Organizing User Search Histories

Pramod Shetty, Sayalee Rane, Mithilesh Tiwari, Prof. Kanchan Doke

Computer Engineering, Bharati Vidyapeeth College of Engineering
Room No-3, Omnath Aptt., Plot No – D-39, Sector – 20, Nerul(W), Navi Mumbai – 400 706, Maharashtra, India.

pramodshetty001@gmail.com
sayaleerane@gmail.com
mithi.tiwari@gmail.com
kanchankdoke@gmail.com

***Abstract*- Nowadays users have started making complex and task- oriented jobs on the web like making online transactions, booking travelling tickets etc. For the purpose of doing this, they use search engines and they normally break their work into some dependent tasks which contributes to the whole task and issue multiple queries around these steps repeatedly over a long period of time. To better support users in their information quests on the web, search engines keep track of their queries and clicks while searching online. In this paper, we study organizing a user's historical queries into groups in a dynamic and automated fashion. Dynamically identifying the relationship between queries and grouping them depending on how similar they are help the user in many tasks, such as query suggestions, result ranking, query alterations, sessionisation, and collaborative search. We will be providing two algorithms in this paper, one of which groups the queries inserted by the user and the other algorithm describes how to use this query group to provide more search result to the user.**

***Keywords* - User history, search history, queries clustering, query reformulation, task identification.**

I. INTRODUCTION

As the size and content of information on the web increases, the variety and the complexity of tasks that users try to accomplish online also increases. Users are no longer concerned with issuing simple navigational queries over the web. Various studies on query logs (e.g., Yahoo's and AltaVista's) reveal that only about 20 percent of queries are navigational. The rest are informational or transactional in nature. This is because users are pushing much broader informational and task oriented goals such as arranging future travel trips, managing their finances, or planning their purchase decisions. However, the primary means of accessing information online is still to search the keyword into a search engine. A complex task such as travel arrangement has to be broken down into a number of codependent steps over a period of time by the users. 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.

It is really important to manage queries entered by the users to make a more complex and reflexive system to provide better search results. It mostly happens that users try to search about a particular topic but do not remember the actual keyword to be searched for to get a more precise and reasonable results. Thus grouping queries together based on the similarities between them provides a better search technique to understand the user and his or her needs.

One important step toward enabling services and features that can help the users during their complex search and arrangements online is the capability to identify and group related queries together. Recently, some of the major search engines have introduced a new and interesting feature named as "Search History", which allows users to track and view their online searches by recording their queries and clicks. For example, Fig. 1 illustrates a portion of a user's search history as it is shown by the Google search engine.

This history includes a sequence of four queries displayed in reverse chronological order together with their corresponding clicks.

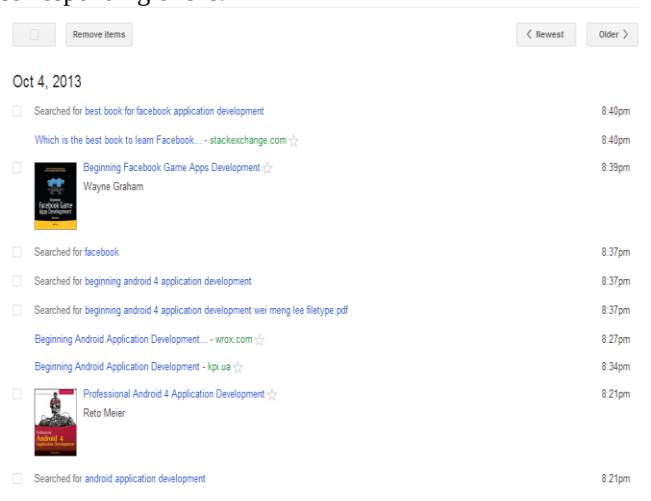

Fig 1. Portion of a user's Search History

This feature of 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. First of all query grouping allows the search engine to better understands the user's session and behavior.

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. This will help to improve the quality of key components of search engines such as query suggestions, result ranking, query alterations, sessionization, and collaborative search.

II. PRELIMINARIES

A. Goal

Here our objective is to group the queries entered by the user depending on the relation between this queries. This query grouping will be such that two queries which are related to each other and would probably give almost same results are kept together. For instance, let us consider 4 queries which are entered by user are "apple iPod", "apple ipad", "Microsoft windows" and "Bill Gates". Then the query "apple iPod" and "apple ipad" will be grouped together and "Microsoft windows" and "Bill Gates" will be in same group. These queries are related to each other by considering different factors related to when, how, and where they are fired. As a habitual nature, users mostly make related queries at a particular period depending on their need. They break down their task into many sub tasks such that all these tasks combine to fulfill the final objective of the user. Hence consideration of the time when the query is made and how many times they have been fired makes sense in this algorithm.

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. Each query group corresponds to an atomic information need that may require a small number of queries and clicks related to the same search goal. For example, in the case of navigational queries, a query group may involve as few as one query and one click (e.g., "cnn" and www.cnn.com). For broader informational queries, a query group may involve a few queries and clicks.

So in all the goals of presenting this paper is to provide a method that can be used to understand the user and his or her needs. This method can make out what actually the user want to search and thus depending on his query more related keywords are matched which are then combined together, fed to the system and final result depending on the all matched group queries are displayed. Main application where this method should be used is query suggestion, better result generation, Users activity monitoring etch. Query suggestion can help to provide more relevant query to to user. For example if the user has forgot a word or two of a new upcoming movie then this method can help him by providing the full movie name. Better search results can be implemented by adding all the results generated by the queries in the matched query group. In this way, results which would have been kept out will also be added to the list of final results.

The algorithm for query grouping is shown below where the queries will be grouped depending on the value of its relationship similarity between them. In this algorithm a similarity value is calculated between the queries entered by the user. Then these values are compared with threshold value which is predefined by the system. Whether to place these queries in the same query or not depends on the result of this comparison. The algorithm can be explained as follows –

A) A query group is an ordered list of queries, qi, together with the corresponding set of clicked URLs, clk_i of q_i. A query group is denoted as s = ($\{q_1, clk_1\} \ldots \{q_k, clk_k\}$).

B) Given: a set of existing query groups of a user, S = $\{s_1, s_2, \ldots s_n\}$, 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 $\{q_c, clk_c\}$.

A. SelectBestQueryGroup.

Input:

A. The current query group s_c containing the current query q_c and the set of clicks clks.
B. A set of existing query groups $s=\{s_1, s_2, \ldots\ldots, s_m\}$
C. A similarity threshold value T_{sim} , $0<T_{sim}<1$.

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

Steps

1. S=null(Ø)
2. $T_{max} = T_{sim}$
3. For i=1 to n
4. If $sim(s_c,s_i) > T_{max}$
5. $s=s_i$
6. $T_{max} = sim(s_c,s_i)$
7. If s=Ø
8. $S=S \cup s_c$
9. $s=s_c$
10. Return s;

Initially a new query group is formed and named s which is assumed to be null since it contains no query in it. This query group s will be the final group where the query along with other will be placed. And another query group s_c is also made which contains only the newly entered query by the user. Then the maximum threshold to be compared with is considered to be equal to the similarity threshold which is predefined by the system administrators. Let us assume that the system already contains some n number of groups each containing sets of queries within them. Here the query group s_c is compared with all the query groups that are already present in the system i.e. s_1, s_2,…. The comparison between the query groups gives a mathematical relation between them. If this similarity value is found to be more than the threshold value which is predefined then the two groups s_c and s_i are placed together. Else the s_c is made to be a new independent query group.

III. QUERY RELEVANCE

Now we develop the mechanism or the formula to find relation between the queries. This relationship or similarity is done on the basis of 4 major factors of a query-

A. Depending on the similarity in the keywords in the queries. i.e. two queries are assumed to be similar if they contain more keywords in common. Thus considering the keyword of the query to find similarity between them is a much better way.
B. Depending on the time in which the queries are made. Two related queries has a very high probability of being called within a particular interval of time.
C. Depending on the frequency in which the query are made together. Queries which are exactly related or can be considered as the sides of the same coin are mostly called together. So considering whether two queries have been called together most of the times whenever they are called makes good sense.
D. Depending on the results that are displayed by the queries. This is another important factor which relates two queries. Queries that have common result sets are actually two same queries with different wordings.

A. Jaccard

One may assume that s_c and s_i are somehow relevant if the queries appear similar if the query keywords are considered. More the number of similar words in the query, more the queries are related to each other.

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 (Jaccard similarity) or characters (Levenshtein similarity). We can thus define the following two text based relevance metrics that can be used in place of sim.

$Sim_{jaccard}(s_c, s_i)$ is defined as the fraction of common words between q_c and q_i as follows:

$$Sim_{jaccard}(s_c,s_i)= \frac{|words(q_c) \cap words(q_i)|}{|words(q_c) \cup words(q_i)|} \quad (1)$$

For example, if we consider two queries "American hustle" and "American idol", then with the above formula we have Similarity measurement value to be equal to 1/3 since 1 word is common in both the queries and the total no. of distinct words in the two queries are 3. If the threshold similarity we took is less than 0.33 then these two above queries will be placed in the same group.

B. Time

It is assumed that related queries are always fired within a fixed interval of time by the same user. This characteristic of user's psychology is used here to group the queries depending on the time the queries have been fired.

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. In this case, we define the following time-based relevance metric sim_{time} that can be used in place of sim.

$Sim_{time}(s_c, s_i)$ is defined as the inverse of the time interval (e.g., in seconds) between the times that q_c and q_i are issued, as follows:

$$Sim_{time}(s_c,s_i) = \frac{1}{|time(q_c) - time(q_i)|} \quad (2)$$

For example, if two queries named "Oppa Ganganam style" and "Psy songs" are fired between a fixed interval of time which is the similarity threshold then this two queries will be placed together. The possibility that two related queries will be always placed together is more since there is a higher probablity of their reference within an interval of time.

C. Coefficient of Retrieved Pages (CoR)

The results a query generate makes species the scope of that query. If another query generates almost the same set of results as that of the previous query, then we can say that the two queries are similar and ought to be in the same group.

CoR is based on the principle that pair of queries is similar if they tend to retrieve similar pages on a search engine. This approach is similar to the ones discussed above.

$Sim_{cor}(s_c, s_i)$ is the Jaccard coefficient of q_c's set of retrieved pages retrieved(q_c) and q_i's set of retrieved pages retrieved(q_i) and is defined as:

$$Sim_{cor}(s_c,s_i) = \frac{|retrieved(q_c) \cap retrieved(q_i)|}{|retrieved(q_c) \cup retrieved(q_i)|} \quad (3)$$

For example, consider two queries "Google nexus" and "Google phones". Since there is large possibility of these two queries to retrieve similar results, the two queries will be placed in the same group. As in the above example, If the common results found by the two queries will be 15 and union of the results by the two queries are 30 then the above similarity value will be 0.5.

D. ATSP

Two queries can be assumed to be lying in the same group if both of these are frequently called after each other. This phenomenon makes it sure that both these queries have some relationship.

This technique is based on the principle that two queries issued in succession in the search logs are closely related. The authors present a solution that first reorders a sequence of user queries to group similar queries together by solving an instance of the ATSP. Once the queries are reordered, query groups are generated by determining "cut points" in the chain of queries, i.e., two successive queries whose similarity is less than a threshold. Note that ATSP needs to operate on the whole set of queries that we are interested in grouping as it involves an initial reordering step.

$Sim_{ATSP}(s_c, s_i)$ is defined as the number of times two queries, q_c and q_i, appear in succession in the search logs over the number of times q_c appears :

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

For example, two queries are frequently fired back to back in a search engine then the two of them will be grouped together.

IV. RESULT GENERATION USING GROUPS

Result displaying algorithm based on the query groups

Steps:

1. Entered Query (q_c) is mapped using SelectBestQueryGroup algorithm.
2. Mapped query groups (q_c) contain query groups (q_1, q_2, q_3…., q_m).
3. ResultSet=null.
4. For i = 1 to m
   If (q_c (sim (qi)))
   ResultSet+=Result (q_i);
5. ResultSet+=Result(q_c)
6. Display ResultSet.

This algorithm is used for the purpose of presenting the result of the entered query by the user using the query groups in which the entered query lies.

Suppose the entered query is q_c and the matched query group is s_c containing query sets q_1,q_2,q_3, … q_m. The ResultSet in the algorithm is the final result to be displayed by the system. Initially, the system assumes the ResultSet to be null.

Then the system searches for each query in the query group to contain similar keyword as q_c. If such query is found in the query group, then the result of that matched query is added to the final ResultSet.

Finally when this process is completed, the query q_c is searched directly in the database to find any exempted result. All this results are then finally displayed to the user in which more indexing priority is given to the ResultSet obtained by the query group.

V. CONCLUSION

The query grouping depending on the relation between them contains useful information on user behavior and motive when making a search online. Here, in this paper, we show how information like dependency between queries can be used effectively for organizing user search histories into query groups. 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. Thus grouping queries together based on the similarities between them provides a better search technique to understand the user and his or her needs.

ACKNOWLEDGMENT

Our most sincere appreciation are to all the people who have helped and inspired us throughout the working of this project. Firstly we are thankful to our principle Dr. M. Z. Shaikh for his help. We are extremely grateful for his friendly support and professionalism. We express our heartfelt gratitude to our Head of Department Prof. D. R. Ingle and our project coordinator Prof. B. W. Balkhande for their help and support. This task would not have been possible without the help and guidance of our project guide Prof. Kanchan Doke. We are also convening special thanks to all staff of Computer Engineering Department for their support and help. Last but not least, we are very much thankful to our friends who directly or indirectly helped us in completion of the project report.

REFERENCES

[1]. J. Teevan, E. Adar, R. Jones, and M.A.S. Potts, "Information Re-Retrieval: Repeat Queries in Yahoo's Logs," *Proc. 30th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '07)*, pp. 151-158, 2007.

[2]. A. Broder, "A Taxonomy of Web Search," *SIGIR Forum*, vol. 36, no. 2, pp. 3-10, 2002.

[3]. W. Barbakh and C. Fyfe, "Online Clustering Algorithms," *Int'l J. Neural Systems*, vol. 18, no. 3, pp. 185-194, 2008.

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