

國立交通大學

資訊工程學系

碩士論文

個人化網頁排序演算法之研究

PPR : Personalized PageRank for Web Search



研究生：林裕欽

指導教授：彭文志教授

中華民國九十四年八月

個人化網頁排序演算法之研究
PPR : Personalized PageRank for Web Search

研究生：林裕欽

Student : Yu-Chin Lin

指導教授：彭文志

Advisor : Wen-Chih Peng

國立交通大學
資訊工程學系
碩士論文

A Thesis

Submitted to Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science and Information Engineering

August 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年八月

摘要

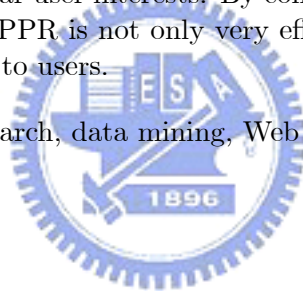
在數量龐大的網頁中搜尋高品質並且與使用者興趣有相關的網頁已經成為一個重要的研究領域。現今有許多以關鍵字為主的搜尋引擎，但是這些搜尋引擎通常都會回傳大量的搜尋的結果，使用者仍然要花許多的時間去找出他們真正需要的網頁。世界上最強大的搜尋引擎 Google 使用 PageRank 演算法用以搜尋結果做排序，以符合需求。雖然可以盡量的找到了許多的搜尋結果，但是 Google 並不能提供個人化的網頁排序。因此，將搜尋結果以符合使用者的興趣做排序已經成為了一個重要的議題。在這篇論文當中，我們首先將實做一個用戶端的模組，此模組會記錄使用者瀏覽網頁的行為，並且使用資料探勘的方法找出頻繁瀏覽網頁的行為。我們將從使用者的經常瀏覽行為中找出使用者的興趣。在這篇論文當中，依據使用者的興趣，我們提出了一個個人化網頁排序演算法，簡稱 PPR。這個演算法將分成四個步驟，第一個步驟會依照使用者的興趣給予搜尋結果的網頁不同的初始值。第二個步驟是依照使用者的興趣在搜尋的結果中增加虛擬的網頁與虛擬的鏈結。第三個步驟則是依照使用者先前的點選行為來加以調整。為了增加個人排序的精準度，我們在第四步驟加上了群組推薦的技術。從實驗結果中，證明演算法 PPR 在提供個人化的排序上不但是相當的有效率，並且也可適時地動態調整網頁順序以符合個人化之需求。

關鍵字：個人化搜尋，資料探勘，網路探勘

Abstract

With a huge amount of Web pages, searching web pages for high quality and relevance to user interests has become an important research field. Nowadays, there are many keyword-based search engines available for this purpose. However, the amount of search results found by these search engines usually is very large and users still spend a lot of time to find what they really want after searching. Note that the most powerful search engine Google explores algorithm PageRank in ranking the search results. Through finding as many pages as possible, Google cannot provide a personalized Web ranking. Thus, ranking search results that satisfy users interests has become a growing importance topic, which is the very problem we shall address. In this paper, we first implement a client-side module to capture user browsing behavior and then exploit the technique of data mining to mine frequent browsing access patterns from user browsing behavior. In light of frequent browsing access patterns, we propose a method to extract user interests as user preferences. In this paper, we propose a new algorithm with the idea of adjusting the ranking scores of Web pages. The adjustments are in accordance with user preferences mined from user browsing behavior. Specifically, the algorithm, referred to as algorithm PPR(standing for Personalized PageRank), is divided into four phases. The first phase assigns the initial weights based on user interests. In the second phase, the virtual links and hubs are created according to user interests. By observing user click streams, our proposed algorithm will incrementally reflect users favors for the personalized ranking in the third phase. To improve the accuracy of ranking, collaborative filter is taken into considerations when the query with similar keywords are submitted by users having similar user interests. By conducting simulation experiments, we have shown that algorithm PPR is not only very effective but also very adaptive in providing personalized ranking to users.

Keywords — Personalized search, data mining, Web mining



誌 謝

寫完這份論文花了我許久的時間，有太多的人必須要感謝的，他們有的給了我啟示，有的給我鼓勵。有他們的幫忙，今天才有這份論文的誕生。

首先最重要的是我的指導教授彭文志老師，我在他的身上不但學到了許多的知識，更學到了做學問的態度。感謝他在我寫論文的期間給予了我許多寶貴的意見與指導，在這兩年的研究當中，讓我對資料探勘這門學問有了一些心得。還有要謝謝跟我一起工作的伙伴們，謝謝實驗室的同學們，在我有問題的時候都會提出他們的看法，讓我在研究的路上不至於跌跌撞撞。感謝交大資工系提供了我這麼好的一個環境，引領著我由淺入深的走入了資訊工程這領域。最後還要感謝一路上支持著我的家人們與朋友，有他們的鼓勵與支持，我才得以寫完這份論文。還要感謝我的女朋友洪佳君，在我心情低落的時候，能夠不斷的鼓勵我，讓我又有了繼續研究的動力，有了她陪伴在我身邊，才能讓我可以完成我的研究。謝謝所有關心我的親朋好友們，謹將這篇論文，獻給各位。



Contents

1	Introduction	1
2	Preliminaries	3
2.1	Algorithm PageRank	4
2.2	Algorithm VIPAS	4
3	Algorithms for Personalized Web Search	5
3.1	The Whole Framework of a Personalized Web Search	5
3.2	Procedure of Extracting User Interests	6
3.3	Design of Algorithm PPR	8
3.3.1	The overview of Algorithm PPR	10
3.3.2	Assigning Weights to Web Pages	11
3.3.3	Adding Virtual hubs and virtual links	13
3.3.4	Exploring Feedbacks from User Clickstreams	15
3.3.5	Exploiting Feedbacks from Collaborative Filtering	17
4	Performance Study	22
4.1	Simulation model	22
4.2	Simulation results	25
5	Conclusions	38



List of Figures

1	The simple example for PageRank	4
2	The framework of the Personalized Web Search	6
3	The flow chart of extracting user preferences.	8
4	A user's frequent access pattern and the keywords in each page	8
5	Another example access pattern.	9
6	The search result for a query q.	12
7	The relation between user preferences and search results.	13
8	(a) The original search result graph. (b) The web graph after adding virtual links and virtual hubs.	14
9	The ontology tree of the test data set.	24
10	The results of user 1 queries "coding"	27
11	The results of user 1 queries "facilities"	28
12	The results of user 24 queries "hotlist"	28
13	The results of user 32 queries "informatics".	29
14	The result of user 42 queries "administrative"	29
15	The comparison of discrepancy coefficient.	30
16	The discrepancy coefficient of different user query the same keyword.	32
17	The importance of each category in each algorithm.	33
18	The discrepancy coefficient in different weight.	33
19	The comparison of the number of virtual hubs added.	35
20	The discrepancy in different number of VHs added.	35
21	The importance in different length of user query history.	36
22	The discrepancy coefficient in different length of user query history.	36
23	The difference while user changes his interests.	37



List of Tables

1	The results of getting keyword with weight higher than 0.4.	9
2	The original PageRank score of the search results.	15
3	The user's preference table.	15
4	The rank with virtual hubs and virtual links addition.	16
5	The group with 20 members and query history of 10 users with the same category query	19
6	The final personalized ranking against the original PageRank	21
7	The ranking when the user's long term interests and short term interests are not the same.	22
8	The table of page distribution	23
9	The table of keyword distribution	23
10	The keywords for test.	24
11	The sample of user preference table.	25
12	The member of each cluster.	25
13	The query results by different user query the same keyword.	31
14	The importance of each category for querying the same keyword.	31
15	The weight in each algorithm.	32
16	The factor fp in different length of user query history.	35



1 Introduction

There are many keyword-based search engines available for Web search and these search engines often return a long list of search results, many of which are not what the user wants. Thus, users may spend lots of time on running through all links of the list to find the truly relevant information. One way to reduce the time spending on browsing search results is to provide personalized Web search, in which those web pages relevant to user interests will be ranked in the front of the search result list, thus leading to a quick process for the users to just access those links ranked in the front of the search list. While the amount of Web pages is growing at rapid speed, the issue of devising a personalized Web search is of increasing importance.

Note that in order to provide personalized ranking for search results, one should first extract user interests. User interests could be in a way that users fill in his/her user preferences. However, the drawback of this approach is that an ordinary user is not always able to fill the user preferences all the time. Moreover, the user interests may change as time goes on. In a reality, users will not always compose precise his/her interests in his/her preferences. Consequently, in this paper, we explore data mining techniques to automatically extract user interests. The purpose of data mining is to discovery knowledge from huge amount of data by various mining algorithms such as frequent pattern mining, clustering and classification. Since the Web is viewed as a huge databases, applying data mining to the Web is referred to as Web mining. Generally speaking, Web mining is divided into three categories: Web content mining, Web structure mining and Web usage mining. Web mining has attracted a lot of research works due to the characteristics of Web and the importance of Web. The focus of this paper is to combine Web usage mining and Web structure mining for developing personalized ranking of search results. To be more specifically, we first explore the technique of Web usage mining to extract user interests and then in accordance of user interests, we

devise a personalized ranking algorithm, which is an extension of algorithm PageRank.

Due to the importance of personalized Web searching, alternatives to keyword based searching have arisen recently. Research efforts, such as HITS [4], PageRank [2], VIPAS [7] and PPV [6], explore the link analysis to determine the authority of Web pages. In HITS [4], there are two scores devised (i.e., the authority score and the hub score). The authority score of a Web page is the measurement to indicate the quality of the Web page relevance to the keyword queried. PageRank [2] also explores the link structure among web pages. By utilizing the feature of random walk, algorithm PageRank is able to appropriately rank the Web pages. The topic-sensitive PageRank builds a topic-oriented PageRank, in which a set of PageRank vectors based on 16 main topics of Open Directory Project (ODP) are computed [5]. When a query is submitted to the server, the search engine computes the similarity between each topic and the query submitted, the search results are ranked by the similarity scores calculated. The authors in [2] exploited personal PageRank vector (denoted as PPV) to speed up the calculation of PageRank, where PPV is a personalized view of importance of pages on the Web [6]. However, the main theme of the paper in [6] is the scalability when computing the ranking process given PPV. Note that users must explicitly specify user preferences so as to set up PPV. Although many research works such as HITS [4], PageRank [2], VIPAS [7] and WebQuery [3] have explored the link analysis to determine the authority of Web pages, none of these research works take the personalization into consideration when ranking the Web pages in the search list. Moreover, the prior work in [6] is not adaptive in that once the PPV is set, the ranking scores are fixed unless the PPV has changed and scores are re-computed. Note that user interests are divided into long term interests and short term interests. Furthermore, user interests will be changed with time. How to accurately extract user interests is a challenging problem. Once obtaining the user interests, the next problem is that how to devise an algorithm to take the personalized view into ranking algorithm able to dynamically and adaptive rank Web pages in accordance with the changes of user interests.

After all, the problem of personalized Web search is still an important research field worthy of studying.

In order to remedy the difficulties stated above, we first implement a client-side module to capture user browsing behavior and then exploit the technique of data mining to mine frequent access patterns from user browsing behavior. Once frequent access patterns are mined, we propose a solution procedure to extract user interests. In this paper, we propose a new algorithm with the idea of adjusting the ranking scores of Web pages. The adjustments are in accordance with user interests mined from user browsing behavior. Specifically, the algorithm, referred to as algorithm PPR (standing for Personalized PageRank), is divided into four phases. The first phase assigns the initial weights based on user interests. In the second phase, the virtual links and hubs are created according to user interests. By observing user click streams, our proposed algorithm will incrementally reflect users' favors for the personalized ranking in the third phase. To improve the accuracy of ranking, collaborative filter is taken into consideration when the query with similar keywords are submitted by users having similar user interests. By conducting simulation experiments, we have shown that algorithm PPR is not only very effective but also very adaptive in providing personalized ranking to users.

This paper is organized as follows. Preliminaries are given in Section 2. Section 3 will describe the detailed steps of algorithm PPR. Experimental results are presented and analyzed in Section 4. This paper concludes with Section 5.

2 Preliminaries

Ranking the Web pages to satisfy the user interest is very essential for the design of personalized web search. Two famous ranking algorithms are presented in this section

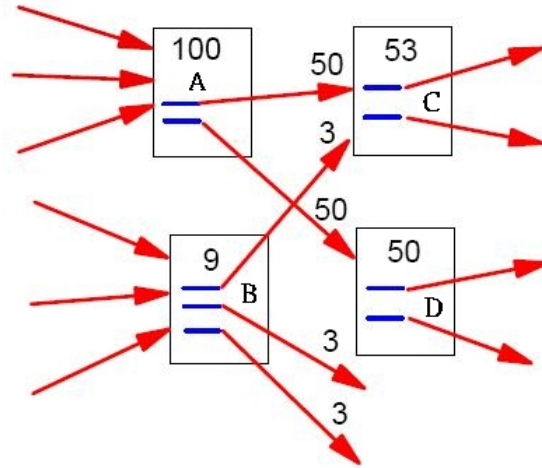


Figure 1: The simple example for PageRank

2.1 Algorithm PageRank

Algorithm PageRank is based on the link structure of Web pages. The score of a page is calculated by the pages links to it. Figure.1 shows a simple example, where the scores of page C and page D are given by page A and page B. Also, the algorithm of PageRank also contains the "random walk" feature. The feature of "random walk" refers to the browsing behavior of users in which users may browse other pages randomly without following the links provided by the current page. The ranking formula of PageRank can be simply formulated as follow:

$$PR(v) = (1 - c) \sum_{u \text{ links to } v} \frac{PR(u)}{O(u)} + c,$$

where c is the random suffer constant and $O(u)$ is the number of outlinks of page u .

After some iteration of calculation, the score of each page will converge and the rank of each page can be derived by sorting the scores of all pages.

2.2 Algorithm VIPAS

The goal of algorithm VIPAS is to take user browsing behavior into consideration when ranking the Web pages. VIPAS is the extension of algorithm HITS. By observing the user behavior in

browsing pages, virtual links and virtual hubs are then created according to the user browsing behavior. The detailed of algorithm VIPAS is shown as follows:

Algorithm VIPAS: (Virtual Linked Powered Authority Search)

/* Initialization Phase: */

1. For a query term, perform the regular HITS analysis
2. Collect a base set of pages with computed authority and hub scores and store them in the data base.

/* Vitrual Link Collection Phase: */

3. Monitor the user behavior to see whether a URL in the list is clicked by the user or not.
4. After a period of user behavior observation, put URLs that are often accessed into the "hot set"
5. Create vitrual links for pages in the hot set.

/* Refinement Phase: */

6. For each page in the hot set, compute ot's new authority and hub scores.
7. Run several iterations of score update for pages in the base set.

3 Algorithms for Personalized Web Search

The goal of our work is to build a personalized Web search. Specifically, the problems we face can be divided into the following. First of all, we must be able to capture users' interests from their browsing behaviors. In light of user interests, a new ranking algorithm is required. Explicitly, in section 3.1, the system framework is described. The user interesting extraction module is developed in section 3.2. The basic idea of our ranking algorithm is presented in section 3.3.

3.1 The Whole Framework of a Personalized Web Search

As mentioned before, the framework of a personalized Web search is divided into two phases: data collection phase and data ranking phase. In the first phase, we intend to capture users' interests in the two ways: bookmarks and users's browsing behavior. For privacy issue, we implement client side module to log user browsing history. The browsing behavior of users reflects the interests of users, which are extracted by our client side collection module. Specifically, we first explore data mining techniques to mine frequent user browsing patterns

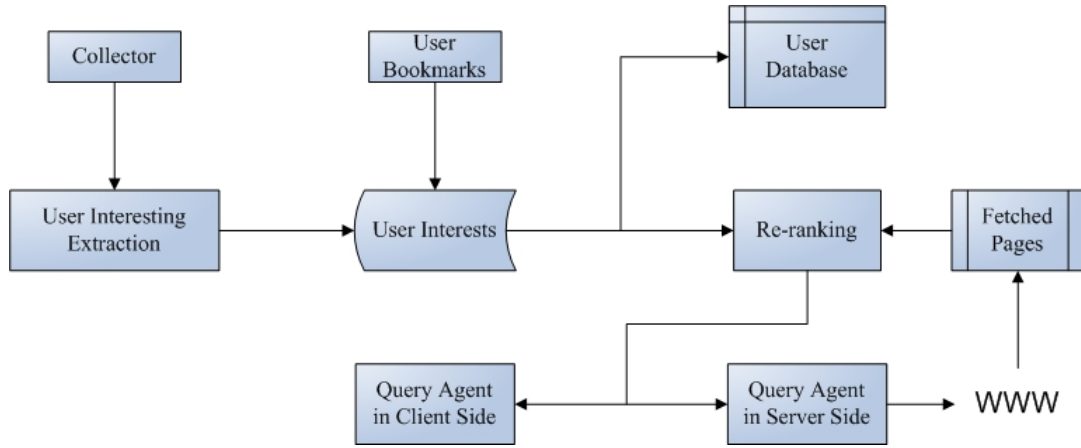


Figure 2: The framework of the Personalized Web Search

from browsing logs. Furthermore, we can utilize information retrieval methods to find out the keywords representing user interests. These keywords extracted by our module represents user preferences. When a user submits a query , our proposed ranking algorithm would list those pages that are likely to satisfy user interests in the front of search list.

In this framework, the collector module will log each web site the user has visited, and store as an URL in database. The user interesting extraction module will extract users' interests and store in the user interests module. The user database module stores all users' interests. According to user's preferences, we can provide a recommendation system in which the system will recommend pages satisfying user's interests. Query agent module is an agent embedded in the browser, help users to key in queries.

3.2 Procedure of Extracting User Interests

From the log of user browsing behavior, we intend to extract user interests. The procedure of extracting user interests is shown in Figure 3. Since users usually daily browse Web, the log of browsing behavior may contains a significant amount of Web browsing data. Note that user interests represent frequent access behavior. Thus, frequent access patterns are mined from Web browsing log. In order to mine frequent access patterns, WAP-mine algorithm[8] is

implemented. Intuitively, user interests refer to the keywords frequently appearing in access patterns mined. As such, we have to find out the keywords in each page of access patterns. Traditional TF/IDF method is used to extract important keywords for each page of access patterns. Since the number of keywords in each page is usually large, each keyword in a Web page is assigned one weight, which is defined as term frequency/ number of Web pages indexed by Google. For each access pattern, the represented keywords of one access patterns is the union of keywords with their corresponding weights larger than one threshold.

Once each access pattern has the represented keywords, user's interests are able to generated by exploiting TF/IDF method in that each access pattern is viewed as one document with its own keywords (i.e., represented keywords of each access pattern). We can classify the access patterns by the pages the access patterns have. If most the pages in the access pattern belong to category 1 and the access pattern is viewed as the browsing of category 1. And then we collect the access patterns with the same browsing category and find out the user's interests category by category. The basic idea of classify the access patterns is that user may browsing page in different order in different access patterns but their concept may be concrete. That is, although the keywords user interested in may appear in different order in different access patterns, but that's what user interested in. The represented keyword set of each access patterns can be the set of the total keywords in each pages in the access pattern. The represented keyword set of each access pattern can be viewed as a document and we use regular TF/IDF to retrieve the keywords user interested in. Figure 4 and Figure 5 shows an illustrative example of the access patterns viewing the same category for the user. Assume that the user has the only two patterns in the category, according to TF/IDF we can obtain the user interest terms in the category in table ???. In order to simply the calculation in our algorithm, we store the distribution of the user interest in to a personal preference vector. The preference vector is an m dimension row vector, and m is the number of global categories. For example, if the number of global categories is 4 and the user has 60% of interest terms in

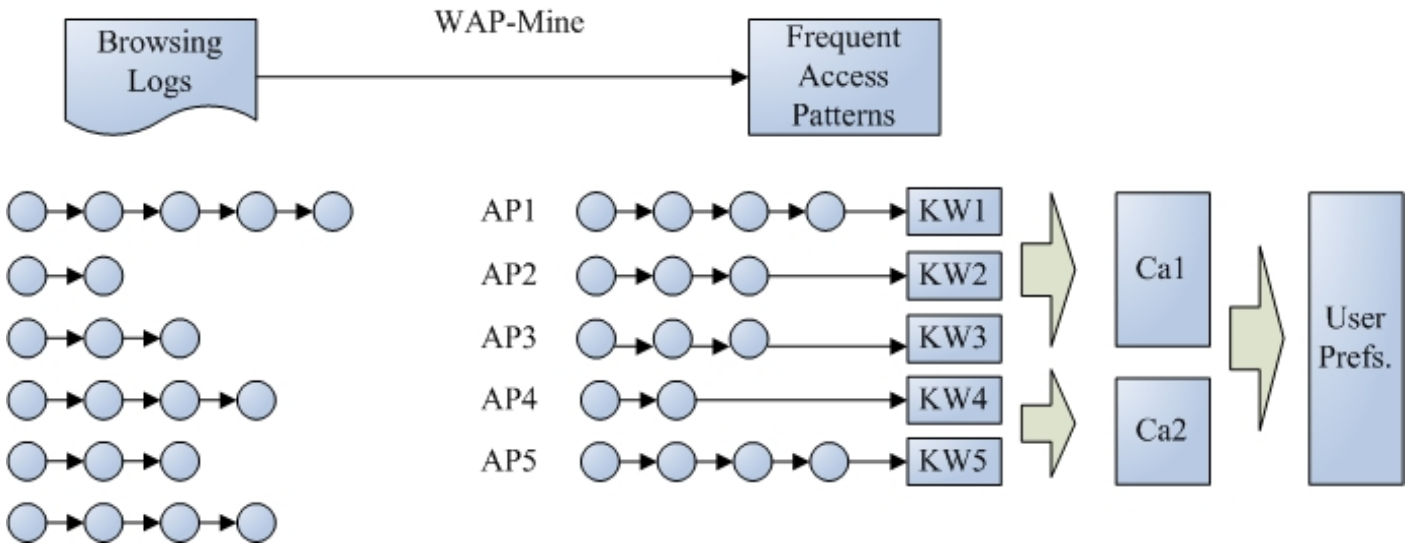


Figure 3: The flow chart of extracting user preferences.

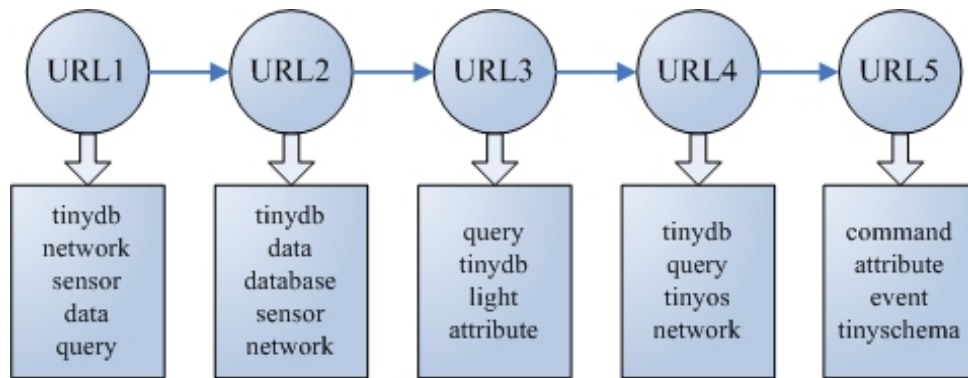


Figure 4: A user's frequent access pattern and the keywords in each page

category 1, 15% in category 2, 10% in category 3, 15% in category 4, the personal preference vector will be $\{0.6, 0.15, 0.1, 0.15\}$.

3.3 Design of Algorithm PPR

In Section 3.3.1, the overview of algorithm PPR is presented. The detailed steps of algorithm PPR are then described in the following sections.

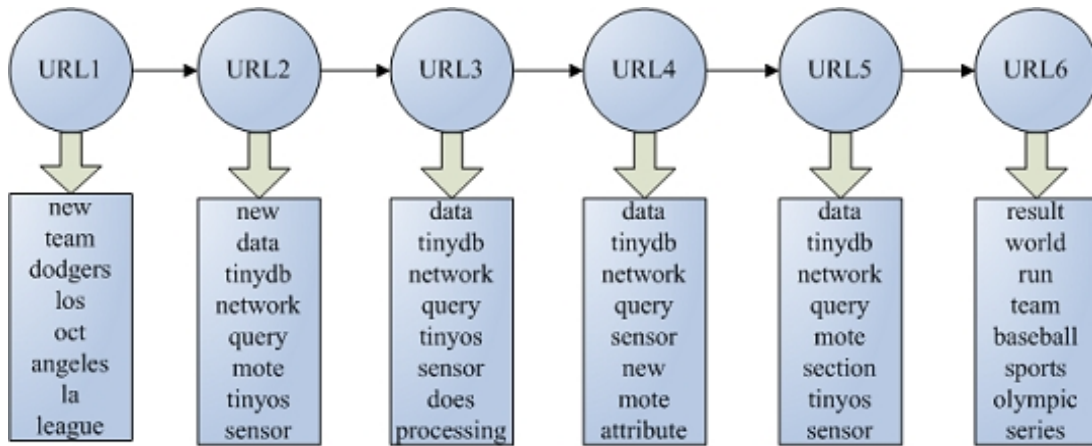


Figure 5: Another example access pattern.



id	keyword
1	data
2	tinydb
3	network
4	query
5	mote
6	tinyos
7	sensor
8	attributes
9	section
10	simple
11	result

Table 1: The results of getting keyword with weight higher than 0.4.

3.3.1 The overview of Algorithm PPR

Given user interests obtained from the above procedure, in algorithm PPR, we take the personalized view when ranking the search list. The algorithm devised is call PPR (Personalized PageRank), whose procedure is divided into four phases. The first phase, called the initial phase, performs the regular HITS analysis. Once we obtain the set of Web pages, these Web pages are clustered into m categories, where m is the number of global categories provided. Then, we assign larger weights to those web pages whose categories are similar to user interests. The second phase is the virtual hubs and virtual links addition phase, which further assigns the weights to those web pages whose categories satisfy the user interests and the query term. Furthermore, user interests may vary with time. In order to quickly capture the change of user interests, algorithm PPR monitors the user behavior to observe whether the user interests are changing or not. The final phase is user group history adjustment phase in which collaborative filtering is used to generate a new rank recommended by others having similar interests. The algorithmic form is shown below.

Algorithm PPR (Personalized PageRank):

Input: User interests categories U_c , a user query q , the query result R

Output: A personalized ranking for the result R

begin

 /* **Re-weighting phase:** */

1. $R' \leftarrow$ Apply Kleinberg Extension on R

2. Cluster R' into m categories named G_c /* m is the number of global categories. */

3. Compute the $sim(U_c, G_c)$ and re-assign the initial weight to each category.

 /* **Virtual hubs and virtual links addition phase:** */

4. Compute the $sim(q, U_c)$ and add virtual hubs and virtual links for the top k clusters.

5. Rebuild the result graph.

 /* **User query history adjustment phase:** */

6. Choose the least n similar queries in user query history.

7. Compute the individual adjustment factor F_p according to user's query history

 /* **User group history adjustment phase:** */

8. Find out the group the user belongs to and compute the group adjustment factor F_g .

9. Assign the weight for the global weight w_{global} , personal weight w_p , group weight w_{group} .

10. **repeat**

 for all v in R'

$$PPR(v) = w_{global} * ((1 - c) \sum_{u \text{ links to } v} \frac{PPR(u)}{O(u)} + c) + w_p F_p * p(v) + w_{group} F_g * p(v)$$

11. **until** all scores converge

12. sort the results by the personal ranking in descending order

13. **return** the result to the user

This algorithm can be divided into four phases: (1) Re-weighting for each page in search result, (2) Virtual hubs and virtual links addition phase, (3) User query history adjustment phase, and (4) User group history adjustment phase. The detailed descriptions for each phase are in the following subsection.

3.3.2 Assigning Weights to Web Pages

After the data collection phase, we obtain user interests stored in our database. The user interests can be viewed as the interesting degree for an user to the categories. While a query sent to the server, the query agent would be sent the query to several search engines and obtain the search results. The search results stored in the "Fetched Pages" module and the search result R can be clustered into categories to find out the distribution of R . Figure 6 shows some selected search results, where the light nodes are in the search result, and the dark ones are the pages extended by the "Kleinberg Extension" introduced in HITS[4]. The "Kleinberg Extension" is to expand the graph in two criteria: (1) add the nodes which link to the nodes in the graph, and (2) add the nodes linked by the nodes in the graph.

Algorithms PageRank and HITS give each page the same initial value (i.e., 1), which does not address the personalized preferences. Therefore, in order to reflect user interests in search results, the initial value of each page should be modified. By the user interest vector U_c , the search result cluster G_c and the query q , we perform a weight function later to find out the relationship between U_c and G_c . While the search engine receives a query q , we shall find out which cluster may satisfy the user interests. There are some issues in finding out the cluster. For example, if a query q has terms A and B, there may be two or more categories having these terms in user preferences. We should determine which category is more similar to user's interests. The search engine can easily know which categories the query is in and then we compare these categories in user preference and search results to find out the similarity

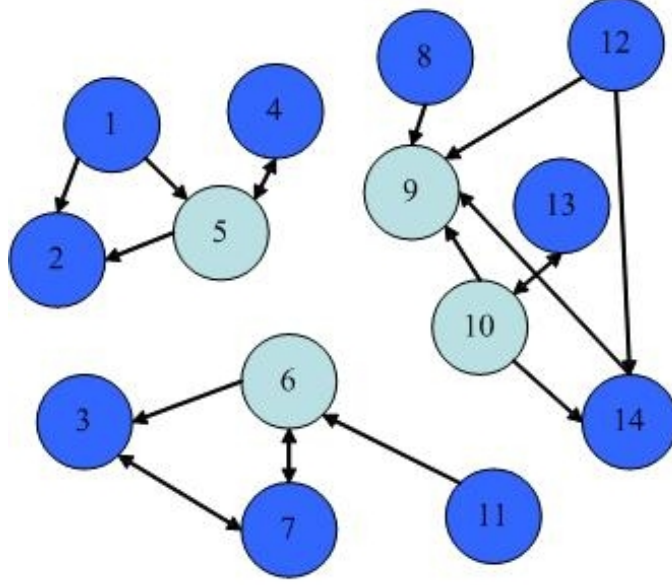


Figure 6: The search result for a query q .

between user preference and search results. The weight of each category in result clusters G_c is the summation of the similarity to each category in user interests U_c . The weight of category i in result clusters G_c can be defined as follow:

$$w(G_{c,i}) = \sum_{\forall \text{ category } j \text{ in user cluster } U_c} sim(U_{c,j}, G_{c,i})$$

where $U_{c,j}$ is denoted as category j in user interests vector U_c , and $sim(U_{c,j}, G_{c,i})$ is formulated as

$$sim(U_{c,j}, G_{c,i}) = \frac{|U_{c,j} \cap G_{c,i}|}{|U_{c,j} \cup G_{c,i}|}$$

Both $U_{c,j}$ and $G_{c,i}$ will contain some keywords when clustering user access patterns or resulting Web pages. Thus, similar to the work in [9], the similarity between $U_{c,j}$ and $G_{c,i}$ is defined as the ratio of the number of the intersection set of keywords in $U_{c,j}$ and $G_{c,i}$, and the number of the union set of keywords in $U_{c,j}$ and $G_{c,i}$.

After the calculation in each category in G_c , the initial weight of the page is the weight

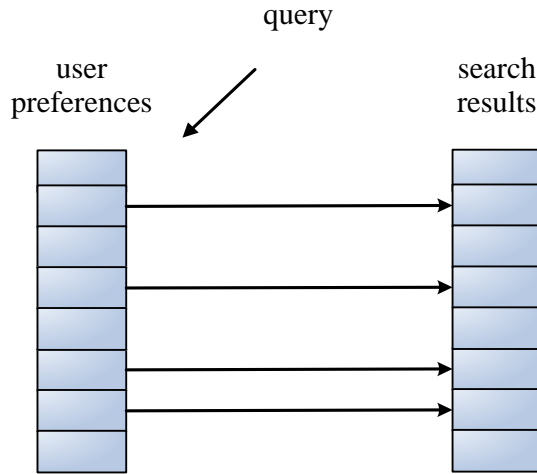


Figure 7: The relation between user preferences and search results.

which category it belongs to. Figure 7 shows the concept of the assigning weight phase. The query fits in some category in user preference, and then we calculate the similarity between the user preference and search results.

3.3.3 Adding Virtual hubs and virtual links

Note that the scores of Web pages calculated by PageRank is mainly based on the link structure of Web pages. The links between Web pages are fixed after being created. Though assigning different weights to Web pages, we will further to adjust the scores so as to emphasize the importances of those Web pages similar to user interests. Since the link analysis is used in PageRank, we come up with the idea of putting virtual links and virtual hubs goes to the pages that are very similar to user interests.

When a user submits a query term q , we will compute the similarity between q and each category i in user interests vector U_c . Only top k categories similar to query q are selected, where the number of k is a controlled parameter to show the personalized degree. With the smaller values of k , only very similar categories are chosen. Once selecting the top k categories, we check whether the search result categories are in the top k categories of user interests. If the search result categories (e.g., $G_{c,i}$) is exactly matched with the top k categories (e.g., $U_{c,j}$),

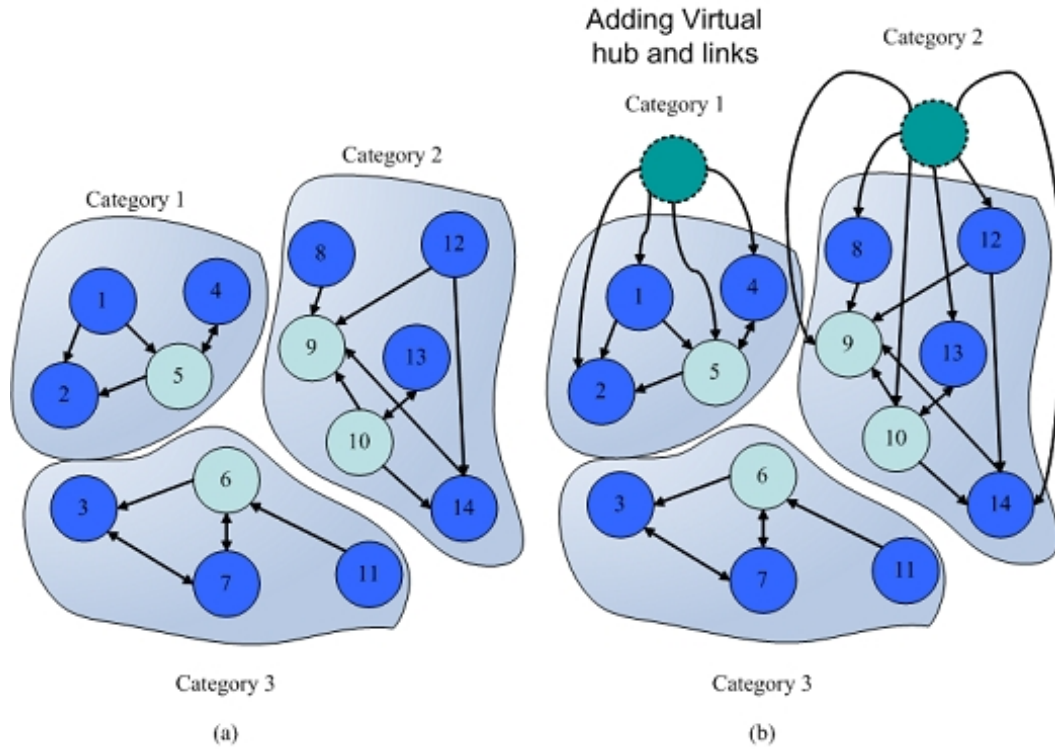


Figure 8: (a) The original search result graph. (b) The web graph after adding virtual links and virtual hubs.

the virtual hubs and links are added to the Web pages whose categories belong to $G_{c,i}$ with the weights of virtual links being $sim(U_{c,j}, G_{c,i})$. The weights of virtual links are set to the values of categories in user interest vector.

Assume that different pages in different categories. Then, when a user's query was sent to the server we have the result graph shown in Figure 8 (a). By algorithm PageRank, page 3, 7, 6 in the bottom have higher rank values. The detailed PageRank scores are shown in Table 2. The score of page v is $PR(v) = c + (1 - c) \sum_{u \text{ links to } v} \frac{PR(u)}{O(u)}$, where $O(u)$ is the number of outlinks in page u and c is the random walk constant.

Assume that the user's preference is shown in Table 3. We choose the categories which have score larger than 0.4 to add the virtual hubs and virtual links. So, we add 2 virtual hubs and add virtual links to the pages in category 1 and category 2. The modified graph is in Figure. 8 (b). If the weight of the virtual hub is 0.5, after the PageRank algorithm, the detailed rank is shown in Table 4.

Page	PRscore
7	1.627362
3	1.253509
6	0.969129
9	0.714537
5	0.534247
2	0.440805
4	0.377055
10	0.365532
14	0.317318
13	0.253568
1	0.15
8	0.15
11	0.15
12	0.15

Table 2: The original PageRank score of the search results.

Category	preference
1	0.6
2	0.4
3	0

Table 3: The user's preference table.

In table 2, page 3, 6, 7 in category 3 have higher ranks in the PageRank algorithm. But user is interested in category 1 and category 2. After the addition of virtual hubs and virtual links the pages in category 1 and category 2 have higher ranks shown in Table 4. The example shows us that the addition of virtual hubs and virtual links is useful in personalized ranking.

3.3.4 Exploring Feedbacks from User Clickstreams

Besides the re-weighting and the virtual hubs and virtual links addition, the feedback by the user is an important factor and the feedback should be considered. The first two steps of our algorithm can be viewed as user long term interests adjustment. But if the user change his interest, the first two steps can't make the real-time adjustment, so we need a real-time adjustment mechanism to reflect user short term interests. Based on the mechanism, the re-ranking algorithm will not always make the same prediction with the feedback of the user.

Page	PRscore
9	3.096329
5	2.315068
2	1.910154
4	1.633904
7	1.627362
10	1.583974
14	1.375043
3	1.253509
13	1.098793
6	0.969129
1	0.65
8	0.65
12	0.65
11	0.15

Table 4: The rank with virtual hubs and virtual links addition.

User clicking streams will be logged to observe the browsing behavior of users. From the user clicking streams, we could clearly justify whether the search results fulfilled the requirements of users or not. For example: assume that the user's interest is belong to C_1 . When this user submits a query q , which falls into four categories. We will log the category which the user click the most to represent the query.

While query q is sent to the server, we'll try to find the query history and find out which category user really choose when facing the queries in the same category. The query history can be transformed by a clickstream history. We emphasis on the correctness of our prediction. If the query history for category which query q falls in is $\{C_2, C_1, C_2, C_1, C_1\}$ and the system predicts the user is interested in C_1 , how to assign the history adjustment to each category? The appear times of a category and the order of a category should be considered in the adjustment. The importance is relative to the order The newest log is the most important and the oldest log has the less importance. Thus, the adjustment function can be designed as follow:

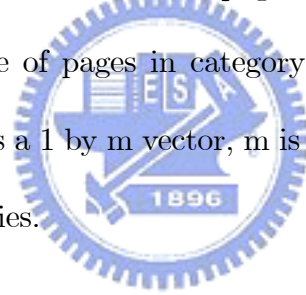
$$F_p(i) = \frac{n_i}{n} * \frac{\sum_{j \in \text{category } i} \text{weight}(j)}{\sum_{j=1}^n \text{weight}(j)}$$

where n is the number of elements in a clickstream,

n_i is the number of appearances of category i ,

and $\text{weight}(j)$ is the order of category j appeared

For example, the query history is $\{C_2, C_1, C_2, C_1, C_1\}$ and our prediction category for this query is C_1 . The order is sorted by the query time in ascending, the first record in the query history is the oldest one. The importance of each category in query history will be $\{1, 2, 3, 4, 5\}$. So the personal adjustment score of pages in category C_1 is $\frac{3}{5} * \frac{(2+4+5)}{15} = 0.44$ and the personal adjustment score of pages in category C_2 is $\frac{2}{5} * \frac{(1+3)}{15} = 0.10667$. These scores are stored in vector F_p . F_p is a 1 by m vector, m is the number of global categories and contains the scores of each categories.



3.3.5 Exploiting Feedbacks from Collaborative Filtering

Besides the user's feedback can be helpful to the adjustment of the personalized ranking, the group has the same interests with the user can provide a recommendation to the user when a query comes. In order to achieve user group collaborative adjustment, grouping user is the first step. There is an important issue in the clustering: "If a user query q is in category C_i , but there are 10 users has another interests with category C_n and 20 users has another interests with category C_m , which group can help the user the most?". Because each user has his own preference, and the preference can be viewed as a point in a m -dimensional spaces where m is the number of global categories. In the m -dimensional spaces, each axis represents a different category. The density based clustering can be used in clustering users. The distance function

used in density based clustering can be the Euclidean distance. The Euclidean distance between two user u_1 and u_2 is $\sqrt{(u_{11} - u_{21})^2 + (u_{12} - u_{22})^2 + \dots + (u_{1m} - u_{2m})^2}$ where u_{ij} is the preference of user i in category j . If the distance is less than a threshold σ , the two users can be in the same cluster. This will result in several clusters and each cluster has similar interests with each other users, so the collaborative filtering can give user in the same group more useful advisories. The clustering algorithm will execute periodically in order to maintain the correctness of user clusters. Although clustering users make users with similar interests in the same group, but there are still a little difference between each members. Before calculating the group's preference, we should check the members in the group and remove the members who may change his interests to another categories in order to provide more precisely ranking. For example, Table 5 shows a group of 20 members but only 10 members has the history of querying in the same category with user's query q . In this example, the user query is in category C_1 . User 3 fails(doesn't click category C_1) 3 times in 4 queries, and we think the user may has another interests, so we want to remove user 3 in order to keep the contribution of group adjustment. The contribution for a user i in a category j can be defined as:

$$con_u(i, j) = \frac{n_{i,q} * n_i C_j}{\sum n_{i,q}}$$

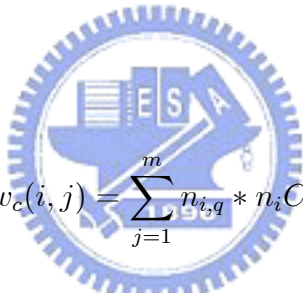
where $n_{i,q}$ represents the number of total queries for user i . $n_i C_j$ represents the number of user i clicks the category j in the query history.

That is: if a user has more queries and more correct ones, the user will have higher contribution. And if the user has the contribute value less than $1/k$, k is the number of members has the same query in the same category with user's query q , the user's contribution will not be considered in group adjustment. In this example, the contribution value of user 3 and 5 will be $4/48 = 0.08\bar{3}$. It's smaller than $1/10 = 0.1$, so the two users will not be considered in group adjustment.

$N_o.$	N_q	C_1	C_2	C_3	C_4
1	10	6	0	1	3
2	5	3	1	1	0
3	4	1	2	0	1
4	6	4	1	1	0
5	2	2	0	0	0
6	3	2	0	0	1
7	5	2	2	1	0
8	6	5	1	0	0
9	4	3	0	0	1
10	3	2	0	0	1
sum	48	30	7	4	7

Table 5: The group with 20 members and query history of 10 users with the same category query

After removing the users with lower contribution, we have to calculate the total contribute value for every category in query history. The weight of user i with category j can be defined as follow:



$$w_c(i, j) = \sum_{j=1}^m n_{i,q} * n_i C_j$$

For example the weight of category C_1 will be $10 * 6 + 5 * 3 + \dots + 3 * 2 = 163$

And the contribution of the category k can be define as:

$$con_c(k) = \frac{\sum_{i=1}^n n_{i,q} * n_i C_j}{\sum_{j=1}^m \sum_{i=1}^n n_{i,q} * n_i C_j}$$

where n is the number of selected users,

and m is the number of categories.

The weight of the category is based on their weight, and the weight is based on the number of queries and the number of hits. According to these two functions the weight of the four categories in the example are: $con_c(C_1) = 163 / (163 + 17 + 11 + 25) = 0.758$,

$con_c(C_2) = 17/(163 + 17 + 11 + 25) = 0.074$, and $con_c(C_3) = 0.051$, $con_c(C_4) = 0.116$. The weight of each category can be saved in the vector F_g . F_g is a 1 by m vector represents the group's adjustment to each category.

Last, we perform modified PageRank algorithm but insert two personalized vector F_p and F_g to compute the personalized ranking for different users and different groups. The PageRank's formula performs a global view of web pages, and in order to add personalized view into the formula. We add a personal vector F_p , the group's collaborative vector F_g . The F_p is shown in last subsection, the function will return the user's preference to the page and F_g will return the group's adjustment to the category the page is in. Finally, the personal PageRank formula can be written as follow:

$$PPR(v) = w_{global} * ((1 - c) \sum_{u \text{ links to } v} \frac{PPR(u)}{O(u)} + c) + w_p F_p * p(v) + w_{group} F_g * p(v)$$

where $p(v)$ is m by 1 vector and m is the number of categories.

The web graph has modified by the "Assigning Weights to Web Pages" step and "Adding Virtual hubs and virtual links" step. There are three parameters, w_{global} , w_p , and w_{group} . The sum of these three parameters will be 1. If the user has a new query to a category, the weight of w_{global} and w_{group} will be higher so the collaborative-filtering can help user to obtain personalized ranking. If the user has enough query histories, the weight of w_{global} will be less so that the personalized ranking will be satisfy user's interests.

According to the examples the user adjustment in four categories will be $\{0.44, 0.107, 0, 0\}$ the group adjustment in four categories will be $\{0.758, 0.074, 0.051, 0.116\}$, the $w_{global} = 0.3$, $w_p = 0.5$, $w_{group} = 0.2$, after the computation, the personalized ranking is shown in Table 6. We can easily see the difference in the rank and the scores between PageRank and our algorithm. In algorithm PageRank, page 3,7,6 have higher scores and ranks but these pages don't satisfy user interests. In our algorithm, page 3, 7, 6 have lower scores and ranks. The

Page	PR()	Page	PPR()
7	1.627362	5	1.065921
3	1.253509	9	0.998399
6	0.969129	2	0.944446
9	0.714537	4	0.861571
5	0.534247	1	0.5664
2	0.440805	10	0.544692
4	0.377055	7	0.498009
10	0.365532	14	0.482013
14	0.317318	13	0.399138
13	0.253568	3	0.385853
1	0.15	6	0.300539
8	0.15	8	0.2645
11	0.15	12	0.2645
12	0.15	11	0.0548

Table 6: The final personalized ranking against the original PageRank

pages user interested in has higher scores and ranks.

Our algorithm can also make the correctness when the user change interests. For example, if a user's long term interests is in C_1 but the last five of the query history is $\{C_1, C_3, C_1, C_3, C_3\}$. It means that the user has short term interests in C_3 . Although the first two steps of our algorithm adjust the search result by the long term interests, but the "Ranking based on User Clicking Streams" step can adjust the short term interests. The "Ranking based on Collaborative Filtering" step will remove the user in several period of time because the user doesn't fit the group. We use the same example mentioned before, the user's long term interests are in C_1 and C_2 , but the short term interest is in C_3 . Table 7 shows the ranking by our algorithm. Compare with the ranks in 6, page 3, 7, 6 in C_3 arise their ranks. Although they can't be the highest ones but the rank will more satisfy the user's interests when the user clustering algorithm re-cluster the user and assign the user to the right cluster.

Page	PRP()
9	0.987
5	0.7289
6	0.6573
7	0.6573
2	0.6014
3	0.5298
10	0.5195
14	0.477
4	0.4739
13	0.3495
1	0.3464
11	0.2748
8	0.2645
12	0.2645

Table 7: The ranking when the user’s long term interests and short term interests are not the same.

4 Performance Study

In order to validate our proposed mechanism, we have implemented one Web search system according to principles described in the previous sections. Our implemented Web environment is described in Section 4.1, and Section 4.2 is devoted to experimental results and comparison with PageRank and VIPAS.

4.1 Simulation model

We use the data set provided by the WebKB project[1] to model the web environment. This data set contains 5 major categories: “Cornell”, “misc”, “Texas”, “Washington” ,and “Wisconsin”. Each category in the data set contains the web pages of related to the corresponding university and the category "misc" has some other pages of other universities. Each major category could be further divided 7 minor categories: “course”, “department”, “faculty”, “others”, “projects”, “staff”, “student”. Figure 9 shows the ontology tree of the data set. The data set contains 8277 different pages, 777050 keywords contains 68188 different keywords. Table 8 and Table 9 shows the distribution of pages and keywords. For a brevity purpose, we

Class	# of pages	Percentage
A	867	10.47%
B	4118	49.75
C	826	9.98%
D	1205	14.56%
E	1261	15.23%

Table 8: The table of page distribution

Class	# of keywords	Percentage
A	98931	12.73%
B	351538	45.24%
C	89314	11.49%
D	112045	14.42%
E	125222	16.12%

Table 9: The table of keyword distribution

use A to represent the category "Cornell", B to "misc", C to "Texas", D to "Washington" and E to "Wisconsin".

Based on the data set, we randomly choose 10 keywords from all keywords as testing keywords to run the simulation. Table 10 shows the test keywords.

The user's preferences are simulated by randomly select the interest of the user and then generate the access patterns randomly. The category user interested in will contains more access patterns. In this simulation we generate 43 different user preferences. The number of users simulated is 43, and users with IDs from 1 to 12 are interested in class A, users with IDs from 13 to 18 are interested in class B, users with IDs from 19 to 30 are interested in class C, users with IDs 31 to 35 are interested in class D and users with IDs from 36 to 43 are interested in class E. Each user has his user interests randomly generated. Table ?? shows some samples of user interests. According to user interests, we will simulate user

browsing behavior. From the user browsing behavior, we could mine access patterns.

By our proposed procedure of extracting user interests, we could cluster the number of users into 5 groups shown in Table 12. It can be seen in Table 12 that the cluster groups of

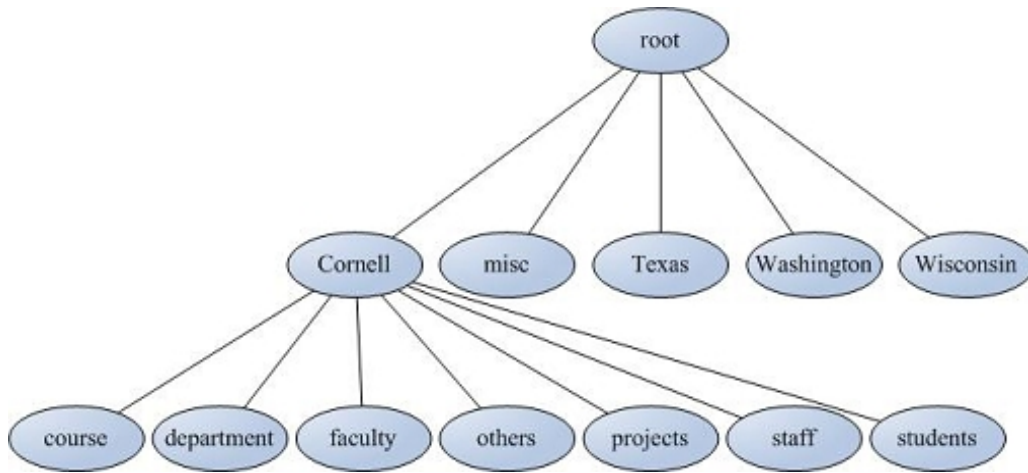


Figure 9: The ontology tree of the test data set.



keyword
minibase
coding
facilities
hotlist
dartmouth
informatics
alexander
administrative
smooth
guestbook

Table 10: The keywords for test.

user id	A	B	C	D	E
1	0.55	0.113	0.1	0.133	0.11
3	0.93	0.015	0.02	0.013	0.023
4	0.21	0.2	0.213	0.193	0.183
13	0.083	0.69	0.1	0.057	0.07
15	0.05	0.71	0.088	0.07	0.083
21	0.13	0.117	0.48	0.123	0.15
30	0.13	0.098	0.5	0.143	0.135
34	0.173	0.177	0.177	0.32	0.153
36	0.1	0.108	0.135	0.098	0.56
42	0.15	0.11	0.195	0.125	0.42

Table 11: The sample of user preference table.

Cluster	user
1	1,2,3,5,6,7,8,9,10,11,12
2	13,14,15,18
3	19,20,21,22,23,24,25,26,27,28,30
4	4,16,17,29,31,21,33,34,35
5	36,37,38,39,40,41,42,43

Table 12: The member of each cluster.

users are almost the same with our simulation parameters for users, showing the effectiveness of our procedure of extracting user interests.

4.2 Simulation results

The simulation can be divided into two major parts, the first part is algorithm comparisons. In this part, we will compare the query results in the same keyword but ranking in different algorithms. Besides, we'll also compare the results of querying the same keyword in different users and different algorithms. The last part of simulation is the discussion in parameters, we will observe the difference while we set the different value in some parameters. Here are the parameters we will discuss: (1)the weight of w_{global} , w_p and w_{group} . (2) The number of added virtual hubs in the virtual hubs and virtual links addition step. (3) The length in user query history.

In order to measure the ranking performance in the search results, we introduce coefficients

named *imp* and *discrepancy*. The *imp* coefficient is used to measure the importance of each page. The traditional The *imp* of page i can be derived as follow:

$$imp(i) = (n - rank(i)) * score(i)$$

n is the number of total documents retrieved by a query q , $rank(i)$ is the rank of page i compute by the algorithm. $score(i)$ is the score computed by the ranking algorithm. The higher *imp* the page has, the more important the page is. In order to simplify the items showed in the result graph, we only shows the average *imp* scores of each category in our simulation results In order to remove the effect by the page which has only outlinks but no inlinks, we add an restriction: In the regular PageRank, we'll drop the page with score 0.15(the lowest score in PageRank), and in our algorithm, we'll drop the pages with the lowest score in each category.

The *discrepancy* coefficient is introduced in VIPAS, and it's use to compare the difference between real rank and ideal rank. Real rank is the rank by the algorithm and the ideal rank is the rank in the user interested category. The *discrepancy* coefficient can be computed by the formula:

$$\mu = \frac{\sum_{\forall \text{ page } i \text{ in category } c \text{ in search result}} r(i) - ideal(i)}{n_c}$$

where $r(i)$ is the real rank of page i , $ideal(i)$ is the ideal rank of page i and n_c is the number of results in category c .

Assume that user 1 queries the keyword "minibase", "coding", "facilities" and we add the category A and C in virtual hubs addition. User 24 queries the keyword "hotlist", "dartmouth", "informatics" and we add the category C and D in virtual hubs addition. User 32 queries "alexander", "administrative" and we add the category B and D in virtual hubs addition. User 42 queries "smooth", "guestbook" and we add the category E in virtual hubs

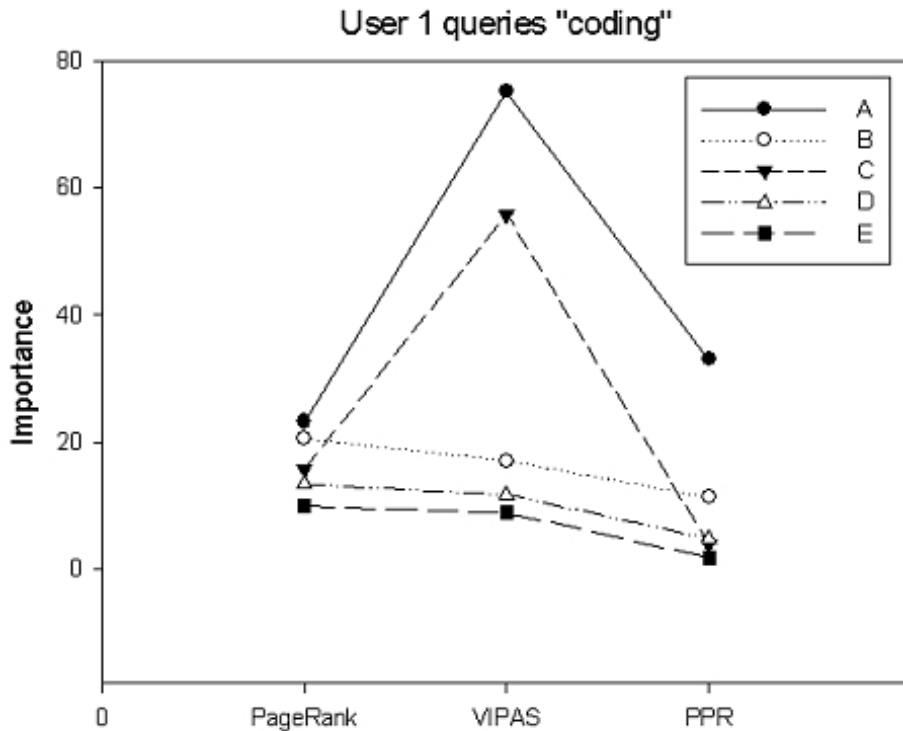
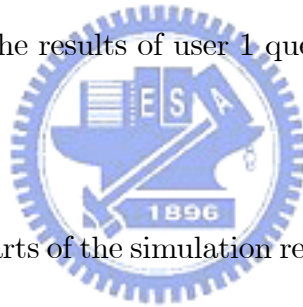


Figure 10: The results of user 1 queries "coding"



addition.

Figure .10 to Figure.14 shows parts of the simulation results. Here are the results computed by three different algorithms: "Regular PageRank", "VIPAS", "Our algorithm", and the *imp* of each category is the average of each page in the category. In order to get the precisely results, we will ignore the page has the lowest value in each category. By ignoring these pages, we can obtain the more important pages. According to the results, we can easily observe the difference between the regular PageRank and our algorithm. We rank the pages user may like higher and more ideal satisfy user's interesting in our algorithm. Figure.10 to Figure.14 show the change of the rank and the score after our algorithm. The class user interested has the highest importance and the other class may has higher importance after ignoring the lowest pages. In algorithm VIPAS, the importance of the category user interested in higher than the importance in our algorithm. It is because VIPAS just adds weights to the pages, but in our algorithm the weights of virtual hubs and virtual links are lower score than VIPAS Figure 15

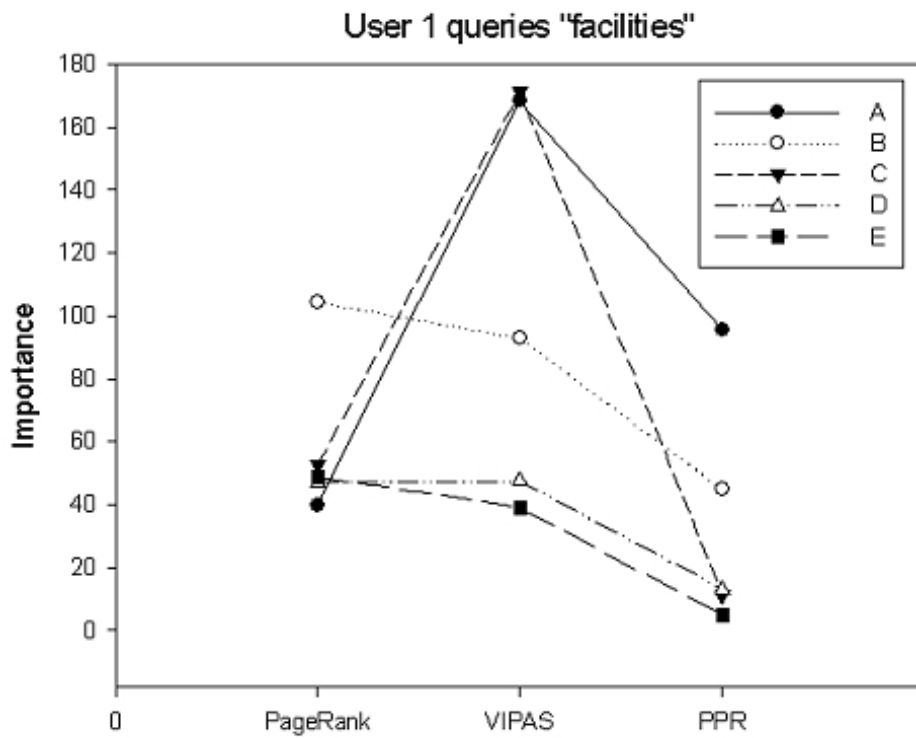


Figure 11: The results of user 1 queries "facilities"

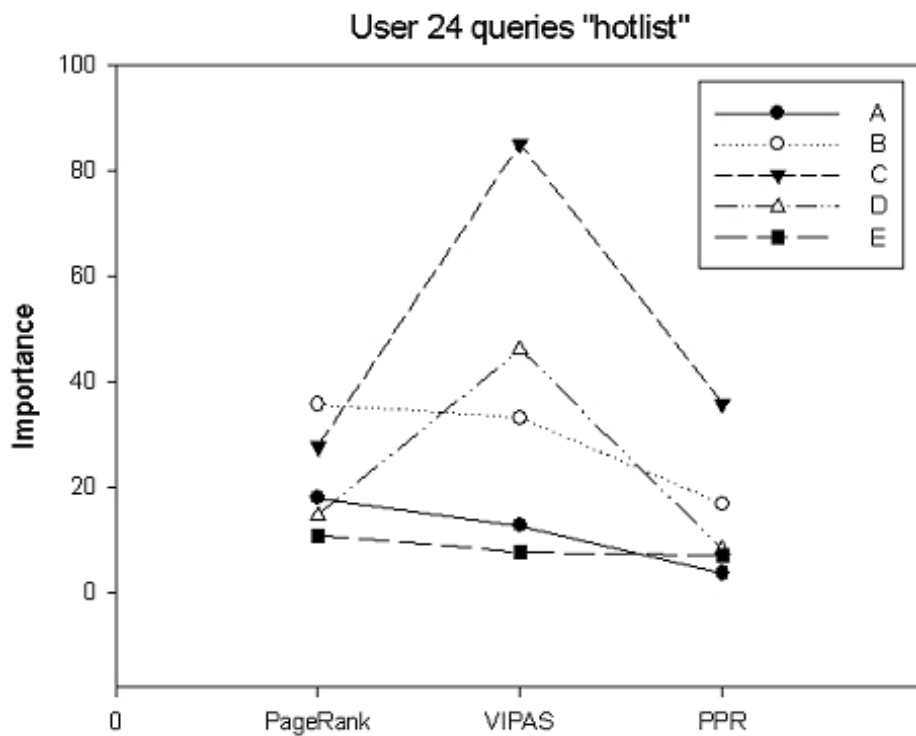


Figure 12: The results of user 24 queries "hotlist".

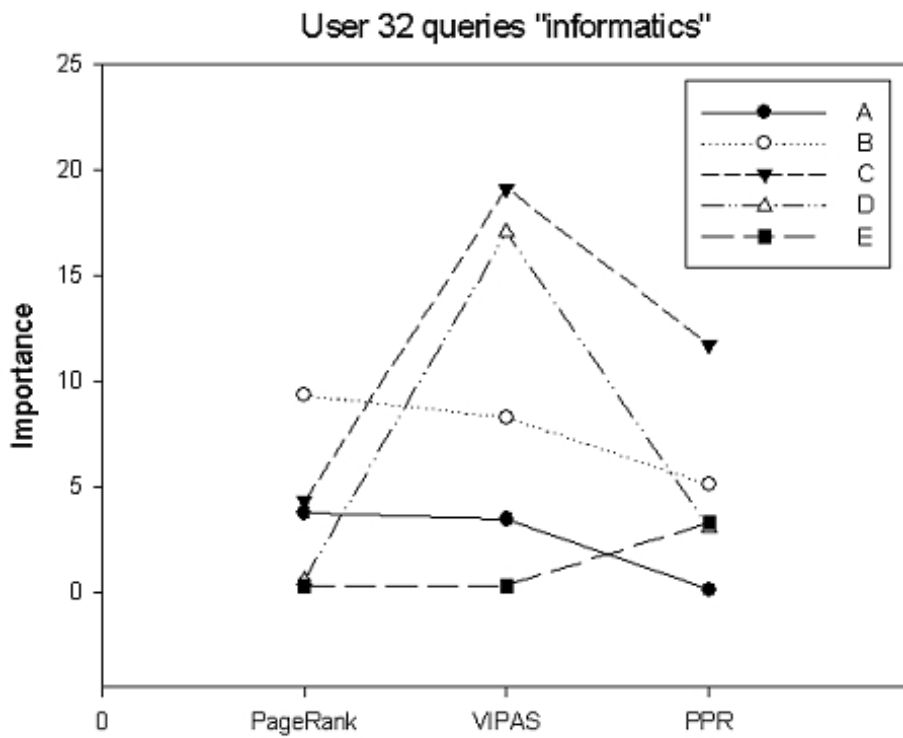


Figure 13: The results of user 32 queries "informatics".

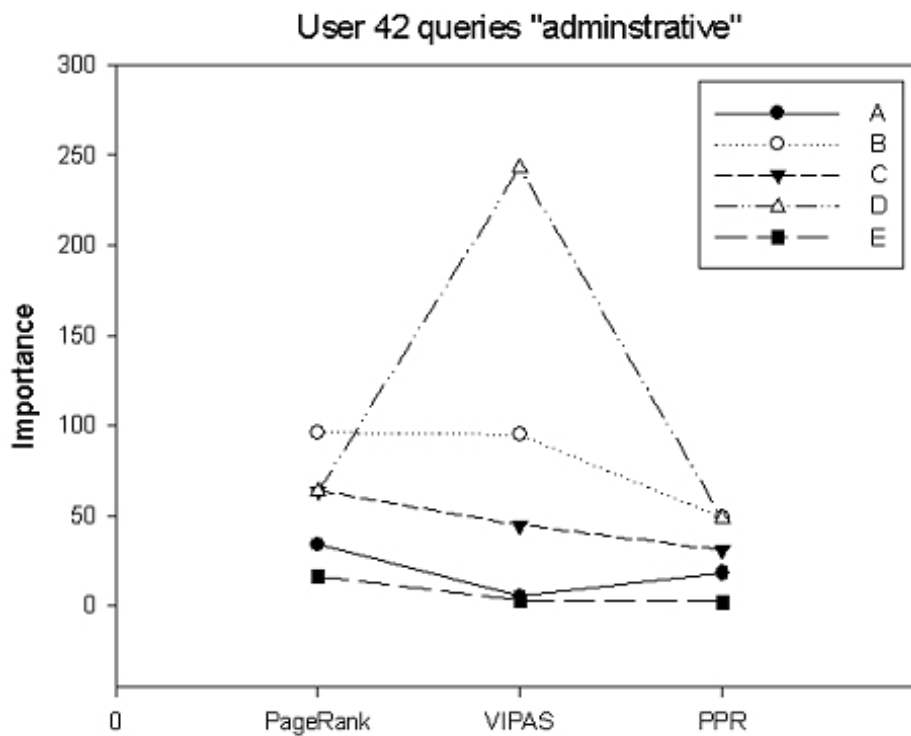


Figure 14: The result of user 42 queries "administrative"

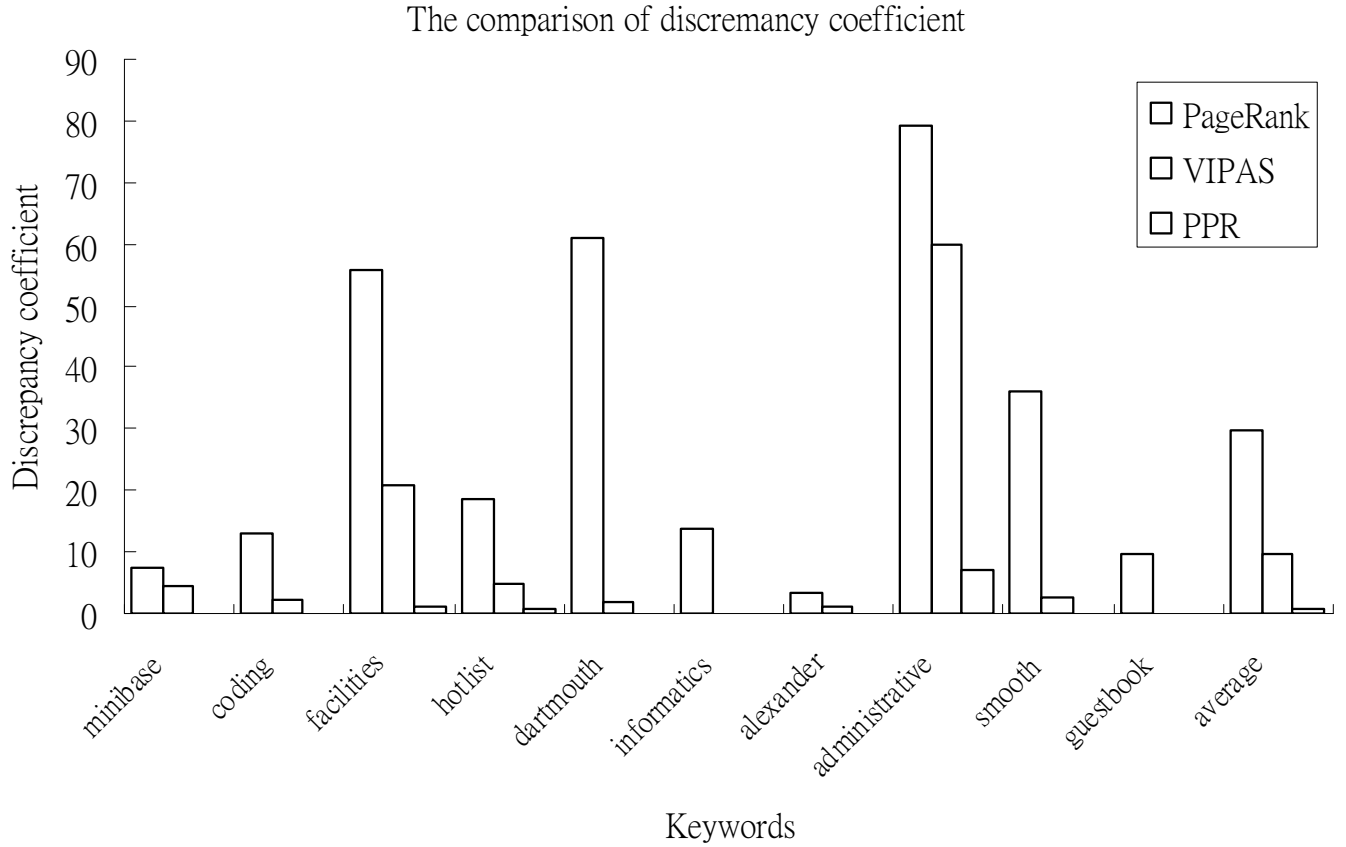


Figure 15: The comparison of discrepanct coefficient.



shows the difference between the real rank and the ideal rank. The discrepancy coefficient may be much lower in our algorithm, it means the rank in our algorithm satisfies users well. The average discrepancy coefficients of algorithm PageRank, VIPAS and ours are {29.785912, 9.7983765 and 0.8869048}.

After comparing the results with different users and different query keywords, we can easily observe that our algorithm gives higher rank and score to the category user interested in. Besides, we compare that different kind of users query the same keyword. Table 13 and Table 14 shows the query results and the importances in each category and each user. We can observe that our algorithm still preforms well in the situation. Figure 16 shows the discrepancy coefficient in different users. According to the simulation results, our algorithm will recommend the pages user may interested in although there are different kind of users.

Regular PR	VIPAS
http://www.tc.cornell.edu/Visualization/Education...	http://www.tc.cornell.edu/Visualization/Education..
http://www.cs.cornell.edu/Info/People/rdz/dissolve..	http://www.cs.cornell.edu/Info/People/rdz/dissolve..
http://metacrawler.cs.washington.edu:8080/config..	http://www.cs.utexas.edu/users/vin/cs384m.htm
http://www.cs.rice.edu/~adve/	http://www.cs.rice.edu/~adve/
user 1	user 24
http://www.cs.cornell.edu/Info/People/rdz/dissolve..	http://www.cs.utexas.edu/users/vin/cs384m.html
http://www.tc.cornell.edu/Visualization/Education/...	http://www.cs.utexas.edu/users/gunnels/Transpose/..
http://www.cs.cornell.edu/Info/Courses/Current/CS...	http://www.cs.cornell.edu/Info/People/rdz/dissolve..
http://www.cs.cornell.edu/Info/Courses/Spring-95/...	http://www.cs.utexas.edu/users/vl/teaching/descript..
user 32	user 42
http://metacrawler.cs.washington.edu:8080/conf...	http://www.cs.wisc.edu/~lukas/tuftslist.html
http://www.cs.cornell.edu/Info/People/rdz/dissolv..	http://www.cs.wisc.edu/~markhill/cs752/
http://www.tc.cornell.edu/Visualization/Educatio..	http://www.cs.wisc.edu/~cs537-1/project1.html
http://www.cs.washington.edu/homes/vass/MVis	http://www.cs.wisc.edu/~cs354-2/cs354/homework..

Table 13: The query results by different user query the same keyword.



	A	B	C	D	E
user 1	21.5194	12.2987	2.33538	2.41803	10.3766
user 24	9.14515	9.83225	23.3807	5.24498	15.137
user 32	7.88751	3.56802	4.88086	18.3439	9.40968
user 42	5.34132	4.05488	7.70857	3.73662	22.5525

Table 14: The importance of each category for querying the same keyword.

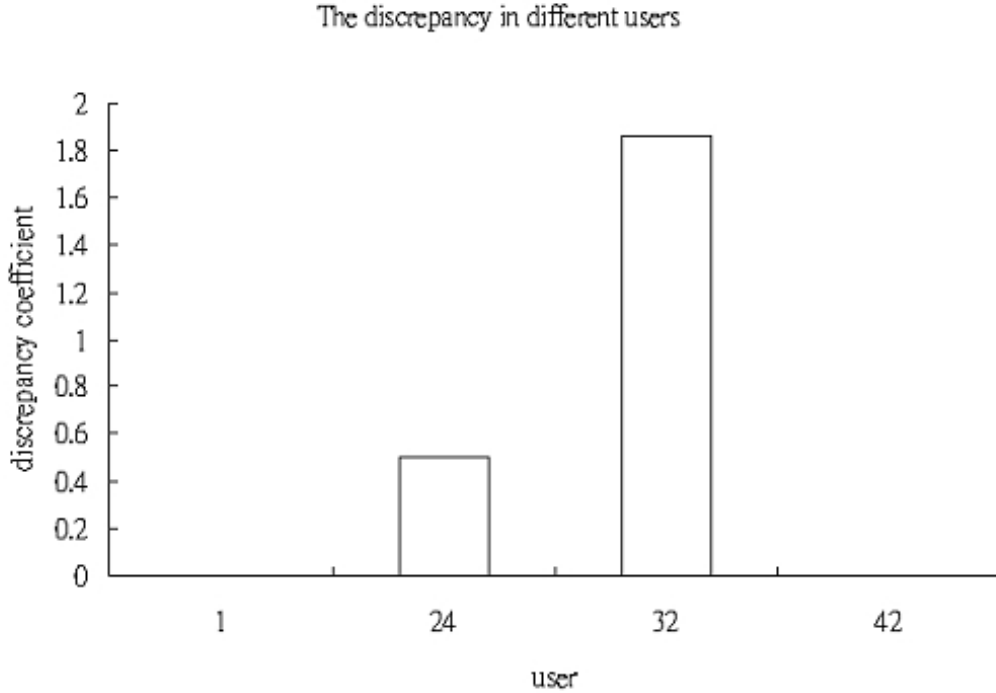


Figure 16: The discrepancy coefficient of different user query the same keyword.

algo	w_{global}	w_p	w_{group}
1	0.5	0.3	0.2
2	0.2	0.5	0.3
3	0.3	0.2	0.5

Table 15: The weight in each algorithm.

After comparing to other algorithms, we'll discuss the effect of changing parameters. We'll change the weight of the parameters w_{global} , w_p and w_{group} in our ranking formula. These three parameters play important roles in our ranking algorithm. If the user doesn't have the similar queries, than the weight of w_{global} and w_{group} will be higher to help the user to retrieve the information he needs and time goes by, the user has enough query histories and the weight of w_{global} and w_{group} will be lower to achieve the personalized goal.

We choose the user 1 queries "coding" as the example and observe the change of importance in each category and the discrepancy coefficient. We compare three kinds of distribution in w_{global} , w_p and w_{group} . The weight of each algorithm is in Table 15.

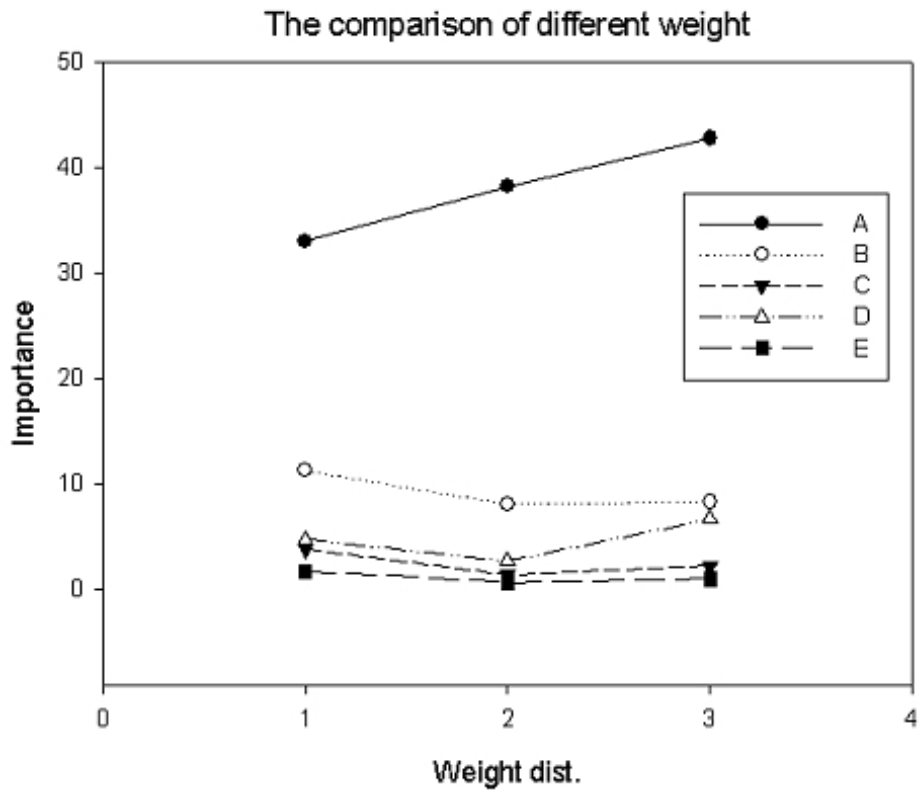


Figure 17: The importance of each category in each algorithm.

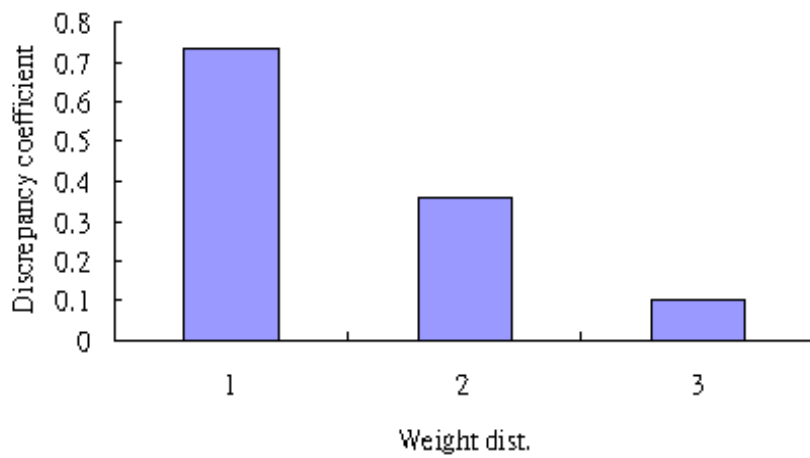


Figure 18: The discrepancy coefficient in different weight.

Figure 17 and Figure 18 shows the results. We can see that the importance of category A is raising and the importance of other categories don't change much. The discrepancy coefficient in the three algorithm is low. That is, the change weight of parameters just change the score of the pages but the ideal rank doesn't change too much.

The next parameter in our discussion is the number of virtual hubs added in the virtual hubs and virtual links addition. Assume that user interests category order is A,C,B,D,E. That is if we only add one virtual hub, we'll choose category A and if we add two virtual hubs, we'll add category A and C. Assume that we choose user 1 and keyword "facilities" for the simulation. Figure 19 shows that the importance of the category which user interested in (category A) won't change too much while the number of VHs are different. The importance of the other categories will arise when the virtual hub is added to the category. Figure 20 shows that although the importance of category A doesn't change too much, but the discrepancy coefficient will arise when the number of VHs arise. That means proper number of virtual hubs added will increase the performance but when we add too much virtual hubs, the performance will descent.

The last coefficient in our discussion is the length of the user query history we choose for computing the vector F_p . Assume that the query history for user 1 is $\{C_3, C_1, C_2, C_1, C_2, C_2, C_1, C_1, C_1, C_3, C_1, C_1, C_2, C_2, C_1, C_4, C_1, C_1, C_1, C_1\}$ the length in our simulation are 5, 10 and all. We choose user 1 and keyword "facility" for the simulation. Table 16 shows the vector F_p of three different length. Figure 21 and Figure 22 shows the simulation results. The shorter the user query history we use, the category user interested in will rank higher. If we choose shorter user query history and the user changes interests, our algorithm can make the rank more satisfy the user newest interests.

At the last of our simulation, we'll show the flexibility of our algorithm. Assume that user's long term interest is in category A, when a query q sent to the server, the server may rank the pages belong to the long term interests higher. But if the user has a short term

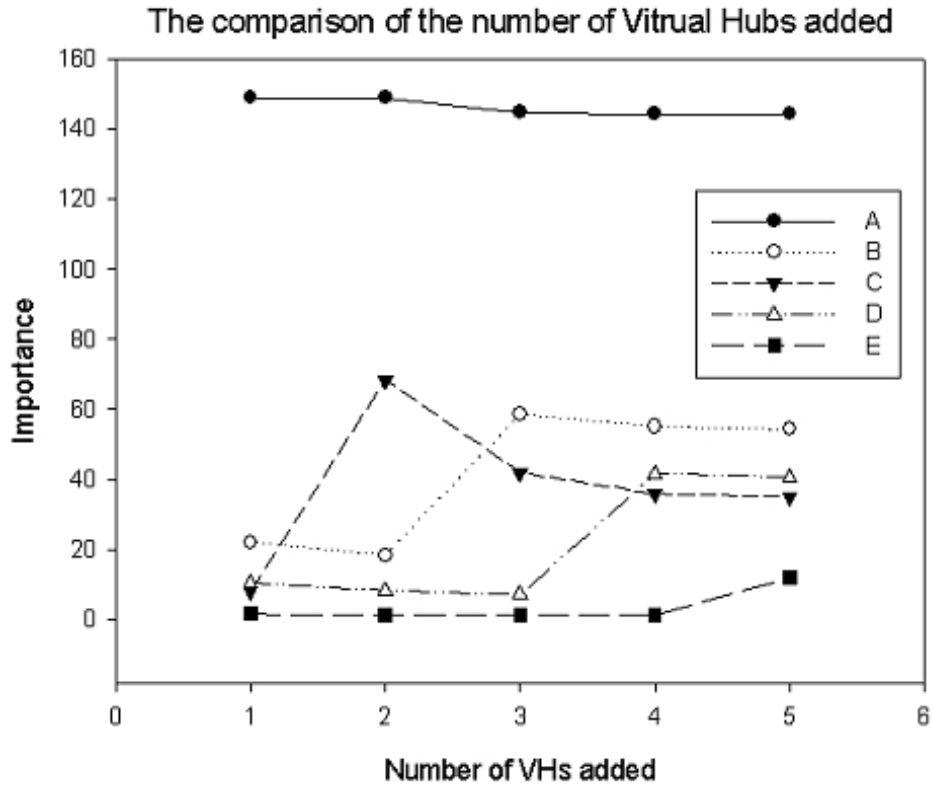


Figure 19: The comparison of the number of vitrual hubs added.

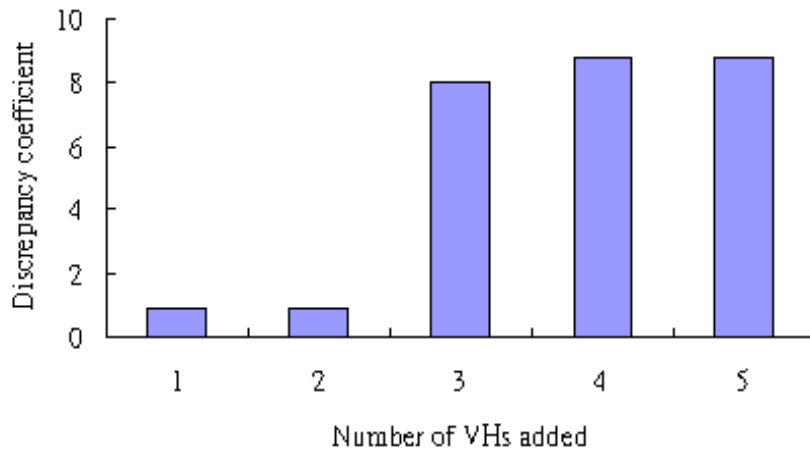


Figure 20: The discrepancy in different number of VHs added.

	A	B	C	D	E
5	0.7466	0	0	0.0133	0
10	0.5345	0.0254	0	0.0109	0
ALL	0.4057	0.0488	0.0052	0.0038	0

Table 16: The factor f_p in different length of user query history.

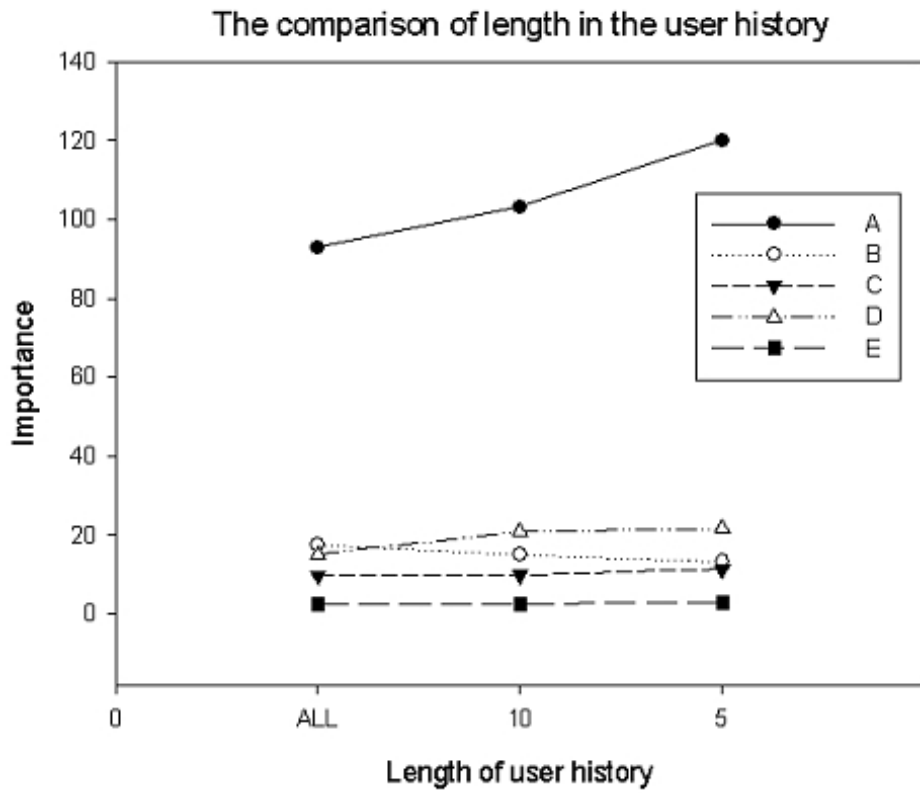


Figure 21: The importance in different length of uwer query history.

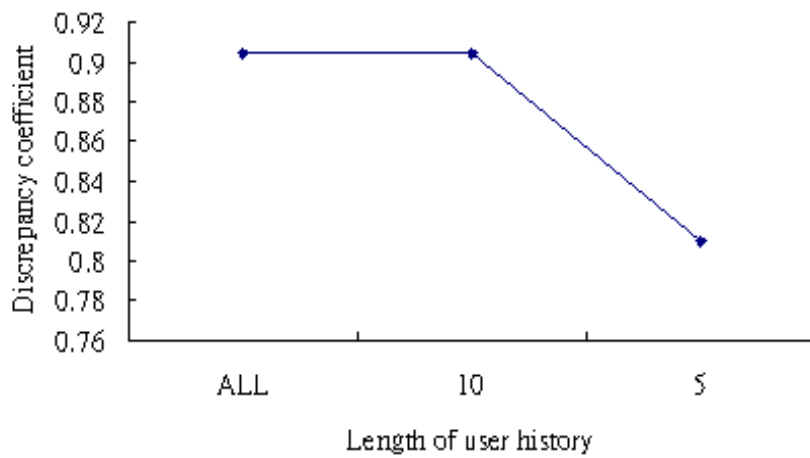


Figure 22: The discrepancy coefficient in different length of user query history.

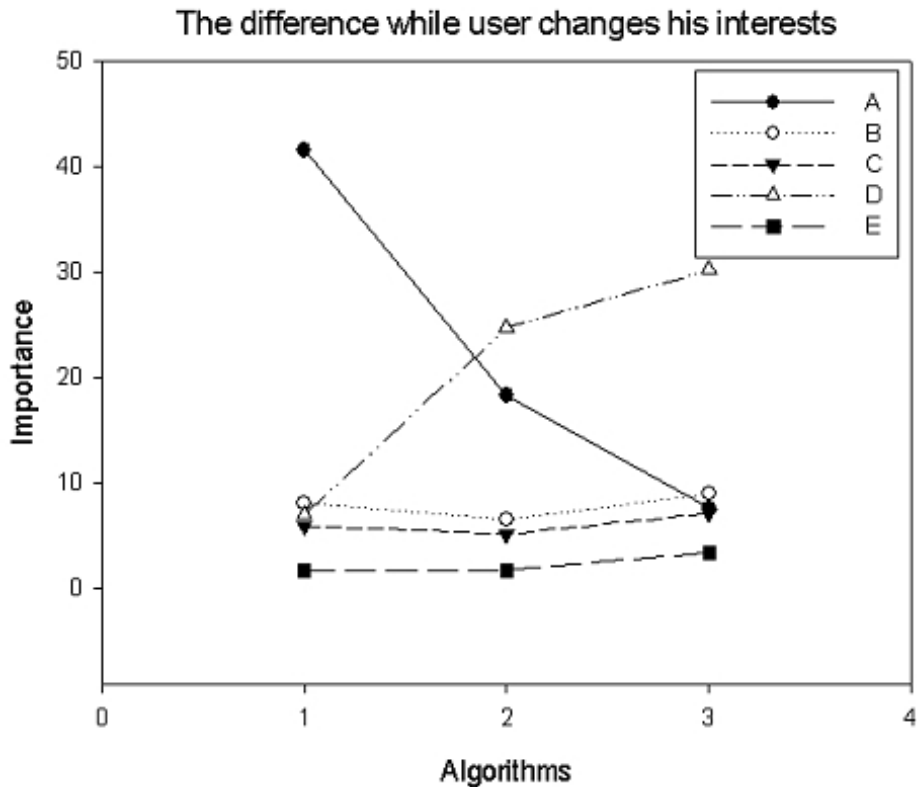


Figure 23: The difference while user changes his interests.

interest in another category, the pages in the short term interest will not as important as the pages in the long term interests. The "Ranking based on User Clicking Streams" part in our algorithm can reduce the tragedy of prediction fail. Assume that the user's long term interest is in category A and the user's short term interest is in category D, we'll show the difference while user changes his interests. Algorithm 1 represents user is in group A and the user click the pages in category A. Algorithm 2 represents user is in group A but the user change his interests to category D. Algorithm 3 represents that the user change his group to category D and the user has short term interests in category D, too Assume that user 1 in category A queries the keyword "coding". Figure 23 and Figure 24 show the result of the difference. We can easily observer that the importance of category D is arising, it means our algorithm can observe the change of user's interest and return the rank that satisfy the user more.

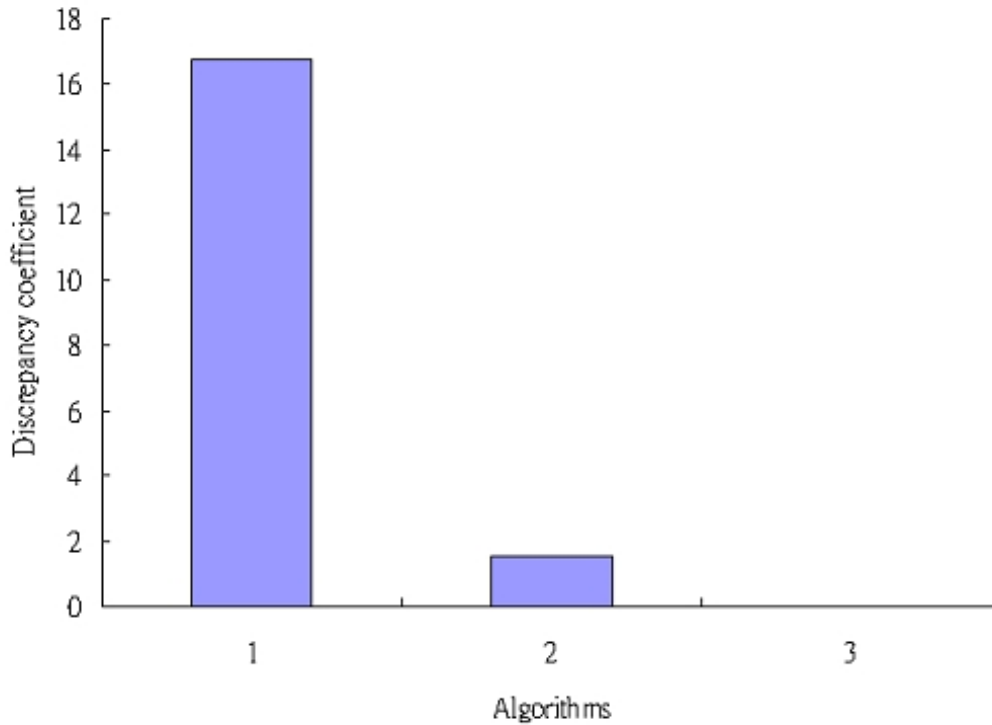


Figure 24: The discrepancy coefficient while user changes his interests.

5 Conclusions



In this paper, we have implemented a client-side module that is able to capture user browsing behavior and then exploited the technique of data mining to mine frequent browsing access patterns from user browsing behavior. In light of frequent browsing access patterns, we proposed a method to extract user interests as user preferences. In this paper, we have developed a new algorithm with the idea of adjusting the ranking scores of Web pages. The adjustments are in accordance with user preferences mined from user browsing behavior. Specifically, algorithm PPR is divided into four phases. The first phase assigns the initial weights based on user interests. In the second phase, the virtual links and hubs are created according to user interests. By observing user click streams, our proposed algorithm will incrementally reflect users favors for the personalized ranking in the third phase. To improve the accuracy of ranking, collaborative filter is taken into considerations when the query with similar keywords are

submitted by users having similar user interests. By conducting simulation experiments, we have shown that algorithm PPR is not only very effective but also very adaptive in providing personalized ranking to users.

References

- [1] Webkb project <http://www.webkb.org/>.
- [2] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. In *Proc. of the 7th International World Wide Web Conference*, 1998.
- [3] J. Carriere and R. Kazman. WebQuery: Searching and visualizing the Web through connectivity. In *Proc. of the 6th International World Wide Web Conference*, 1997.
- [4] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleinberg. Automatic resource compilation by analyzing hyperlink structure and associated text. In *Proc. 7th International World Wide Web Conference*, April 1998.
- [5] T. Haveliwala. Topic-Sensitive PageRank. In *Proceedings of the Eleventh International World Wide Web Conference*, 2002.
- [6] G. Jeh and J. Widom. Scaling personalized web search. In *Proc. of the 12th International WWW Conference, 2003.*, 2002.
- [7] C.-C. Lin and M.-S. Chen. VIPAS: Virtual Link Powered Authority Search in the Web. In *Proc. of the 29th Intern'l Conf. on Very Large Data Bases*, 2003.
- [8] J. Pei, J. Han, B. Mortazavi-Asl, and H. Zhu. Mining access patterns efficiently from web logs. In *Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications*, pages 396–407. Springer-Verlag, 2000.

- [9] D. J. Zhao, D. L. Lee, and Q. Luo. A meta-search method with clustering and term correlation. In *DASFAA*, pages 543–553, 2004.

