Volume 4, Issue 7, July 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

A Novel Approach for Re-ranking Search Results Using User Search History

Rupali D. Navagire *
Department of Computer Science,SKNCOEP,
Pune University
India

Prof. T. H. Gurav
Department of Computer Science,SKNCOEP,
Pune University
India

Abstract— As the size of information on the Web growing, the variety and the complexity of tasks that users try to accomplish online is also growing. But Search Engines often return a large volume of pages in response to user queries, while the user always wants to get the best in a short span of time. Many Search engines keep track of their clicks and queries to help users in their information search .We are interested in how this user's search history can be used to improve the search results quality for users having same interest. However, the search histories are not organized into related groups. In this paper, our goal is to automatically and dynamically organize user's search history into query groups containing related queries and propose an approach for collaborative re-ranking of the search results using these query groups.

Keywords— *Click graph, queries reformulation, clustering, search engine, search history, ranking.*

I. INTRODUCTION

Users are performing various tasks-oriented goals on the web such as making travel arrangements, managing finances, or planning purchases. To complete such tasks user has to issue multiple queries. To help users in their search some search engines allow users to track their online searches by recording their queries and clicks but it is manual process and is a difficult job to maintain such a large history in an organized way.

In fact, identifying related query groups has applications beyond helping the users to make sense and keep track of queries and clicks in their search history .Query grouping can help to improve some of the features and services provided by search engine such as query suggestions, result ranking, query alterations and collaborative search. So in this paper we study the organization of user's 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.

In this work 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.

The rest of the paper is organized as follows. In Section 2, we state problem statement of our paper. In Section 3, we review the related work. In Section 4, we explain how to construct behaviour graphs, how to use them to determine relevance between queries or query groups within a user's history and an algorithm to perform query grouping using query fusion graph logs .In Section 5 we explain Re-ranking algorithm. Experimental results are discussed in section 6 and we conclude with a discussion on future research directions in Section 7.

II. PROBLEM STATEMENT

Our goal is to automatically and dynamically organize a user's search history into query groups and use these groups efficiently to increase the quality of search results returned by search engines. So we have to solve two problems to develop re-ranking system.

1. Grouping of relevant queries and
2. Re-ranking search results using relevant 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}, {q_k, clk_k}).

The specific formulation of our first problem is as follows:

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

The core of the solution is a measure of relevance between two queries (or query groups) that not only rely on time or text but also we propose a relevance measure based on signals from search logs. For second problem, considering Google ranks importance for query and similarity between Google results and relevant query group is a big problem.

III. LITERATURE SURVEY

At the In recent work, Jones and Klinkner [2] worked on the search-task identification problem. He constructed a query flow graph to solve the problem. Our work differs from these prior works as we consider query pairs having common clicked URLs and we exploit both co-occurrence and click information through a combined query fusion graph.
Some prior work [5] and [6] proposed segmentation of a user's query streams into "sessions" based on a "time-out threshold". As discussed in section I, time is not a good basis for identifying query groups because related queries may not appear close to one another. Radlinski and Joachims[7] identified query sequences by employing a classifier. They used both timeout threshold and textual similarity features of the queries to train the classifier. While text similarity may work in some cases, it may fail to capture cases where there is "semantic" similarity between queries.
Query clustering [4] and [14] problem is also related to query grouping problem. The authors in [4] found query clusters by relying on both text and click features. In Beeferman and Berger [15] and Baeza-Yates and Tiberi [4], commonly clicked URLs on query-click bipartite graph are used to cluster queries. While these prior work make use of click graphs, our approach is much better in that we use the click graph in combination with the reformulation graph.
Wen et al. [14] proposed a clustering algorithm that makes use of both query contents and URL clicks. They suggested that two queries should be in the same cluster, if they contain the same or similar terms, and lead to the selection of the same documents. However, since Web search queries are usually short and common clicks on documents are rare (see discussion below), Wen et al.'s method may not be effective for disambiguating Web queries.

IV. QUERY RELEVANCE COMPUTATION AND QUERY GROUP CREATION

We assume that queries that frequently appear together are relevant. Also queries that have induced the users to click on similar sets of pages are relevant. So we are considering both these important properties of relevant queries to measure query relevance. We first construct behaviour graphs that capture the above mentioned properties of relevant queries and then we show how we can use these graphs to compute query relevance.

A. Constructing QFG

The query reformulation graph, QRG, captures the first important property of related queries. We construct the query reformulation graph, QRG= (V_Q, E_{QR}) as follows: For each query pair (q_i, q_j), we count the number of such occurrences in the query logs, denoted $count_r(q_i, q_j)$. We removed all the query pairs whose count is less than threshold value. For each (q_i, q_j) with $count_r$ greater than threshold value, we add a directed edge from q_i to q_j to EQR.

We construct a query click graph, QCG=(V_Q, E_{QC}) by constructing CG =(VQ U VU; EC), used by Fuxman et al.[6] and then we derive our query click graph, QCG = (V_Q, E_{QC}). In QCG ,the vertices are the queries. If there exists at least one URL, that both qi and qj link to in CG, we draw a directed edge from qi to qj in QCG.We construct QFG =(V_Q, E_{QF}), by combining QRG and QCG into a single graph, that we refer to as the query fusion graph.

B. Calculating Query Relevance Using QFG

After constructed QFG, we now compute the relevance between two queries. The edges in QFG correspond to pairs of relevant queries extracted from the query logs and the click logs. The following Algorithm is used for calculating the query relevance by simulating random walks over the query fusion graph

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, rel^F_q

```
1) Initialize $rel^F_q = 0$
2) numWalks = 0; numVisits = 0
3) while numWalks < numRWs
4) numHops = 0; v = q
5) while v = NULL ^ numHops < maxHops
6) numHops++
7) $rel^F_q(v)$++; numVisits++
8) v = SelectNextNodeToVisit (v)
9) numWalks++
10) For each v, normalize $rel^F_q(v) = rel^F_q(v)/numVisits$
```

This algorithm computes the fusion relevance vector of a given query q, rel^F_q. A jump vector of q, g_q, specifies the probability that a query is selected as a starting point of a random walk. The algorithm works as follows: jump vector g_q is used to pick up the starting point for the random walk. At each node v, the random walk either continues by following one of the outgoing edges of v or stops or restarts at one of the starting points in gq. The selection of the next node to visit is based on the outgoing edges of the current node v in QFG.

C. Creating Query Group Using QFG

In this section, we explain our proposed similarity function sim_{rel} to be used in the online query grouping process. For each query, we maintain a query image. Query image contain all the queries related to the query and for each query group, we maintain a context vector. The similarity between the query group and the user's latest singleton query group is computed by using 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 relevance between the user's latest singleton query group $s_c = (q_c, clk_c)$ and an existing query group s_i Є S will be calculated as follow.

$Sim_{rel}(s_c,s_i) = \sum_{q \epsilon I(sc) \cap I(si)} rel_{(qc,clkc)}(q) \ * \ \sum_{q \epsilon I(sc) \cap I(si)} ctx_{si}(q)$ ----------------------(1)

Where,

S_c = singleton group

S_i = existing group

I = Image of query group

Q = current query

Cxt_{si} = Context vector of query group s_i.

$rel(q_s, clk_s)$ = relevance between query q and corresponding url clk

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

V. RE-RANKING ALGORITHM

In this section we explain re-ranking algorithm. For each user query we fetch the top N results returned by search engines such as Google. Then we propose the following formula to calculate each web's importance score.

$$importance(i) = \frac{1-\frac{i-1}{tot}}{\log 2(i+1)} \quad \text{---------------} \quad (2)$$

Where *i* is the original Page Rank serial number (i.e., the original ranking position) and *tot* is the number of the fetched web pages for a query. The formula indicates that the top results have significant importance to the search keywords and there-by are much valuable for web users. Then using query and a single click, we can find the most relevant query group. Now following Re-ranking algorithm is applied to re-rank search results.

Similarity score calculation

1) Now matched the title with group query, if matched then
   Score1 =Original rank *i* + 1;
2) If contain matched then
   Score2= Original rank *i* + 5;
3) If url matched then
   Score3= Original rank *i* + 10;
4) Sim-score=(score1+score2+score3)/16;

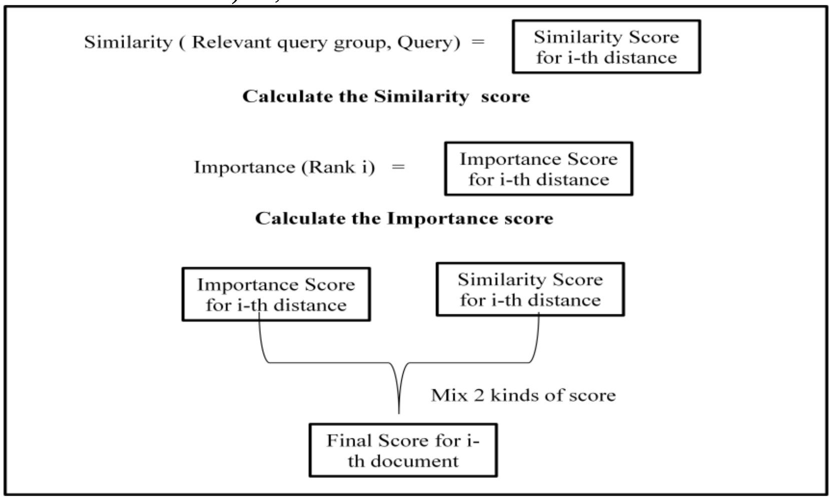
Similarity (Relevant query group, Query) =
Similarity Score for i-th distance
Calculate the Similarity score
Importance (Rank i) =
Importance Score for i-th distance
Calculate the Importance score
Importance Score for i-th distance
Similarity Score for i-th distance
Mix 2 kinds of score
Final Score for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

Fig.1. Re-ranking method

VI. A PROPOSED SYSTEM

A proposed System has following major steps:

1. Creating groups of relevant queries
2. Appling re-ranking algorithm

Re-ranking Algorithm:

1. When a user submits a query, it fetches the top N results returned by search engines such as Google for that queries
2. Then it find the best relevant group for the query and gets all the URL's from that group that are clicked most. This will help us to find users preferences.
3. Convert ranking position to an importance score for each Google candidate using formula (2).
4. Similarity scores are calculated for each Google candidate.
5. Then the similarity score is combined with this initial importance score and finally we get the new ranks.
6. Display the results in descending order of new rank.

VII. EXPERIMENTAL SETUP AND EVALUATION OF RE-RANKING ALGORITHM

In this section, we talk about data set for the experiment, performance metrics used to measure the quality of proposed system and evaluation results.

Data. We used the search activity of 30 users for the implementation. Table 1 shows Search Engine Query Log Sample .In our database; we have over 900 queries entered by different users. To reduce the effect of noise and outliers, we considered query pairs that appeared at least two times and query click edges that had at least 3 clicks. Thus produced query and click graphs are smaller compared to their original respective graphs. More than 23 groups of relevant queries are created. These groups are used for re-ranking search results. Table 1 shows sample of user search history log.

Table 1. User search history Log Sample

ID	User	Query	Clicked URL	Timestamp
q1	u1	Flash player	Get.adobe.com/.../...	2014-07-08 15:12:41
q2	u1	flash		2014-07-08 11:13:44
q3	u1	Photoshop	www.brothersoft.com/.../...	2014-07-08 11:14:21
q4	u1	Flash		2014-07-07 07:13:01
q5	u1	Blade series	En.wikipedia.org/.../...	2014-07-07 19:13:01
q6	u2	Superhero	www.superherodb.com/	2014-07-08 09:44:26
q7	u2	Flash	www.imdb.com/.../...	2014-07-08 14:35:14
q8	u2	adobe	www.adobe.com/.../...	2014-07-07 14:36:26

Performance Metric. We are using precision measure to evaluate the performance of our re-ranking method. Precision is the fraction of the documents retrieved that are relevant to the user's information need. The formula for precision is given below.

$$precision = \frac{|\{\text{Relevant Urls}\} \cap \{\text{Retrieved Urls}\}|}{|\{\text{Retrieved Urls}\}|}$$

Evaluation Results.
To evaluate results, we have considered six queries. These queries are Q1: Apple, Q2: big data, Q3: cricket, Q4: Data mining, Q5: java download and Q6: Samsung Mobile .For each query, precision values of Google searches and re-ranking search is calculated. These values are plotted in the graph. From above graph it is clear that Re-ranking algorithm gives more relevant data based on User search history.

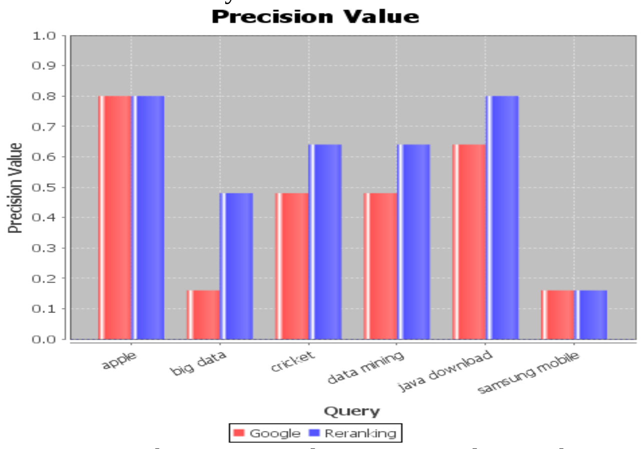

Fig 2. Comparison between Google precision and Re-rank precision

VIII. CONCLUSIONS

In this paper, we propose a method to improve the search quality by re-ranking result based on user search history. As we know that user search history contains useful information on user search behaviour. In this paper, we organized queries in user search history into groups of related queries by considering query reformulation and query click information then we proposed a method to uses these group for re-ranking search results. For that we combined importance score of each Google candidate and similarities between these candidates and the most relevant query group candidate to re-rank the results. As future work, we intend to a factor that considers user's changing interests to re-rank search results.

ACKNOWLEDGMENT

Sincerely thank the all anonymous researchers for providing us such helpful opinion, findings and recommendations. I would like to acknowledge and thank my guide, Prof. Trupti Gurav, Professor of Computer Engineering at SKNCOE, Pune for her valuable guidance, support and motivation.

REFERENCES

[1] Heasoo Hwang, Hady W. Lauw, Lise Getoor, and Alexandros Ntoulas" Organizing User Search Histories", IEEE Transactions On Knowledge And Data Engineering, vol. 24, no. 5, May 2012

[2] R. Jones and K.L. Klinkner, "Beyond the Session Timeout: Automatic Hierarchical Segmentation of Search Topics in Query Logs," Proc. 17th ACM Conf. Information and Knowledge Management (CIKM), 2008.

[3] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna, "The Query-Flow Graph: Model and Applications," Proc. 17th ACM Conf. Information and Knowledge Management (CIKM), 2008.

[4] R. Baeza-Yates and A. Tiberi, "Extracting Semantic Relations from Query Logs," Proc. 13th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD), 2007.

[5] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz, "Analysis of a Very Large Web Search Engine Query Log," SIGIR Forum, vol. 33, no. 1, pp. 6-12, 1999.

[6] H.C. Ozmutlu and F. C¸ avdur, "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," Proc. ACM Conf. Knowledge Discovery and Data Mining (KDD), 2005.

[8] Lin LI, Zhenglu YANG, and Masaru KITSUREGAWA" Rank Optimization of Personalized Search" DEIM Forum 2009 A9-5

[9] Neelam Duhan and A. K. Sharma "A Novel Approach for Organizing Web Search Results using Ranking and Clustering" *International Journal of Computer Applications (0975 – 8887)Volume 5– No.10, August 2010.*

[10] Ziming Zhuang and Silviu Cucerzan "Re-Ranking Search Results Using Query Logs"*ACM CIKM '06*, November 6–11, 2006, Arlington, VA, USA.

[11] Ruofan Wang, Shan Jiang and Yan Zhang "Re-Ranking Search Results Using Query Logs".

[12] Zheng Lu, Hongyuan Zha, Xiaokang Yang, Weiyao Lin, ZhaohuiZheng, "A New Algorithm for Inferring User Search Goals withFeedback Sessions", IEEE Transactions on Knowledge and Data Engineering.

[13] Xinye Li "An improved method in clustering Web retrieval result based on relevance feedback", Computer Science and Service System (CSSS), IEEE International Conference ,pp. 3000 - 3003,2011

[14] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang, "Query Clustering Using User Logs," ACM Trans. in Information Systems, vol. 20, no. 1, pp. 59-81, 2002.

[15] D. Beeferman and A. Berger, "Agglomerative Clustering of a Search Engine Query Log," Proc. Sixth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD), 2000.