A Survey on Organizing user Search Histories

Rupali D. Navagire
Department of Computer Engineering
Smt. Kashibai Navale College of Engineering.
Pune, India

Prof. Trupti H. Gurav
Department of Computer Engineering
Smt. Kashibai Navale College of Engineering.
Pune, India

***Abstract*— As the size of information on web is growing, the use of it to accomplish variety of task is also growing. Peoples are started using Internet to do many tasks such as managing finances, planning purchase decision etc. To help users in their information search, search engines maintains user search history consisting of their clicks and query. However, the search histories are not well organized. In this paper, we study the problem of organizing user's historical queries into groups of related queries in a dynamic and automated fashion.**

Keywords*— *click graph, queries reformulation , query clustering, search engine, user log.

I. INTRODUCTION

Large amount of data is available on the web and is continuously growing every day. Out of this morass of data, users typically search for the relevant information that they want by posing search queries to search engines. End users are no longer content with issuing simple queries. Various studies on query logs reveal that only about 20 percent of queries are Hierarchal. The remaining queries are informational or transactional. The problem that the search engines face is that the queries are very diverse. Most of the individual queries may refer to a single concept, while a single query may correspond to several techniques.

To increase usability of internet, most commercial search engines provide some additional services such as query recommendation or query suggestion. These services make it more convenient for users to issue queries and obtain accurate results from the web search engine, and thus it is quite valuable. From the search engine view, efficient group of search queries is a necessary pre-requisite for these services to function well. Following are the some applications where groupings of related queries can be used.

1) To help the users keep track of clicks and queries in their search history.
2) Query grouping can also help other users. For example, if a set of query groups created by expert users is given, we can recommend queries that are highly relevant to the current user's query activity.
3) Query grouping can also help to improve some of the features and services provided by search engine such as query suggestions, result ranking, query alterations etc.

So in this paper we study the problem of organization of user's search history into a set of related query groups. Our approach is to generate these query groups automatically and dynamically.

It is challenging to organize queries into related query group. Following are the some of the reasons. Firstly, it is possible that related queries may not appear close to one another .They may be separated by many unrelated queries. In this case, the approaches that rely on time or sequence to identify related queries may not work efficiently. Therefore, it is not good to rely solely on time based approaches. Secondly, they may not be textually similar but have the same semantics. Therefore, approaches relying solely on string similarity will not work also. Our approach does not rely on time-based and text-based measure but uses information from query log.

The rest of the paper is organized as follows. In Section 2, we state the goal of our paper. In Section 3, we review the related work. In Section 4, we explain how to construct the query fusion graph from search logs. Also we explain how to determine relevance value for each query in user's history and perform query grouping. We describe proposed system in section 5. We conclude with a discussion on our results and future research directions in Section 6.

II. PRELIMINARIES

A. Goal

Our goal is to automatically organize a user's search history into query groups. A query group is an ordered list of queries, q_i, together with the corresponding set of clicked URLs, clk_i of q_i. A query group is denoted as $s = (\{q_1, clk_1\}, \{q_k, clk_k\})$.

The specific formulation of our problem is as follows:

- Given: a set of existing query groups of a user, $S= \{s_1, s_2, ..., s_n\}$ and her current query and clicks $\{q_c, clk_c\}$.
- Find: the query group for $\{q_c , clk_{c}\}$, which is either one of the existing query groups in S that it is most related to, or a new query group $s_c=\{q_{c,} clk_{c}\}$ if there does not exist a query group in S that is sufficiently related to $\{q_c, clk_c\}$.

The core of the solution is a measure of relevance between two queries (or query groups) that not only rely on time or text but also we propose a relevance measure based on signals from search logs.

B. Dynamic Query Grouping

One approach to identify query groups is by using online clustering algorithms [2]. In this approach first we place the current query and clicks into a singleton query group $s_c = \{q_c, clk_c\}$, and then compare it with each existing query group s_i within a user's history (i.e., $s_i \in S$) to find the best match. SelectBestQueryGroup algorithm is used to select the query group that is the most similar to the given query and clicked URLs as given below.

SelectBestQueryGroup
Input:

1) The current singleton query group sc 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 τ_{sim}
4) Query fusion vector $rel_{(qc,clkc)}$

Output: The query group s that best matches s_c, or a new one if necessary

1) $\tau_{max} = \tau_{sim}$
2) **for** all group s_i in query groups set S
3) **if** $sim(s_c, s_i) > \tau_{max}$
4) $s = s_i$
5) $\tau_{max} = sim(s_c, s_i)$
6) **if** $s = \Phi$ and $rel_{(qc,clkc)} > \tau_{sim}$
7) $S = S \cup s_c$
8) $s = s_c$
9) **return** s

C. Query Relevance

It is important to have a suitable relevance measure sim between the current query singleton group s_c and an existing query group $s_i \in S$. There are a number of possible approaches to determine the relevance between s_c and s_i. Many relevance matrics are either Time-based or text-based [6] and [7].But these relevance metrics may work well in some cases, they cannot capture certain aspects of query similarity as discussed in section I. Therefore, we need a more robust relevance measure. Our approach makes use of search logs in order to determine the relevance between query groups more effectively. We will discuss our proposed relevance measure in greater detail in Sections 4 and 5.

III. RELATED WORK

In recent work, Jones and Klinkner [8] worked on search-task identification problem. He constructed a query flow graph to solve the problem. Our work is different from these prior works as we consider query pairs having common clicked URLs and we also exploit both co-occurrence and click information through a combined query fusion graph. Some prior work [9] and [10] proposed segmentation of a user's query streams into "sessions" based on a "time-out threshold". But time is not a good basis for identifying query groups because related queries may not appear close to one another. Keyword-based query grouping has provided interesting results. However, because, specifically the queries submitted to the web search engines usually are very short; in many cases it is hard to deduce the semantics from the query itself. Therefore, keywords alone do not provide a reliable basis for grouping queries effectively.

Radlinski and Joachims [11] employed a classifier that combines a timeout threshold with textual similarity features of the queries to identify query sequences. While text similarity may work in some cases, it may fail to capture cases where there is "semantic" similarity between queries.

The problem of query clustering [12] and [13] is also related to online query grouping. In Beeferman and Berger [12] and Baeza-Yates and Tiberi [13], commonly clicked URLs on query-click bipartite graph are used to cluster queries. Wen et al. [14] proposed a query clustering algorithm that considers both query contents and URL clicks. They assumed that two queries are related to each other, if they contain the same or similar terms, and lead to the selection of the same documents. However, since Web search queries contain less keyword and common clicks on documents are rare, Wen et al.'s method may not be very effective. While these prior work make use of click graphs, our approach is much better in that we use the click graph in combination with the reformulation graph.

IV. QUERY RELEVANCE

A. Constructing Query Graphs

We assume that queries that frequently appear together are relevant. Also queries that have induced the users to click on similar sets of pages are relevant. So we are considering both these important properties of relevant queries to measure query relevance. We derive three types of graphs from the search logs of a commercial search engine. These three graphs are: Query Reformulation Graph, Query Click Graph and Query Fusion Graph.

The query reformulation graph, $QRG = (V_Q, E_{QR})$, captures the first important property of related queries. We construct a query click graph, $QCG = (V_Q, E_{QC})$ by constructing CG =(VQ U VU; EC), used by Fuxman et al.[6] and then we derive our query click graph, $QCG = (V_Q, E_{QC})$. In QCG ,the vertices are the queries. If there exists at least one URL that both qi and qj link to in CG we draw a directed edge from qi to qj in QCG.

We construct $QFG = (V_Q, E_{QF})$, by combining QRG and QCG into a single graph, that we refer to as the query fusion graph.

B. Computing Query Relevance

Having constructed QFG, we now compute the relevance between two queries. Relevance Algorithm is used for calculating the query relevance by simulating random walks over the query fusion graph.

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, rel^F_q

1) Initialize $rel^F_q = 0$
2) numWalks = 0; numVisits = 0
3) while numWalks < numRWs
4) numHops = 0; v = q
5) while v = NULL ^ numHops < maxHops
6) numHops++
7) $rel^F_q(v)$++; numVisits++
8) v = SelectNextNodeToVisit (v)
9) numWalks++
10) For each v, normalize $rel^F_q(v) = rel^F_q(v)/numVisits$

This algorithm computes the fusion relevance vector of a given query q, rel^F_q.

The algorithm works as follows: jump vector g_q is used to pick up the starting point for the random walk. At each node v, the random walk either continues by following one of the outgoing edges of v or stops or restarts at one of the starting points in gq. The selection of the next node to visit is based on the outgoing edges of the current node v in QFG.

C. Creating query group using QFG

In this section, we explain our proposed similarity function sim_{rel} to be used in the online query grouping process. For each query, we maintain a query image. Query image contain all the queries related to the query and for each query group, we maintain a context vector. The similarity between the query group and the user's latest singleton query group is computed by using context vector. The context vector for a query group s, denoted cxt_s, is obtained by aggregating the fusion relevance vectors of the queries and clicks in s.The relevance between the user's latest singleton query group $s_c = (q_c, clk_c)$ and an existing query group $s_i \in S$ will be calculated as follow.

$$sim_{rel}(s_c, s_i) = \sum_{q \in \mathbf{I}(s_c) \cap \mathbf{I}(s_i)} \mathbf{rel}_{(\mathbf{q_c}, \mathbf{clk_c})}(q) * \sum_{q \in \mathbf{I}(s_c) \cap \mathbf{I}(s_i)} \mathbf{cxt}_{\mathbf{s_i}}(q).$$

Where,

S_c = singleton group
S_i = existing group
I = Image of query group
Q = current query
Cxt_{si} = Context vector of query group s_i.
rel (q_s, clk_s) = relevance between query q and corresponding url clk

This relevance metric sim_{rel} is used in the Step (5) of the SelectBestQueryGroup algorithm .

V. A PROPOSED SYSTEM

A proposed System architecture is shown in fig 1. It consists of the following major steps.

1. When a user submits a query, those queries and its associated clicks along with other information is stored in database.
2. Query Reformulation Graph and Query click graph are constructed. Using both graph third graph, Query Fusion Graph is constructed.
3. Query Fusion Graph is used to calculate fusion relevance vector for given query. Also query Images are maintained for each query.
4. Context vector is calculated for each existing group.
5. Fusion relevant vector of given query is compared with Context vector of each group to find the best match. If best match is found query is merged to that group otherwise new group is formed.

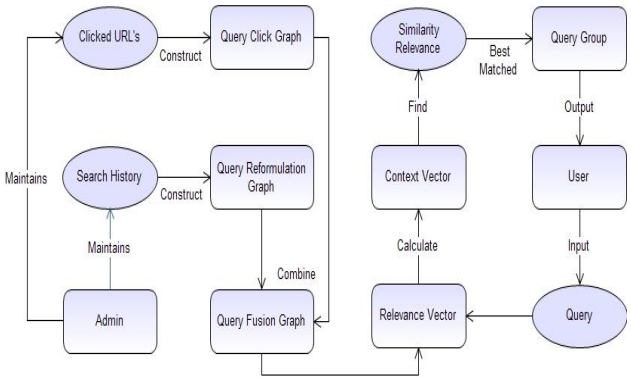

Fig.1. System architecture.

VI. CONCLUSION

Search engines maintain historical data but it is not well organized. Organizing user search histories have very important utilities. In this paper, we study a method to organize user search history logs into groups of related queries. The approach described in this paper is fully capable of grouping search engine queries. There are several directions for future work, including developing better treatment of ambiguous queries, and developing methods that uses the knowledge gained from these query groups to improve the search experience and to provide query suggestion. Our method of query similarity calculation can also be used in different contexts.

ACKNOWLEDGMENT

I would like to acknowledge and thank my guide, Prof. Trupti Gurav, Professor of Computer Science Department at SKNCOE, Pune for her valuable guidance, support and motivation.

REFERENCES

[1] Heasoo Hwang, Hady W. Lauw, Lise Getoor, and Alexandros Ntoulas" Organizing User Search Histories", IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 24, NO. 5, MAY 2012

[2] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna, "The Query-Flow Graph: Model and Applications," Proc. 17th ACM Conf. Information and Knowledge Management (CIKM), 2008.

[3] A. Fuxman, P. Tsaparas, K. Achan, and R. Agrawal, "Using the Wisdom of the Crowds for Keyword Generation," Proc. the 17th Int'l Conf. World Wide Web (WWW '08), 2008.

[4] Ji-Rong Wen And Jian-Yun Nie," Query Clustering Using User Logs", ACM Transactions on Information Systems, Vol. 20, No. 1, Pages 59–81, 2002.
[5] Wilfred Ng, Lin Deng and Dik Lun Lee," Mining User Preference Using Spy Voting for Search Engine Personalization", ACM Transactions on Internet Technologies, Vol. 7, No. 3, 2007.
[6] Lecture Notes in Data Mining, M. Berry, and M. Browne, eds. World Scientific Publishing Company, 2006.
[7] V.I. Levenshtein, "Binary Codes Capable of Correcting Deletions,Insertions and Reversals," Soviet Physics Doklady, vol. 10, pp. 707-710, 1966
[8] R. Jones and K.L. Klinkner, "Beyond the Session Timeout: Automatic Hierarchical Segmentation of Search Topics in Query Logs," Proc. 17th ACM Conf. Information and Knowledge Management (CIKM), 2008.
[9] D. He, A. Goker, and D.J. Harper, "Combining Evidence for Automatic Web Session Identification," Information Processing and Management, vol. 38, no. 5, pp. 727-742, 2002.
[10] R. Jones and F. Diaz, "Temporal Profiles of Queries," ACM Trans.Information Systems, vol. 25, no. 3, p. 14, 2007.
[11] F. Radlinski and T. Joachims, "Query Chains: Learning to Rank from Implicit Feedback," Proc. ACM Conf. Knowledge Discovery and Data Mining (KDD), 2005.
[12] D. Beeferman and A. Berger, "Agglomerative Clustering of a Search Engine Query Log," Proc. Sixth ACM SIGKDD Int'l Conf.Knowledge Discovery and Data Mining (KDD), 2000.
[13] R. Baeza-Yates and A. Tiberi, "Extracting Semantic Relations from Query Logs," Proc. 13th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD), 2007.
[14] A. Broder, "A Taxonomy of Web Search," SIGIR Forum, vol. 36,no. 2, pp. 3-10, 2002.