PERSONALIZING (RE-RANKING) WEB SEARCH RESULTS USING INFORMATION PRESENT ON A SOCIAL NETWORK

BY

SUSHRUTH PUTTASWAMY

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE & ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

DECEMBER 2006

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Dr. Gautam Das, for his constant guidance and support, and for giving me a wonderful opportunity to work on such a challenging project, which involved in depth knowledge of probability and statistics, and also for discussing various practical and interesting aspects of the problem in hand.

I am also grateful to Dr. Nan Zhang and Dr. Leonidas Fegaras for serving on my committee.

I would also like to thank Manos Pappagelis, Dr. Nick Koudas for helping me throughout the design and implementation of the conference paper version of this work. I would also like to thank my parents, my sister and all my friends for their constant help and support for this project and throughout my academic career.

Finally I would like to thank Arjun Dasgupta for his consistent support during my thesis work.

November 8, 2006

ABSTRACT

PERSONALIZING (RE-RANKING) WEB SEARCH RESULTS USING INFORMATION PRESENT ON A SOCIAL NETWORK

Sushruth Puttaswamy, MS

The University of Texas at Arlington, 2006

Supervising Professor: Dr. Gautam Das

We describe a social search engine paradigm which can be built on top of a classic search engine (e.g. Google, Yahoo, etc.) and a social information network (such as FriendSter). In this thesis, the objective was to design algorithms and develop methods to efficiently combine information available in the underlying systems (Search Engine & Social Information Network) to better satisfy the search needs of a user. We are interested on how to efficiently employ social information to re-order a list of URLs retrieved by querying a search engine. The objective was to re-order the list of URLs in a way that favors URLs that are more relevant to user's interest towards a personalized search engine. We conduct a thorough user study & come up with certain experiments to show some of the functionality of the system.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

CHAPTER 1

INTRODUCTION

Millions of users seek content using the search engines. They issue queries asking for content meeting specific criteria, follow some of the links in the results, click on ads, explore the pages, reformulate their queries, and perform other necessary actions. Search engines index large numbers of documents and let users query desired documents. However, most of the available search engines are not designed to meet individual user preferences. [7] Noted that almost half of the documents returned by search engines are deemed irrelevant by their users. Current information retrieval and data mining research tries to enhance user's web experience from several aspects. Two of them are,

- Create a better structural model of the web, such that it can interface more efficiently with search engines.
- Model user behavior as to predict users' interests better.

Search engines have become a modern paradox. While intended to make the Web more accessible and easier to use, more often than not, more questions than answers tend to crop up, as they retrieve thousands, if not millions of links, besides producing advertisements and sponsored links which are of little or no relevance to the user. Search engine results have improved over the past couple of years to a certain degree, but these improvements typify just a tiny fraction of what's needed, and what's possible in the world of web search engines.

As the amount of information on the Web increases rapidly, it gives rise to many new challenges for Web search. When the same query is submitted by different users, a typical search engine returns the same result, regardless of who submitted the query. This may not be desirable for users with different information needs. Clearly, work in user profiling is closely related to building better personalized systems. Different methods of gathering user data is often coupled with various personalization systems. Personalized search is the fine-tuning of search results and advertising based on an individual's preferences, demographic information and other factors. Presumably, the better a search engine understands a user's interests and preferences, the better it is able to target search results, advertising, sponsored links, etc.

For example, suppose a user searches for information to help him plan a vacation to California. Without personalized search, the user receives thousands of content links and ads based purely on their global relevance i.e., how relevant they are to consumers in general, without regard for this individual consumer's demographic data, lifestyle and interests. With personalized search, the results take on an entirely new level of relevance. Results returned to a budget-conscious traveler vacationing with his wife and children are different from those returned to, for example, a more upscale traveler, a Native American who vacations with friends or he may be a single traveling alone. These personalized results are first filtered according to a search engine's basic criteria, such as popularity, keyword relevance, etc., and then they are automatically tuned to meet the individual's specific needs.

In this thesis we initiate a study of personalizing search by using data available on a social network. Rather than just using the user's search profile to give him better results, why don't we take into account all people whom we think would share his interest? This is where a social network comes into picture. A **social network** is a social structure made of nodes which are generally individuals or organizations. It indicates the ways in which they are connected through various social familiarities ranging from casual acquaintance to sharing similar interests. The essential idea is that people make decisions based primarily on a few *people* whom they trust. The average person has a set of experts whom they consult in designated areas: the computer expert, the car expert, the fashion expert, the financial expert. If the opinions of these experts can be collected, they are incredibly useful: it is this metadata (data about other data) that gives the most intelligent filtering and sorting of the information on the internet. In particular we make the following contributions:

- We initiate the study of personalized search, based on information present on a social network. We present a framework within which each user gets results pertaining to his interest & his neighboring friends interests.
- We show the architecture of the system designed along with the individual components involved, the database schema.

- We present efficient algorithms based on similarity between queries, tokens & URL's to obtain better search results pertaining to the user's interest.
- We show how our algorithms can be applied to extract similar results among the neighbors of a user.
- We present the results of a thorough experimental evaluation of our algorithms using data provided from a set of users who have registered in our system.

This thesis is organized as follows: Chapter 2 reviews related work. In Chapter 3 we present our overall framework and demonstrate several important properties of Social Networks & how they are useful in personalizing search results. In Chapter 4 we present the architecture of the system including the individual components of the system. In Chapter 5 we present our algorithms and methodology to enable search results personalization. In Chapter 6 we provide some snapshots of the actual social search engine. In Chapter 7 we present the results of our experimental validation of our overall methodology. We conclude in Chapter 8.

CHAPTER 2

RELATED WORK

Personalization applications cover a gamut of spectrum. We have filtering systems at one end of this spectrum, which filter input from an information resource; from the input, information of possible interest is marked. One such example of a filtering system is SmartPush [8] which combines different novel ideas together. The system filters through information using semantic meta-data embedded in news articles. In addition, it builds a user profile using a hierarchical concept model.

At present, many available systems allow for personalization to some extent. Some systems like 'Yahoo' allow users to specify page content, while some others recommend web pages, books, music, etc. [9] Contains a survey of many of the available systems and their methodologies.

[7] Explores the utility of incorporating noisy implicit feedback obtained in a real web search setting to improve web search ranking. Two alternatives of incorporating implicit feedback into the search process are proposed here, namely re-ranking with implicit feedback and, incorporating implicit feedback features directly into the trained ranking function.

[2] Describes a strategy for personalization of web search by making use of the following concepts:

(1) A user's search history can be collected without direct user involvement.

(2) The user's profile can be constructed automatically from the user's search history.

(3) The user's profile is augmented by a general profile which is extracted automatically from a common category hierarchy.

The common feature in all these approaches is that the data of only the user who instigates the search session is recorded & processed upon. Again the search pattern, preferences & objectives of the user in question is the one who posed the query. Our approach extends this idea by incorporating such kind of information from people whom the user considers relevant to him & based on that does re-ranking. This way not only are we taking into consideration the user’s personal interest but we are also trying to refine & add more value to his profile by looking at all the neighbors of this user whom he considers relevant.

CHAPTER 3

FRAMEWORK

In this chapter we describe algorithms which do re-ranking based on user & his friends profiles. Our actual algorithms are described in more detail in the subsequent chapters.

3.1 Methodology

Before describing the re-ranking algorithm we define two components that will be useful in the algorithm: (a) the User Query History Document (UQHD) and (b) the user interconnectivity property.

3.1.1 User Query History Document (UQHD)

UQHD may be considered a virtual and dynamic document that is created by appending to a file all query information submitted by a user. In the simple case it is enough to think of it as a document that is compiled by appending to it submitted user query terms. Query terms may be repeated.

Conference conference management system Confious social search search engines infocious teoma database research trends social networks Papagelis collaborative filtering indexing database group Greece kos island Toronto ttc routes patio world cup zidane personalization google checkout …

Figure 3.1 Example of a simple UQHD for a random User u.

Bear in mind that people are not static as they have many fleeting and seasonal interests. A student might intensely research Abraham Lincoln for a school project but may care nothing at all about it later. S/he may read about spectacular tragedies such as a recent fire in Paraguay that led to hundreds of deaths, but s/he is not generally concerned with death, fire, Paraguay, or supermarkets. Seasonal phenomena like elections, the Olympics, sports leagues, etc. also lead to variable interests.

In a more complex case the UQHD may include much more useful information for each query (query aging, query locality, query disambiguation, query frequency, etc.). This depends in our policy and the nature of algorithms we would like to design in the future. Sliding Window algorithms may be designed to identify current trends in user's search behavior.

Social information should be able to easily identify the click-through behavior of users in my network for a recent period of time and be able to suggest better URLs than classic link-based algorithms (e.g., Google). Therefore, we may assume that a UQHD is associated with each user, or otherwise a UQHD index exists in the system.

3.1.2 User Interconnectivity

Consider a subset of users in the system that have at least one common query term in their UQHDs. It is able to assess the interconnectivity of these users by comparing the similarity between their UQHDs (many methods are available to assess the similarity of two documents). Based on the similarity of two UQHDs we can infer the similarity of two users' query behavior. More similar UQHDs imply more similar users. A natural extension of this algorithm should take into account the whole click-through analysis of the form user→query→clicked-pages and is expected to be more accurate in assessing interconnectivity between users for a specific query *q*.

3.1.3 Re-Ranking Algorithm

The following steps describe a re-ordering algorithm that takes into account social information of a user's network. The focus here is on the conceptualization of an algorithm that is able to assess social information to re-order a ranked list of search results. Optimizations should be considered to make it efficient, mainly based on random sampling and offline computations.

Input: Query q is submitted by active user a

Output: A ranked list L' of relevant to the query URLs that exploits social information

A ranked list L_q of (all) relevant to query q URLs is retrieved by redirecting the query q to a link analysis based search engine (e.g., Google)

A subset S of (all) users that have previously submitted the query q is retrieved by scanning through the UQHD index. User a and users in S have at least one common query q in their UQHDs, thus it is able to compute the similarity of user a and each of the users in S by computing the similarity of their associated UQHDs. The result is a ranked list $N_{a,q}$ of users in the network of the active user a based on the query q. The list N can be considered as the list of neighbors of user a when the query q is submitted and the ranking of the list indicates which users in the social network of user a have more similar query behavior. This implies a weighting scheme of the users in the virtual dynamic network. The list $N_{a,q}$ needs to be normalized (e.g., weights should sum to 1)

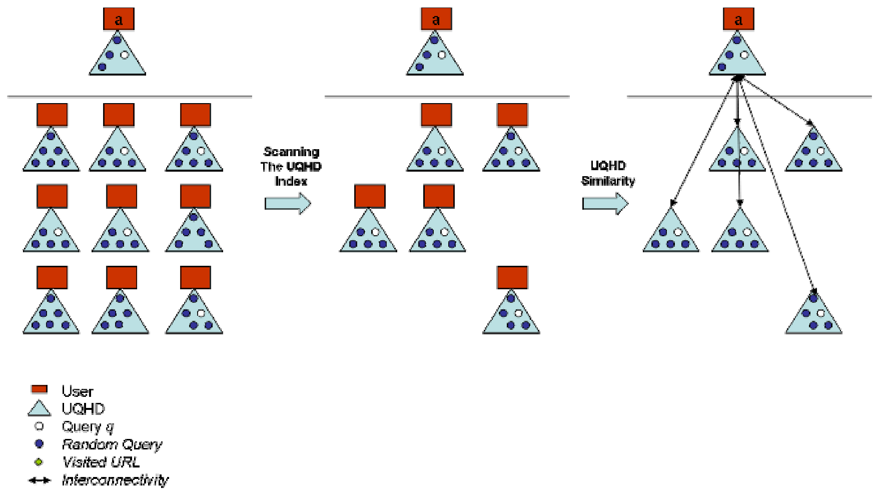
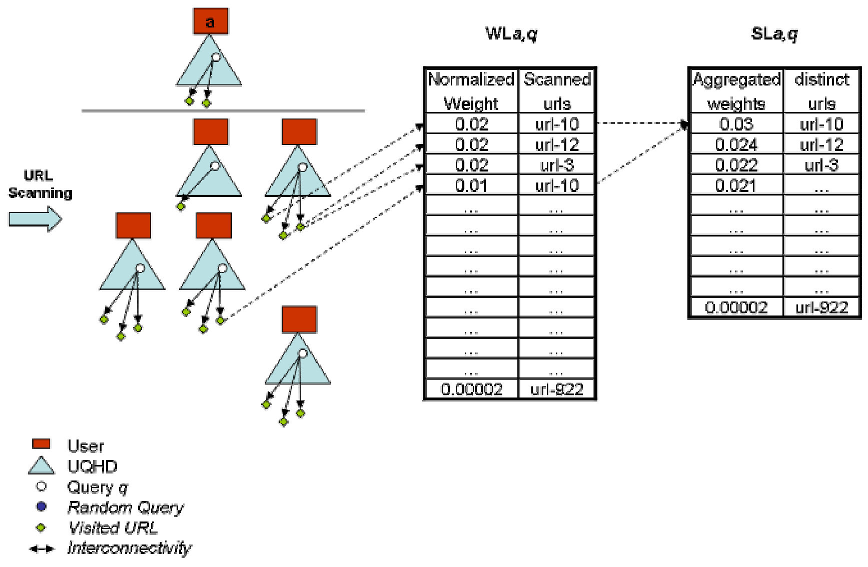


Figure 3.2 Re-ranking algorithm: Users selected based on query.

For each of the users in the list $N_{a,q}$ of the active user *a* and for query *q* we need to scan the set of URLs that this user has visited (this can be liked/ saved/ bookmarked/ tagged/ or whatever) when submitted the query *q* (in case of multiple occurrences of q a user's UQHD). Each of these URLs is associated with the user's interconnectivity weight as it is assessed in the list $N_{a,q}$. The set of these URLs can be represented as a weighted list of URLs $\mathrm{WL}_{a,q}$. Thus $WL_{a,q}$ represents a list of URLs as they have been retrieved based on query *q* and user's *a* prior query behavior. Bear in mind that the list $WL_{a,q}$ may contain duplicates (Identical URLs as they have been assessed by different users in the network of user *a* as formed by query *q*.). To make sure that each URL coming from network information exists only once in the list we need to aggregate the weights of URLs that appear in $WL_{a,q}$ more than once. This results in a ranked list $SL_{a,q}$ of URLs. $SL_{a,q}$ provides a ranked list of all URLs that are relevant to a user's *a* query *q* by assessing social information.

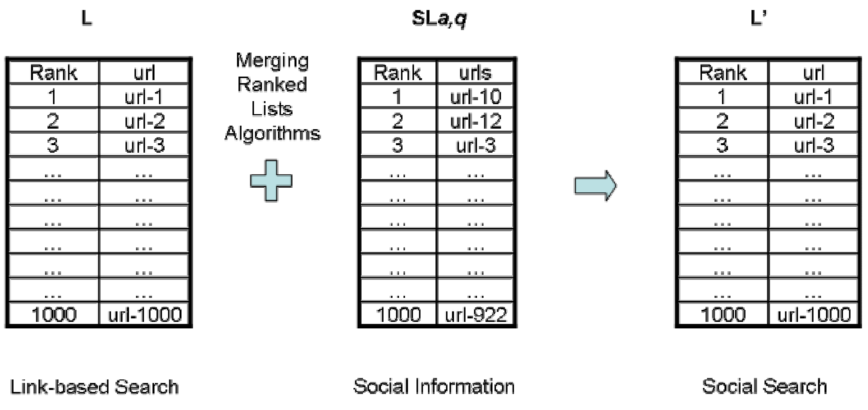


Normalized Weight	Scanned urls
0.02	url-10
0.02	url-12
0.02	url-3
0.01	url-10
...	...
...	...
...	...
...	...
...	...
...	...
...	...
...	...
...	...
0.00002	url-922

Aggregated weights	distinct urls
0.03	url-10
0.024	url-12
0.022	url-3
0.021	...
...	...
...	...
...	...
...	...
...	...
0.00002	url-922

Figure 3.3 Re-ranking algorithm: A ranked list of URLs.

Based on the list $SL_{a,q}$ that has been compiled using social information it is possible to employ algorithms that merge overlapping ranked (ordered) lists and compile a re-ordered ranked list *L'* (e.g., similar to the way that meta search engines combine results from many search engines, or similar to the work of determining the top-*k* queries from multiple sources)



Rank	url
1	url-1
2	url-2
3	url-3
...	...
...	...
...	...
...	...
...	...
...	...
1000	url-1000

Rank	urls
1	url-10
2	url-12
3	url-3
...	...
...	...
...	...
...	...
...	...
...	...
1000	url-922

Rank	url
1	url-1
2	url-2
3	url-3
...	...
...	...
...	...
...	...
...	...
...	...
1000	url-1000

Figure 3.4 Re-ranking algorithm: Merging of the two ranked lists.

CHAPTER 4

ARCHITECTURE

We present the architecture of *Social Search*, an online social search engine at University of Texas at Arlington. Through the *Social Search* site we were able to study a reflection of the real world community structure & ways of using it to do re-ranking. *Social Search* was introduced at UTA in the summer of 2006. Members can use *Social Search* to send e-mail and invitations, chat, find friends with similar interests. But primarily the system was designed to provide user's the option to search on the web.

4.1 User Registration Data

Users could register in the system by automatically signing up for it or by using invitations sent by already registered members. Upon registering, users were required to supply their names, e-mail addresses, first & last names (for searching other people on the network).

In the second registration step, users were asked to list their friends and acquaintances. In *Social Search* these people are called 'search-buddies'. Users identified their search-buddies by searching for them in the registered list of user's or by entering their names manually.

If a user adds a search-buddy who is already registered, the search-buddy will get a notification that the user has requested to be their search-buddy and can accept or decline the request. If the 'search-buddy' is not yet registered, they will get an invitation to join *Social Search*. The resulting dataset was a social network with profiles for each of the members.

4.2 Searching & Book Marking

Once a user has been registered & all his friends' information gathered up, the next step is to incorporate searching ability for each user & gather book marks. This process is crucial for the system because based on the data gathered during this process itself the re-ranking of URL's is done.

User is presented with a search box similar to Google where in he can enter his search query. The query is then sent to Google making use of the API provided & the results are got which are displayed to the user in groups of 10. User would now go through the link text, snippet below the link & other information & decide to explore a link further which he thinks is relevant to the query by clicking on the link. These clicks on links are recorded & associated with the query issued.

User's can then go to their search history by clicking on the 'My Knowledge' tab which shows them the list of queries issued along with the links clicked on for each of the queries. Here the link along with the query can be book marked by providing a small description regarding the link along with some labels or tags for the link. Again these tags are very important in our algorithms as certain similarity is done between the query & the tags. So, finally for each user we have their search history along with book marking information for certain links.

4.3 Algorithm Selections

Users are provided the option of selecting which algorithm they want to include for getting re-ranked results. These options are represented by checkboxes. User can select which algorithm to be included for processing & get the respective results. The options provided range from algorithm 1 to 4 along with Google's results & a combination of both.

4.4 System Components

Figure showing the whole system as a collection of components.

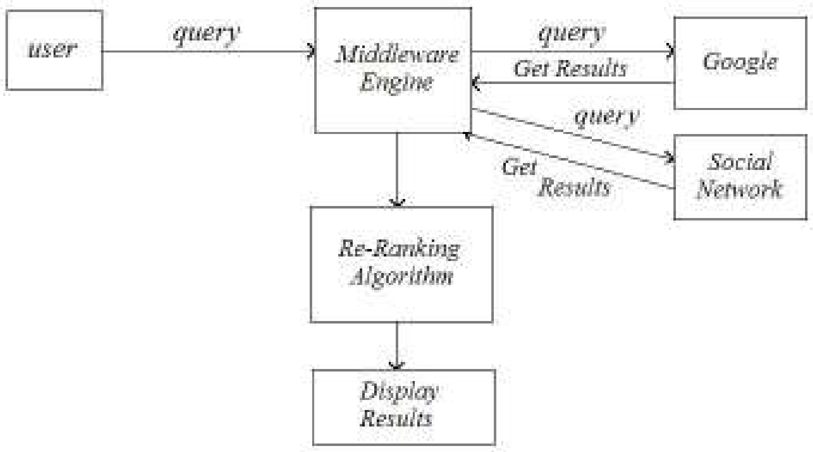


Figure 4.1 System Components

4.5 Similarity Function

Similarity function based on TF/IDF is used to find similar queries. Some of the algorithms involve mapping user queries to similar queries present in the user log. Such kind of mapping is also done based on the similarity between user query & a token also. This section explains the similarity function provided by the spider software which is made use to find similar words or tokens.

In this section, we present a detailed description of tf-idf (term frequency, inverse document frequency) and cosine similarity for matching against the values in a single relational attribute.

Let Base denote a base table with a string-valued attribute sva against which the flexible matching needs to be performed, and let Search denote the table containing the search strings (this may consist of just a single record with a single attribute value, or may be more complex). Flexible string matching is performed in two stages:

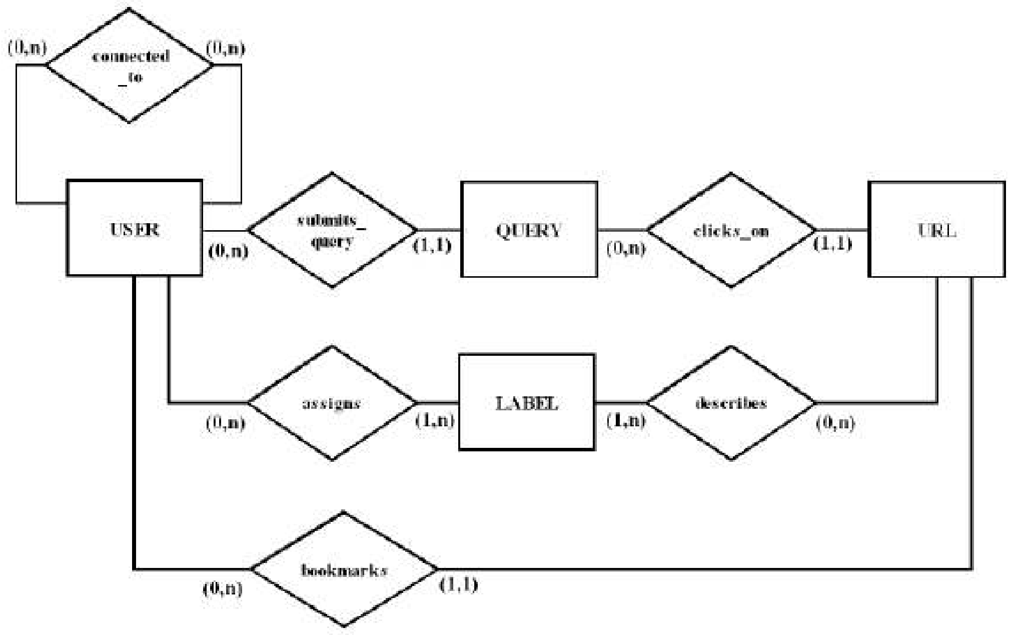
- At *pre-processing time*, the Base table is preprocessed, and tokens (words, q-grams, etc.) are extracted from each database string in Base.sva. A variety of auxiliary tables get created, to compute the idf's of each token, and ultimately to associate each database string with a (normalized) weight vector (incorporating both tf and idf) corresponding to the tokens extracted from it.

- At *query time*, a similar process is first done with respect to the Search table. Then, an SQL query that operates on the auxiliary tables created from Base and Search is executed, which identifies the matching records, along with their similarity score. Essentially, this query computes the cosine similarity (inner product) of the weight vectors of the search string with the weight vectors of the database strings in Base.sva, taking the weights of the common tokens

4.6 Database Schema

Figure below describes the schema of the database that was used in the xxxx system. The name of the database is socialsearch

Entity-Relationship Model



SocialSearch Database Relational Model

USER (<u>user_login</u>, user_password, user_fname, user_lname, user_email, user_timestamp, confirmation_key)

QUERY (<u>query_id</u>, query_text, query_timestamp, user_login)

LABEL (<u>label_id</u>, label_text, label_timestamp)

URL (<u>url_id</u>, url_text, url_domain, url_timestamp, url_description, url_title, url_local_path, url_rank, url_vote, query_id, user_id)

CONNECTED_TO (<u>user_login_ping</u>, <u>user_login_pong</u>, link_timestamp, link_status)

ASSIGNS (<u>user_login</u>, <u>label_id</u>)

DESCRIBES (<u>label_id</u>, <u>url_id</u>)

Figure 4.2 Database Schema of the Social Search system

CHAPTER 5

ALGORITHMS

We present an analysis of the algorithms that are used as part of the re-ranking process. Totally the numbers of algorithms which are present are 4 with variations of 2, 3 & 4 being present. Below is the description of each of the algorithms along with an analysis of the time & space constraints for each of them.

For easy understandability & readability the following annotations have been used.

Annotation:

Ls: An ordered list of web pages based on social information

U: A set of users

Q: A set of queries

T: A set of tags

W: A set of web pages

$u_{a:}$ The active user

q_a: The active query submitted by the active user

t_a: The active tag assigned by the active user for an webpage

w_a: The active webpage that the active user u_a visited for the submitted query q_a

v_a: The vote of the active user u_a to the active webpage w_a (either 0 or 1)

$N_a(k)$: Set of Neighbors of user u_a in a neighborhood of size k. (includes u_a)

$Wu_{a,}q_q$: Set of WebPages visited by user u_a for the query q_a

Wu_a,t_a: Set of WebPages assigned by user u_a with the tag t_a

S_q: The set of queries similar to the query q_a (based on 3-grams)

S_t: The set of tags similar to the query q_a (based on 3-grams)

Method 1:

This method is the simplest & intuitively the most obvious one. In this method when a query is submitted, a sequence of events happen which starts of by finding all the users who are in the neighborhood of the active user who submitted the query. Then for each of the users in the neighborhood we find the set of web pages visited by that user for that particular query. We get a list of URL's which have been visited by all the neighbors of the active user for that particular query. Next step is to aggregate the votes given by users to each of these URL's. Finally we sort the list of URL's based on the number of votes obtained for each of them. So, we have a set of URL's along with the votes for each of them obtained from the social network with respect to the query issued.

INPUT: Submitted query q_a

OUTPUT: A sorted list of WebPages Ls

```
Compute $N_a(3)$ for user $u_a$
For each user $u_a$ in $N_a(3)$
  Compute $Wu_a,q_a$
  For each $w_a$ in $Wu_a,q_a$
     Aggregate user (weighted) vote $v_a$ to the list Ls
Sort Ls
```

Method 2:

This method is similar to method one expect that along with getting the URL's visited by the neighbors for the query in question, we also get the URL's visited by the user & his neighbors for queries which are similar to the initial query based on the similarity function. As in method one, we find all the neighbors of the user who are at a distance of 3-hops. Then we find all queries similar to the issued query based on the similarity function. Once this is done we then find the list of URL's visited by each of the users in the neighborhood for the query issued & each of the queries returned by the similarity function. As similar to method 1 we aggregate the votes for each URL & then sort the list of URL's based on votes.

INPUT: Submitted query q_a

OUTPUT: A sorted list of WebPages Ls

```
Compute $N_a(3)$
Compute $S_q$
For each user $u_a$ in $N_a(3)$
  For each $q_a$ in $S_q$
  Compute $Wu_a,q_a$
      For each $w_a$ in $Wu_a,q_a$
           Aggregate user (weighted) vote $v_a$ to the list Ls
Sort Ls
```

Method 3:

In this method instead of finding queries which are similar to the issued query we find the tokens which are similar to the issued query. The set S_t is the set obtained by finding similarity between the issued query & tokens present in the database. After this set is found the method is similar to method 2. For each of the neighbors we find all the URL's visited for each of the keywords in the set S_t. Finally the votes are aggregated & the list of URL's are sorted based on the vote values.

INPUT: Submitted query q_a

OUTPUT: A sorted list of WebPages Ls

```
Compute $N_a(3)$
Compute $S_t$
```

```
For each user $u_a$ in $N_a(3)$
  For each $q_a$ in $S_t$
  Compute $Wu_a,q_a$
     For each $w_a$ in $Wu_a,q_a$
        Aggregate user (weighted) vote $v_a$ to the list Ls
Sort Ls
```

Method 4:

This method makes use of the concepts present in both method 2 & method 3. What this means is that this method makes use of similarity between the issued query to both the queries & tokens present in the database & based on this does aggregation of URL's.

Method 4 (2 Union 3):

```
INPUT: Submitted query $q_a$
OUTPUT: A sorted list of WebPages Ls

Compute $N_a(3)$
Compute $S_q$
For each user $u_a$ in $N_a(3)$
```

```
  For each $q_a$ in $S_q$
  Compute $Wu_a,q_a$
     For each $w_a$ in $Wu_a,q_a$
        Aggregate user (weighted) vote $v_a$ to the list Ls
Compute $S_t$
For each user $u_a$ in $N_a(3)$
  For each $q_a$ in $S_t$
  Compute $Wu_a,q_a$
     For each $w_a$ in $Wu_a,q_a$
        Aggregate user (weighted) vote $v_a$ to the list Ls
Sort Ls
```

Re-ranking

The re-ranking algorithm is basically a sort-merge between the lists obtained from Google & social information.

Ls: An ordered list of WebPages based on social information

Lg: An ordered list of WebPages based on link-analysis (e.g., retrieved from Google)

L: An ordered list of WebPages based on link-analysis and social information

w: A webpage

Rw: The rank of w_i in a list

IRw: The inverted rank of w_i in a list

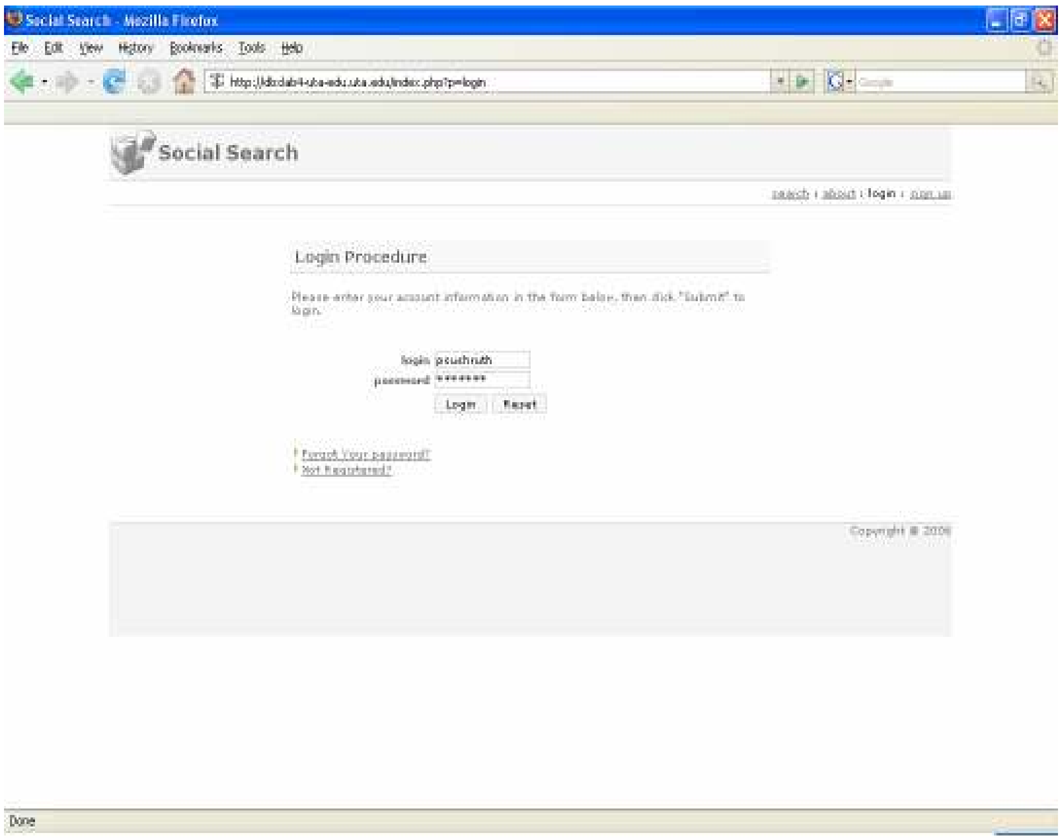
```
For each (or top-k) w in Ls
  IRw ← 1/Rw
  Aggregate IRw to the list L

For each (or top-k) w in Lg
  IRw ← 1/Rw
  Aggregate IRw to the list L
Sort L
```

CHAPTER 6

SNAPSHOTS

Here are some snapshots of the Social Search system.


Social Search - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://dbclab4-uta-edu.uta.edu/index.php?p=login
Social Search
search | about | login | sign up

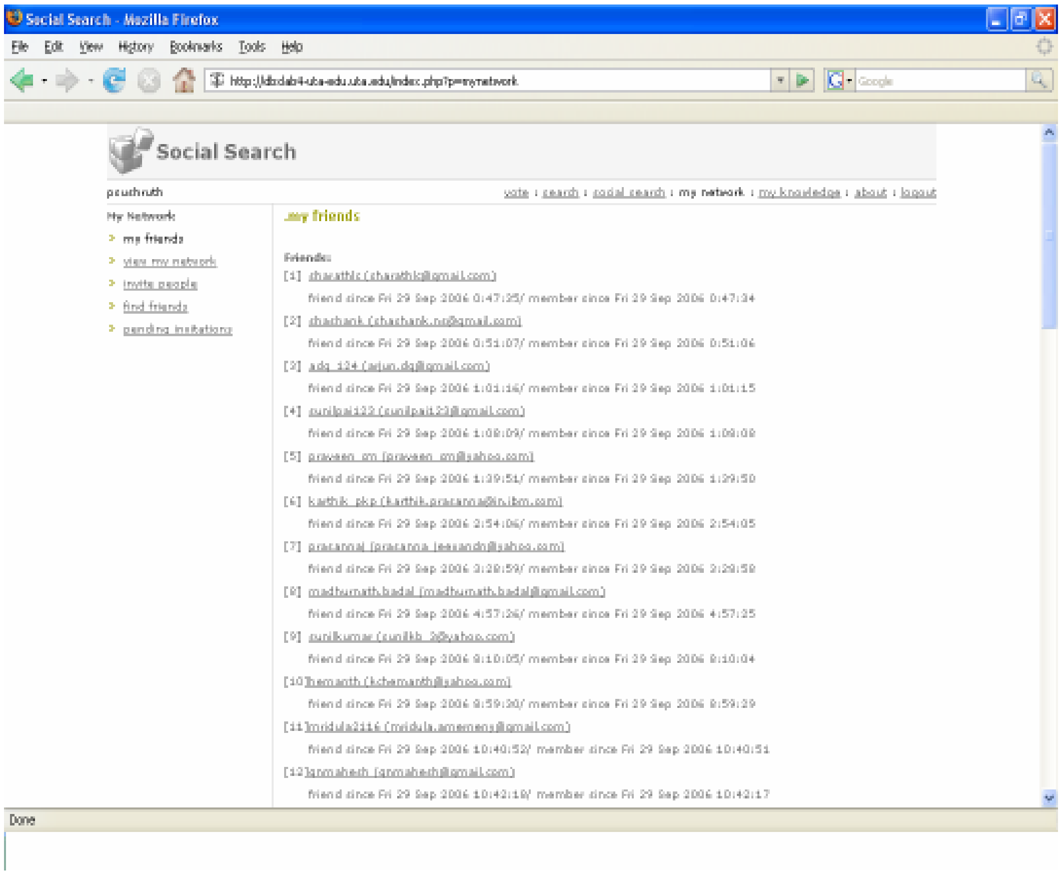
Login Procedure

Please enter your account information in the form below, then click "Submit" to login.

login pcushruth
password *******

Login Reset
Forgot Your password?
Not Registered?
Copyright © 2006
Done

Figure 6.1 Login screen to the Social Search system.


Social Search - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://dbxlab4-uta-edu.uta.edu/index.php?p=mynetwork
Google
Social Search
vote : search : social search : my network : my knowledge : about : logout
My Network
my friends
view my network
invite people
find friends
pending invitations
Done

psushruth

..my friends

Friends:

[1] sharathk (sharathk@gmail.com)
friend since Fri 29 Sep 2006 0:47:25/ member since Fri 29 Sep 2006 0:47:24

[2] shashank (shashank.nr@gmail.com)
friend since Fri 29 Sep 2006 0:51:07/ member since Fri 29 Sep 2006 0:51:06

[3] adq_124 (arjun.dg@gmail.com)
friend since Fri 29 Sep 2006 1:01:16/ member since Fri 29 Sep 2006 1:01:15

[4] sunilpai123 (sunilpai123@gmail.com)
friend since Fri 29 Sep 2006 1:08:09/ member since Fri 29 Sep 2006 1:08:08

[5] praveen_gm (praveen_gm@yahoo.com)
friend since Fri 29 Sep 2006 1:39:51/ member since Fri 29 Sep 2006 1:39:50

[6] karthik_pkp (karthik.prasanna@in.ibm.com)
friend since Fri 29 Sep 2006 2:54:06/ member since Fri 29 Sep 2006 2:54:05

[7] prasannaj (prasanna_jeevandn@yahoo.com)
friend since Fri 29 Sep 2006 3:28:59/ member since Fri 29 Sep 2006 3:28:58

[8] madhumath.badal (madhumath.badal@gmail.com)
friend since Fri 29 Sep 2006 4:57:26/ member since Fri 29 Sep 2006 4:57:25

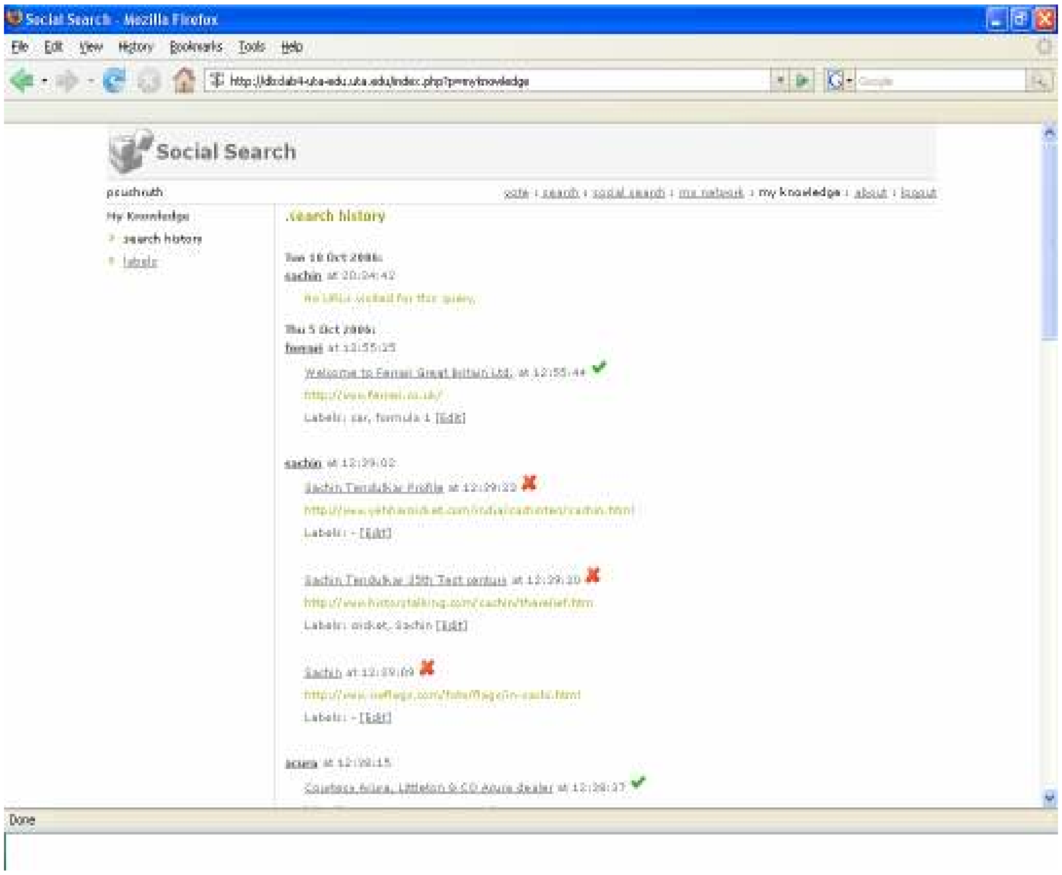
[9] sunilkumar (sunilkb_3@yahoo.com)
friend since Fri 29 Sep 2006 8:10:05/ member since Fri 29 Sep 2006 8:10:04

[10] hemanth (kchemanth@yahoo.com)
friend since Fri 29 Sep 2006 8:59:30/ member since Fri 29 Sep 2006 8:59:29

[11] mridula2116 (mridula.amemens@gmail.com)
friend since Fri 29 Sep 2006 10:40:52/ member since Fri 29 Sep 2006 10:40:51

[12] gnmahesh (gnmahesh@gmail.com)
friend since Fri 29 Sep 2006 10:42:18/ member since Fri 29 Sep 2006 10:42:17

Figure 6.2 “My Network” option within the Social Search system.


Social Search - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://idir.dab4-uta-edu.uta.edu/index.php?p=myknowledge
Social Search
vote | search | social search | my network | my knowledge | about | logout
My Knowledge
search history
labels
Done

pcushruth

.search history

Tue 10 Oct 2006:
sachin at 20:24:42
No URLs visited for this query.

Thu 5 Oct 2006:
ferrari at 12:55:25
Welcome to Ferrari Great Britain Ltd. at 12:55:44 ✓
http://www.ferrari.co.uk/
Labels: car, formula 1 [Edit]

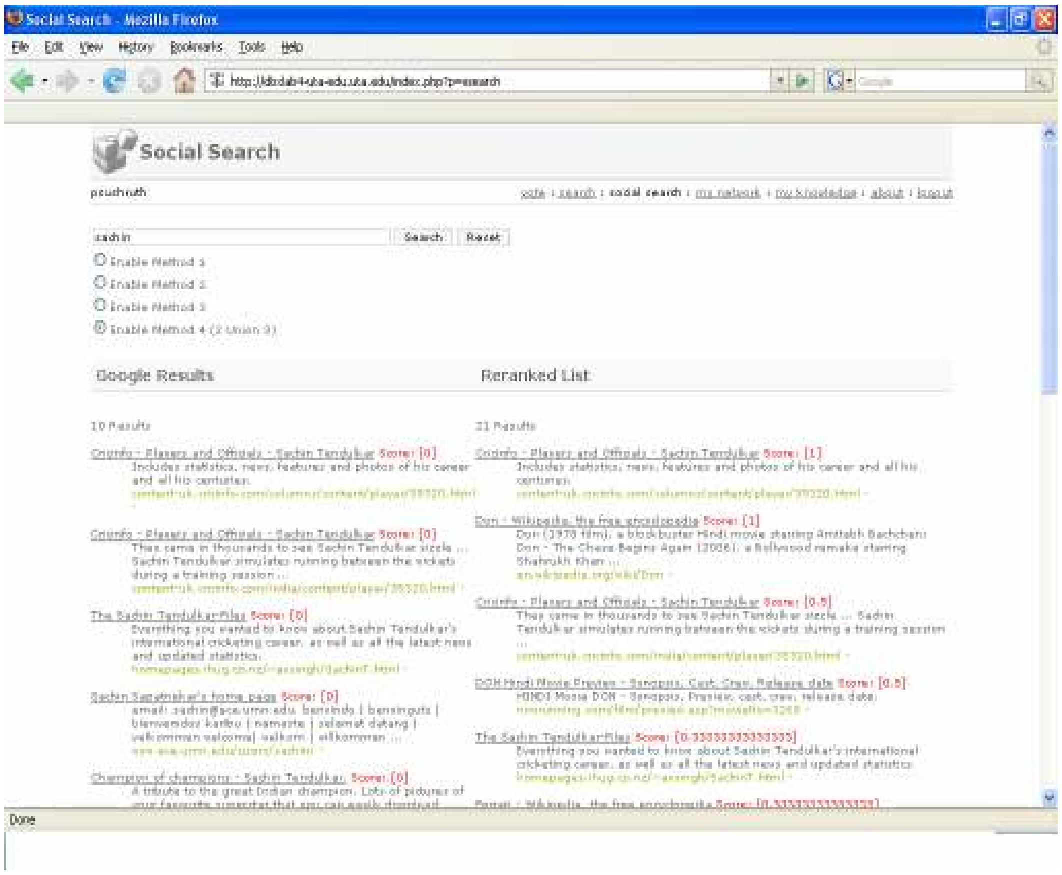
sachin at 12:29:02
Sachin Tendulkar Profile at 12:29:22 ✗
http://www.yehhaicricket.com/india/cachintendulkar/sachin.html
Labels: - [Edit]

Sachin Tendulkar 35th Test century at 12:29:20 ✗
http://www.historytalking.com/cachin/thaerlef.htm
Labels: cricket, Sachin [Edit]

Sachin at 12:29:09 ✗
http://www.ineflags.com/fotw/flags/in-sachi.html
Labels: - [Edit]

acura at 12:28:15
Courtesy Acura, Littleton & CO Acura dealer at 12:28:37 ✓

Figure 6.3 Query Log for the user.


Social Search - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://idbclab4-uta-edu.uta.edu/index.php?p=research
Social Search
pcushruth
vote | search | social search | my network | my knowledge | about | logout
sachin
Search
Reset
Done

( ) Enable Method 1
( ) Enable Method 2
( ) Enable Method 3
[✓ Enable Method 4 (2 Union 3)]

Google Results

10 Results

Cricinfo - Players and Officials - Sachin Tendulkar Score: [0]
Includes statistics, news, features and photos of his career and all his centuries.
content-uk.cricinfo.com/columns/content/player/35320.html -

Cricinfo - Players and Officials - Sachin Tendulkar Score: [0]
They came in thousands to see Sachin Tendulkar sizzle ... Sachin Tendulkar simulates running between the wickets during a training session ...
content-uk.cricinfo.com/india/content/player/35320.html -

The Sachin Tendulkar-Files Score: [0]
Everything you wanted to know about Sachin Tendulkar's international cricketing career, as well as all the latest news and updated statistics.
homepages.ihug.co.nz/~asingh/SachinT.html -

Sachin Sapatnekar's home page Score: [0]
email: sachin@ece.umn.edu. bienvenido | benvinguts | bienvenidos karibu | namaste | selamat datang | velkommen welcome| welkom | willkommen ...
www.ece.umn.edu/users/sachin/ -

Champion of champions - Sachin Tendulkar Score: [0]
A tribute to the great Indian champion. Lots of pictures of your favourite superstar that you can easily download ...

Reranked List

21 Results

Cricinfo - Players and Officials - Sachin Tendulkar Score: [1]
Includes statistics, news, features and photos of his career and all his centuries.
content-uk.cricinfo.com/columns/content/player/35320.html -

Don - Wikipedia, the free encyclopedia Score: [1]
Don (1978 film), a blockbuster Hindi movie starring Amitabh Bachchan; Don - The Chase Begins Again (2006), a Bollywood remake starring Shahrukh Khan ...
en.wikipedia.org/wiki/Don -

Cricinfo - Players and Officials - Sachin Tendulkar Score: [0.5]
They came in thousands to see Sachin Tendulkar sizzle ... Sachin Tendulkar simulates running between the wickets during a training session ...
content-uk.cricinfo.com/india/content/player/35320.html -

DON Hindi Movie Preview - Synopsis, Cast, Crew, Release date Score: [0.5]
HINDI Movie DON - Synopsis, Preview, cast, crew, release date.
nowrunning.com/film/preview.asp?movieId=1268 -

The Sachin Tendulkar-Files Score: [0.333333333333333]
Everything you wanted to know about Sachin Tendulkar's international cricketing career, as well as all the latest news and updated statistics.
homepages.ihug.co.nz/~asingh/SachinT.html -

Parzan - Wikipedia, the free encyclopedia Score: [0.333333333333333]

Figure 6.4 Google & Social Search results for a query.

CHAPTER 7

EXPERIMENTS

In this chapter we present an experimental evaluation of our frame- work. Our implementation of our techniques is in PHP and Mysql and our evaluations are performed on an Intel Pentium 2GHz processor system with 1GB of memory. The system was hosted on a Linux machine with Apache as the web server. Some of the other technologies made use within the system are JavaScript for client side scripting, Ajax, CSS for a consistent user interface. For comparison and accuracy measures, we report the algorithms presented in chapter 5.

The feedback provided by users provides a baseline measurement to compare the accuracy of our methods. All of our experiments provide information on the accuracy & some interesting features about the system. In all the experiments separate studies are done for each of the algorithms & analyzed.

7.1 Experimental methodology

All of our results have been taken by the study of the xxx system. The data set consists of real world data provided by actual users. Experiments were carried out in two steps.

In the first step the system was initially launched & invitations were sent out to people asking them to sign up. People were asked to login to the system after signing up & conduct certain specific search queries on the system. The algorithms were not included in the system initially because of lack of user query logs. So users were given specific topics like Cars, Cricket & asked to make searches specific on these topics. Users were also asked to bookmark those links which they thought were relevant to the query. This was simulated by initial click on the link & then by providing much more information about the link by specifying certain tags & a small description for each of the link clicked. This way the initial part of the experiments included collecting user queries & building up the query log.

After the end of the first part of the experiments the system had sufficient users, queries, URL's & query log for the algorithms to be run. In the second part of the experiments the algorithms were included in the system & users were asked to search for something which they had already searched or something to similar to their previous query. Users were given the option for selecting which of the algorithms they wanted to be included in the re-ranking process. Based on the selected algorithm the corresponding results were displayed.

The results were displayed in two columns where in the left column had the original Google results for the issued query & the right column had the results obtained from the social search engine. The results also display the score that was obtained for each of the URL. Currently only the first 10 results from Google & a combination of Google & social search results are displayed.

Users were also provided a checkbox where in they could give out their opinion regarding the results obtained. For each query issued & the results obtained users could compare the results obtained by Google & the social search engine & give out their satisfaction by checking on the checkbox based on whether they are satisfied or not. So for each of the queries a survey was conducted as to whether the user was satisfied with the social search results compared to the Google results.

Lastly, an option to vote for the best algorithm was also included within the system. Six different radio buttons were provided for each of the algorithms & users were asked to vote for which algorithm they thought was the best. Every time user issued a query he was asked to make a selection as to which algorithm gave him the best results.

All these data was collected over a period of 2 months. As of the first of November there were 105 people on the network with a total of 400 queries & around 600 bookmarked URL's. The system also had around 100 tags associated with the URL's. The diameter of the network was 4.

7.2 Accuracy based results

These experiments give a rough estimate of the accuracy of the social search system. As mentioned above all the experiments were carried out on real data provided by actual users. The accuracy was calculated based on the feedback provided by users. The feedback was in the form of a Yes or a No based on whether the user was satisfied with the results obtained by the social search system when compared to Google. The number of Yes's was recorded over a period of 2 months.

Figure 7.1 shows the feedback of users over time for each of the algorithms. To give an accurate estimate of the actual number of Yes's, average of the number of Yes's was used. The graph shows an estimate as to how the user's opinion about the algorithms varied over time. As seen by the graph during the initial stages of the experiment the accuracy of the system was relatively low. As the number of users increased & considerable amount of log was recorded over time the accuracy of the system began to improve.

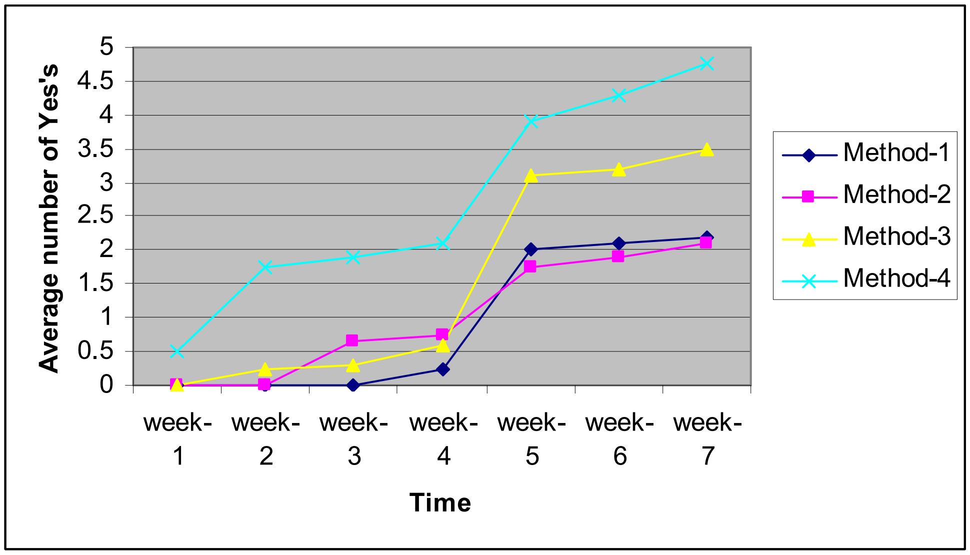


Figure 7.1 Accuracy of each algorithm based on variation time

Figure 7.2 shows the feedback of users varying over the number of records in the query log. Put in simple words this shows how the accuracy of the system changed as the log size increased. The log size shown is the average number of records within the query log per user. Here again the log size kind of directly proportional to the accuracy of the system. This experiment shows that for large number of records within the query log of each user the system performs really well & has high accuracy.

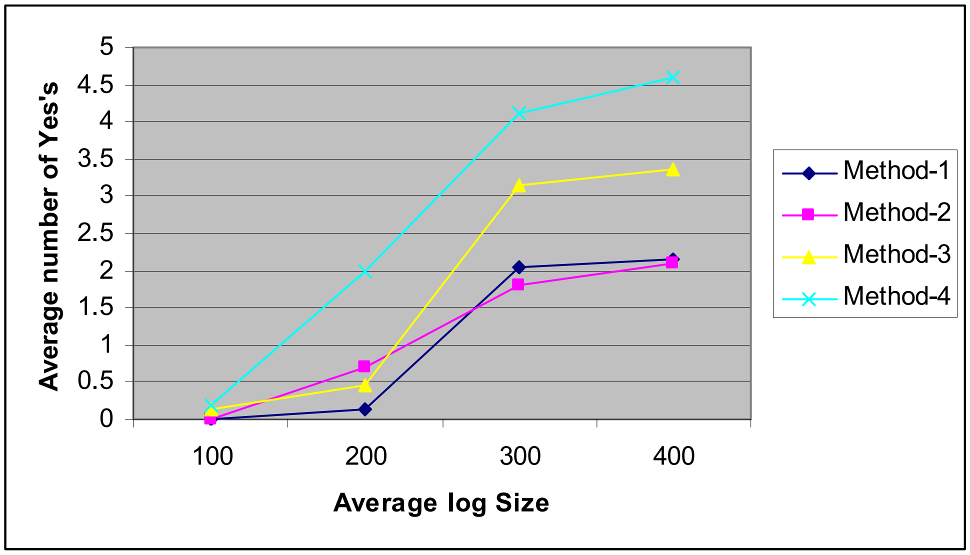


Figure 7.2 Accuracy of each algorithm based on variation Log size

7.3 Evaluation based results

These experiments are intended at providing some insight into the performance of the system. Main purpose is to find out how fast or slow the system is. The particular attribute that we focus on is the time taken. We compare the time taken for each of the algorithms along with Google. The situations are varied based on the log size, depth of the network & the average number of users in the network.

Figure 7.3 shows the variation of the time taken by each of the algorithms with respect to the average number of records within the query log per user. As show by the graph when the log size is relatively smaller the algorithms are pretty fast. When the log size exceeds 100 the time taken by the algorithms is more.

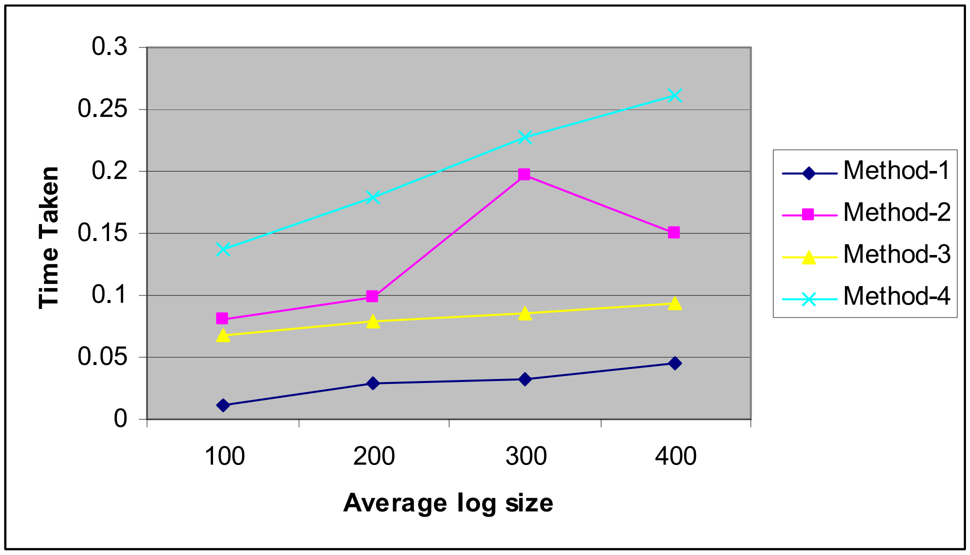


Figure 7.3 Time taken by each algorithm based on varying Log size

Figure 7.4 shows the variation of the time taken by each of the algorithms with respect to the depth of the network used within the network. As mentioned in chapter 5 the algorithms define a value called ‘k’ which basically says till what depth in the network, user logs are searched. The graph does a study on the variation of the time taken by each of the algorithms when the ‘k’ value is varied. As shown in the graph the time taken increases as ‘k’ value increases because the numbers of user logs which need to be scanned grow almost exponentially.

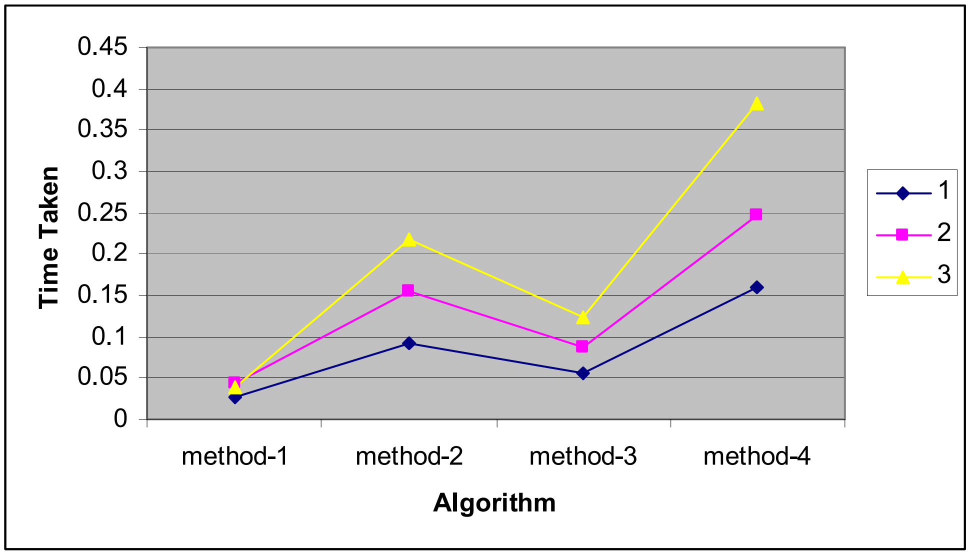


Figure 7.4 Time taken by each algorithm based on varying the level ‘k’

7.4 Interesting observations

The following set of experiments shows certain properties & interesting features within the network. Certain observations made during the evolution of the network & the query logs are shown below.

Figure 7.5 shows an experiment showing what percentage of the queries in the logs are same for users within their same network. For a particular user we make an observation as to how many queries & tags are similar based on the spider software. This gives as estimate as to the search pattern of users who are connected directly in a social network.

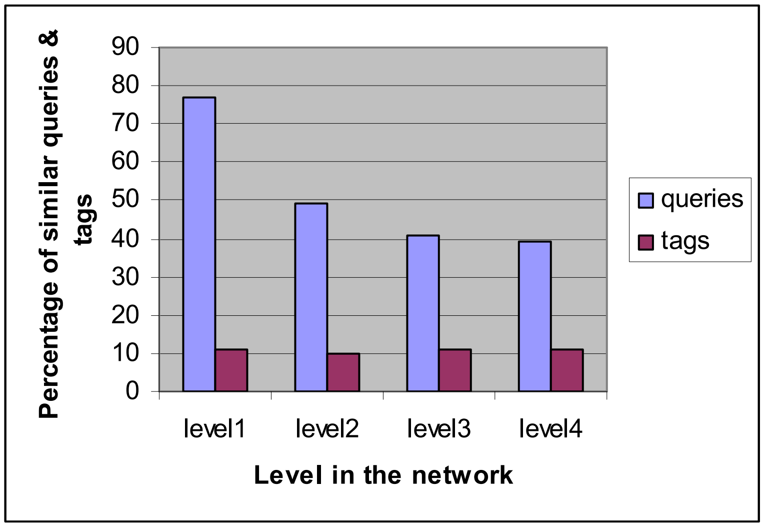


Figure 7.5 Percentage of similar queries/tags based on varying level 'k'

Figure 7.6 shows an estimate as to how the network is distributed. It compares the number of users at each level. As mentioned before the length of the longest path in the graph is 3, so estimation up to level 3 is shown.

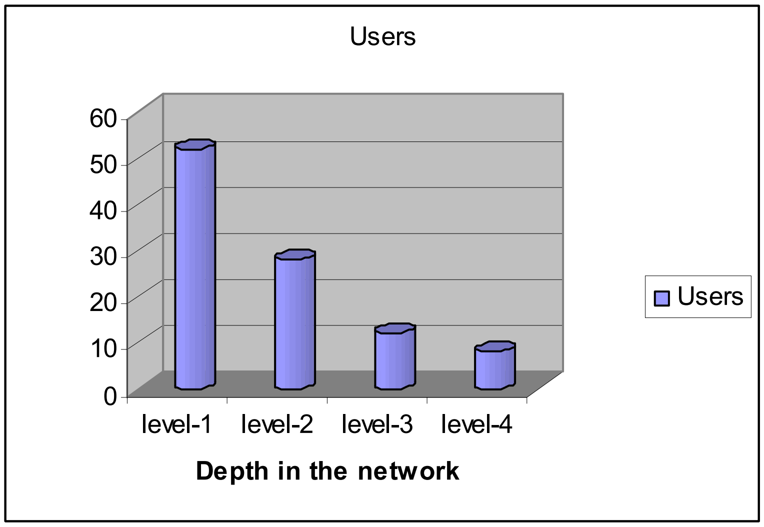


Figure 7.6 Number of users based on varying level 'k'

The experimental results suggest that the overhead approach is relatively small. The actual Re-ranking process takes very little time as shown by the graphs. Moreover the results vary a lot as users are added onto the network. The process is very subjective in that a new user might contribute a lot to the system. Our experiments show that the probability of good contribution made by such kind of a user is very likely. Still this is a very subjective kind of study where a lot of things depend on the user involved. But based on the above experimental user study the behavior of the system can be predicted to certain levels.

CHAPTER 8

CONCLUSION

In this thesis we have presented an approach for personalizing search results using information present on a social network. Our framework can be applied on any kind of social network. We have discussed and analytically demonstrated several properties of our framework regarding the behavior of the system. Through a detailed experimental study we have demonstrated the practical utility of our approach.

REFERENCES

[1] F. Tanudjaja, and L. Mui. *Persona*: A contextualized & personalized web search. Proceedings of the Hawaii International Conference on System Sciences, Copyright IEEE 2002.

[2] F. Liu, C. Yu, and W. Meng. Personalized Web Search by Mapping User Queries to Categories. CIKM, Nov. 2002.

[3] E. Agichtein, E. Brill, and S. Dumais. Improved Web Search Ranking by Incorporating User Behavior Information. SIGIR, Aug. 2006.

[4] L. A. Adamic, O. Buyukkokten, and E. Adar. A Social Network Caught in the Web. Proceedings of ICDE, Apr. 2002.

[5] N. Koudas, A. Marathe, and D. Srivastava. Flexible String Matching against Large Databases. VLDB, 2004.

[6] M. Pazzani, and D. Billsus. Learning & Revising User Profiles: The Identification of Interesting Websites . Proceedings of ICDE, Apr. 2002.

[7] A. Pretschner, and S. Gauch, *Ontology Based Personalized Search*, Proceedings of the Eleventh IEEE International Conference on Tools with Artificial Intelligence, 1999

[8] T. Kurki, S. Jokela, R. Sulonen, and M. Turpeinen, *Agents in Delivering Personalized Content Based on Semantic Metadata*, Proceedings of AAAI Spring SymposiumWorkshop on Intelligent Agents in Cyberspace, 1999.

[9] A. Pretschner, and S. Gauch, *Personalization on the Web*, University of Kansas, ITTC-FY2000-TR-13591-01, 199 [10] M. Mcnee, I. Albert, D. Cosley, P. Gopalkrishna, K. Lam, M. Rashid, A. Konstan, and J. Riedl, On the Recommending of Citations for Research Papers, CSCW, 2002.

[10] B. Yu, and P. Singh, Searching Social Networks, AAMAS, 2003.

BIOGRAPHICAL INFORMATION

Sushruth Puttaswamy was born in Karnataka, India in 1983. He received his BE in Computer Science & Engineering from SJCE, Visweswaraiaha Technological University in 2005. His constant interest on search engines & their working influenced him to pursue study on Information Retrieval while doing masters in UT Arlington.