IJDCST @Feb-March Issue- V-2, I-3, SW-04
ISSN-2320-7884 (Online)
ISSN-2321-0257 (Print)

User Evaluation Using Query Pattern Resolution

[1] M V S Sandeep ,[2] G J Sunny Deol.
[1]Mtech, Dept. of CSE, Sri Mittapalli College of Engineering,Tummalapalem,Gutur.
[2]Assistant Professor , Sri Mittapalli College of Engineering,Tummalapalem,Gutur.

Abstract: However, the primary means of accessing information online is still through keyword queries to a search engine. 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. One important step towards enabling services and features that can help users during their complex search quests online is the capability to identify and group related queries together. 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 click, Bing search engine on February of 2010. This history includes a sequence of four queries displayed in reverse chronological order together with their corresponding clicks. In addition to viewing their search history, users can manipulate it by manually editing and organizing related queries and clicks into groups, or by sharing them with their friends. While these features are helpful, the manual efforts involved can be disruptive and will be untenable as the search history gets longer over time.

Keywords: query group, query group relevance, query logs and query clustering.

I. Introduction

Information seeking skills have become more important in the last few decades as large amounts of easy-to-access information in everyday life became prevalent through electronic means and end users started searching for information in computerized sources. Factors enhancing and supporting information seeking vary from physical tools (print and electronic) to human and electronic intermediaries and specific skills and knowledge. The overall goal of this dissertation is to make searchers' jobs easier in finding information in electronic environments.

The dissertation sets out to examine searchers' behavior in order to identify and describe search history use and areas of potential use. A thorough description of the nature and role of search histories will form a theoretical framework on which to base interface designs. This framework will be developed through several iterations. History information in information-seeking environments can be applied in many different areas. This research aims to identify potential areas of use for automatically and manually recorded history information to enhance information-seeking interfaces [1].

Information seeking as a process is part of the larger task of the user. When searchers look for

information using the same computer they use for creating documents or for other tasks, the search system should help seamlessly embed searching into the larger work process context. Recording the history of actions not only in searching, but also in other processes can help create a continuum between the various tasks. The recorded search history can also help customize systems to users' needs by analyzing log of previous actions.

Another dimension of integration is extending or sharing search histories with other users. Recorded histories are good candidates for sharing with others, searchers often record this information in order to share it. Although this topic is not at the center of the dissertation, the implications of sharing search histories are too strong to ignore and are discussed.

The goal of the dissertation is to provide a foundation for designing improved information-seeking user interfaces that incorporate search history data. Search histories provide a continuity between past, present and future actions through making these more easily available. History information can be utilized in human computer interfaces in three ways. Explicit search history displays can give users overviews of the whole of the search process, navigation aids between the different step and even tools for further query formulation or information exploration. Search history information can be integrated in other parts of information-seeking interfaces as well. They can enhance displays by showing relationships between steps (e.g.) result lists by showing what items have been returned previously, can contribute to relevance feedback and recommendation systems, and so on. This implicit use of history information needs to be part of any consideration of interface designs building on this information. A third area of application for search histories involves interface tools built on the availability of search histories, or tools provided to complement and further manage search histories. Tools in this category can include features to transfer information from finding to using or tools to help searchers organize results collected [1].

II. Related Work

Teevan and Eytan Adar stated that People often repeat Web searches, both to find new information on topics they have previously explored and to re-find information they have seen in the past. The query associated with a repeat search may differ from the initial query but can nonetheless lead to clicks on the same results. This paper explores repeat search behavior through the analysis of a one-year Web query log of 114 anonymous users and a separate controlled survey of an additional 119 volunteers. Our study demonstrates that as many as 40% of all queries are re-finding queries. Re-finding appears to be an important behavior for search engines to explicitly support, and we explore how this can be done. We demonstrate that changes to search engine results can hinder re-finding, and provide a way to automatically detect repeat searches and predict repeat clicks.

Amanda Spink and Minsoo Park stated that A users single session with a Web search engine or information retrieval (IR) system may consist of seeking infor mation on single or multiple topics, and switch between tasks or multitasking information behavior. Most Web search sessions consist of two queries of approximately two words. However, some Web search sessions consist of three or more queries.

We present findings from two studies. First, a study of two-query search sessions on the AltaVista Web search engine, and second, a study of three or more query search sessions on the AltaVista Web search engine. We examine the degree of multitasking search and information task switching during these two sets of AltaVista Web search sessions. A sample of two-query and three or more query sessions were filtered from AltaVista transaction logs from 2002 and qualitatively analyzed. Sessions ranged in duration from less than a minute to a few hours. Findings include: (1) 81% of two-query sessions included multiple topics, (2) 91.3% of three or more query sessions included mu tiple topics, (3) there are a broad variety of topics in multitasking search sessions, and (4) three or more query sessions sometimes contained frequent topic changes. Multitasking is found to be a growing element in Web searching. This paper proposes an approach to interactive information retrieval (IR) contextually within a multitasking framework.

Rosie Jones and Kristina Lisa Klinkner stated that Most analysis of web search relevance and performance takes a single query as the unit of search engine interaction. When studies attempt to group queries together by task or session, a timeout is typically used to identify the boundary. However, users query search engines in order to accomplish tasks at a variety of granularities, issuing multiple queries as they attempt to accomplish tasks. In this work we study real sessions manually labeled into hierarchical tasks, and show that timeouts, whatever their length, are of limited utility in identifying task boundaries, achieving a maximum precision of only 70%. We report on properties of this search task hierarchy, as seen in a random sample of user interactions from a major web search engine's log, annotated by human editors, learning that 17% of tasks are interleaved, and 20% are hierarchically organized. No previous work has analyzed or addressed automatic identification of interleaved and hierarchically organized search tasks. We propose and evaluate a method for the automated segmentation of users' query streams into hierarchical units. Our classifiers can improve on timeout segmentation, as well as other previously published approaches, bringing the accuracy up to 92% for identifying fine-grained task boundaries, and 89-97% for identifying pairs of queries from the same task when tasks are interleaved hierarchically. This is the first work to identify, measure and automatically segment sequences of user queries into their hierarchical structure. The ability to perform this kind of segmentation paves the way for evaluating search engines in terms of user task completion.

Paolo Boldi and Francesco Bonchi stated that Query logs record the queries and the actions of the users of search engines, and as such they contain valuable information about the interests, the preferences, and the behavior of the users, as well as their implicit feedback to searchengine results. Mining the wealth of information available in the query logs has many important applications including query-log analysis, user profiling and personalization, advertising, query recommendation, and more. In this paper we introduce the query-flow graph, a graph representation of the interesting knowledge about latent querying behavior. Intuitively, in the query-flow graph a directed edge from query qi to query qj means that the two queries are likely to be part of the same "search mission". Any path over the query-flow

graph may be seen as a searching behavior, whose likelihood is given by the strength of the edges along the path.

III. Existing System

Dynamic Query Grouping:

One approach to the identification of query groups is to first treat every query in a user's history as a singleton query group, and then merge these singleton query groups in an iterative fashion (in a k-means or agglomerative way). However, this is impractical in our scenario for two reasons. First, it may have the undesirable effect of changing a user's existing query groups, potentially undoing the user's own manual efforts in organizing her history. Second, it involves a high computational cost, since we would have to repeat a large number of query group similarity computations for every new query. As in online clustering algorithms [9], we perform the grouping in a similar dynamic fashion, whereby we first place the current query and clicks into a singleton query group sc = {qc, clkc}, and then compare it with each existing query group si within a user's history (i.e., si 2 S). The overall process of identifying query groups is presented in Figure. Given sc, we determine if there are existing query groups sufficiently relevant to sc. If so, we merge sc with the query group s having the highest similarity max above or equal to the threshold sim. Otherwise, we keep sc as a new singleton query group and insert it into S.

Query (or Query Group) Relevance:

To ensure that each query group contains closely related and relevant queries and clicks, it is important to have a suitable relevance measure sim between the current query singleton group sc and an existing query group si 2 S. There are a number of possible approaches to determine the relevance between sc and si. Below, we outline a number of different relevance metrics that we will later use as baselines in experiments. We will also discuss the pros and cons of such metrics as well as our proposed approach of using search logs . Time. One may assume that sc and si are somehow relevant if the queries appear close to each other in time in the user's history. In other words, we assume that users generally issue very similar queries and clicks within a short period of time. In this case, we define the following time-based relevance metric simtime that can be used in place of sim in Figure.

Select Best Query Group
Input:
1) the current singleton query group sc containing the current query qc and set of clicks clkc
2) a set of existing query groups S = {s1, . . . , sm}
3) a similarity threshold sim, 0 sim 1
Output: The query group s that best matches sc, or a new one if necessary
(0) s = ;
(1) max = sim
(2) for i = 1 to m
(3) if sim(sc, si) > max
(4) s = si
(5) max = sim(sc, si)
(6) if s = ;
(7) S = S [sc
(8) s = sc
(9) return s

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

IV. Proposed System

QUERY RELEVANCE USING SEARCH LOGS

We now develop the machinery to define the *query relevance* based on Web search logs. Our measure of relevance is aimed at capturing two important properties of relevant queries, namely: (1) queries that frequently appear together as reformulations and (2) queries that have induced the users to click on similar sets of pages. We start our discussion by introducing three search behavior graphs that capture the aforementioned properties. Following that, 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.

Computing Query Relevance:

Having introduced the search behavior graphs in the previous section, we now compute the relevance between two queries. More specifically, for a given user query q, we compute a relevance vector using QFG, where each entry corresponds to the relevance value of each query qj ϵ VQ to q.

The edges in QFG correspond to pairs of relevant queries extracted from the query logs and the click logs. However, it is not sufficiently effective to use the pairwise relevance values directly expressed in QFG as our query relevance scores. Let us consider a vector rq, where each entry, rq(qj), is wf (q, qj) if there exists an edge from q to qj in QFG, and 0 otherwise. One straightforward approach for computing the relevance of qj to q is to use this rq(qj) value. However, although this may work well in some cases, it will fail to capture relevant queries that are not directly connected in QFG (and thus rq(qj) = 0).

Therefore, for a given query q, we suggest a more generic approach of obtaining query relevance by defining a Markov chain for q, MCq, over the given graph, QFG, and computing the stationary distribution of the chain.We refer to this stationary distribution as the fusion relevance vector of q, relF q , and use it as a measure of query relevance throughout this paper.

In a typical scenario, the stationary probability distribution of MCq can be estimated using the matrix multiplication method, where the matrix corresponding to MCq is multiplied by itself iteratively until the resulting matrix reaches a fixpoint. However, given our setting of having thousands of users issuing queries and clicks in real-time and the huge size of QFG, it is infeasible to perform the expensive matrix multiplication to compute the stationary distribution whenever a new query comes in. Instead, we pick the most efficient Monte Carlo random walk simulation method among the ones presented in, and use it on QFG to approximate the stationary distribution for q. Figure 2 outlines our algorithm.

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

Output: the fusion relevance vector for q, relF
q
(0) Initialize relF
q = 0
(1) numWalks = 0; numVisits = 0
(2) **while** numWalks < numRWs
(3) numHops = 0; v = q
(4) **while** v 6= NULL ^ numHops < maxHops
(5) numHops++
(6) relF
q (v)++; numVisits++
(7) v = SelectNextNodeToVisit (v)
(8) numWalks++
(9) For each v, normalize relF
q (v) = relF
q (v)/numVisits

Fig. 2. Algorithm for calculating the query relevance by simulating random walks over the query fusion graph.

V. Experimental Results

Experimental Setup:

We study the behavior and performance of our algorithms on partitioning a user's query history into one or more groups of related queries. For example, for the sequence of queries "Caribbean cruise"; "bank of America"; "expedient"; "financial statement", we would expect two output partitions: first, {"Caribbean cruise", "expedia"} pertaining to travel-related queries, and, second, {"bank of America", "financial statement"} pertaining to money-related queries.

Using Search Logs

Our query grouping algorithm relies heavily on the use of search logs in two ways: first, to construct the query fusion graph used in computing query relevance, and, second, to expand the set of queries considered when computing query relevance. We start our experimental evaluation, by investigating how we can make the most out of the search logs. In our first experiment, we study *how we should combine* the query graphs coming from the query reformulations and the clicks within our query log.

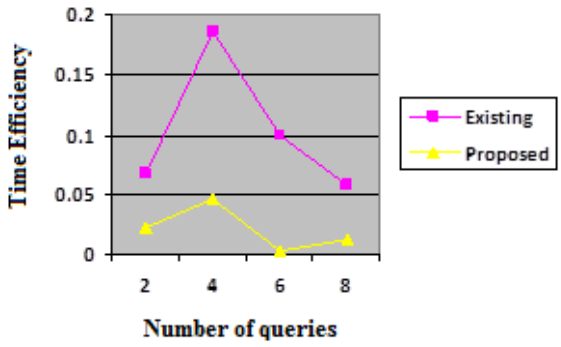

Figure 3: Varying query results in both existing and proposed approaches.

Above graph describes the horizontal axis represents _ (i.e., how much weight we give to the query edges coming from the query reformulation graph), while the vertical axis shows the performance of our algorithm in terms of the RandIndex metric.

VI. Conclusion

The Query formulations based on click graphs contain useful information on user behavior when searching online. For this process we are using different informative techniques like page rank operations for analyzing the user histories. In this paper we propose to develop the efficient data extraction based on click graph results. We also find

value in combining our method with keyword similarity-based methods, especially when there is insufficient usage information about the queries. As future work, we intend to investigate the usefulness of the knowledge gained from these query groups in various applications such as providing query suggestions and biasing the ranking of search results.

VII. References

[1]. SEARCH HISTORY FOR USER SUPPORT IN INFORMATION-SEEKING INTERFACES by Anita Hajnalka Komlódi, Doctor of Philosophy, 2002.

[2]. Information Re-Retrieval: Repeat Queries in Yahoo's Logs by Jaime Teevan , Eytan Adar.

[3]. Multitasking during Web search sessions by Amanda Spink and Minsoo Park.

[4]. Beyond the Session Timeout: Automatic Hierarchical Segmentation of Search Topics in Query Logs by Rosie Jones and Kristina Lisa Klinkner.

[5]. The Query-flow Graph: Model and Applications by Paolo Boldi and Francesco Bonchi.

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

[7] P. Anick, "Using terminological feedback for web search refinement: A log-based study," in *SIGIR*, 2003.

[8] B. J. Jansen, A. Spink, C. Blakely, and S. Koshman, "Defining a session on Web search engines: Research articles," *Journal of the American Society for Information Science and Technology*, vol. 58, no. 6, pp. 862–871, 2007.