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

[1]Shaikh Ambreen Mohd Ibrahim, [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 increasing size of information on the web, the various complex tasks and information search that users perform is also increasing. Users depends on the Web for various activities like for seeking knowledge, solving problems and performing tasks. 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, Click graph, Search History, ranking.**

I. INTRODUCTION

Search engines are facing problems and struggle to provide relevant queries for current search query. As the web is growing user interaction on the web has also increased and users carry out many complex task related operations over the web. The various complex task related goals can be for eg planning a holiday tour, making purchase online and managing finances etc. Query grouping is used to provide related queries to users for their current query search. The problem that come across while creating query groups is, if the same query represents different information (eg Blackberry query related to 'Blackberry phone manufacturer' or 'Blackberry fruit'). Also, if different queries represent same search result (eg. Bank of India and financial statement). Thus Query grouping will help in solving this problem, as Query groups collects queries that are syntactically as well as semantically related. Earlier to create query groups, string similarity functions were used. Text, Time, Jaccard, Levenshtein, CoRetrieval and Asymmetric Traveler Sale Problem (ATSP) are string similarity functions[1]. K-Means, Modified K-Means and Online clustering algorithms are also used to group similar queries. Bi-Partile graph construction and Monte Carlo Tree Search methods are recent techniques useful in similarity calculation. In addition to this to improve the search result we apply re-ranking algorithm on search result. First we get the top N results returned by search engine for a user query, then find the most relevant query group for that query and use similarites between the candidate and query group candidate to re-rank the results. First we convert the ranking position of each search engine candidate to an importance score. Then we combine the similarity score with the initial importance score and thus we get the new ranks of the search results.

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

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 clicks , $\{q_c,clk_c\}$.

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 exixt a query group in s that is not 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 graphs that capture those properties. Following that we learn how we can use these graphs to compute query relevance and how we can incorporate the clicks following a users query in order to enhance our relevance score.

Search Graphs

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

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 q_i is searched before query qj, we count the number of such occurrences in the query logs and denote it $count_r(q_i, q_j)$. 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 (q_i,q_j) with count greater than threshold is calculated as follows:

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

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, 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 q_i to q_j exists if both q_i and q_j results in click on the URL U_k. The edge weight is calculated as follows:

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

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 (q_i,q_j) in QFG, $w_f(qi, qj)$ is calculated to be linear sum of the edge's weights $w_r(qi, qj)$in EQR and $w_c(qi, qj)$in EQC 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 .

IV. USING QFG CALCULATING QUERY RELEVANCE

The following algorithm can be used for calculating the query relevance by simulating random walks on query fusion graph

Algorithm for calculating query relevance

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, $\mathbf{rel}_{\mathbf{q}}^{\mathbf{F}}$

(0)Initialize $\mathbf{rel}_{\mathbf{q}}^{\mathbf{F}} = 0$
(1)numWalks = 0; numVisits = 0
(2)**while** numWalks<numRWs
(3) numHops = 0; v = q
(4) **while** v ≠ NULL ^ numHops < maxHops
(5) numHops++
(6) $\mathbf{rel}_{\mathbf{q}}^{\mathbf{F}}(v)$++; numVisits++
(7) v = SelectNextNodeToVisit(v)
(8) numWalks++
(9) For each v, normalize $\mathbf{rel}_{\mathbf{q}}^{\mathbf{F}}(v) = \mathbf{rel}_{\mathbf{q}}^{\mathbf{F}}(v)$/numVisits

This algorithm gives the fusion relevance vector of a given query q, rel_q^F. The jump vector "g" is used to pick the random walk starting point. The random walk rather continues by starting at node v, for a damping factor d by following one of the outgoing edges of v with probability of d, or stops and then restarts with probability (1-d) at one of the starting points in g. Each outgoing edge (v,q_i) is selected with probability $w_f(v,q_i)$, and the random walk restarts if no outgoing edges of v exist. Selection of next node to visit of current node v of QFG and damping factor d is performed by step 7 of algorithm. In addition to this user activities also take into account the clicks on URLs by query after the query submission in the search engine.

Algorithm for selecting the next node to visit

SelectNextNodeToVisit(v)

Input:

(1) The query fusion graph, QFG
(2)The jump vector, g
(3)The damping factor, d
(4)The current node, v

Output: the next node to visit, q_i
(1)**if** random()<d
(2) V={q_i | $(v,q_i) \in$ EQF}
(3) pick a node $q_i \in$ V with probability $w_f(v,q_i)$
(4)**else**
(5) v={q_i | $g(q_i) > 0$}
(6) pick a node $q_i \in$ V with probability $g(q_i)$
(7)**return** q_i

V. CREATING QUERY GROUP USING QFG

Here we explain the proposed similarity function sim_{rel} which can be used in the dynamic online query grouping process. For each query there is query image. Query image of a query is denoted I(q), here q is the set of queries in VQ that are highly relevant to q. We generate I(q) by taking every query q' whose relevance value to q, $\mathrm{rel}_q(q')$, is within top percentage. Query image contain all the queries related to the query and for each query group, we maintain a 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 similarity between the users singleton query and the already existing query group is calculated by using context vector.

The relevance between the users latest singleton query group $s_c = (q_c, clk_c)$ where q_c is the current query and clk_c is its corresponding URL clicks and an existing query group $s_i \in S$ will be calculated as follows:

$$\text{Eq. 4} \qquad \mathrm{Sim}_{\mathrm{rel}}(s_c, s_i) = \sum_{q \in I(sc) \cap I(si)} \mathrm{rel}_{(qc,clkc)}(q) * \sum_{q \in I(sc) \cap I(si)} \mathrm{cxt}_{si}(q)$$

Where,

s_c = singleton group
s_i = existing group
I = Image of query group
q = query
Cxt_{si} = Context vector of query group s_i
$\mathrm{rel}(q_s, clk_s)$ = relevance between query q and corresponding url clk

In this way the latest current query will be attached to the query group that has highest similarity $sim_{rel.}$

VI. 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.

$$\text{Eq. 5} \qquad \text{importance(i)} = \frac{1 - \frac{i-1}{tot}}{\log 2(i+1)}$$

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 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;

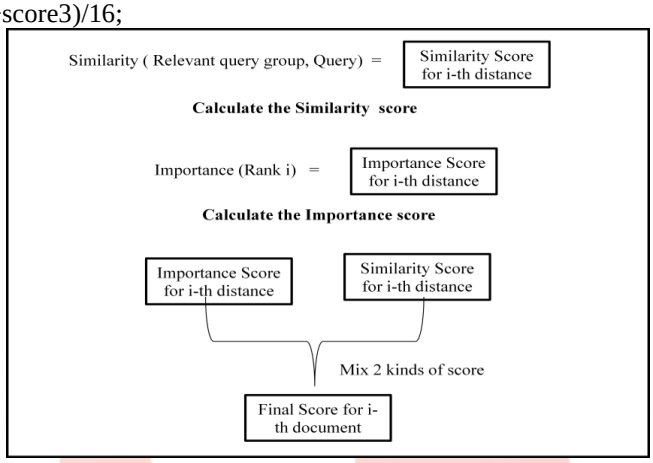

Fig. 1 Re-ranking method

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.

VII. CONCLUSION

Query reformulation and query click graphs 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 graphs i.e query reformulation graph and query click graph into a query fusion graph. 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 and a factor that considers users changing interests to re-rank search results.

VIII. 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. Hea, 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.