An Iterative Implicit Feedback Approach to Personalized Search

Yuanhua Lv [1], Le Sun [2], Junlin Zhang [2], Jian-Yun Nie [3], Wan Chen [4], and Wei Zhang [2]
[1, 2] Institute of Software, Chinese Academy of Sciences, Beijing, 100080, China
[3] University of Montreal, Canada
[1] lvyuanhua@gmail.com
[2] {sunle, junlin01, zhangwei04}@iscas.cn
[3] nie@iro.umontreal.ca [4] chenwan@nus.edu.sg

Abstract

General information retrieval systems are designed to serve all users without considering individual needs. In this paper, we propose a novel approach to personalized search. It can, in a unified way, exploit and utilize implicit feedback information, such as query logs and immediately viewed documents. Moreover, our approach can implement result re-ranking and query expansion simultaneously and collaboratively. Based on this approach, we develop a client-side personalized web search agent PAIR (Personalized Assistant for Information Retrieval), which supports both English and Chinese. Our experiments on TREC and HTRDP collections clearly show that the new approach is both effective and efficient.

1 Introduction

Analysis suggests that, while current information retrieval systems, e.g., web search engines, do a good job of retrieving results to satisfy the range of intents people have, they are not so well in discerning individuals' search goals (J. Teevan et al., 2005). Search engines encounter problems such as query ambiguity and results ordered by popularity rather than relevance to the user's individual needs.

To overcome the above problems, there have been many attempts to improve retrieval accuracy based on personalized information. Relevance Feedback (G. Salton and C. Buckley, 1990) is the main post-query method for automatically improving a system's accuracy of a user's individual need. The technique relies on explicit relevance assessments (i.e. indications of which documents contain relevant information). Relevance feedback has been proved to be quite effective for improving retrieval accuracy (G. Salton and C. Buckley, 1990; J. J. Rocchio, 1971). However, searchers may be unwilling to provide relevance information through explicitly marking relevant documents (M. Beaulieu and S. Jones, 1998).

Implicit Feedback, in which an IR system unobtrusively monitors search behavior, removes the need for the searcher to explicitly indicate which documents are relevant (M. Morita and Y. Shinoda, 1994). The technique uses implicit relevance indications, although not being as accurate as explicit feedback, is proved can be an effective substitute for explicit feedback in interactive information seeking environments (R. White et al., 2002). In this paper, we utilize the immediately viewed documents, which are the clicked results in the same query, as one type of implicit feedback information. Research shows that relative preferences derived from immediately viewed documents are reasonably accurate on average (T. Joachims et al., 2005).

Another type of implicit feedback information that we exploit is users' query logs. Anyone who uses search engines has accumulated lots of click through data, from which we can know what queries were, when queries occurred, and which search results were selected to view. These query logs provide valuable information to capture users' interests and preferences.

Both types of implicit feedback information above can be utilized to do result re-ranking and query expansion, (J. Teevan et al., 2005; Xuehua Shen. et al., 2005) which are the two general approaches to personalized search. (J. Pitkow et al., 2002) However, to the best of our knowledge, how to exploit these two types of implicit feedback in a unified way, which not only brings collaboration between query expansion and result re-ranking but also makes the whole system more concise, has so far not been well studied in the previous work. In this paper, we adopt HITS algorithm (J. Kleinberg, 1998), and propose a

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 585–592,
Sydney, July 2006. ©2006 Association for Computational Linguistics

HITS-like iterative approach addressing such a problem.

Our work differs from existing work in several aspects: (1) We propose a HITS-like iterative approach to personalized search, based on which, implicit feedback information, including immediately viewed documents and query logs, can be utilized in a unified way. (2) We implement result re-ranking and query expansion simultaneously and collaboratively triggered by every click. (3) We develop and evaluate a client-side personalized web search agent PAIR, which supports both English and Chinese.

The remaining of this paper is organized as follows. Section 2 describes our novel approach for personalized search. Section 3 provides the architecture of PAIR system and some specific techniques. Section 4 presents the details of the experiment. Section 5 discusses the previous work related to our approach. Section 6 draws some conclusions of our work.

2 Iterative Implicit Feedback Approach

We propose a HITS-like iterative approach for personalized search. HITS (Hyperlink-Induced Topic Search) algorithm, first described by (J. Kleinberg, 1998), was originally used for the detection of high-score hub and authority web pages. The Authority pages are the central web pages in the context of particular query topics. The strongest authority pages consciously do not link one another[1] — they can only be linked by some relatively anonymous hub pages. The mutual reinforcement principle of HITS states that a web page is a good authority page if it is linked by many good hub pages, and that a web page is a good hub page if it links many good authority pages. A directed graph is constructed, of which the nodes represent web pages and the directed edges represent hyperlinks. After iteratively computing based on the reinforcement principle, each node gets an authority score and a hub score.

In our approach, we exploit the relationships between documents and terms in a similar way to HITS. Unseen search results, those results which are retrieved from search engine yet not been presented to the user, are considered as "authority pages". Representative terms are considered as "hub pages". Here the representative terms are the terms extracted from and best representing the implicit feedback information. Representative terms confer a relevance score to the unseen search results — specifically, the unseen search results, which contain more good representative terms, have a higher possibility of being relevant; the representative terms should be more representative, if they occur in the unseen search results that are more likely to be relevant. Thus, also there is mutual reinforcement principle existing between representative terms and unseen search results. By the same token, we constructed a directed graph, of which the nodes indicate unseen search results and representative terms, and the directed edges represent the occurrence of the representative terms in the unseen search results. The following Table 1 shows how our approach corresponds to HITS algorithm.

Approaches	The Directed Graph		
	Nodes		Edges
HITS	Authority Pages	Hub Pages	Hyperlinks
Our Approach	Unseen Search Results	Representative Terms	Occurrence[2]

Table 1. Our approach versus HITS.

Because we have already known that the representative terms are "hub pages", and that the unseen search results are "authority pages", with respect to the former, only hub scores need to be computed; with respect to the latter, only authority scores need to be computed.

Finally, after iteratively computing based on the mutual reinforcement principle we can re-rank the unseen search results according to their authority scores, as well as select the representative terms with highest hub scores to expand the query. Below we present how to construct a directed graph to begin with.

2.1 Constructing a Directed Graph

We can view the unseen search results and the representative terms as a directed graph $G = (V, E)$. A sample directed graph is shown in Figure 1:

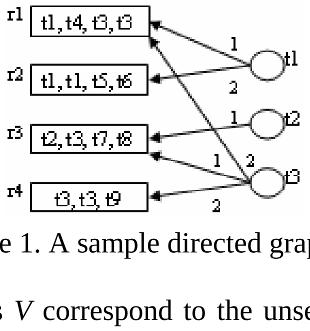


Figure 1. A sample directed graph.

The nodes V correspond to the unseen search results (the rectangles in Figure 1) and the repre-

[1] For instance, There is hardly any other company's Web page linked from "http://www.microsoft.com/"

[2] The occurrence of the representative terms in the unseen search results.

sentative terms (the circles in Figure 1); a directed edge "$p \to q \in E$" is weighed by the frequency of the occurrence of a representative term p in an unseen search result q (e.g., the number put on the edge "$t_1 \to r_2$" indicates that t_1 occurs twice in r_2). We say that each representative term only has an out-degree which is the number of the unseen search results it occurs in, as well as that each unseen search result only has an in-degree which is the count of the representative terms it contains. Based on this, we assume that the unseen search results and the representative terms respectively correspond to the authority pages and the hub pages — this assumption is used throughout the proposed algorithm.

2.2 A HITS-like Iterative Algorithm

In this section, we present how to initialize the directed graph and how to iteratively compute the authority scores and the hub scores. And then according to these scores, we show how to re-rank the unseen search results and expand the initial query.

Initially, each unseen search result of the query are considered equally authoritative, that is,

$$y^0_1 = y^0_2 \cdots = y^0_{|Y|} = 1/|Y| \tag{1}$$

Where vector Y indicates authority scores of the overall unseen search results, and $|Y|$ is the size of such a vector. Meanwhile, each representative term, with the term frequency tf_j in the history query logs that have been judged related to the current query, obtains its hub score according to the follow formulation:

$$x^0_j = tf_j \Big/ \sum_{i=1}^{|X|} tf_i \tag{2}$$

Where vector X indicates hub scores of the overall representative terms, and $|X|$ is the size of the vector X. The nodes of the directed graph are initialized in this way. Next, we associate each edge with a weight:

$$w(t_i \to r_j) = tf_{i,j} \tag{3}$$

Where $tf_{i,j}$ indicates the term frequency of the representative term t_i occurring in the unseen search result r_j; "w($t_i \to r_j$)" is the weight of edge that link from t_i to r_j. For instance, in Figure 1, w($t_1 \to r_2$) = 2.

After initialization, the iteratively computing of hub scores and authority scores starts.

The hub score of each representative term is re-computed based on three factors: the authority scores of each unseen search result where this term occurs; the occurring frequency of this term in each unseen search result; the total occurrence of every representative term in each unseen search result. The formulation for re-computing hub scores is as follows:

$$x'^{(k+1)}_i = \sum_{\forall j: t_i \to r_j} y^k_j \frac{w(t_i \to r_j)}{\sum_{\forall n: t_n \to r_j} w(t_n \to r_j)} \tag{4}$$

Where $x`_i^{(k+1)}$ is the hub score of a representative term t_i after *(k+1)th* iteration; y_j^k is the authority score of an unseen search result r_j after *kth* iteration; "$\forall j$: $t_i \to r_j$" indicates the set of all unseen search results those t_i occurs in; "$\forall n$: $t_n \to r_j$" indicates the set of all representative terms those r_j contains.

The authority score of each unseen search result is also re-computed relying on three factors: the hub scores of each representative term that this search result contains; the occurring frequency of each representative term in this search result; the total occurrence of each representative term in every unseen search results. The formulation for re-computing authority scores is as follows:

$$y'^{(k+1)}_j = \sum_{\forall i: t_i \to r_j} x^k_i \frac{w(t_i \to r_j)}{\sum_{\forall m: t_i \to r_m} w(t_i \to r_m)} \tag{5}$$

Where $y`_j^{(k+1)}$ is the authority score of an unseen search result r_j after *(k+1)th* iteration; x_i^k is the hub score of a representative term t_i after *kth* iteration; "$\forall i$: $t_i \to r_j$" indicates the set of all representative terms those r_j contains; "$\forall m$: $t_i \to r_m$" indicates the set of all unseen search results those t_i occurs in.

After re-computation, the hub scores and the authority scores are normalized to 1. The formulation for normalization is as follows:

$$y_j = \frac{y'_j}{\sum_{k=1}^{|Y|} y'_k} \text{ and } x_i = \frac{x'_i}{\sum_{k=1}^{|X|} x'_k} \tag{6}$$

The iteration, including re-computation and normalization, is repeated until the changes of the hub scores and the authority scores are smaller than some predefined threshold θ (e.g. 10^{-6}). Specifically, after each repetition, the changes in authority scores and hub scores are computed using the following formulation:

$$c = \sum_{j=1}^{|Y|} (y_j^{(k+1)} - y_j^k)^2 + \sum_{i=1}^{|x|} (x_i^{(k+1)} - x_i^k)^2 \tag{7}$$

The iteration stops if $c<\theta$. Moreover, the iteration will also stop if repetition has reached a predefined times *k* (e.g. 30). The procedure of the iteration is shown in Figure 2.

As soon as the iteration stops, the top *n* unseen search results with highest authority scores are selected and recommended to the user; the top *m* representative terms with highest hub scores are selected to expand the original query. Here *n* is a predefined number (in PAIR system we set *n*=3, *n* is given a small number because using implicit feedback information is sometimes risky.) *m* is determined according to the position of the biggest gap, that is, if $t_i - t_{i+1}$ is bigger than the gap of any other two neighboring ones of the top half representative terms, then *m* is given a value *i*. Furthermore, some of these representative terms (e.g. top 50% high score terms) will be again used in the next time of implementing the iterative algorithm together with some newly incoming terms extracted from the just now click.

```
Iterate (T, R, k, θ)
    T: a collection of m terms
    R: a collection of n search results
    k: a natural number
    θ: a predefined threshold
    Apply (1) to initialize Y.
    Apply (2) to initialize X.
    Apply (3) to initialize W.
    For i = 1, 2…, k
        Apply (4) to ($X^{i-1}$, $Y^{i-1}$) and obtain $X^{`i}$.
        Apply (5) to ($X^{i-1}$, $Y^{i-1}$) and obtain $Y^{`i}$.
        Apply (6) to Normalize $X^{`i}$ and $Y^{`i}$, and respectively
           obtain $X^{i}$ and $Y^{i}$.
        Apply (7) and obtain c.
        If $c<θ$, then break.
    End
Return (X, Y).
```

Figure 2. The HITS-like iterative algorithm.

3 Implementation

3.1 System Design

In this section, we present our experimental system PAIR, which is an IE Browser Helper Object (BHO) based on the popular Web search engine Google. PAIR has three main modules: Result Retrieval module, User Interactions module, and Iterative Algorithm module. The architecture is shown in Figure 3.

The Result Retrieval module runs in backgrounds and retrieves results from search engine. When the query has been expanded, this module will use the new keywords to continue retrieving.

The User Interactions module can handle three types of basic user actions: (1) submitting a query; (2) clicking to view a search result; (3) clicking the "Next Page" link. For each of these actions, the system responds with: (a) exploiting and extracting representative terms from implicit feedback information; (b) fetching the unseen search results via Results Retrieval module; (c) sending the representative terms and the unseen search results to Iterative Algorithm module.

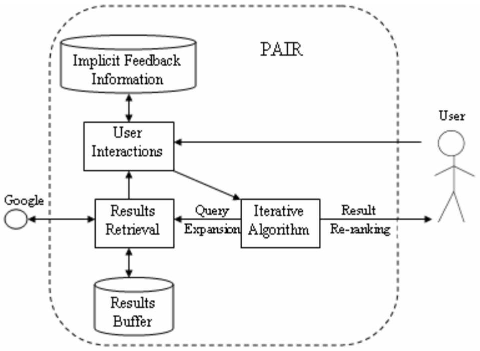


Figure 3. The architecture of PAIR.

The Iterative Algorithm module implements the HITS-like algorithm described in section 2. When this module receives data from User Interactions module, it responds with: (a) iteratively computing the hub scores and authority scores; (b) re-ranking the unseen search results and expanding the original query.

Some specific techniques for capturing and exploiting implicit feedback information are described in the following sections.

3.2 Extract Representative Terms from Query Logs

We judge whether a query log is related to the current query based on the similarity between the query log and the current query text. Here the query log is associated with all documents that the user has selected to view. The form of each query log is as follows

<query text><query time> [clicked documents]*

The "clicked documents" consist of URL, title and snippet of every clicked document. The reason why we utilize the query text of the current query but not the search results (including title, snippet, etc.) to compute the similarity, is out of consideration for efficiency. If we had used the search results to determine the similarity, the computation could only start once the search engine has returned the search results. In our method, instead, we can exploit query logs while search engine is doing retrieving. Notice that although our system only utilizes the query logs in the last 24 hours; in practice, we can exploit much more because of its low computation cost with respect to the retrieval process performed in parallel.

	Google result	PAIR result	
	query = "jaguar"	query = "jaguar" After the 4th result being clicked	query = "jaguar" "car" ∈ query logs
1	Jaguar www.jaguar.com/	Jaguar www.jaguar.com/	**Jaguar UK - Jaguar Cars** **www.jaguar.co.uk/**
2	Jaguar CA - Jaguar Cars www.jaguar.com/ca/en/	Jaguar CA - Jaguar Cars www.jaguar.com/ca/en/	**Jaguar UK - R is for…** **www.jaguar-racing.com/**
3	Jaguar Cars www.jaguarcars.com/	Jaguar Cars www.jaguarcars.com/	**Jaguar** **www.jaguar.com/**
4	Apple - Mac OS X www.apple.com/macosx/	Apple - Mac OS X www.apple.com/macosx/	Jaguar CA - Jaguar Cars www.jaguar.com/ca/en/ **-2**
5	Apple - Support … www.apple.com/support/...	**Amazon.com: Mac OS X 10.2…** **www.amazon.com/exec/obidos/...**	Jaguar Cars www.jaguarcars.com/ **-2**
6	Jaguar UK - Jaguar Cars www.jaguar.co.uk/	**Mac OS X 10.2 Jaguar…** **arstechnica.com/reviews/os…**	Apple - Mac OS X www.apple.com/macosx/ **-2**
7	Jaguar UK - R is for… www.jaguar-racing.com/	**Macworld: News: Macworld…** **maccentral.macworld.com/news/…**	Apple - Support … www.apple.com/support/... **-2**
8	Jaguar dspace.dial.pipex.com/…	Apple - Support… www.apple.com/support/... **-3**	Jaguar dspace.dial.pipex.com/…
9	Schrödinger -> Home www.schrodinger.com/	Jaguar UK - Jaguar Cars www.jaguar.co.uk/ **-3**	Schrödinger -> Home www.schrodinger.com/
10	Schrödinger -> Site Map www.schrodinger.com/...	Jaguar UK - R is for… www.jaguar-racing.com/ **-3**	Schrödinger -> Site Map www.schrodinger.com/...

Table 2. Sample results of re-ranking. The search results in boldface are the ones that our system recommends to the user. "-3" and "-2" in the right side of some results indicate the how their ranks descend.

We use the standard vector space retrieval model (G. Salton and M. J. McGill, 1983) to compute the similarity. If the similarity between any query log and the current query exceeds a predefined threshold, the query log will be considered to be related to current query. Our system will attempt to extract some (e.g. 30%) representative terms from such related query logs according to the weights computed by applying the following formulation:

$$w(t_i) = tf_i \big/ idf_i \tag{8}$$

Where tf_i and idf_i respectively are the term frequency and inverse document frequency of t_i in the clicked documents of a related query log. This formulation means that a term is more representative if it has a higher frequency as well as a broader distribution in the related query log.

3.3 Extract Representative Terms from Immediately Viewed Documents

The representative terms extracted from immediately viewed documents are determined based on three factors: term frequency in the immediately viewed document, inverse document frequency in the entire seen search results, and a discriminant value. The formulation is as follows:

$$w(x_i) = tf_{x_i}^{d_r} \times idf_{x_i}^{d_N} \times d(x_i) \tag{9}$$

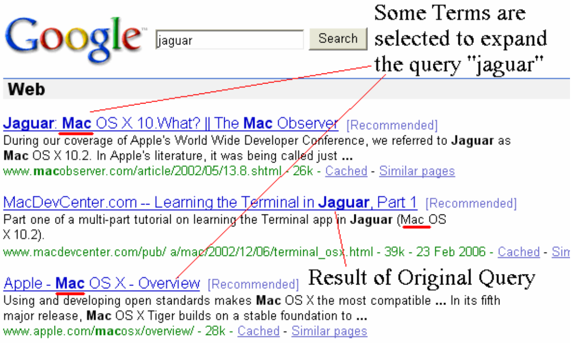
Where tf_{xi}^{dr} is the term frequency of term x_i in the viewed results set d_r; tf_{xi}^{dr} is the inverse document frequency of x_i in the entire seen results set d_N. And the discriminant value $d(x_i)$ of x_i is computed using the weighting schemes F2 (S. E. Robertson and K. Sparck Jones, 1976) as follows:

$$d(x_i) = \ln \frac{r/R}{(n-r)/(N-R)} \tag{10}$$

Where r is the number of the immediately viewed documents containing term x_i; n is the number of the seen results containing term x_i; R is the number of the immediately viewed documents in the query; N is the number of the entire seen results.

3.4 Sample Results

Unlike other systems which do result re-ranking and query expansion respectively in different ways, our system implements these two functions simultaneously and collaboratively — Query expansion provides diversified search results which must rely on the use of re-ranking to be moved forward and recommended to the user.



Jaguar: Mac OS X 10.What? || The Mac Observer [Recommended]
During our coverage of Apple's World Wide Developer Conference, we referred to Jaguar as Mac OS X 10.2. In Apple's literature, it was being called just ...
www.macobserver.com/article/2002/05/13.8.shtml - 26k - Cached - Similar pages

MacDevCenter.com -- Learning the Terminal in Jaguar, Part 1 [Recommended]
Part one of a multi-part tutorial on learning the Terminal app in Jaguar (Mac OS X 10.2).
www.macdevcenter.com/pub/ a/mac/2002/12/06/terminal_osx.html - 39k - 23 Feb 2006 - Cached - Sim

Apple - Mac OS X - Overview [Recommended]
Using and developing open standards makes Mac OS X the most compatible ... In its fifth major release, Mac OS X Tiger builds on a stable foundation to ...
www.apple.com/macosx/overview/ - 28k - Cached - Similar pages

Figure 4. A screen shot for query expansion.

After iteratively computing using our approach, the system selects some search results with top highest authority scores and recommends them to the user. In Table 2, we show that PAIR successfully re-ranks the unseen search results of "jaguar" respectively using the immediately viewed documents and the query logs. Simultaneously, some representative terms are selected to expand the original query. In the query of “jaguar” (without query logs), we click some results about “Mac OS”, and then we see that a term “Mac” has been selected to expand the original query, and some results of the new query “jaguar Mac” are recommended to the user under the help of re-ranking, as shown in Figure 4.

4 Experiment

4.1 Experimental Methodology

It is a challenge to quantitatively evaluate the potential performance improvement of the proposed approach over Google in an unbiased way (D. Hawking et al., 1999; Xuehua Shen et al., 2005). Here, we adopt a similar quantitative evaluation as what Xuehua Shen et al. (2005) do to evaluate our system PAIR and recruit 9 students who have different backgrounds to participate in our experiment. We use query topics from TREC 2005 and 2004 Hard Track, TREC 2004 Terabyte track for English information retrieval,[3] and use query topics from HTRDP 2005 Evaluation for Chinese information retrieval.[4] The reason why we utilize multiple TREC tasks rather than using a single one is that more queries are more likely to cover the most interesting topics for each participant.

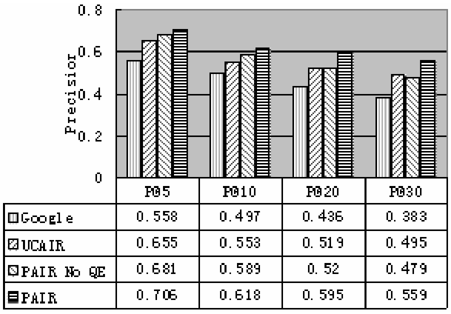
Initially, each participant would freely choose some topics (typically 5 TREC topics and 5 HTRDP topics). Each query of TREC topics will be submitted to three systems: UCAIR [5] (Xuehua Shen et al., 2005), “PAIR No QE” (PAIR system of which the query expansion function is blocked) and PAIR. Each query of HTRDP topics needs only to be submitted to “PAIR No QE” and PAIR. We do not evaluate UCAIR using HTRDP topics, since it does not support Chinese. For each query topic, the participants use the title of the topic as the initial keyword to begin with. Also they can form some other keywords by themselves if the title alone fails to describe some details of the topic. There is no limit on how many queries they must submit. During each query process, the participant may click to view some results, just as in normal web search.

Then, at the end of each query, search results from these different systems are randomly and anonymously mixed together so that every participant would not know where a result comes from. The participants would judge which of these results are relevant.

At last, we respectively measure precision at top 5, top 10, top 20 and top 30 documents of these system.

4.2 Results and Analysis

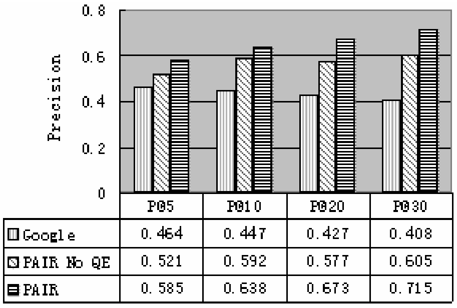
Altogether, 45 TREC topics (62 queries in all) are chosen for English information retrieval. 712 documents are judged as relevant from Google search results. The corresponding number of relevant documents from UCAIR, “PAIR No QE” and PAIR respectively is: 921, 891 and 1040. Figure 5 shows the average precision of these four systems at top n documents among such 45 TREC topics.



	P@5	P@10	P@20	P@30
□Google	0.558	0.497	0.436	0.383
▨UCAIR	0.655	0.553	0.519	0.495
▨PAIR No QE	0.681	0.589	0.52	0.479
▤PAIR	0.706	0.618	0.595	0.559

Figure 5. Average precision for TREC topics.

45 HTRDP topics (66 queries in all) are chosen for Chinese information retrieval. 809 documents are judged as relevant from Google search results. The corresponding number of relevant documents from “PAIR No QE” and PAIR respectively is: 1198 and 1416. Figure 6 shows the average precision of these three systems at top n documents among such 45 HTRDP topics.



	P@5	P@10	P@20	P@30
□Google	0.464	0.447	0.427	0.408
▨PAIR No QE	0.521	0.592	0.577	0.605
▤PAIR	0.585	0.638	0.673	0.715

Figure 6. Average precision for HTRDP topics.

PAIR and “PAIR No QE” versus Google

We can see clearly from Figure 5 and Figure 6 that the precision of PAIR is improved a lot comparing with that of Google in all measure-

[3] Text REtrieval Conference. http://trec.nist.gov/

[4] 2005 HTRDP Evaluation. http://www.863data.org.cn/

[5] The latest version released on November 11, 2005. http://sifaka.cs.uiuc.edu/ir/ucair/

ments. Moreover, the improvement scale increases from precision at top 10 to that of top 30. One explanation for this is that the more implicit feedback information generated, the more representative terms can be obtained, and thus, the iterative algorithm can perform better, leading to more precise search results. "PAIR No QE" also significantly outperforms Google in these measurements, however, with query expansion, PAIR can perform even better. Thus, we say that result re-ranking and query expansion both play an important role in PAIR.

Comparing Figure 5 with Figure 6, one can see that the improvement of PAIR versus Google in Chinese IR is even larger than that of English IR. One explanation for this is that: before implementing the iterative algorithm, each Chinese search result, including title and snippet, is segmented into words (or phrases). And only the noun, verb and adjective of these words (or phrases) are used in next stages, whereas, we only remove the stop words for English search result. Another explanation is that there are some Chinese web pages with the same content. If one of such pages is clicked, then, occasionally some repetition pages are recommended to the user. However, since PAIR is based on the search results of Google and the information concerning the result pages that PAIR can obtained is limited, which leads to it difficult to avoid the replications.

PAIR and "PAIR No QE" versus UCAIR

In Figure 5, we can see that the precision of "PAIR No QE" is better than that of UCAIR among top 5 and top 10 documents, and is almost the same as that of UCAIR among top 20 and top 30 documents. However, PAIR is much better than UCAIR in all measurements. This indicates that result re-ranking fails to do its best without query expansion, since the relevant documents in original query are limited, and only the re-ranking method alone cannot solve the "relevant documents sparseness" problem. Thus, the query expansion method, which can provide fresh and relevant documents, can help the re-ranking method to reach an even better performance.

Efficiency of PAIR

The iteration statistic in evaluation indicates that the average iteration times of our approach is 22 before convergence on condition that we set the threshold $\theta = 10^{-6}$. The experiment shows that the computation time of the proposed approach is imperceptible for users (less than 1ms.)

5 Related Work

There have been many prior attempts to personalized search. In this paper, we focus on the related work doing personalized search based on implicit feedback information.

Some of the existing studies capture users' information need by exploiting query logs. For example, M. Speretta and S. Gauch (2005) build user profiles based on activity at the search site and study the use of these profiles to provide personalized search results. F. Liu et al. (2002) learn user's favorite categories from his query history. Their system maps the input query to a set of interesting categories based on the user profile and confines the search domain to these categories. Some studies improve retrieval performance by exploiting users' browsing history (F. Tanudjaja and L. Mu, 2002; M. Morita and Y. Shinoda, 1994) or Web communities (A. Kritikopoulos and M. Sideri, 2003; K. Sugiyama et al., 2004) Some studies utilize client side interactions, for example, K. Bharat (2000) automatically discovers related material on behalf of the user by serving as an intermediary between the user and information retrieval systems. His system observes users interacting with everyday applications and then anticipates their information needs using a model of the task at hand. Some latest studies combine several types of implicit feedback information. J. Teevan et al. (2005) explore rich models of user interests, which are built from both search-related information, such as previously issued queries and previously visited Web pages, and other information about the user such as documents and email the user has read and created. This information is used to re-rank Web search results within a relevance feedback framework.

Our work is partly inspired by the study of Xuehua Shen et al. (2005), which is closely related to ours in that they also exploit immediately viewed documents and short-term history queries, implement query expansion and re-ranking, and develop a client-side web search agents that perform eager implicit feedback. However, their work differs from ours in three ways: First, they use the cosine similarity to implement query expansion, and use Rocchio formulation (J. J. Rocchio, 1971) to re-rank the search results. Thus, their query expansion and re-ranking are computed separately and are not so concise and collaborative. Secondly, their query expansion is based only on the past queries and is implemented before the query, which leads to that their query expansion does not benefit from user's click through data. Thirdly, they do not compute the relevance of search results and the relativity of expanded terms in an iterative fashion. Thus, their approach does not utilize the relation among search results, among expanded terms, and between search results and expanded terms.

6 Conclusions

In this paper, we studied how to exploit implicit feedback information to improve retrieval accuracy. Unlike most previous work, we propose a novel HITS-like iterative algorithm that can make use of query logs and immediately viewed documents in a unified way, which not only brings collaboration between query expansion and result re-ranking but also makes the whole system more concise. We further propose some specific techniques to capture and exploit these two types of implicit feedback information. Using these techniques, we develop a client-side web search agent PAIR. Experiments in English and Chinese collections show that our approach is both effective and efficient.

However, there is still room to improve the performance of the proposed approach, such as exploiting other types of personalized information, choosing some more effective strategies to extract representative terms, studying the effects of the parameters used in the approach, etc.

Acknowledgement

We would like to thank the anonymous reviewers for their helpful feedback and corrections, and to the nine participants of our evaluation experiments. Additionally, this work is supported by the National Science Fund of China under contact 60203007.

References

A. Kritikopoulos and M. Sideri, 2003. The Compass Filter: Search engine result personalization using Web communities. In *Proceedings of ITWP*, pages 229-240.

D. Hawking, N. Craswell, P.B. Thistlewaite, and D. Harman, 1999. Results and challenges in web search evaluation. *Computer Networks*, 31(11-16):1321–1330.

F. Liu, C. Yu, and W. Meng, 2002. Personalized web search by mapping user queries to categories. In *Proceedings of CIKM*, pages 558-565.

F. Tanudjaja and L. Mu, 2002. Persona: a contextualized and personalized web search. *HICSS*.

G. Salton and M. J. McGill, 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill.

G. Salton and C. Buckley, 1990. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288-297.

J. J. Rocchio, 1971. Relevance feedback in information retrieval. In *The SMART Retrieval System : Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall Inc.

J. Kleinberg, 1998. Authoritative sources in a hyperlinked environment. *ACM*, 46(5):604–632.

J. Pitkow, H. Schutze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, and T. Breuel, 2002. Personalized search. *Communications of the ACM*, 45(9):50-55.

J. Teevan, S. T. Dumais, and E. Horvitz, 2005. Personalizing search via automated analysis of interests and activities. In *Proceedings of SIGIR*, pages 449-456.

K. Bharat, 2000. SearchPad: Explicit capture of search context to support Web search. *Computer Networks*, 33(1-6): 493-501.

K. Sugiyama, K. Hatano, and M. Yoshikawa, 2004. Adaptive Web search based on user profile constructed without any effort from user. In *Proceedings of WWW*, pages 675-684.

M. Beaulieu and S. Jones, 1998. Interactive searching and interface issues in the okapi best match retrieval system. *Interacting with Computers*, 10(3):237-248.

M. Morita and Y. Shinoda, 1994. Information filtering based on user behavior analysis and best match text retrieval. In *Proceedings of SIGIR*, pages 272–281.

M. Speretta and S. Gauch, 2005. Personalizing search based on user search history. *Web Intelligence*, pages 622-628.

R. White, I. Ruthven, and J. M. Jose, 2002. The use of implicit evidence for relevance feedback in web retrieval. In *Proceedings of ECIR*, pages 93–109.

S. E. Robertson and K. Sparck Jones, 1976. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129-146.

T. Joachims, L. Granka, B. Pang, H. Hembrooke, and G. Gay, 2005. Accurately Interpreting Clickthrough Data as Implicit Feedback, In *Proceedings of SIGIR*, pages 154-161.

Xuehua Shen, Bin Tan, and Chengxiang Zhai, 2005. Implicit User Modeling for Personalized Search. In *Proceedings of CIKM*, pages 824-831.