to a certain threshold whether the user already queried the same query previously.

By using the random walks over query fusion graph input is in fig2.We use the jump vector gq to pick the random walk starting point. At each node v the random walk either continues by following one of the outgoing edges of v with a probability of d for a given damping factor d. Each outgoing edge i.e. (v, qi) is selected with probability and the random walk always re-starts if v has no outgoing edge.

```
( 0) Initialize $\mathrm{rel}_q^F = 0$
( 1) $numWalks = 0$; $numVisits = 0$
( 2) while $numWalks < numRWs$
( 3)   $numHops = 0$; $v = q$
( 4)   while $v \neq NULL \wedge numHops < maxHops$
( 5)     $numHops{+}{+}$
( 6)     $\mathrm{rel}_q^F(v){+}{+}$; $numVisits{+}{+}$
( 7)     $v = SelectNextNodeToVisit(v)$
( 8)   $numWalks{+}{+}$
( 9) For each $v$, normalize $\mathrm{rel}_q^F(v) = \mathrm{rel}_q^F(v)/numVisits$
```

Fig.3. Output relevance Vector

Selection of the next node to visit based on the outgoing edges of the current node v in QFG and the damping factor d is performed by the Select next node to visit process in Step (7) in the fig.3.

If the user did, then search engine further retrieves click points from user search history repositories and reformulates query results by generating click graphs. Click graphs contain useful information on user behavior when searching online. This step is called query fusion graph. Uses random walk propagation over the query fusion graph instead of time-based and keyword similarity based approaches. This entire process is called organizing user search histories into query groups. This approach helps users to pursue complex search quests online. The clicks of a user may further help us infer her search interests behind a query q and thus identify queries and query groups relevant to q more effectively.

IV. PROPOSED SYSTEM

Random walk propagation over the query fusion graph methods support complex search quests in IR systems. For making the IR Systems effective and dynamic we propose to use these search quests as auto complete features in similar query propagations. Biasing the ranking of search results can also be provided using any ranking algorithms (top-k algorithms). Supporting these methods yields dynamic performance in IR systems, by providing enriched user querying experience. The main characteristics that guide our choice are the following: As query logs usually are more, the algorithm approach should be capable of handling a large data set within reasonable time and space constraints.

Our goal is to automatically organize a user's search history into query groups, each containing one or more related queries and their corresponding clicks. Query group corresponds to an atomic information need that may require a small number of queries and clicks related to the same search goal. They highlight the importance of external problem representation and evaluation in problem solving that can be supported by search histories. The History displays have to incorporate both analytical searches and hypertext browsing in full-text systems.

Our study investigates the effectiveness of personalized search based upon user profiles constructed from user search histories returned results and the Web pages selected from results retrieved is collected [7]. Search results are also classified into the same concept hierarchy, and the match between the user profile concepts and result concepts are used to re-rank the search results. User interests are collected in a completely non-invasive and search personalized is based upon data readily available to the search engine. We do not require the user to install a boot or use a proxy server to collect and share their browsing histories.

```
1: Let $k$ be the number of results requested;
2: Let $w_{max}$ be the maximum weight of a string in the
   dataset;
3: Let $f \geq 1$ be a multiplication factor;
4: Let $R \leftarrow \phi$ be the range-search-result set;
5: Let $\tau$ be the initial similarity threshold;
   {Step 1: Computing initial candidates}
6: while $size(R) < f \cdot k$ do
7:    $R \leftarrow ApproxRangeSearch(\tau)$;
8:    if $size(R) < f \cdot k$
9:    then Decrease $\tau$;
10: end while
   {Step 2: Finalizing results}
11: Compute scores for elements in $R$ and keep the first $k$;
12: Let $\tau_1$ be the minimum similarity for which
    $Score(\tau_1, w_{max}) > Score(R[k])$;
13: if $\tau_1 < \tau$ then
14:    $R \leftarrow ApproxRangeSearch(\tau_1)$;
15:    Compute scores for elements in $R$ and keep the first
       $k$;
16: end if
17: Return $R[1..k]$;
```

Fig.4TOP-K algorithm

STEP 1: Computing Initial Candidates

In the fig 4 lines 6-10 are step 1

The goal of step 1 is to compute at least f. K results, where f ≥ 1 is a multiplication factor. We call a function "ApproxRangeSearch" to run an approximate-string-range-search algorithm of our choice, and find the strings that pass the similarity threshold T. Next, we decrease the similarity threshold, depending on the number of results we got.

This step ends when we get at least f. k results.

STEP 2: Finalizing Results

In fig 4 lines 11-16 are step 2.

Here we compute the score of each element computed in step1, and keep the first k elements ordered by their scores. Next, we want to be certain that these k elements are indeed the best results. Consider one element e that was not seen before, and it has the maximum possible weight in the dataset. We compute how similar e needs to be to the query in order to have a better score than the current k^{th} element.

4.1 QUERY RELEVANCE USING SEARCH LOGS:

Our measure of relevance is aimed at capturing two important properties of relevant queries, namely:

- Queries that frequently appear together as reformulations
- Queries that have induced the users to click on similar sets of pages

We show how we can use these graphs to compute query relevance and how we can incorporate the clicks following a user's query in order to enhance our relevance metric. To identify relevant queries is to consider query reformulations that are typically found within the query logs of a search engine. To measure the relevance between two queries issued by a user makes use of the interval between the timestamps of the queries within the user's search history.

V. EXPERIMENTAL ANALYSIS

We study the behavior and performance of our algorithms on partitioning a user's query history into one or more groups of related queries. As we considered the example Bank of America can useful to our case study for the sequence of queries "caribbean cruise"; "bank of america"; "expedia"; "financial statement". We would expect two output partitions: first, {"Caribbean cruise", "expedia"} pertaining to travel-related queries. Second we would expect two output partitions: first, {"Caribbean cruise", "expedia"} pertaining to travel-related queries. Providing a continuously growing history record in the user interface is the most common use of search histories on the role of search histories formed the basis for designing search history interfaces. Interface design recommendations for displaying search history data are presented to feed the recorded information back to the user. User interface prototypes are included and described to illustrate some of the design recommendations. Search-history-based user interface functions are described organized around a scratchpad and a results collection tool. Query grouping algorithm relies heavily on the use of search logs in two ways:

- To construct the query fusion graph used in computing query relevance
- To expand the set of queries considered when computing query relevance

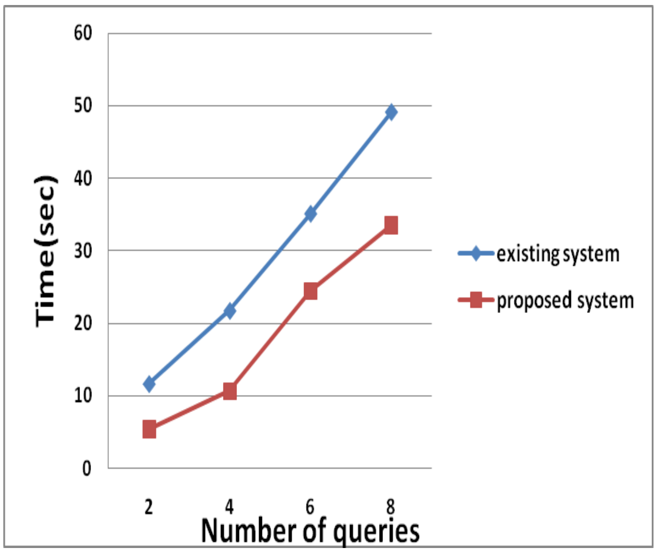

Fig 5: Comparison results for query fusion with top-k products.

As shown in the above figure

Horizontal axis is queries and vertical axis is time (seconds).

In the fig time efficiency is high in query fusion graph data extraction where as in our top–k preferences it will give less time when compared to prior approach. For example using query fusion graphs enter text data will be take more time for extracting user information from various data bases [4]. For decreasing these features of the extracting information we will develop biased ranking procedures in displaying keywords for extracting information from data base.

We use a time-based method that groups queries based on whether the time difference between a query and the most recent previous query is above a threshold. Since our QFG method relies on the accurate estimation of a query image within the query fusion graph. It is expected to perform better when the estimation was based on more information and is therefore more accurate. So by the top-k algorithm the results will be more speed than the fusion graph.

VI CONCLUSION

Web data extraction is the major concept in present day's for retrieving relevant information from various repositories. By using this context we show how such information can be used effectively for the task of organizing user search histories into query groups. For this instance traditionally developed techniques like query fusion graphs give appropriate information regarding this extraction. But it will take more time for retrieving that relevant information key word extraction. In this paper we propose extend biased ranking application on query fusion graph extension, then it automatically displays the keywords suggestion when user entered data by using top-k algorithm so it syntactically derives details of the query fusion graph representation. Experimental results show efficient data extraction based in the top-k key word arrangement in data repository.

REFERENCES

[1] Devang Karavadiya, Purnima Singh," User Specific Search Using Grouping and Organization", In Proceedings of International Journal of Emerging Trends & Technology in Computer Science, Volume 1, Issue 4, November – December 2012.

[2] Heasoo Hwang, Hady W. Lauw, Lise Getoor and AlexandrosNtoulas, "Organizing User Search Histories", IEEE 2012 Transactions on Knowledge and Data Engineering, Volume: 24 , Issue: 5.

[3] A. Spink, M. Park, B. J. Jansen, and J. Pedersen,"Multitasking during Web search sessions", Information Processing and Management, 2006.

[4] E. Sadikov, J. Madhavan, L. wang and A. Halevy,"Clustering query refinements by user intent," in WWW, 2010.

[5] J. Yi and F.Maghoul, "Query clustering using click-through graph," in WWW, 2009

[6] D. Beeferman and A. Berger, "Agglomerative clustering of a search engine query log," in KDD, 2000.

[7] J. Teevan, E. Adar, R. Jones, and M. A. S. Potts, "Information reretrieval: repeat queries in yahoo's logs," in SIGIR. New York, 2007.

[8] Komlodi, A," Search history for user support in information seeking interfaces, University of Maryland"; College Park -2002.

[9] T. Joachims," Optimizing search engines using click through data", In Proceedings of SIGKDD 2002

[10] D. Kelly and J. Teevan," Implicit feedback for inferring user preference: A Bibliography", SIGIR Forum, 2003.

[11] J. Han and M. Kamber, "Data Mining: Concepts and Techniques",Morgan Kaufmann, 2000.

[12] A. Broder, "A taxonomy of web search," SIGIR Forum, 2002.