Dynamic Adaptation Strategies for Long-Term and Short-Term User Profile to Personalize Search

Lin Li, Zhenglu Yang, Botao Wang, and Masaru Kitsuregawa

Institute of Industrial Science, The University of Tokyo,
4-6-1 Komaba, Meguro-Ku, Tokyo 153-8305, Japan
{lilin,yangzl,botaow,kitsure}@tkl.iis.u-tokyo.ac.jp

Abstract. Recent studies on personalized search have shown that user preferences could be learned implicitly. As far as we know, these studies, however, neglect that user preferences are likely to change over time. This paper introduces an adaptive scheme to learn the changes of user preferences from click-history data, and a novel rank mechanism to bias the search results of each user. We propose independent models for long-term and short-term user preferences to compose our user profile. The proposed user profile contains a taxonomic hierarchy for the long-term model and a recently visited page-history buffer for the short-term model. Dynamic adaptation strategies are devised to capture the accumulation and degradation changes of user preferences, and adjust the content and the structure of the user profile to these changes. Experimental results demonstrate that our scheme is efficient to model the up-to-date user profile, and that the rank mechanism based on this scheme can support web search systems to return the adequate results in terms of the user satisfaction, yielding about 29.14% average improvement over the compared rank mechanisms in experiments.

1 Introduction

With the advent of the era of the information explosion, never before have there been so many information sources availably indexed by search engines on the Internet. Ideally, users should be able to take advantage of the wide range of the valuable information while being able to find only those which are appealing to them. On the contrary, it becomes more difficult than ever to obtain desired results due to the ambiguity of user needs. Moreover, present search engines generally handle search queries without considering user preferences or contexts in which users submit their queries. For example, suppose that a database researcher who wants to search for information about a conference on Mobile Data Management and a banker who is interested in searching for the MDM bank, both input "MDM" on Google. Regardless of the different intentions of the two users on the same query, the results turn out to be a multimedia software company, a broadband services company, a national observatory, a conference on mobile data management, and so on. Current search engines prove unfortunately inadequate for this situation.

To address this problem, personalized search has recently become an active on-going research field. Studies [2, 14] have focused on requiring users to explicitly enter their contextual preferences including interest topics, bookmarks, etc., and these contextual preferences are used to expand user queries or re-rank search results. Forcing users to submit their contextual preferences would be a task that few users would be willing to do. Furthermore, it is very difficult for users to define their own contextual preferences accurately. Much attention has been paid in [13, 16, 18, 19] to learn user preferences transparently without any extra effort from users. These studies place emphasis on modelling user profiles or user representations to indicate user preferences automatically. Speretta et al. [18] created user profiles by classifying some information into concepts from the ODP [12] taxonomic hierarchy and then re-ranked search results based on the conceptual similarity between the web page and the user profile. The authors, however, have not taken the hierarchy structure of the ODP into account when calculating the conceptual similarity.

In this paper, we focus on studying learning user profiles and utilizing the learned user profiles to re-rank search results. Most studies on learning user profiles have deemed user profiles to be static. A related problem occurs when user preferences change over time. For instance, if a user changes her vocation from being an IT specialist to a lawyer, her interests will naturally shift with this change. It becomes important to keep the user profile up-to-date, and for a search engine to adapt accordingly. In addition, a user profile covers both short-term and long-term user preferences, which may increase or reduce respectively and co-relatedly with time. Using one model to represent two differently featured parts of the user profile will be far from perfect. Accordingly, suitable strategies are needed to capture the accumulation and degradation of changes of user preferences, and then adapt the content and the structure of a user profile to these changes. For re-ranking search results, our rank mechanism is similar to that proposed by [2] in which a semantic similarity measure is introduced with consideration to the hierarchy of the ODP structure. Meanwhile, the technique proposed in [2] suffers from the problem of requiring users to select topics which best fit their interests from the ODP, and other shortcomings we will address in Section 4.

Our contributions in this paper could be summarized as follows.

(1) We devise independent models for long-term and short-term user preferences.
(2) Dynamic adaptation strategies for modelling user profiles automatically are proposed. These strategies are based on click-history data while considering the accumulation and degradation changes of user preferences.
(3) When user preferences change, our user profiles, not only in contents, but also in structures, are modified to adapt to the changes.
(4) Finally, we propose a novel rank mechanism by measuring hierarchy semantic similarities between up-to-date user profiles and web pages. About 29.14% average improvement is gained over existing rank mechanisms.

The rest of this paper is organized as follows. In Section 2, we review the related work. In Section 3 we describe two independent models and dynamic adaptation strategies for user profiles. Rank mechanisms and evaluation metrics are addressed in Section 4. Section 5 presents the experimental results. Finally, we conclude in Section 6 with some directions for future work.

2 Related Work

2.1 Context Search

Kraft et al. [8] state that the context, in its general form, refers to any additional information associated with the query in the web search field, and also present three different algorithms to implement the contextual search instead of modelling user profiles. Generally speaking, if the context information is provided by an individual user in any form, whether automatically or manually, explicitly or implicitly, the search engine can use the context to custom-tailor search results. The process is named as a personalized search.

In this way, such a personalized search could be either server-based or client-based. The system in [4] is an available server-based search engine that unifies a hierarchical web-snippet clustering system with a web interface for the personalized search. Google and Yahoo! also supply personalized search services. With the cost of running a large search engine already very high, however, it is likely that the server-based full-scale personalization is too expensive for the major search engines at present.

On a client-based personalized search, studies [3, 16, 19] focus on capturing all the documents edited or viewed by users through computation-consuming procedures. Allowing for scalability, the client-based personalized search could learn user contexts more accurately than the server-based personalized search, while it is unavoidable that keeping track of user contexts has to be realized by a middleware in the proxy server or the client. Users, however, may feel unsafe to install such a kind of softwares even if they are guaranteed to be non-invasive, and may intend to enjoy the services provided by search engines instead. Moreover, if a user at home uses her private computer which is different from that in her office, keeping her contexts consistent becomes a problem. Therefore, our work is server-based.

In this paper, we focus on the use of suitable strategies to learn user profiles in a trade-off between scalability and accuracy for the server-based personalized search.

2.2 User Profile

There have been vast schemes of learning user profiles to figure user preferences from text documents. We notice that most of them model user profiles represented by bags of words without considering term correlations [1, 9, 17, 20]. To overcome the drawbacks of the bag of words, the taxonomic hierarchy, particularly constructed as a tree structure, has been widely accepted in [2, 11, 15]. Schickel-ZuberF et al.[15] score user preferences and concept similarity based on the structure of ontology. But their work needs users to express their preferences by rating a given number of items explicitly.

Meanwhile, these studies omit that user interests could change with time. Some topics will become more interesting to the user, while the user will completely or to varying degrees, lose interest in other topics. Studies [1, 9, 20] suggest that relevance feedback and machine learning techniques show promise in adapting to changes of user interests and reducing user involvements, while still overseeing what users dislike and their interest degradation. In [9] a two-level approach is proposed to learn user profiles for information filtering. While the lower level learns the stationary user preferences, the higher level detects changes of user preferences. In [20] a multiple three-descriptor representation is introduced to learn changes in multiple interest categories, and it also needs positive and negative relevance feedback provided by users explicitly.

Our work, particularly our dynamic adaptation strategies for user profiles, are based on the idea that sufficient contextual information is already hidden in the web log with little overhead, and all the visited pages can be considered as user preferences to various degrees because the users have accessed them. This contextual information motivates us to capture the accumulation and degradation changes of user preferences implicitly, to learn user profiles automatically.

3 User Profile and Dynamic Adaptation Strategies

As indicated in [20], for user profiles, long-term user preferences generally hold user preferences and the degree of preferences accumulated by experiences over a long time period. Hence it is fairly stable. On the other hand, short-term user preferences are unstable by nature. For instance, interests in current hot topics could change on a day-to-day basis. It is crucial to design a temporal structure for shot-term user preferences. Based on these features, we propose two novel models for long-term and short-term user preferences respectively and discuss them together with the adaptation strategies for their close correlations. Our strategies are in accord with the changes of user preferences in nature.

3.1 Long-Term Model of User Profile

The taxonomic hierarchy for our long-term model is a part of the Google Directory [5]. This part is composed of topics that have only been associated with the clicked search results, instead of the whole Google Directory. And these topics are linked as a tree structure to form our long-term model that is also called the user topic tree from now on. In other words, each node in the user topic tree means a topic in the Google Directory. We use search results and web pages interchangeably when referring to the URLs returned from the web search engine on a specific query.

In the Google Directory, each web page is classified into a topic [1]. In the "adding" operation, topics associated with the clicked pages are added into the user topic tree click by click. Moreover, each node in the user topic tree has a

[1] If necessary, all the symbolic links may be loaded into memory or the shortest distance on the graph is computed.

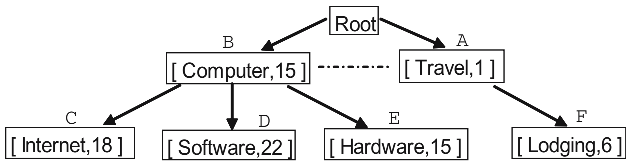

Fig. 1. Schema of Long-Term User Profile

value of the number of times the node has been visited. This value is called the "$TopicCount$", and represents the degree of preferences. The "deleting" operation is effected by the changes of the short-term model. It will be addressed in Section 3.2. Figure 1 illustrates the schema of the user topic tree. For example, node C is represented by the $[Internet, 18]$ which means one user has clicked a page associated with the topic "$Internet$" and the user has visited the "$Internet$" 18 times before this search. In our experiments node C is actually stored as the $[\backslash Root\backslash Compuetr\backslash Internet, 18]$ with a full path in the Google Directory.

3.2 Short-Term Model of User Profile

We frame the Page-History Buffer (PHB) for the short-term model. The PHB caches the most recently clicked pages with a fixed size that is determined by the ability of the search engine. We now meet the same problem as the cache in the processor, and that is how to kick off the "old" pages in time to keep up with the changes of short-term user preferences. As it is known, in the cache management, there are popular cache replacement algorithms that are all designed for the processor, the web cache and the database disk buffering. No such research could be available in the personalized search, especially in the short-term model of the user profile. Our goal, keeping track of the most recent accesses of search results in the PHB, is basically similar to that in the cache management. As a result, the LFU (Least Frequently Used), one of these replacement algorithms, is adjusted to our scheme, which is named the Least Frequent Used Page Replacement (LFUPR). The details are shown in Table 1. The LFUPR reflects the changes of the short-term model, including how to add (line 3 $\sim$ line 6) and replace (line 10 $\sim$ line 12) web pages in the PHB.

From Figure 1 and the LFUPR algorithm in Table 1, our dynamic adaptation strategies maintain user profiles such that the short-term model is updated by the LFUPR (line 1 $\sim$ line 15), while the degree of preferences in the long-term model could be degraded (line 13) when the page in the PHB is replaced, and could be accumulated when the user clicks the page ("adding" operation). On the other hand, if the user accesses the web page whose associated topic is not in the current user topic tree, the new node could be added into the tree ("adding" operation). From line 16 to line 18, if the "$TopicCount$" of one node becomes zero, the node would be deleted from the tree. This procedure is called the "deleting" operation. The "adding" and "deleting" operations dynamically

Table 1. LFUPR Algorithm

Input:	current short-term model, current long-term model, search results
Output:	updated user profile
Parameters:	PageCount=Vector of the number of clicked times for pages in the PHB TopicCount=Vector of the number of clicked times for nodes in the user topic tree BufferPages=Vector of pages in the PHB Results=Vector of pages returned by a search engine UserTopics=Vector of nodes in the user topic tree

```
1.    For i=1 to Size(Results)
2.    Begin Loop
3.        If Result[i] is the $n$th page IN the PHB
4.          PageCount[n]++;
5.        Else If PHB is NOT FULL
6.          Add the clicked page into the end of the PHB;
7.        Else
8.          Begin
9.            For j=1 to Size (PHB)
10.             BufferPages[k] ← Find one page in the BufferPages with the Minimum
11.             PageCount[j];
12.           Replace the BufferPages[k] with the Results[i];
13.           TopicCount[m]- -; //BufferPages[k] is the $m$th node IN the UserTopics
14.         End
15.   End loop
16.   For t=1 to Size (UserTopics)
17.   If TopicCount[t] ==0
18.       Clear the UserTopics[t] out from the user topic tree
```

adapt the structure of the long-term model to the user click behaviors. Although we design independent models for short-term and long-term user preferences, our strategies ensure that the inherent correlations between them are not ignored, and that the changes of the short-term model have an even influence on the long-term model. Here, the meaning of "even" is that we degrade the "$TopicCount$" not on an hour-to-hour or a day-to-day basis, only after a period of time during which the user has not accessed the *topic* in the whole search process.

4 Rank Mechanisms and Evaluation for Personalized Search

4.1 Distance Metrics

The tree distance which we deal with, is the distance between each search result and the user topic tree, as described in [2]. The search result with the shorter distance, meaning the higher similarity to user preferences, should be put in the topmost position of the ranking list. For each search result, there is an associated node in the Google Directory. The user topic tree is also composed of nodes. The distance computation is actually how the distance between two nodes in the tree structure is measured.

Chirita et al. [2] point out that the main drawback of the naïve tree distance is that it overlooks the depth of the subsumer (the deepest node common to two nodes). With the help of Figure 1, let us explain the problem clearly. $sub_{i,j}$ represents the subsumer of the node i and the node j. $Edges(i, sub_{i,j})$ represents the number of edges between the node i and the node $sub_{i,j}$. The naïve distance is defined as

$$Distance(i,j) = Edges(i, sub_{i,j}) + Edges(j, sub_{i,j})\,. \tag{1}$$

$Distance(A,B)$ is 2, which is the same as $Distance(C,D)$, making it difficult to re-order search results by Equation (1).

4.2 Hierarchy Semantic Similarity

Li et al. [10] takes the depth of the subsumer h and the naïve distance between two nodes l into the calculation. α and β are the parameters scaling the contribution of the naïve distance and the depth respectively. The semantic similarity is defined as

$$Sim(i,j) = e^{-\alpha\cdot l} \cdot \frac{e^{\beta\cdot h} - e^{-\beta\cdot h}}{e^{\beta\cdot h} + e^{-\beta\cdot h}}\,, \quad \alpha \geq 0\,, \beta > 0\,. \tag{2}$$

Their experiment results show that the optimized values of the two parameters are, α=0.2 and β=0.6. For example, $Sim(A,B)$ is unequal to $Sim(C,D)$ based on Equation (2). Because the subsumer of A and B, i.e., "*Root*", is in the different level from the subsumer of C and D, i.e., "*Computer*". However, Equation (2) only solves problem partially. Let us see another example. Due to the same value (i.e., 3) between $Distance(A,C)$ and $Distance(B,F)$, and the same subsumer (i.e., "Root") between the pairs (A, C) and (B,F), $Sim(A,C)$ is equal to $Sim(B,F)$.

Under this situation, Chirita et al. [2] separate l into l_1 and l_2, and then gives different weights to the two variables through the parameter δ defined as

$$Sim^*(i,j) = ((1-\delta)\cdot e^{-\alpha\cdot l_1} + \delta\cdot e^{-\alpha\cdot l_2}) \cdot \frac{e^{\beta\cdot h} - e^{-\beta\cdot h}}{e^{\beta\cdot h} + e^{-\beta\cdot h}}\,. \tag{3}$$

Equation (3) can work well for common cases. However, we find that the parameter δ is sensitive to the semantic meanings between the two topics, as illustrated in [2]. Furthermore, even if we compute the similarity by Equation (3), $Sim(C,D)$ is still equal to $Sim(E,D)$ because of the same value between l_1 and l_2. In our system, we extend Equation (2) in another way, as the "$TopicCount$" has much better effect on the overall performance than the weak parameter δ. Comparative experiments are in Section 5.

4.3 Our Rank Mechanism

When a user submits a query to the search engine, the search results are reranked by our semantic similarity defined as

$$CSim(i,j) = WT(i) * Sim(i,j)\,, \tag{4}$$

the degree by which the search result is similar to the user profile. i is a node in the user topic tree ($i = 1, 2, \cdots, size(UserTopics)$). j is the associated node with a search result in the Google Directory ($j = 1, 2, \cdots, size(Results)$). $WT(i)$ is defined as $TopicCount(i)/\sum_{i=1}^{size(UserTopics)} TopicCount(i)$, weighing the degree of preferences of a node in the user topic tree. The larger the WT is, the more interested the user is in one topic. For one search result, the number of the $CSim$ values is $size(UserTopics)$ in Equation (4). One user topic tree represents one user. We define the semantic similarity between one search result and the user topic tree as the maximum value among all the values ($i = 1, 2, \cdots, size(UserTopics)$) expressed as

$$CSim^*(User, j) = Max(WT(i) * Sim(i, j))\,. \tag{5}$$

To keep our rank mechanism from missing the high quality pages in Google, Equation (5) is integrated with PageRank as

$$FinalRank(User, j) = (1 - \gamma)CSim^*(User, j) + \gamma * PageRank(j)\,. \tag{6}$$

Here γ is a parameter in [0,1] which blends the two ranking measures. The user could vary the value of γ to merge our rank mechanism and PageRank in different weights. In our experiments, γ is set to 0.5, which gives equal weight to the two measures.

4.4 Evaluation Metrics

Accuracy of User Profile. It is natural to evaluate our user profiles by computing the difference between the real user topic tree and the modelled user topic tree. Equation (2) is suitable for this task and the relative error between the two user profiles is shown as

$$Error(M) = \frac{|Sim(M, R) - Sim(R, R)|}{Sim(R, R)}\,, \tag{7}$$

where $Sim(M, R)$ is denoted by $\sum_{j=1}^{K} Max(Sim(j, i))$. R means the vector of topics in the real user topic tree. M means the vector of topics in the modelled user topic tree. i is a node in R ($i = 1, 2, \cdots, N \rightarrow size(R)$). j is a node in M ($j = 1, 2, \cdots, K \rightarrow size(M)$). A smaller value of $Error(M)$ means a higher accuracy of our modelled user profile.

Quality of Our Personalized Search System. Whether a personalized system is successful or not is determined by the user satisfaction. An effective rank mechanism should place relevant pages close to the top of the rank list. We ask the users to select the pages they considers relevant to their preferences for our evaluation. The quality of our system is measured as

$$AveRank(u, q) = \sum_{p \in S}(R(p))/Count(p)\,. \tag{8}$$

Here S denotes the set of the pages selected by user u for query q, $R(p)$is the position of page p in the ranking list, and $Count(p)$ is the number of selected pages. A smaller $AveRank$ represents better quality.

Table 2. Procedures of Evaluation Experiments

1.	Issuing the query submitted by an online user through the Google API module ;
2.	Re-ranking search results by our rank mechanism based on the current user profile and then going into the Log module;
3.	Adapting the user profile to click-history data provided by the Log module through our strategies:
4.	For the long-term model updating the structure and the degree of preferences by the "adding" operation;
5.	For the short-term model, updating web pages in the PHB by the LFUPR algorithm;
6.	If needed, degrading the long-term model according to the changes of the short-term model by the "deleting" operation.
7.	Waiting until the online user submits a new query, and then going to 1.

5 Experiments

5.1 Experimental Setup

Our rank mechanism could be combined with any search engine. In this study we choose the Google Directory Search [5] as our baseline in that Google applies its patented PageRank technology on the Google Directory to rank the sites based on their importance. It is convenient for us to combine and evaluate our rank mechanism with Google. The necessary steps are depicted in Table 2. Main modules in the experiments are listed as follows.

(1) Google API module: Given a query, we are offered titles, snippets, and page-associated Google directories beside the URLs of web pages by the Google API [6]. Here a Google directory is regarded as a topic in the user topic tree.
(2) Log module: We monitor user click behaviors, recording the query time, clicked search results, associated topics.
(3) User profile: It has been described in Section 3.

In our experiments, due to the large size of the whole Google Directory, only the top 4 levels are encoded into the user topic tree. The size of the PHB is 20 pages. Ideally if we could cache all the clicked web pages in the PHB and utilize the whole levels of the Googe Directory, it would be much easier to personalize a search.

5.2 Dataset

For each search, the Google API module got the order of the top 20 Google results due to the limited number of the Google API licenses we have. We randomized the order of the results before returning the 20 results to the user at run-time. For evaluation, 12 subjects are invited to search through our system. The 12 subjects are graduate students (5 females and 7 males) researching in several fields, i.e., computer, chemistry, food engineering, electrical engineering, art design, medical, math, architecture, and law. These subjects are divided into three types:

- Clear User, searching on queries that usually have one meaning,
- Semi-ambiguous User, searching on queries that have two or three meanings,
- Ambiguous User, searching on queries that have more than three meanings.

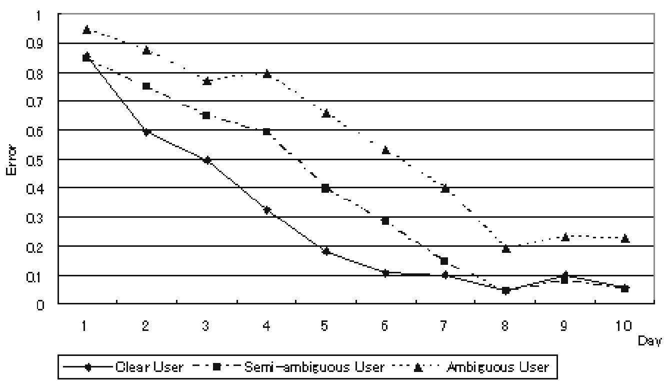

Fig. 2. Accuracy of User Profile

Our search interface was available on the Internet, and convenient for the subjects to access it at any time. They were asked to query topics closely related to their interests and majors. In the first four days, subjects input the queries on their majors, and then in the next three days the queries on their hobbies were searched. Finally, in the last three days, the subjects were required to repeat some queries done before. This repeated procedure gave a clear performance comparison between the current and earlier systems, as user profiles were updated search by search. After the data were collected over a ten-day period (From October 23nd, 2006, to November 1st, 2006), we got a log of about 300 queries averaging 25 queries per subject and about 1200 records of the pages the subjects clicked in total.

5.3 Experimental Results

Results of Accuracy of User Profile. From Figure 2, we see that as the days went on, the relative errors of our user profiles generally kept decreasing. In the last three days they even apparently stopped decreasing. The trend was expected because the subjects were asked to repeat some queries done earlier for comparison. Without a new query for a search, we are not able to learn more about the user preferences. Moreover, relative errors got even slightly larger on these days. Because the subjects might click pages different from those of the early search on the same query. This further indicates the importance of adaptation strategies to learn the changes in user preferences. Figure 2 also shows that it is easier and quicker to learn the user profile of a "Clear User" than that of a "Semi-ambiguous User" and slowest to learn the user profile of an "Ambiguous User". For example, when day=4, Error(Clear User)=0.3, Error(Semi-ambiguous User)=0.6, and Error(Ambiguous User)=0.8. Although the learning procedure of the "Ambiguous User" is slower than the other two kinds of users, as long as its user profile is converged relatively, it yields the best improvement in terms of quality among all the three kinds of users.

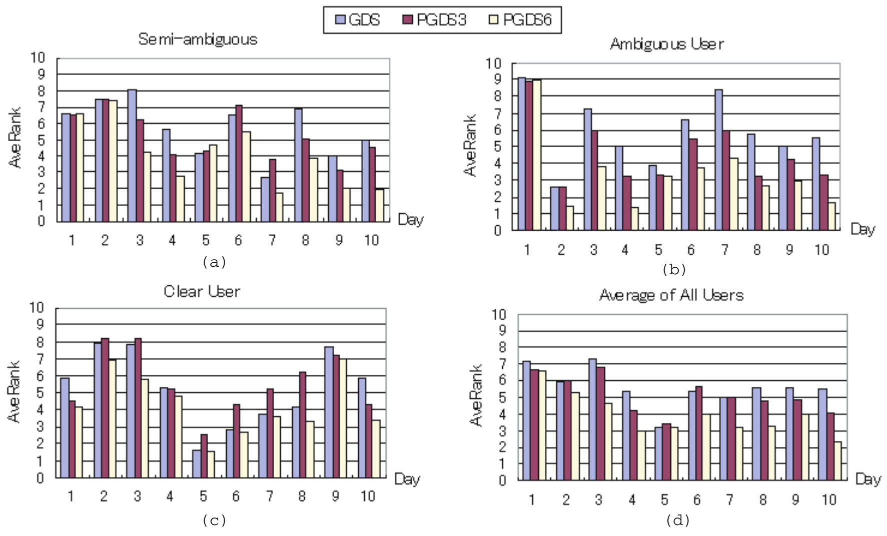

Fig. 3. Quality of Personalized Search System (Lower is better)

Results of Quality of Personalized Search System. Now, we compare the performance improvements of the following three ranking mechanisms:

- the Google Directory Search (GDS), using the Google API,
- the Personalized Google Directory Search (PGDS3), combing Equation (3) and the PageRank,
- the Personalized Google Directory Search (PGDS6), using Equation (6).

Evaluated by Equation (8), how they performed day by day is shown in Figure 3. By using the GDS as a baseline, the performance improvement of our PGDS6 in Figure 3(b) is 42.37 %, which outperforms those in both Figure 3(a) (i.e., 28.86%) and Figure 3(c)(i.e., 16.27%). The little improvement in Figure 3(c) indicates that GDS has done well with the "Clear User". However, for the "Semi-ambiguous User" and the "Ambiguous User", the significant improvements in Figure 3(a) and Figure 3(b)illustrates that GDS works worse than our strategies.

Figure 3(d) illustrates the average improvement over all users. As a result of requiring the subjects to change queries from their majors to hobbies, we see that from the fourth day to the fifth day, the values of AveRank experience a sudden increase. But after three days on learning the changes, our PGDS6 shows better results than the GDS and the PGDS3. More accurately, compared with the GDS, our PGDS6 outperforms the PGDS3 with a 60% improvement for the tenth day, while for the fifth day the improvement is only around 2%. This difference demonstrates that the changes of user preferences will lower the improvement that our strategy could achieve. Nevertheless, our rank mechanism still greatly improves over the GDS and the PGDS3 overall. The average improvements of our PGDS6 and the PGDS3 over the GDS, are 29.14% and 7.36% respectively.

6 Conclusion

In this paper we introduced how to capture the changes of user profiles from click-history data and how to use the user profiles to re-rank the search results, thus creating personalized views of the web. First, we designed independent models for short-term and long-term user preferences to consist of a user profile. Then, we adapted the user profile, including the content and the structure, to the accumulation and degradation changes of user preferences by our dynamic strategies. Finally, we proposed a novel rank mechanism to re-rank search results. Experimental results on real data demonstrate that our dynamic adaptation strategies are effective and our personalized search system performs better than the selected rank mechanisms, especially for the "Semi-ambiguous User" and the "Ambiguous User".

In the future, we plan to do some comparative experiments when the user varies the value of γ in Equation (6). In addition, when computing for the node distance in the tree, we plan to consider the edge distance, assigning a different weight for each edge, because each pair of two nodes linked by an edge has different semantic similarity. As Kelly et al. [7] summarize key papers that cover a range of approaches on implicit feedback techniques, we will study more user implicit information to construct the user profile, such as the time interval between two clicks, browsing patterns, and so on.

References

[1] D. Billsus and M. J. Pazzani. A hybrid user model for news story classification. In *Proc. of the 7th Int'l Conf. on User modeling (UM'99)*, pages 99–108, Secaucus, NJ, USA, 1999.

[2] P. A. Chirita, W. Nejdl, R. Paiu, and C. Kohlschütter. Using ODP metadata to personalize search. In *Proc. of the 28th Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR'05)*, pages 178–185, Salvador, Brazil, 2005.

[3] S. T. Dumais, E. Cutrell, J. J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. Stuff I've seen: A system for personal information retrieval and re-use. In *Proc. of the 26th Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR'03)*, pages 72–79, Toronto, Canada, 2003.

[4] P. Ferragina and A. Gulli. A personalized search engine based on web-snippet hierarchical clustering. In *Proc. of the 14th Int'l Conf. on World Wide Web - Special interest tracks and posters (WWW'06)*, pages 801–810, Chiba, Japan, 2005.

[5] Google Directory. http://directory.google.com.

[6] Google Soap Search API(Beta). http://code.google.com/apis/soapsearch.

[7] D. Kelly and J. Teevan. Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum*, 37(2):18–28, 2003.

[8] R. Kraft, C. C. Chang, F. Maghoul, and R. Kumar. Searching with context. In *Proc. of the 15th Int'l Conf. on World Wide Web (WWW'06)*, pages 477–486, Edinburgh, Scotland, UK, 2006.

[9] W. Lam, S. Mukhopadhyay, J. Mostafa, and M. J. Palakal. Detection of shifts in user interests for personalized information filtering. In *Proc. of the 19th Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR'96)*, pages 317 – 325, Zurich, Switzerland, 1996.

[10] Y. Li, Z. Bandar, and D. McLean. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Trans. Knowl. Data Eng.*, 15(4):871–882, 2003.

[11] B. Markines, L. Stoilova, and F. Menczer. Bookmark hierarchies and collaborative recommendation. In *Proc. of The 21st National Conf. on Artificial Intelligence and the 8th Innovative Applications of Artificial Intelligence Conference (AAAI'06)*, Boston, Massachusetts, USA, 2006.

[12] Open Directory Project(odp). http://dmoz.org.

[13] F. Qiu and J. Cho. Automatic identification of user interest for personalized search. In *Proc. of the 15th Int'l Conf. on World Wide Web (WWW'06)*, pages 727–736, Edinburgh, Scotland, 2006.

[14] H. rae Kim and P. K. Chan. Personalized ranking of search results with learned user interest hierarchies from bookmarks. In *Proc. of the 7th WEBKDD workshop on Knowledge Discovery from the Web (WEBKDD'05)*, pages 32–43, Chicago, Illinois, USA, 2005.

[15] V. Schickel-Zuber and B. Faltings. Inferring user's preferences using ontologies. In *Proc. of The 21st National Conf. on Artificial Intelligence and the 8th Innovative Applications of Artificial Intelligence Conference (AAAI'06)*, Boston, Massachusetts, USA, 2006.

[16] X. Shen, B. Tan, and C. Zhai. Implicit user modeling for personalized search. In *Proc. of the 2005 ACM CIKM Int'l Conf. on Information and Knowledge Management (CIKM'05)*, pages 824–831, 2005.

[17] S. J. Soltysiak and I. B. Crabtree. Automatic learning of user profiles- towards the personalisation of agent services. *BT Technology Journal*, 16(3):110–117, 1998.

[18] M. Speretta and S. Gauch. Personalized search based on user search histories. In *Proc. of the IEEE / WIC / ACM Int'l Conf. on Web Intelligence (WI'05)*, pages 622–628, Compiegne, France, 2005.

[19] J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *Proc. of the 28th Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR'05)*, pages 449–456, Salvador, Brazil, 2005.

[20] D. H. Widyantoro, T. R. Ioerger, and J. Yen. Learning user interest dynamics with a three-descriptor representation. *JASIST*, 52(3):212–225, 2001.