Maintaining User Search Accounts using Page Rank Algorithm

Prabhakar Reddy Gangireddy
M.Tech CSE Dept
Institute of Aeronautical Engineering
Hyderabad- 500043, *Andhra Pradesh, India*

Dr.N. Chandra Sekhar Reddy
Professor, CSE Dept.,
Institute of Aeronautical Engineering,
Hyderabad -500043, Andhra Pradesh, India

G. Praveen Babu
Associate Professor
Dept., of Computer Science & Engineering,
School of Information Technology,
JNT University Hyderabad, *Andhra Pradesh, India.*

Er. Sai Prasad Kashi
M.Tech CSE Dept
Institute of Aeronautical Engineering
Hyderabad- 500043, *Andhra Pradesh, India*

Abstract - Users are increasingly pursuing complex task-oriented goals on the Web, such as making travel arrangements, managing finances or planning purchases. Many approaches to creating user profiles collect user information through proxy servers (to capture browsing histories). User profiles, and interests, can be used by search engines to provide personalized search results. Both these techniques require participation of the user to install the proxy server. We extend previous work by proposing a number of techniques for filtering previously viewed content that greatly improve the user model used for personalization. Automatically identifying query groups is helpful for a number of different search engine components and applications, such as query suggestions, result ranking, query alterations, sessionization, and collaborative search. In this approach, we go beyond approaches that rely on textual similarity or time thresholds, and we propose a more robust approach that leverages search query logs. We experimentally study the performance of different techniques, and showcase their potential, especially when combined together. To better support users in their long-term information quests on the Web, search engines keep track of their queries and clicks while searching online. In this paper, we study the problem of organizing a user's historical queries into groups in a dynamic and automated fashion.

Keywords — Furl Toolbar, Page Rank, Surf Saver Toolbar, Search Query Logs, Search Results, User Profiles.

I. INTRODUCTION

User profiles were created by classifying the collected information (queries or snippets) into concepts in a reference concept hierarchy. These profiles were then used to re-rank the search results and the rank-order of the user-examined results before and after re-ranking were compared. Our study found that user profiles based on queries were as effective as those based on scraps. We also found that our personalized re-ranking resulted in a 38% improvement in the rank order of the user-selected results. Automatically identifying query groups is helpful for a number of different search engine components and applications, such as query suggestions, result ranking, query alterations, sessionization, and collaborative search. In this approach, we go beyond approaches that rely on textual similarity or time thresholds, and we propose a more robust approach that leverages search query logs. We

experimentally study the performance of different techniques, and showcase their potential, especially when combined together.

II. LITERATURE SURVEY

2.1 Motivation-

Companies that provide marketing data report that search engines are used more and more as referrals to websites, rather than direct navigation via hyperlinks. As search engines perform a larger role in commercial applications, the desire to increase their effectiveness grows. However, search engines order their results based on the small amount of information available in the user's queries and by web site popularity, rather than individual user interests. Thus, all users see the same results for the same query, even if they have wildly different interests and backgrounds. To address this issue, interest in personalized search had grown in the last several years, and user profile construction is an important component of any personalization system. Explicit customization has been widely used to personalize the look and content of many web sites1but we concentrate on personalized search approaches that focus on implicitly building and exploiting user profiles. Another issue facing search engines is that natural language queries are inherently ambiguous.

Our goal is to show that user profiles can be implicitly created out of short phrases such as queries and snippets collected by the search engine itself. We demonstrate that profiles created from this information can be used to identify, and promote, relevant results for individual users.

2.2 Current Technologies and Problems-

In general, personalization can be applied to search in two different ways:
1. By providing tools that help users organizing their own past searches, preferences, and visited URLs;
2. By creating and maintaining sets of user's interests, stored in profiles that can be used by retrieval process of a search engine to provide better results.
The first approach is applied by many new toolbars and browser add-ons. The Seruku Toolbar and the SurfSaver are examples of tools that try to help users to organize their search histories in a repository of URLs and web pages visited. Furl is another personalization tool that stores web pages including topics which users are interested in, however it was developed as a server-side technology. In recent times, search engines have been improved with personalization features. One among them is goggle, A9 launched by amazon.com, where users are identified through a login + cookie technology. All queries submitted can be viewed, organized and reused in future searches. Submitted queries are also used to do a full text search on the books available at amazon.com to locate and suggest the best books related to the query topic. Ujiko.com is also a new interesting search engine that identifies users through cookies and has an appealing interface that allows users to give explicit judgments about specific results to store submitted queries to organize browsed results to be helped in “refining” their searches augmenting queries with special terms suggested. All these systems have interesting features that can guide users to find better information but they represent the user with overall profile rather than trying to identify simple specific topics of interest. Our study focuses on personalization in search based on implicit feedback. Many implicit feedback systems capture browsing histories through proxy servers or desktop activities through the installation of bots on a personal computer. These technologies require the technology rather than a desktop toolbar.

In this study, we explore the use of a less-invasive means of assembling user information for personalized search. In particular, we build user profiles based on activity at the search site itself and study the use of these profiles to provide adapted search results. Desktop bots can capture all activity whereas proxy servers can capture all Web activity. In contrast, cookies can only capture the activity at one specific site, the one that issues the cookie. Our goal is to show that user profiles can be implicitly created out of the limited amount of information available to the search engine itself; the queries submitted and snippets of user-selected results. We demonstrate that profiles created from this information can be used to identify, and promote, relevant results for individual users.

III. EXISTING SYSTEM

However, this is impractical in our scenario for two reasons:-

First, it may have the undesirable effect of changing a user's existing query groups, potentially undoing the user's own manual efforts in organizing her history.

Second, it involves a high computational cost, since we would have to repeat a large number of query group similarity computations for every new query.

Disadvantages:

We motivate and propose a method to perform query grouping in a dynamic fashion. Our goal is to ensure good performance while avoiding disruption of existing user-defined query groups.

IV. PROPOSED SYSTEM

We investigate how signals from search logs such as query reformulations and clicks can be used together to determine the relevance among query groups. We study two potential ways of using clicks in order to enhance this process by fusing the query reformulation graph and the query click graph into a single graph that we refer to as the query fusion graph, and by expanding the query set when computing relevance to also include other queries with similar clicked URLs. We show through comprehensive experimental evaluation the effectiveness and the robustness of our proposed search log-based method, especially when combined with approaches using other signals such as text similarity.

Advantages:

1. Our focus on improvising effectiveness in obtaining query relevance.

2. Relevance Measure

3. Online query grouping process

4. Similarity function

System Architecture-

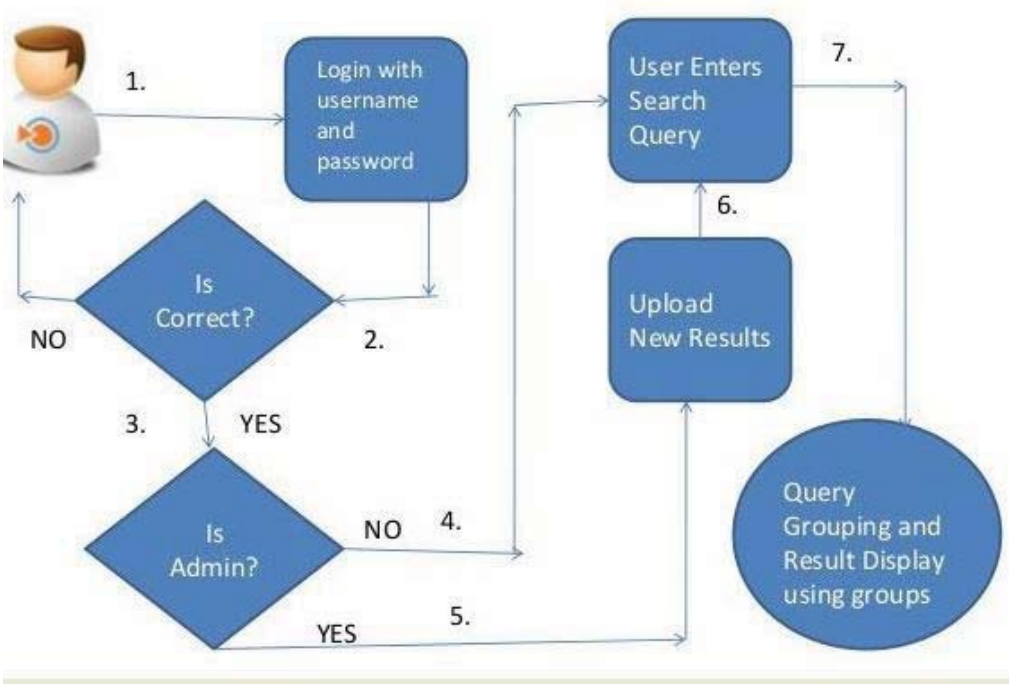


Figure 1. Architecture for Query grouping

From Fig: 1 we can understand that once a user login and enters search queries and admin can upload any results, process query grouping and we can display results using groups.,

V. MODULE DESCRIPTION

5.1 Query Group-

We need a relevance measure that is robust enough to identify similar query groups beyond the approaches that simply rely on the textual content of queries or time interval between them. Our approach makes use of search logs in order to determine the relevance between query groups more effectively. In fact, the search history of a large number of users contains signals regarding query relevance, such as which queries tend to be issued closely together (query reformulations), and which queries tend to lead to clicks on similar URLs (query clicks). Such signals are user-generated and are likely to be more robust, especially when considered at scale. We suggest measuring the relevance between query groups by exploiting the query logs and the click logs simultaneously.

Time. simtime(s_c, s_i) is defined as the inverse of the time interval (e.g. in seconds) between the times that qc and qi are issued, as follows:

$$\text{simtime}(s_c, s_i) = \frac{1}{|time(q_c) - time(q_i)|}$$

Select Best Query Group

Algorithm for selecting the query group that is the most similar to the given query and clicked URLs.

Input:

1) The current singleton query group sc containing the current query qc and set of clicks clk_c

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 sc, or a new one if necessary

(0) s = ø;

(1) *T*max = *T* sim

(2) **For** i = 1 **to** m

(3) **If** sim (sc, si) > *T*max

(4) s = si

(5) *T*max = sim (sc, si)

(6) **if** s s = ø ;

(7) S = S ∪ sc

(8) s = sc

(9) **Return** s

The queries qc and qi are the most recent queries in sc and si respectively. Higher sim time values imply that the queries are temporally closer.

5.2 Search History-

We study the problem of organizing a user's search history into a set of query groups in an automated and dynamic fashion. Each query group is a collection of queries by the same user that are relevant to each other around a common informational need. These query groups are dynamically updated as the user issues new queries, and new query groups may be created over time.

5.3 Query Relevance and Search logs-

We now develop the machinery to define the query relevance based on Web search logs. Our measure of relevance is aimed at capturing two important properties of relevant queries, namely: (1) queries that frequently appear together as reformulations and (2) queries that have induced the users to click on similar sets of pages. We start our discussion by introducing three search behavior graphs that capture the aforementioned properties. Following that, we show how we can use these graphs to compute query relevance and how we can incorporate the clicks following a user's query in order to enhance our relevance metric.

5.4 Dynamic Query Grouping-

One approach to the identification of query groups is to first treat every query in a user's history as a singleton query group, and then merge these singleton query groups in an iterative fashion (in a k-means or agglomerative way). However, this is impractical in our scenario for two reasons. First existing query groups, potentially doing the user's own manual efforts in organizing her history. Second, it involves a high computational cost, since we would have to repeat a large number of query group similarity computations for every new query.

VI. IMPLEMENTATION

6.1 Algorithm Used: Page Rank Algorithms-

Page Rank is a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page. Page Rank can be calculated for collections of documents of any size. It is assumed in several research papers that the distribution is evenly divided among all documents in the collection at the beginning of the computational process. The Page Rank computations require several passes, called "iterations", through the collection to adjust approximate Page Rank values to more closely reflect the theoretical true value.

Page Rank can be calculated for collections of documents of any size. It is assumed in several research papers that the distribution is evenly divided among all documents in the collection at the beginning of the computational process. The Page Rank computations require several passes, called "iterations", through the collection to adjust approximate Page Rank values to more closely reflect the theoretical true value.

According to web in short Page Rank is a "vote", by all the other pages on the Web, about how important a page is. A link to a page counts as a vote of support. If there's no link there's no support (but it's only an abstention from voting rather than a vote against the page). Quoting from the original Google paper, Page Rank is defined like this:

We assume page A has pages T1…Tn which point to it (i.e., are citations). The parameter d is a damping factor which can be set between 0 and 1. We usually set d to 0.85. There are more details about d in the next section. Also C(A) is defined as the number of links going out of page A. The Page Rank of a page A is given as follows:

PR(A) = (1-d) + d (PR(T1)/C(T1) + … + PR(Tn)/C(Tn))

Note that the Page Ranks form a probability distribution over web pages, so the sum of all web pages\' PageRanks will be one.

Page Rank or PR(A) can be calculated using a simple iterative algorithm, and corresponds to the principal eigenvector of the normalized link matrix of the web.

PR(Tn) – Each page has a notion of its own self-importance. That's "PR(T1)" for the first page in the web all the way up to "PR(Tn)" for the last page

C(Tn) – Each page spreads its vote out evenly amongst all of its outgoing links. The count, or number, of outgoing links for page 1 is "C(T1)", "C(Tn)" for page n, and so on for all pages.

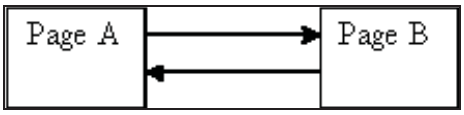
PR(Tn)/C(Tn) – so if our page (page A) has a back link from page "n" the share of the vote page A will get is "PR(Tn)/C(Tn)"

6.2 Calculation of Page Rank-

This is where it gets tricky. The PR of each page depends on the PR of the pages pointing to it. But we won't know what PR those pages have until the pages pointing to them have their PR calculated and so on… And when you consider that page links can form circles it seems impossible to do this calculation! But actually it's not that bad. Remember this bit of the Google paper:

Page rank or PR(A) can be calculated using a simple iterative algorithm, and corresponds to the principal eigenvector of the normalized link matrix of the web.

What that means to us is that we can just go ahead and calculate a page's PR without knowing the final value of the PR of the other pages. That seems strange but, basically, each time we run the calculation we're getting a closer estimate of the final value. So all we need to do is remember the each value we calculate and repeat the calculations lots of times until the numbers stop changing much. Let’s take the simplest example network: two pages, each pointing to the other:



Each page has one outgoing link (the outgoing count is 1, i.e. C(A) = 1 and C(B) = 1).

6.3 Personalization Strategies-

In this section, we describe our approach. The first step consists of constructing a user profile, that is then used in a Second phase to re-rank search results.

User profile Generation: A user is represented by a list of terms and weights associated with those terms, a list of visited URLs and the number of visits to each, and a list of past search queries and pages clicked for these search queries. This profile is generated as shown in Figure 1. First, a user's browsing history is collected and stored as (URL, HTML content) pairs. Next, this browsing history is processed into six different summaries consisting of term lists. Finally, the term weights are generated using three different weighting algorithms. We now describe each of these steps in detail.

Data Capture: To obtain user browsing histories, a Firefox add-on called AlterEgo was developed. To respect the users' privacy as Much as possible, a random unique identifier is generated at installation time. This identifier is used for all data exchange between the add-on and the server recording the data.

Term List Filtering: To reduce the number of noisy terms in our user representation, we also tried filtering terms by removing infrequent words or words not in Word Net. However, neither of these where found to be beneficial. Therefore we do not discuss term list filtering further.

General Search history personalization: By personalizing your results based on your search history, we hope to deliver you the most useful, relevant content for your search. Search history personalization is just one of the ways that we show you more personalized search results.

Search results from your friends and connections: When you search on Google, you can see search results from the public web along with pages, photos, and Google+ posts from your friends. For example, if you're planning a trip to Italy, and you search for Google, you may see photos or articles from your friends about Florence in your search results. These results make it easy to explore their recommendations and strike up a conversation about which sights to see.

The below Figure 2. shows us the results obtained based on Dates V/S links to URL's.

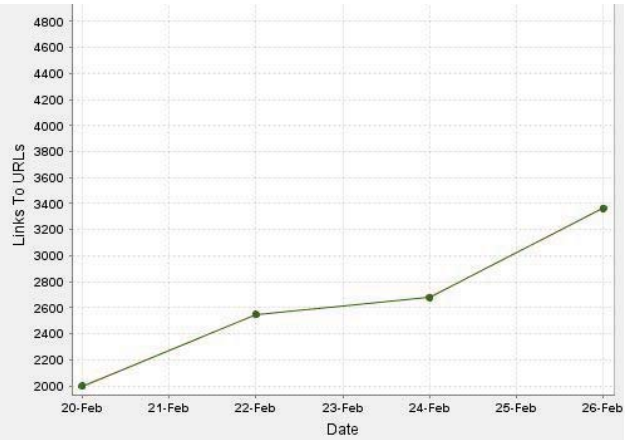


Figure 2. Results(Dates V/S links to URL's)

Signed-in personalization: When you're signed in, Google personalizes your search experience based on your Google Web History. If you don't want to see results personalized based on your Google Web History while you're signed in, you can turn off Google Web History and remove it from your Google Account. You can also view and remove individual items that you don't want from your Google Web History.

You can also turn off personal results to prevent personalization based on your Google Web History. Turning off personal results will also disable several other personalization features, such as the ability to search across content shared by your friends and connections. Personal results are currently not available in all languages.

Signed-out personalization: When you're not signed in, Google personalizes your search experience based on past search information linked to your browser, using a cookie. Because many people might search from a single computer, the browser cookie may be associated with more than one person's search activity. For this reason, we don't provide a method for viewing this signed-out search activity. If you don't want to see results personalized

based on your search history while you are signed out, you can turn off search history personalization Here's an illustration of the information we use in each case:

	Signed-in search history personalization	Signed-out search history personalization
Where the data we use to customize is stored	In Google Web History, linked to your Google Account	On Google's servers, linked to an anonymous browser cookie
Which searches are used to customize	Only signed-in search activity, and only if you have Google Web History enabled	Only signed-out search activity
How to turn off	Turn off search history Personalization ("Signed in searches" section)	Turn off search history Personalization ("Signed out searches" section)

VII. CONCLUSION

Personalization technology has matured in the last few years to the degree that large-scale personalized search systems can now be deployed. Various algorithms and techniques have been developed and tested in mostly specialized and restricted domains, some of which have been reviewed here. Relevance feedback, a technique dating to the early days of IR, is being incorporated into result ranking and is being enhanced and adapted to take advantage of the unique features of web search. Moreover, newer systems are beginning to combine several different techniques to improve the user's overall experience of search personalization. The challenge facing any personalization system is that to succeed commercially on a large scale the loyalty and trust of users must be won. We see three main components that must be addressed in overcoming this challenge. Firstly, the system's behavior should be predictable and its workings transparent. Lack of predictability is an issue for any adaptive system where following the same sequence of actions at different times could lead to different results, as users like to have a degree of knowledge of what to expect. Any lack of predictability should be compensated for by providing an explanation of why the results have been re-ranked in the order they have, or why a particular site has been recommended.

Secondly, the system needs to be highly scalable and very robust. Server-side personalized search systems present scalability problems over and above those of standard web search because the profiles for all users will need to be stored, these must be retrievable quickly, and the additional computational load of running a personalization algorithm will need to be catered for. No system with slow response times or periods of unreachability due to server overload will gain the loyalty of its users.

VIII. FUTURE WORK

We are planning to conduct empirical studies to evaluate the combination of binning and ranking. In particular, we are interested to see if presenting refinements in this way reduces the cognitive load as compared to our previous study. In addition, we plan to evaluate a system that simply ranks the refinements without the use of bins.

REFERENCES

[1] Organizing User Search Histories, IEEE Transactions On Knowledge And Data Engineering, VOL. 24, NO. 5, January 2010 Heasoo Hwang, Hady W. Lauw, Lise Getoor and Alexandros Ntoulas

[2] Personalizing Web Search using Long Term Browsing History Nicolaas Matthijs, University of Cambridge 15 JJ Thomson Avenue Cambridge, UK nm417@cam.ac.uk. Filip Radlinski, Microsoft 840 Cambie Street Vancouver, BC, Canada filiprad@microsoft.com.
[3] IEEEJ021 Organizing User Search Histories
[4] Query Chains: Learning to Rank from Implicit Feedback Filip Radlinski Department of Computer Science Cornell University Ithaca, NY, USA filip@cs.cornell.edu.
[5] Thorsten Joachims Department of Computer Science Cornell University Ithaca, NY, USA tj@cs.cornell.edu.
[6] "Personalizing Search Based on User Search Histories" By Mirco Speretta B.Sc. , Udine University, Udine, Italy 2000
[7] The Ranking of Query Refinements in Interactive Web-based Retrieval R. McArthur and P.D. Bruza Distributed Systems Technology Centre University of Queensland, Brisbane, Australia mcarthur@dstc.edu.au
[8] "Clustering Query Refinements by User Intent" Eldar Sadikov Jayant Madhavan Lu Wang Alon Halevy Stanford University Google Inc. Google Inc. Google Inc. eldar@cs.stanford.edu jayant@google.com. luwang@google.com halevy@google.com.