User Search History Using Query Clustering for Information Retrieval

Smruti Patel
Computer Engineering,
Silver Oak College of Engineering & Technology, Ahmedabad, Gujarat, India

***Abstract* - 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 online. Users are increasingly pursuing complex task oriented goals on the Web, such as making travel arrangements, managing finances or planning purchases. The existing search engines organize such queries only in chronological order. However, when the quires are grouped together based on the relevancy that might be very useful to users as they can reuse queries with ease In this paper, we study the problem of organizing a user's historical queries into groups in a dynamic and automated fashion. This organization of user search histories can have various real time utilities such as result ranking, query alternations, query suggestions, sessionization and collaborative search. We experimentally study the performance of different techniques, and showcase their potential, especially when combined together.**

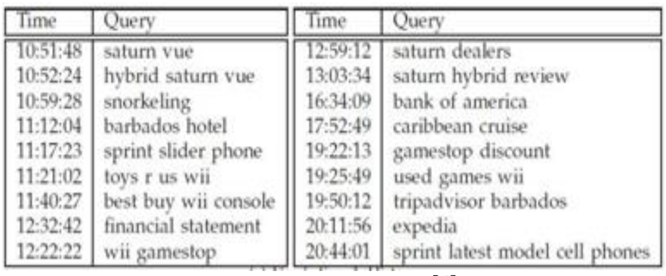
***Index Terms*- search history, query grouping, query reformulation, click graph**

I. INTRODUCTION

Clustering is a useful technique for the discovery of data distribution and patterns in the underlying data. The goal of clustering is to discover both the dense and the sparse regions in a data set. Clustering is the process of organizing data into meaningful groups, and these groups are called clusters. [3] Query Clustering itself means Query grouping. Query grouping allows the search engine to better understand a user's session and potentially tailor that user's search experience according to her need.

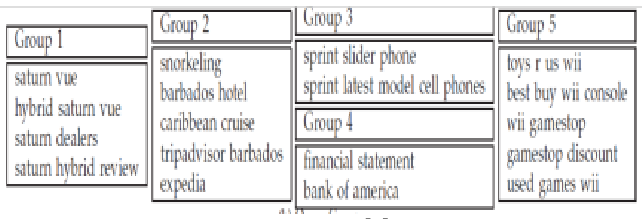
In this paper we study the problem of organizing a users search history into a set of query groups in an automatically and dynamic fashion. Each group is a collection of a query by the same user that is relevant to each other around a common information need. This query groups are dynamically update as the user issues new query, and query groups may be created over time.

World Wide Web is rich in information as it accumulates vast data every day from various sources. As there is content of all walks of life, people make searches in order to get required information. Thus the search engines are playing a great role in obtaining required information. It is very common that users give input to search engines in the form of key words. Search engines take the queries as input and come up with results. The users can make queries that can be reused when they are organized well. At present the search engines organize the history of searches in hierarchical fashion and in chronological order. [4]



Time	Query	Time	Query
10:51:48	saturn vue	12:59:12	saturn dealers
10:52:24	hybrid saturn vue	13:03:34	saturn hybrid review
10:59:28	snorkeling	16:34:09	bank of america
11:12:04	barbados hotel	17:52:49	caribbean cruise
11:17:23	sprint slider phone	19:22:13	gamestop discount
11:21:02	toys r us wii	19:25:49	used games wii
11:40:27	best buy wii console	19:50:12	tripadvisor barbados
12:32:42	financial statement	20:11:56	expedia
12:22:22	wii gamestop	20:44:01	sprint latest model cell phones

(a) User's Search History[4]



Group 1
saturn vue
hybrid saturn vue
saturn dealers
saturn hybrid review

Group 2
snorkeling
barbados hotel
caribbean cruise
tripadvisor barbados
expedia

Group 3
sprint slider phone
sprint latest model cell phones

Group 4
financial statement
bank of america

Group 5
toys r us wii
best buy wii console
wii gamestop
gamestop discount
used games wii

(b) Query groups [4]

Google kind of search engines is capable of organizing various search histories made by end users. However, the chronological order is not much useful to end users. They wanted to view the related queries together so that they can reuse queries.

As shown in figures (b), it is evident that the related quires are grouped together. This will help users to reuse such queries easily. Besides, the search engines can make use of these lists for various operations such as sessionization, query processing, query modifications, collaborative Search and so on. This kind of approach is also followed in for session identification and in for query clustering. [4] However, in this paper our work extends that in two ways. We use information from click graph and also query reformulation graph for capturing similarity in better way.

For example, if a search engine knows that a current query “financial statement” belongs to a {“bank of America”, “financial statement”} query group, it can boost the rank of the page that provides information about how to get a Bank of America statement instead of the Wikipedia article on “financial statement”, or the pages related to financial statements from other banks.

II. PROBLEM DEFINITION

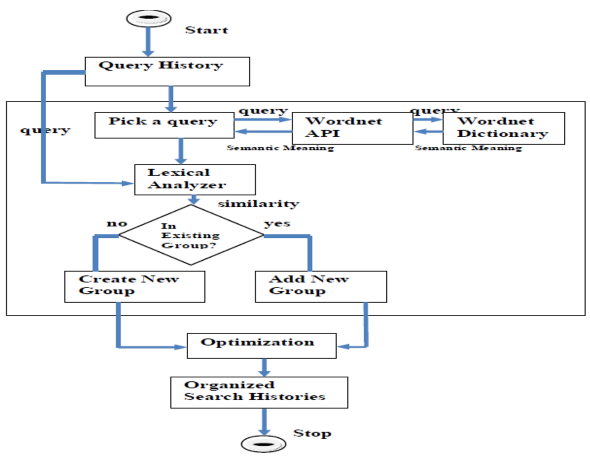
- Current clustering techniques do not address all the requirements adequately (and concurrently).
- It’s had to dealing with large number of data using query grouping techniques.
- The effectiveness of the method depends on the execution of process flow.
- Some query grouping techniques doesn’t give effective results when small number of logs history available.
- Results of query grouping techniques varies based on storage of search history data.

III. RELATED WORKS

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. [2] 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. [1]

The search engines like Google, Yahoo and Bing are organize user search histories in chronological order. The end users are to view the history of queries chronologically or date wise. Human beings issue quires based on their requirements. The queries might have repeated ones and semantically similar ones. In this project, the aim is to mine the queries given by end users over a period of time. Similar queries are to be grouped together. When similar queries are grouped, the results are very useful in many applications. On based on that we can find out that how many query and clicks for particular topic. This application widely used for making real time valuable decisions.

As you can see the semantic similarity approach in below figure. [4]The lexical analysis makes use of natural language processing in order to complete the similarity search. As you can see the flow of the system based on query we can check the similarity using lexical analyzer and based on similarity we can find out that is there any group existing for searched query or not, if yes than add new group otherwise create a new group. After that need to optimize the list and organize search histories.



However, very important thing here is the usage of WordNet dictionary which is a well known dictionary which has semantic meanings of words in English. In this project this dictionary is used in order to get semantic meanings.

Now we proceed towards methodology and terminology of some required terms. [2]

A. Definition: Query Group.

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 $s = (\{q_1, clk_1\}, \ldots, \{q_k, clk_k\})$.

1) 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.

2) 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 s_c and an existing query group s_i □ S. There are a number of possible approaches to determine the relevance between s_c and s_i.

B. Definition: Time.

One may assume that s_c and s_i 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.

C. Definition: Jaccard.

On a different note, we may assume that two query groups are similar if their queries are textually similar.
Textual similarity between two sets of words can be measured by metrics such as the fraction of overlapping
words or characters.

D. Definition: Levenshtein.

This is the same matrics as Jaccard but instead of words we can consider here a character. So basically Textual similarity between two sets of character can be measured by Levenshtein metrics.

E. Definition: CoR.

Co-Retrieval (or CoR) is based on the principle that a pair of queries is similar if they tend to retrieve similar pages on a search engine. Unlike previous metrics which relies on textual comparison, we compare
two queries based on the overlap in pages retrieved. We consider a page to be retrieved by a search engine if it has not only been shown to some users, but has also been clicked at least once in the past one year.

F. Definition: ATSP

This technique is based on the principle that two queries issued in succession in the search logs are closely
related. First reorders a sequence of user queries to group similar queries together by solving an instance of the
Asymmetric Traveler Salesman Problem (ATSP).

F. Search Behavior Graphs

We derive three types of graphs from the search logs of a commercial search engine. The *query reformulation*
graph, QRG, represents the relationship between a pair of queries that are likely reformulations of each other. The *query click graph*, QCG, represents the relationship between two queries that frequently lead to clicks on similar URLs. The *query fusion graph*, QFG, merges the information in the previous two graphs.

1) Query Reformulation Graph

One way to identify relevant queries is to consider *query reformulations* that are typically found within the
query logs of a search engine. If two queries that are issued consecutively by many users occur frequently enough, they are likely to be reformulations of each other. [4] To measure the relevance between two queries issued by a user, the timebased metric, sim_{time}, makes use of the interval between the timestamps of the queries within the user's search history. In contrast, our approach is defined by the statistical frequency with which two queries appear next to each other in the entire query log, over all of the users of the system. [1] [5]

2) Query Click Graph

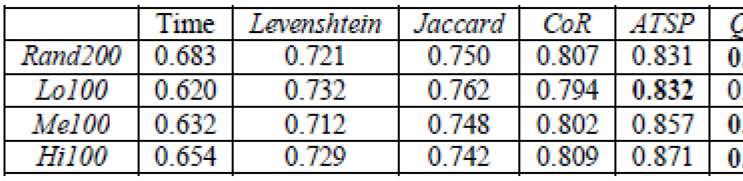
A different way to capture relevant queries from the search logs is to consider queries that are likely to induce
users to click frequently on the same set of URLs. For example, although the queries "ipod" and "apple store" do not share any text or appear temporally close in a user's search history, they are relevant because they are likely to have resulted in clicks about the ipod product. In order to capture such property of relevant queries, we construct a graph called the query click graph, QCG. [5]

3) Query Fusion Graph

The query reformulation graph, QRG, and the query click graph, QCG, capture two important properties of
relevant queries respectively. In order to make more effective use of both properties, we combine the query reformulation information within QRG and the query click information within QCG into a single graph, QFG = (V_Q, E_{QF}), that we refer to as the *query fusion graph*. [2] [4]

IV. PERFORMANCE COMPARISION

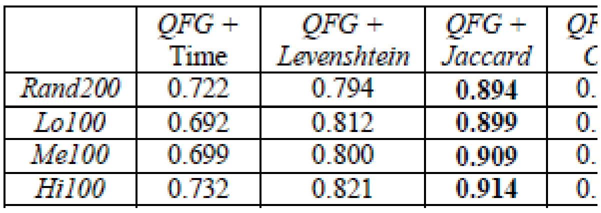
To gain a measure of usage information for a given user, we look at the average outdegree of the user's queries
(average outdegree), as well as the average counts among the outgoing links (average weight) in the query reformulation graph. In order to study the effects of usage information on the performance of our algorithms, we created three additional test sets of 100 users each. The sets were also manually labeled as we described in Section 5.1. The first set, *Lo100* contains the search activity of 100 users, with average outdegree < 5 and average weight < 5. Similarly, *Me100* contains user activity for users having 5 ≤ average outdegree < 10 and 5 ≤ average weight < 10, while *Hi100* contains user activity with average outdegree ≥ 10 and average weight ≥10.



	Time	*Levenshtein*	*Jaccard*	*CoR*	*ATSP*	*Q*
Rand200	0.683	0.721	0.750	0.807	0.831	0.
Lo100	0.620	0.732	0.762	0.794	**0.832**	0.
Me100	0.632	0.712	0.748	0.802	0.857	0.
Hi100	0.654	0.729	0.742	0.809	0.871	0.

TABLE 1. Comparative performance (RandIndex) of our methods. Best
performance in each dataset is shown in bold.

The techniques that we have studied so far fall into different categories and attempt to capture different aspects
of query similarities; *Time* simply looks at the time intervals, *Jaccard* and *Levenshtein* exploit textual similarities of queries, while *CoR*, *ATSP* and QFG use the search logs. Therefore, given the different natures of these algorithms it is reasonable to hypothesize that they do well for different kinds of queries. In particular, 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. On the other hand, if there are queries that are rare in the search logs or do not have many outgoing edges in our graph to facilitate the random walk, the graph-based techniques may perform worse due to the lack of edges. We study how the structure of the graph affects the performance of the algorithms as follows.



	QFG + Time	*QFG + Levenshtein*	*QFG + Jaccard*	*QF* *C*
Rand200	0.722	0.794	**0.894**	0.
Lo100	0.692	0.812	**0.899**	0.
Me100	0.699	0.800	**0.909**	0.
Hi100	0.732	0.821	**0.914**	0.

TABLE 2. Performance (RandIndex) of combined methods. Best
performance in each dataset is shown in bold.

The results of the previous experiment point out the contrast between the performances of the different methods.
This suggests that a combination of two methods may yield better performance than either method individually. We explore combining two methods by merging the output query groups as follows: given the output groups of any two methods, query pairs that belong to a group within one *or* within the other, will belong to the same group in the combined output. Table 2 shows the performance gained by combining QFG with each baseline.

In summary, from the experimental results, we observe that using the click graph in addition to query
reformulation graph in a unified query fusion graph helps improve performance. Additionally, the query fusion graph performs better for queries with higher usage information and handily beats time-based and keyword similarity-based baselines for such queries. Finally, keyword similarity-based methods help complement our method well providing for a high and stable performance regardless of the usage information.

V. CONCLUSION AND FUTURE WORK

The query reformulation and click graphs contain useful information on user behaviour when searching online. In this paper, we show how such information can be used effectively for the task of organizing user search histories into query groups. The future work intend to investigate the usefulness of the knowledge gained from these query groups in various applications such as providing query suggestions and ranking of search results.

REFERENCE

[1] Devang Karavadiya, Purnima Singh, "User Specific Search Using Grouping and Organization", International Journal of Emerging Trends & Technology in Computer Science (IJETTCS) 2012.
[2]in IJETT, "Performance Augementation of User Search Histories "Arshiya Kousar , Abdul Haq Syed 2013.
[3] " Design and Implementation of Query Clustering on User Search History For Effective Information Retrieval"Sattari Ajay, Hema K L. Vol. 2, 2014 in IJSRD.
[4] A.S.Saleem Basha, V.Trilik Kumar "Dynamic Grouping of Semantically Similar User Search Histories"(IJCTT) – volume 6 ,2013.
[5] M.Jhansi Rani, M.Ram Bhupal "Robust Approach For Query Grouping Using User Search Logs" International Journal of Research in Computer and Communication Technology, 2013
[6] J. Han and M. Kamber, "Data Mining: Concepts and Techniques",Morgan Kaufmann, 2000.
[7] A. Broder, "A taxonomy of web search," SIGIR Forum, 2002.
[8] A. Spink, M. Park, B. J. Jansen, and J. Pedersen,"Multitasking during Web search sessions", Information Processing and Management, 2006.
[9] Heasoo Hwang, Hady W. Lauw, Lise Getoor and Alexandros Ntoulas," Organizing User SearchHistories",in conf. IEEE Transactions on Knowledge
and Data Engineering,2012.
[10] D. Beeferman and A. Berger, "Agglomerative clustering of a search engine query log," in KDD, 2000.
[11] J. Teevan, E. Adar, R. Jones, and M. A. S. Potts, "Information reretrieval: repeat queries in yahoo's
logs," in SIGIR. New York, 2007.
[12] Komlodi, A," Search history for user support in information seeking interfaces, University of Maryland"; College Park -2002.
[13] T. Joachims," Optimizing search engines using click through data", In Proceedings of SIGKDD

2002, [9] D. Kelly and J. Teevan," Implicit feedback for inferring user preference: A Bibliography", SIGIR Forum, 2003.
[14] M. Speretta," Personalizing Search Based on User Search Histories", Master's thesis, The University of
Kansas, 2004.
[15]E. Agichtein, E. Brill, and S. Dumais, "Improving Web Search Ranking by Incorporating User Behavior
Information," Proc. ACMSIGIR, 2006.
[16] E. Agichtein, E. Brill, S. Dumais, and R. Ragno, "Learning User Interaction Models for Predicting
Web Search Result Preferences," Proc. ACM SIGIR, 2006.
[17]Aula, A., Jhaveri, N., & Käki, M," Information search and re-access strategies of experienced Web users.", Proceedings of the 14th international conference on World Wide Web, New York:2005.