Dynamic Grouping of User Search Histories and Re-Ranking User Search Results

[1]Ambreen Shaikh, [2]Seema Kolkur
[1]Post graduate student, [2]Associate Professor,
[1]Department of Computer Engineering,
[1]Shree L.R. Tiwari College of Engineering, Thane, India

***Abstract* – Due to the ever increasing size of information on the web, the various complex tasks and information search that users perform is also increasing day by day. Users depends on the Web for various activities like for seeking knowledge, solving problems and performing task oriented searches. To help users in their long term information search on the Web, search engines keep track of their search by recording their queries and clicks while searching. We study how the users search history can be used to improve search results quality for users with similar or same interest. One way to achieve better search results is by grouping similar queries that involves same or similar search interests into query groups. In this paper we study how to automatically organize a users search history into query groups, each containing one or more related queries and their corresponding clicks and also we propose an approach for re-ranking of search results using these query groups. Query grouping help us understand users search session better and thus make the users search experience according to her needs. By grouping search queries we can provide better search results to the users in future based on their past search interests. Here in this paper we study how query groups are created in dynamic and automated fashion. Each query group contains related queries and their corresponding clicks and proposing an approach to re-rank search results using the created query groups.**

***Index Terms* –Query reformulation, Query Click , Search History, re-ranking.**

I. INTRODUCTION

Grouping or clustering is a useful technique for the discovery of data distribution and patterns in the underlying data. Grouping or clustering is a process of organizing data into meaningful useful groups, and these groups are called cluster. Query grouping allows the search engine to better understand a users session and potentially tailor the users search experience according to their needs. Query grouping is a useful technique for enhancing the users internet searches. In this paper we study the issue of organizing users search history into query groups in an automatic and dynamic fashion. Each query group is a collection of queries that are relevant to each other and their corresponding clicks that revolves around a common information need. These query groups are dynamically updated as user issues new queries, and new query groups may be created over time. So in this paper we study the organization of users search history into a set of query groups. Our approach is to generate these query groups automatically and dynamically and also each group is organized according to user preferences. It is challenging to organize the 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. Therefore, it is not sufficient to rely solely on string similarity.

Nowadays the search engines keep track of users searches by organizing the users searches in chronological order. However the chronological order is not very useful to end users. As the users want to view the related queries together so that they can reuse the queries or their corresponding clicks.

The vast amount of web pages and the fast growth of web suggests that users are becoming more and more dependent on the search engines ranking ways to discover information relevant to their requirements. Typically, users expect to find the information they need in the top-ranked results, and more often they look at the document snippets in the first few pages.To improve search result we apply re-ranking algorithm on search result. First we fetch the top N results returned by search engines such as Google for user queries, find the most relevant group for the query and use similarities between the candidate and the query group candidate to re-rank the results. We first convert the ranking position to an importance score for each Google candidate. Then the similarity score is combined with this initial importance score and finally we get the new ranks.

II. LITERATURE REVIEW

In [2] is explained compution of similarity relevance between two different strings. The drawbacks that occur in using string similarity functions is that it requires more time and the problem of ambiguity arises. As users carry out various complex task oriented operations over the net[1]. Each task is again further divided into set of subtasks. In [3][4] the author studies the search task identification problem. In [3] the authors consider a search session, a search session consist of number of goals and each goal further consists of sub goals. In [4] the authors implement the construction of query flow graph, in query flow graph two queries are linked by an edge then such queries that are linked by edge are part of the same search mission. In [5][6] authors study the overlap of terms of queries to detect changes in the topic of searches. In [7] the authors study and implement query sequences, i.e called chains by using a classifiers that combines two features like time threshold with textual similarity features of the queries, and the results returned by those queries. Query graphs based on query and click logs [8] have also been used for different applications like query expansion[9], query suggestions[4], ranking[10].

III. PROPOSED SOLUTION

Mission:

Our mission is to automatically and dynamically organize a users search history into query groups, also to find relevant query group out of exixting query groups for users current query and applying re-ranking algorithm to re-rank search results of current user query based on the relevant query group and if there is no relevant existing query group for current query then create new query group. Each query groups contain one or more related queries and their corresponding clicks related to same search goal.

Definition of Query Group:

A query group consists of list of queries, qi, together with the corresponding set of clicked URL's, clk_i of q_i. A query group is denoted as S=({q_1,clk_1},…….., {q_k,clk_k}).

The formulation of the problem is as follows:

Given: A set of already existing query groups of a user, S={s1,s2,s3,s4,…….,sn} and a current query and click , {q_c,clk_c}.

Find: Finding query group for current query and its corresponding click {q_c,clk_c}, which can be either one of the already existing query groups in S that 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 related enough to {q_c,clk_c}. Then apply re-ranking algorithm to re-rank search results of current query.

We now develop a system to define query relevance in order to contruct query groups based on web serach logs. Our measure of relevance is based on two important properties of queries, i.e (1) queries that frequently appear together as reformulations of one another. (2) queries that have induced the users to click on similar sets of webpages .
We now study how to capture the above mentioned properties by implementing three search behavior patterns that capture those properties. Following that we learn how we can use these formats to compute query relevance and how we can incorporate the click following a users query in order to enhance our similarity score.

Search Behavior patterns

There are three types of search behavior patterns that we derive from search logs of a search engine.

Query Reformulation

Query reformulation 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 find query reformulation pattern, 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 it countr(qi,qj). Remove out less frequent query pairs and include only the query pairs whose count is greater than a threshold value Tr. The weight of a query pair (qi,qj) with count greater than threshold is calculated as follows:

$$\text{Eq. 1} \qquad w_r\left(q_i, q_j\right) = \frac{\text{count}_r\left(q_i\,, q_j\right)}{\sum_{(q_i\,, q_j) \in EQR} \text{count}_r(q_i, q_j)}$$

Where EQR denotes the various qi and qj values belonging to same search goal and countr(qi,qj) is the number of times qi is issued before qj.

Query Click

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 neither do they appear temporally close in users search history log, they are relevant because they must have resulted in clicks about the Tata Motors. In order to capture this property of relevant queries we search if there exists any URL's (uk) that both qi and qj click then count (countc) the common clicks by both qi and qj. The weight of query pair (qi,qj) is calculated as follows:

$$\text{Eq. 2} \qquad w_c(q_i, q_j) = \frac{\sum_{uk} \min(count_c(q_i, u_k), count_c(q_j, u_k))}{\sum_{uk} \; count_c(q_i, u_k)}$$

Where countc(qi,uk) is the number of times qi was issued and url uk was clicked.

Query Fusion

In order to make more efficient use of the two properties captured by query reformulation and query click we combine query reformulation information and the query click information into query fusion.

The weight of the edge (qi,qj) in QFG, wf(qi, qj) is calculated to be linear sum of the weights wr(qi, qj) in query reformulation and wc(qi, qj) in query click as follows:

$$\text{Eq. 3} \qquad w_f(q_i, q_j) = \alpha \times w_r(q_i, q_j) + (1 - \alpha) \times w_c(q_i, q_j)$$

The relative contribution of two weights is controlled by α .
In system take α = 0.3.

IV. USING QFG CALCULATING QUERY RELEVANCE

Calculating query relevance:

After construction of Query Fusion we now find the relevant queries that are highly related to the query group. The values of Query Reformulation and Query Click represents pairs of relevant queries extracted from query logs. Calculation of query relevance is done by analyzing values of Query Fusion. We consider and take only some highly relevant wf values i.e top wf values whose sum is less than certain threshold. Query Image is the set of queries that are highly relevant to the query groups. Thus Query Image holds only those highly relevant queries and its wf values. Next step is to calculate cxt (context vector). Context vector is obtained by adding all wf entries present in query image of each different query groups.

$$\text{Eq. 4} \quad cxt = \sum w_f(q_i, q_j)$$

V. CREATING QUERY GROUP USING QFG

The related queries are found from query reformulation and query click values. Thus these queries which are similar are grouped together in different query groups according to the search goal they aim. Next step is to find similarity between users current singleton query and already existing query groups. Thus after finding similarity, the latest singleton query will be attached to the query group that has highest similarity value with the current query. If there does not exist any query group that is similar to current query then new query group will be created for the current query.

Algorithm For Selecting Best Query Group that is similar to the given current query and its clicked URL :

Input to Algorithm: A current query q_c and its corresponding first click clk_c .

Output of Algorithm: Linking current query to already existing query group that is similar to it or creating a new query group if no similar query group found.

```
1: $s = \emptyset$
2: $Tmax = sim(s_c, s_i)$
3: for i= 1 to m
4:    if $sim(s_c , s_i) >= Tmax$
5:         $s = s_i$
6:         $Tmax = sim(s_c , s_i)$
```

```
7: if s = $\emptyset$
8:     $S = S \cup s_c$
9:     $s = s_c$
10: return s
```

Where s_c is current singleton query group containing the current query q_c and its first clicked URL clk_c.
S is a set of existing query groups $S = \{s_1,\ldots,s_i\}$ that are similar to s_c. This can be found by matching the current query keyword q_c and clicked URL clk_c to the existing query groups queries and its corresponding clicks. If match is found that means that query group is similar to s_c.
Where $sim(s_c,s_i)$ is a similarity threshold that is calculated as (summation of w_f) * cxt of queries of query group that is found related to the current query or its click. $Sim(s_c,s_i)$ is calculated as follows:

$$\text{Eq. 5} \qquad sim(sc\,, si) \;=\; \sum w_f\left(q_i, q_j\right) \;*\; cxt$$

Then depending upon which $sim(s_c,s_i)$ of which group is higher, the current query is attached to that query group.
Lastly q_c and clk_c is attached to a query group S that best matches s_c, or a new query group s_c is created if no related query group found in S.

ALGORITHM FOR RANKING

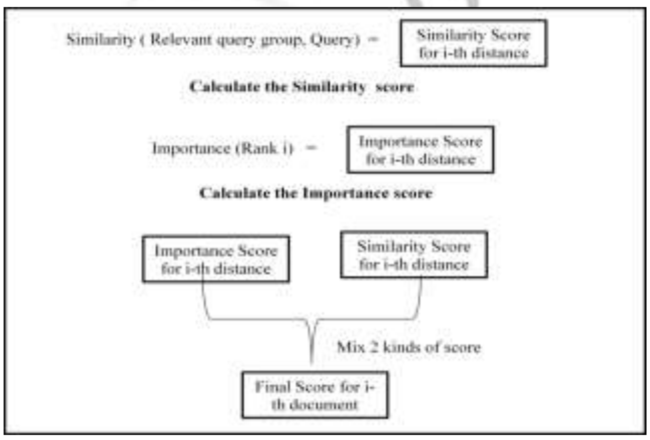
Here we explain re-ranking algorithm. For each query of user we take top N results returned by search engines. Then we use the following formula to calculate each web search results importance score.

$$\text{Eq. 6} \qquad importance(i) \;=\; \frac{1-\dfrac{i-1}{tot}}{\log_2(i+1)}$$

Where *i* is the original Page Rank serial number i.e original page rank position and tot is the number of results web pages fetched for the query. The formula shows that the top results that are fetched have significant importance to the search keywords and thereby are much usefull for web users. Then using query and single click, we can find the most relevant query group for the users current query. Now the following similarity score and re-ranking algorithm is applied to re-rank search results.

Calculation of similarity score

1) If the title of of search result matches with query group, if it matches
   Score 1 = Original rank *i*+ 1;
2) If content matches then
   Score 2 = Original rank *i* + 5;
3) If URL matches then
   Score 3 = Original rank *i* + 10;
4) Sim-score = (score1+score2+score3)/16;



Similarity (Relevant query group, Query) =
Similarity Score for i-th distance
Calculate the Similarity score
Importance (Rank i) =
Importance Score for i-th distance
Calculate the Importance score
Importance Score for i-th distance
Similarity Score for i-th distance
Mix 2 kinds of score
Final Score for i-th document

Fig. 1 Re-ranking method

Re-ranking Algorithm

1) When a user submits a query to search engine, it gets the top N results returned by search engine for that query.
2) Then it finds the most relevant query group for that current query and gets all the URL's from that group that are clicked most of the time. In this way we would know the users preferences.
3) Then convert the ranking position to an importance score for each serach engine candidate using importance formula.
4) Then calculate similarity score for each search engine candidate .
5) Then combine similarity score with initial impotance score and finally this combined score makes the new ranks.
6) Then display the search results in descending order of new rank.

VI. CONCLUSION

Query reformulation and query click represents useful information about user behavior when searching online. In this way explore how such information can be used efficiently for organizing user search histories into query groups. More precisely, we propose combining the two patterns i.e query reformulation and query click into a query fusion pattern. Then we propose a method to use these query groups for re-ranking search results. As future work, we intend to use the knowledge gained from these query groups in other applications like providing query suggestions.

VII. ACKNOWLEDGEMENT

I would like to thank the publishers and researchers for helping me by making their resources available for learning. I would like to thank all our internal guides.

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.

[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 identification, "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. 1–8, 2007.

[9] K. Collins-Thompson and J. Callan, "Query expansion using random walk models," in CIKM, 2005.

[10] N. Craswell and M. Szummer, "Random walks on the click graph," in SIGIR, 2007.