IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 12, Issue 6 (Jul. - Aug. 2013), PP 49-53
www.iosrjournals.org

User Priority Based Search on Organizing User Search Histories with Security

Gogula. Vijay Kumar[1], A. Krishna Chaitanya[2]
[1] *M.Tech (CSE), Vardhaman College of Engineering, Hyd, A.P, INDIA*
[2] *Associate Professor in IT Dept, Vardhaman College of Engineering, Hyd, A.P, INDIA*

Abstract: *Presently, users are facing many complicated and complex task-oriented goals on the search engine. Those are managing finances, making travel arrangements or any other planning and purchases. To reduce this problem, usually break down the tasks into a few codependent steps and issuing multiple queries, and which store repeatedly over a long period of time, whatever the user search in the search engine, that information search engines keep track of their queries and clicks while search in the search engine or online. In this paper we become skilled at the complexity of organizing user's historical queries into in an active and expected manner. Automatic identifying query groups are compassionate for the number of different search engines, deals with applications. Those are result status, query suggestions, query alterations. In this we are proposing security for the related query groups. When we work in the single or any organization, security will provides the security for the user's data or information in the search engine or any data base.*

Keywords *- User history, search history, query clustering, query reformulation, click graph, task recognition, security.*

I. Introduction

Now a days we have large information or data in the web, but we have some complex problems while searching in the online for the required data.whn the user enter a query in the search engine , then the user may or may not get the required information within the short time and minimum clicks. Various studies on query logs like, Yahoo's and AltaVista's has reveal that only 20-30 percent of queries are navigational[1]. That is since user's now pursuing broader information and task leaning goals, such as arranging for outlook tour, organization their assets, scheduling and their obtain decisions. The searching will start by entering a keyword in the search engine. A complicated task such as travel scheduling has to be broken down into a number of codependent steps over a period of time. User may primary search on probable destinations, time lines, proceedings, etc., and then the user may search for most proper planning for rental cars, air tickets, temporary housing and meals etc. Each step requires one or more queries, and each query results in one or more clicks on related pages.

To decrease the trouble on the user's while searching in the search engine. Some of major search engines introduced a new "Search History" feature. Which helps to the users during their complex search quests online is the capability to identify group related queries together, which allows the user's to track their online searches by recording their queries and clicks. For example, user interested to search some queries in the online, that queries will be stored in the search history, which helps to the users to make sense and keep track of queries and clicks in the search history[2]. This query grouping allows the users to better understanding which the user searched before. For example, if the user searched for a query that already in the history log, then that will be displayed in the related query group, by which the user can select required query from the query group. The query group consists of correlated queries, and it will show only significant query as an alternative of Wikipedia.

In this paper we study the problem of organizing a user's search history into a set of query groups in an automated and dynamic fashion. Every query group is a group of queries by the similar user that are related to each other. And the user searched for new query groups may be created over time. In particular, we develop an online query grouping method over the query fusion graph that combines a probabilistic query reformulation graph[3], which captures the relationship between queries frequently issued together by the user's, and query click graph, which captures the relationship between queries and clicks on similar URLs.

Time	Query
01:05:33	Crick info
01:06:21	Hotel
01:10:42	Bangalore
01:12:44	Ipl
01:22:55	Taj banjara hotel
01:45:12	Eenadu

01:56:33	Sitara hotel
02:02:33	Live score
02:05:22	Vaartha
02:06:34	Cricket
02:12:22	Madhura hotel
02:35:34	Hyderabad
03:05:22	Chennai
03:12:23	sakshi
04:34:44	Mumbai

Fig (a): Users search history

Group 1	Group 2	Group 3
Hotels	Cric info	Bangalore
Taj banjara	Ipl	Chennai
Sitara hotel	Cricket	Hyderabad
Madhura hotel	Live score	Mumbai

Fig (b): Search history of a user's one day together with query groups

II. Preliminary Results

2.1 Objectives

Our major objective is organizing a user's search history into correlated query groups automatically, which consists of one or more related queries and their correlated clicks.

Select Best Query Group

Input:

1) The current singleton query group s_c containing query q_c and set of clicks clk_s
2) A set of existing query groups S={s1……,sm}
3) A similarity threshold T_{sim}, 0<_T_{sim}<_1

Output:

The query group s that best matches s_c, or a new one if necessary

(0) S= ϕ
(1) $T_{max}=T_{sim}$
(2) For i=1 to m
(3) If $sim(s_c,s_i)>t_{max}$
(4) $S=s_i$
(5) $T_{max}= sim(s_c,s_i)$
(6) If s= ϕ
(7) S=s s_c
(8) $S=s_c$
(9) Return s

Fig: Algorithm for query group

2.2 Dynamic Query Grouping

In which, identification of user's search history into a related query group and then merge these query groups in an iterative fashion, there will not be any undesirable effect of changing a user's existing query group. So that, which involves high-computational cost for every new query, so it is not possible to create single group to every new query entered by the user. When the user enter a query in the search engine, we first place the current query and clicks into a singleton query group $S_c= \{q_c,clk_c\}$, and then we compare with existing query group s_i within a user's history[4]. If there is an existing query group sufficiently relevant to S_c. If so, we merge with the S_c with the query group S having highest similarity T_{max} above or equal to the threshold T_{sim}, otherwise we keep S_c as a new singleton query group and insert it into S.

III. Query Relevance

If the user entered an existing query in the search engine, that is the query relevant and appear close to each other in time in the user's history, in that situation we define time-based relevance metric as follows,

3.1 Query Import Ants

(3.1.1) Time

$$Sim_{time}(S_c,S_i) = \frac{1}{|time(q_c) - time(q_i)|}$$

Where, $sim_{time}(S_c,S_i)$ is defined as the inverse of the time interval between the times

(3.1.2) Jacquards

If the query group textually similar then we use jaccard similarity, as follows,

$$Sim_{jaccard}(S_c,S_i) = \frac{|words(q_c) \cap words(q_i)|}{|words(q_c) \cup words(q_i)|}$$

$Sim_{jaccard}(S_c,S_i)$ is defined as the fraction of common words between q_c and q_i

Above two time-based and text-based relevance metrics may work well in some cases, but they cannot capture query similarity. If the user is multitasking means, more than one tab opens in his browser. Then we can follow the co-retrieval method as follows,

(3.1.3) Co-Retrieval

$$Sim_{cor}(S_c,S_i) = \frac{|retrieved(q_c) \cap retrieved(q_i)|}{|retrieved(q_c) \cup retrieved(q_i)|}$$

$Sim_{cor}(S_c,S_i)$ is defined set of retrieved pages retrieved (q_c) and (q_i)

3.2 Query Relevance Using Search Logs

Let us know about, how to define the query relevance based on web search logs, means that relevance is aimed at capturing two important properties of relevant queries. Let us know how we can use these graphs to calculate query weight and how we can integrate the click.

Search behavior graphs:

Three types of graphs from the search logs of a commercial search engine. They are

(3.2.1) Query reformulation graph (QRG)

This represents, the correlation between a pair of queries that possible reformulation of each other.

Query reformulation graph, is one way to identify relevant queries that are typically found within the query logs of a search engine. If two queries that are issued repeatedly by the several users happen repeatedly, then they are likely to reformulations of each other[5]. Sim_{time} is a time based metric between two queries and which makes use of the distance between the time stamps of the queries within the users search history. Based on the query logs, we create the query reformulation graph, QRG = $(v_{Q,} \sum_{QR})$ and set of edges $\sum_{QR.}$ Every pair (q_i,q_j), where q_i is issued before q_j and we count number of such occurrences across all users daily activities, denotes $count_r$ (q_i,q_j). If the two pairs are not relevant[6], then we filter out infrequent pairs and include only the query pairs whose counts exceed a threshold value, Tr. For each (q_i,q_j) with $count_r$ (q_i,q_j) >_ Tr, we add a direct edge from q_i to q_j to $\sum_{QR,}$ the edge weight w_r (q_i,q_j),

$$w_r(q_i,q_j) := \frac{count_r(q_i,q_j)}{\sum_{q_i,q_k \in \sum QR} count_r(q_i,q_j)}$$

(3.2.2) Query click graph (QCG)

This represents the correlation between two queries repeatedly lead to clicks on related URLs.

One way to capture relevant queries from the search logs are to be consider queries that are likely to induce users to click frequently on the same set of URLs. First start by considering a bipartite click through graph, CG = (V_q V_u , $\sum_c$) used by fuxman. CG has two different set of nodes related to queries, V_q, and URLs, $V_{u,}$ extracted from the click logs[7]. The edge $(q_i.u_k) \in \sum_{c,}$ if query q_i was issued and URL u_k was clicked by the users. We weight each edge $(q_{i,} u_k)$ by number of times q_i was issued and u_k was clicked, $count_c$ $(q_{i,} u_k)$. the weight edge (q_i,q_j) in QCG, w_c (q_i,q_j) is defined as follows,

$$W_c(q_i,q_j) := \frac{\sum_{u_k} \min(count_c(q_i,u_k), count_c(q_c,u_k))}{\sum_{u_k} count_c(q_i,u_k)}$$

(3.2.3) Query fusion graph (QFG)

This merges the queries in the earlier two graphs. These three graphs have the same set of vertices V_Q, but those edges are defined differently.

We combine both the query reformulation graph and query click graph within QRG and query click information $\sum_{QF}$ the weight of edge(q_i,q_j) in

within QCG into a single graph QFG = (v_q ,) that we refer as the fusion graph.
QRG, $w_f(q_i,q_j)$ and taken to be a linear sum of the edge weights, $w_r(q_i,q_j)$ in E_{QR} and $w_c(q_i,q_j)$ in E_{QC}, as follows:

$$w_f(q_i,q_j) = \alpha * w_r(q_i,q_j) + (1-\alpha) * w_c(q_i,q_j)$$

The relative contribution of the two weights is controlled by[7] α , and we denote a query fusion graph constructed with a particular value of α as QFG (α).

Algorithm for calculate the query relevance
Input:

(1) Query fision graph , QFG
(2) The jump vector, g
(3) The damping factor, d
(4) Total number of random walks, numRWs
(5) The size of neighbourhood, maxHops
(6) The given query, q

Output:

(0) Initialize rel $_q^F$ =0
(1) Numwalks=0; numvisit=0
(2) While numwalks<numRWs
(3) numHops=0;v=q
(4) while v ≠ NULL ∩ numHops < maxHops
(5) numHops ++
(6) rel $_q^F$ (v)++; numvisits++
(7) v=selectnext node to visit(v)
(8) numwalks++
(9) for each v, normalize rel $_q^F$ (v)= rel $_q^F$ (v)/numvisits

SELECT NEXT NODE TO VISIT:
Algorithm for selecting next node to visit
Input:

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

Output:

(0) if random()<d
(1) v={q_i|(v,q_i) ∈ $\sum_{QF}$}
(2) pick a node q_i∈v with probability w_f (u,q_i)
(3) else
(4) v={q_i|g(q_i)>0}
(5) pick a node q_i ∈ v with probability g(q_i)
(6) return q_i

IV. Conclusion

In this paper, we show how such queries are stored in the search history while searching in the online. And we used reformulation graph and click graphs, which contains useful data on use behavior when searching online. We have seen how the historical queries into groups in a active and automatic manner. Frequently identifying query groups are compassionate to the number of users. In addition, we are enhancing security; provide the security to the query group, in an organization security is more important to secure the data or information. The security plays a key role for providing security to the user's data. With no security, data will be losses by the unofficial users. Security, it will provide the data, if we were authenticated otherwise it will not provide anything.

References

[1] Heasoo Hwang, Hady W. Lauw, Lise Getoor, and Alexandros Ntoulas "*Organizing User Search Histories*" ", Knowledge ang Data Engineering, IEEE Transactions on volume:24 ,Issue:5 , 2012.
[2] 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
[3] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang, "*Query Clustering Using User Logs*," ACM Trans. in Information Systems,
[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] J. Han and M. Kamber," *Data Mining: Concepts and Techniques*". Morgan Kaufmann, 2000.
[6] W. Barbakh and C. Fyfe, "*Online Clustering Algorithms*," Int'l J. Neural Systems, vol. 18, no. 3, pp. 185-194, 2008.
[7] Data Mining Concepts M. Berry, and M. Browne, eds. World Scientific Publishing Company, 2006.