Volume 5, Issue 10, October-2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Grouping Queries of Search Logs in to Query Groups

Shaikh Ambreen Mohd Ibrahim*, Prof. Seema Kolkur
Computer Engineering & Mumbai University
Maharashtra, India

Abstract— As the information on the web is increasing, users are also increasingly pursuing complex task based goals on the web such as planning purchases, making travel arrangements and managing finances. To pursue these tasks, users usually break down the tasks into a few co-dependent steps and issue multiple queries around these steps many times for long period of time. To better the users information search on the web, search engines keep track of users queries and clicks while searching online. Identifying related queries into groups has many applications beyond helping the users in keeping track of queries and clicks in their search history. Query grouping allows the search engine to better understand a users session and potentially tailor users search experience according to their needs. After the query groups have been recognized, search engines can have a good understanding and representation of search aim behind the current query using queries and clicks in the corresponding query group. This will help to better the quality of other components of search engines such as query suggestions, result re-ranking, query alterations, sessionization and collaborative search. In this paper we study different techniques that can be used to organize users historical queries into groups. We study different techniques used for identifying related queries.

Keywords— Query Reformulation, Query Clicks, Search History, Textual and Temporal Similarity.

I. INTRODUCTION

As the information on the web grows, the variety and complexity of tasks that users try to accomplish online on web also grows. The users pursue much broader informational and task related goals such as managing finances, planning purchases online and making travelling arrangements. These queries are informational and transactional in nature. The primary means of accessing information online is through keyword queries to search engine. The important step that can help users during their complex online search quests is the capability to identify and group related queries together. Some major search engines have recently introduced a new "Search History" feature that allows users to track their online searches by recording their queries and clicks.

For example in Fig 1 a search engine shows users search history. This history includes a series of user queries displayed in reverse chronological order together with their corresponding clicks.

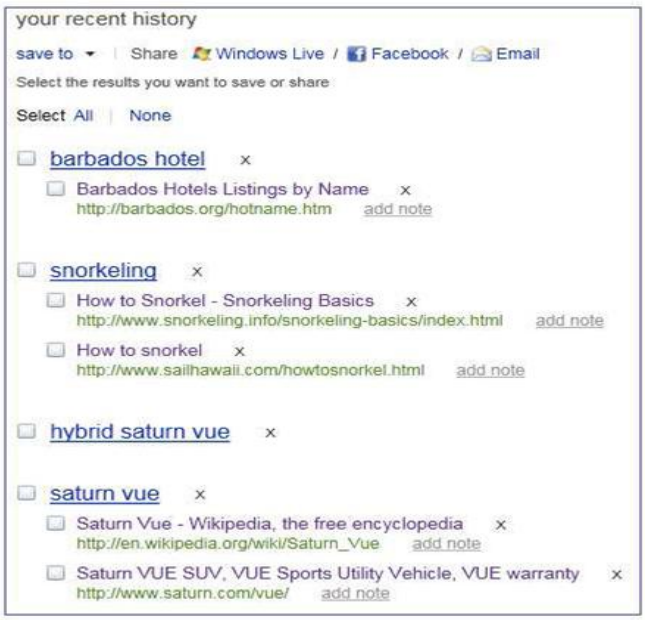

Fig. 1 Example of users search history

Users can view their search history and also can manipulate it by manually editing and organizing related queries and clicks into groups. As the search history gets longer over time manual efforts involved in grouping queries will not be very efficient and will be difficult. Identifying related queries into groups has many applications beyond helping the users

© 2015, IJARCSSE All Rights Reserved

in keeping track of queries and clicks in their search history. Query grouping allows the search engine to better understand a users session and potentially tailor users search experience according to their needs. After the query groups have been recognized, search engines can have a good understanding and representation of search aim behind the current query using queries and clicks in the corresponding query group. This will help to better the quality of other components of search engines such as query suggestions, result re-ranking, query alterations, sessionization and collaborative search. Query grouping can also help in task level collaborative search. For eg given a set of query groups created by users, we can select the ones that are highly relevant to current users activity and suggest them to the user.

Earlier in order to form query groups, string similarity functions were used. The various similarity functions like Text, Time, Jaccard, Levenshtein, CoRetrieval and Asymmetric Traveler Salesman Problem (ATSP) was used [1]. Computation of similarity relevance between two different strings is explained in [2]. Use of string similarity functions has drawbacks that it requires more time and problem of ambiguity arises. Users carry out various complex task oriented operations on web and each task is further divided into subtasks. Search task identification problem is studied for identifying different tasks that revolves around same search goal [3]-[4]. Implementation of query flow graph which links related queries helps in better understanding and grouping of queries [4]. The queries are linked by edge in query flow graph that are part of same search mission. Overlap of terms of queries is studied to detect changes in topic searches [5]-[6]. Implementation of query chains or sequences for grouping similar queries is done by using classifiers that combine two features, time threshold with textual similarity and the results returned [7]. Query groups based on query click logs have been used for different applications like query expansion, query suggestions, ranking search results etc [8]-[9].

II. PRELIMINARIES

Here, we study organization of users search history into query groups using different techniques, we also study problems faced during the organization .

Fig 2(a) shows user search history and Fig 2(b) shows query groups as shown below:

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

(a) User's Search History

Group 1	Group 2	Group 3	Group 4	Group 5
saturn vue	snorkeling	sprint slider phone	financial statement	toys r us wii
hybrid saturn vue	barbados hotel	sprint latest model cell phones	bank of america	best buy wii console
saturn dealers	caribbean cruise			wii gamestop
saturn hybrid review	tripadvisor barbados			gamestop discount
	expedia			used games wii

(b) Query Groups

Fig. 1 Example of Query groups formed from users history

The query groups consists of queries that are related to each other around a common informational need. These query groups are updated as the users issues new queries, and new query groups may be created over time.

Organizing the query groups in users history is challenging for various reasons.

1] First related queries may or may not appear close to one another, as the search task may span days or even weeks. This is further complicated by interleaving of queries and clicks from different search tasks due to users opening multiple browsers tabs, multitasking and frequently changing search topics.

2] The related queries are separated by many unrelated queries. Also the related queries may not be textually similar. Relying solely on string similarity is also insufficient. To achieve more effective query grouping we do not rely only on textual or temporal properties of queries. Instead we focus on search behavioral data as captured in search engines log.

Goal :- The goal is to automatically organize a users search history into query groups. Each query group consists of one or more related queries and their corresponding clicks. Each query group corresponds to information need that includes small number of queries and clicks related to same search goal.

Query Group:- A query group consists of list of queries, qi, together with the corresponding set of clicked URL's, clki of qi. A query group is denoted as $S = (\{q_1,clk_1\},... \{q_k,clk_k\})$.

Dynamic Query Grouping:- The query grouping is performed in 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 users history (i.e si $\in$ S). If there exists query groups sufficiently relevant to sc, we merge sc with the query group present in S having highest similarity, and if there is no query group similar to sc in S we keep sc as a new singleton query group and insert it into S.

Query Group Relevance :- To make sure that each query group contains related and relevant queries and clicks, it is important to have a suitable relevance measure "sim" between the current singleton query group sc and existing query

© 2015, IJARCSSE All Rights Reserved

group si, sj,… ∈ S. There are different possible ways to determine the relevance between current query group sc and an existing query group si. Below we describe a number of different relevance metrics. Also we discuss the merits and demerits of such metrics as well as a different approach of using search logs for grouping queries.

III. QUERY GROUPING BASED ON TIME AS RELEVANCE METRIC

One may believe that sc and si are somehow relevant if the queries appear close to each other in time in the users search history i.e we assume that users issue similar queries and clicks within a short period of time. In this scenario we define a time based relevance metric sim_{time} that can be used in place of sim.

$$sim_{time}(s_c, s_i) = \frac{1}{|time(q_c) - time(q_i)|} \_(1)$$

$sim_{time}(s_c, s_i)$ is defined as the inverse of time interval(in seconds) between the times that q_c and q_i are issued. The queries q_c and q_i are the most recent queries in s_c and s_i respectively. Higher the values of sim_{time} imply that the queries are temporally closer.

Eg. If queries qc , qi, qj are the most recent queries in sc, si and sj query group respectively. And qc, qi, qj are issued at 1258, 1225 and 1217 seconds respectively. Then qc is relevant to and belongs to the group si as qc and qi have higher sim_{time} values.

IV. QUERY GROUPING BASED ON TEXT AS RELEVANCE METRIC

Two query groups are similar if their queries are textually similar. Textual similarity is measured by relevance metrics such as fraction of overlapping words between two sets of words. Jaccard similarity is used to find overlapping words and Levenshtein similarity is used to find overlapping characters. The following two text based relevance metrics can be used in place of sim.

1) Jaccard: $sim_{jaccard}(s_c,s_i)$ is defined as the fragment of common words between q_c and q_i

$$\text{sim}_{\text{jaccard}}(s_c, s_i) = \frac{|\text{words}(q_c) \cap \text{words}(q_i)|}{|\text{words}(q_c) \cup \text{words}(q_i)|} \_(2)$$

Eg. The queries "tata motor cars" and "tata cars" are more similar as they will have higher $sim_{jaccard}$ values as compared to queries "automobiles" and "tata cars". The queries "tata motor cars" and "tata cars" have words in common thus will have higher $sim_{jaccard}$ value as compared to "automobile" query.

2) Levenshtein : $sim_{edit}(s_c,s_i)$ is defined as $|1 - \text{dist}_{\text{edit}}(q_c, q_i)|$. The edit distance $dist_{edit}$ is the number of character insertions, deletions, or substitutions required to transform one series of characters into another.

Eg. Suppose we have two strings x, y

x = kitten, y = sitting . And we want to transform x into y.

We use edit operations:

1. Insertions
2. Deletions
3. Substitutions

First find the common letters between the two strings as follows:

k i t t e n

s i t t i n g

1st step : kitten → sitten (substitution)

2nd step : sitten → sittin (substitution)

3rd step : sittin → sitting (insertion)

Edit distance : It is the minimum number of edit operations required to transform x to y.

In this case $dist_{edit}$ is 3.

Although the above temporal and textual based relevance metrics which may work well in some cases, they cannot capture certain aspects of query similarity like for example if the queries do not appear close to one another temporally or the queries that are not textually similar but are related to one another. Therefore we need a relevance metrics that is robust enough to identify query groups beyond the approaches that simply rely on the textual content of queries or time interval between them.

Next we mention more robust approaches that mine search logs in order to find the relevance between queries in query groups more effectively.

V. QUERY GROUPING BASED ON CO-RETRIEVAL (CoR) PROPERTY

CoR: Co-Retrieval is based on the principal that a pair of queries are similar if they tend to retrieve similar sets of web pages on a search engine.

$Sim_{cor}(s_c,si)$ is the Jaccard coefficient of qc's set of retrieved pages retrieved(qc) and qi's set of retrieved pages retrieved (qi) and is defined as follows:

$$\text{sim}_{\text{cor}}(s_c, s_i) = \frac{|\text{retrieved}(q_c) \cap \text{retrieved}(q_i)|}{|\text{retrieved}(q_c) \cup \text{retrieved}(q_i))|} \_\_\_(3)$$

Here we compare two queries based on the overlap in web pages retrieved. We consider a web page to be retrieved by a search engine if it has not only been shown to some users, but has also clicked at least once in the past one year. This is the stronger baseline to define relevance measure.

© 2015, IJARCSSE All Rights Reserved

Eg. The queries "tata motor cars" and "tata cars" will retrieve more web pages that are similar whereas queries "tata power" and "tata cars" will retrieve less similar web pages. The queries "tata motor cars" and "tata cars" are similar as they will have higher sim_{cor} values as compared to queries "tata power" and "tata cars".

VI. QUERY GROUPING BASED ON ATSP (ASYMMETRIC TRAVELER SALESMAN PROBLEM) PROPERTY

$sim_{ATSP}(s_c, s_i)$ is defined as the number of times two queries, qc and qi , appear in succession in the search logs over the number of times qc appears.

$$sim_{ATSP}(s_c, s_i) = \frac{freq(q_c, q_i)}{freq(q_c)} \quad \text{---(4)}$$

This technique is based on the principal that two queries issued in succession in the search history logs are closely related if their similarity value sim_{ATSP} is greater than a threshold value T.

Eg. Two queries qc = banana and qi = apple , if these queries appear consecutively together in search logs. We remove infrequent query pairs by considering only those whose successive count is greater than 2. Thus if banana and apple appear more than twice consecutively in search log i.e count(qc, qi)>2 we calculate sim_{ATSP}.

VII. QUERY GROUPING BASED ON QUERY REFORMULATION AND QUERY CLICK INFORMATION

Here a different approach of query relevance is shown in order to construct query groups based on web search logs. The measure of relevance is based on two important properties of queries, i.e (1) queries that frequently appear together as reformulations of one another are considered as query reformulations thus are considered relevant to one another. (2) queries that have induced the users to click on similar sets of web pages are considered relevant to one another . We combine these two important properties and group queries based on their results. This approach is the most robust approach out of all to group relevant queries.

Here we introduce three such search behavior graphs that captures the aforementioned properties.

Query Reformulation graph :

Query reformulation graph represents the relationship between a pair of queries that are reformulations of one another. If two queries that are issued consecutively by the user occur frequently enough, they are likely to be reformulations of one another. In our approach we search for queries that appear next to each other in the entire query log. Thus using information from query logs we construct query reformulation graph QRG = (VQ,EQR) where VQ are the set of vertices which represents queries. Where EQR is the set of edges which is constructed as follows: for each query pair (qi,qj) where query qi is searched before query qj, we count the number of such occurrences in the query logs and denote its countr(qi, qj). We also remove out less frequent query pairs and include only the query pairs whose count is greater than a threshold value Tr. The edge weight of a directed edge for a query pair (qi,qj) with count greater than threshold is calculated as follows:

$$w_r(q_i, q_j) = \frac{count_r(q_i, q_j)}{\sum_{(q_i, q_k) \in EQR} count_r(q_i, q_k)} \quad \text{---(5)}$$

Query Click Graph:

Another way to capture relevant queries is to take into account queries that are likely to induce users to click frequently on same set of URLs for eg. queries like "Tata Motors" and "Nano" do not have any text in common neiher do they appear temporally close in users search history log, but they are relevant because they must have resulted in clicks about the Tata Motors. In order to capture this property of relevant queries we construct a query click graph QCG. In QCG (VQ,EQC) VQ is the set of vertices i.e the queries that induce users to click on similar set of URLs and EQC is set of edges where a directed edge from qi to qj exists if both qi and qj results in click on the same URL Uk. The edge weight is calculated as follows:

$$w_c(q_i, q_j) = \frac{\sum_{uk} \min(count_c(q_i, u_k), count_c(q_j, u_k))}{\sum_{uk} count_c(q_i, u_k)} \quad (6)$$

Where $count_c(q_i,u_k)$ is defined as the number of times query q_i is issued and url u_k is clicked

Query Fusion Graph:

In order to make more efficient use of the two properties captured by query reformulation graph QRG and query click graph QCG we combine query reformulation information and the query click information into a single graph query fusion graph QFG (VQ,EQF). Where EQF contains set of edges that is present in either EQR or EQC.

The weight of the edge (qi,qj) in QFG, wf(qi, qj) is calculated to be linear sum of the edge's weights wr(qi, qj) in EQR and wc(qi, qj)in EQC as follows:

$$w_f(q_i, q_j) = \alpha \times w_r(q_i, q_j) + (1 - \alpha) \times w_c(q_i, q_j) \quad \text{---(7)}$$

The relative contribution of two weights is controlled by α

After calculating wf for each queries in QRG and QCG we remove the infrequent pairs of (qi, qj) respectively. For removing the infrequent query pairs we consider only top 10 query pairs i.e query pairs with top 10 wf values. Thus group the top 10 values into a query group.

Example of Query Grouping based on Query Reformulation and Query Click Information is shown below:

Here we take α = 0.3 and count threshold Tr >= 1

© 2015, IJARCSSE All Rights Reserved

TABLE I USERS SEARCH HISTORY

TIME	QUERIES
10:51:48	Tata motor cars
10:52:24	Tata cars
10:59:28	Saturn dealers
11:12:04	Saturn vue
11:17:23	Jaguar animal
11:21:02	Jaguar wild animal
11:40:27	Tata motor cars
12:22:42	Tata nano
12:32:22	Saturn dealers
12:59:12	Saturn vue
13:03:34	Tata motor cars
14:00:00	Tata cars
16:34:09	Jaguar animal
17:52:49	Jaguar wild animal
19:22:13	Saturn dealers
19:25:49	Saturn vue
19:50:12	Saturn dealers
20:11:56	Hybrid Saturn vue
22:44:01	Tata motor cars
25:22:20	Tata nano
25:42:15	flipkart
26:40:15	snapdeal
26:42:12	Saturn dealers
28:14:12	Hybrid Saturn vue

TABLE III EDGE WEIGHT CALCULATION BY QFG

Query	**Queries from QRG and QCG**	
	Queries	**Edge weight**
Tata motor cars	Tata cars	0.6499
	Tata nano	0.6169
Saturn dealers	Saturn vue	0.6799
	Hybrid Saturn vue	0.5869
Jaguar animal	Jaguar wild animal	0.7999

TABLE IIIII QUERY GROUPS FORMED

Query Groups	**Queries of each query group**
	Group(1,2,….n)
Group 1: Tata motor cars	Tata cars
	Tata nano
Group 2: Saturn dealers	Saturn vue
	Hybrid Saturn vue
Group 3: Jaguar animal	Jaguar wild animal

VIII. CONCLUSIONS

In this paper, we consider the organization of user search histories into query groups by considering and studying various grouping techniques.

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, query alterations, search results re-ranking etc.

ACKNOWLEDGMENT

I would like to thank my family, friends and teachers who have been a source of encouragement and inspiration throughout the duration of the paper.

REFERENCES

[1] D. M. Siti Salwa Salleh1, Noor Aznimah Abdul Aziz1 and M Omar, "Combining mahalanobis and jaccard distance to overcome similarity measurement constriction on geometrical shapes," IJCSI International Journal of Computer Science Issues, vol. 9, 2012.

© 2015, IJARCSSE All Rights Reserved

[2] W. Barbakh and C. Fyfe, "Online clustering algorithms", International journal of Neural Systems, vol. 18, no. 03,pp. 185-194, 2008.
[3] R. Jones and K.L. Klinkner, "Beyond the session timeout: Automatic hierarchical segmentation of search topics in query logs," in CIKM,2008.
[4] P.Boldi, F.Bonchi, C.Castillo, D.Donato, A.Gionis, and S.Vigna, "The query-flow graph:Model and applications," in CIKM, 2008.
[5] D. He, A. Goker and D. J. Harper, "Combining evidence for automatic Web session identication," Information Processing and Management, vol.38,no. 5, pp. 727-742,2002.
[6] H.C.Ozmutlu and F.Cavdur,"Application of automatic topic identification on Excite Web search engine data logs," Information Processing and Management,vol.41,no. 5,pp. 1243-1262,2005.
[7] F.Radlinski and T.Joachims,"Query chains:Learning to rank from implicit feedback," in KDD,2005.
[8] R. Baeza-Yates, "Graphs from search engine queries," Theory and Practice of Computer Science (SOFSEM), vol. 4362, pp. 18, 2007.
[9] K. Collins-Thompson and J. Callan, "Query expansion using random walk models," in CIKM, 2005.

© 2015, IJARCSSE All Rights Reserved