Implicit User Modeling for Personalized Search

Xuehua Shen, Bin Tan, ChengXiang Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

ABSTRACT

Information retrieval systems (e.g., web search engines) are critical for overcoming information overload. A major deficiency of existing retrieval systems is that they generally lack user modeling and are not adaptive to individual users, resulting in inherently non-optimal retrieval performance. For example, a tourist and a programmer may use the same word "java" to search for different information, but the current search systems would return the same results. In this paper, we study how to infer a user's interest from the user's search context and use the inferred implicit user model for personalized search . We present a decision theoretic framework and develop techniques for implicit user modeling in information retrieval. We develop an intelligent client-side web search agent (UCAIR) that can perform eager implicit feedback, e.g., query expansion based on previous queries and immediate result reranking based on clickthrough information. Experiments on web search show that our search agent can improve search accuracy over the popular Google search engine.

Keywords

implicit feedback, personalized search, user model, interactive retrieval

1. INTRODUCTION

Although many information retrieval systems (e.g., web search engines and digital library systems) have been successfully deployed, the current retrieval systems are far from optimal. A major deficiency of existing retrieval systems is that they generally lack user modeling and are not adaptive to individual users [17]. This inherent non-optimality is seen clearly in the following two cases: (1) Different users may use exactly the same query (e.g., "Java") to search for different information (e.g., the Java island in Indonesia or the Java programming language), but existing IR systems return the same results for these users. Without considering the actual user, it is impossible to know which sense "Java" refers to in a query. (2) A user's information needs may change over time. The same user may use "Java" sometimes to mean the Java island in Indonesia and some other times to mean the programming language. Without recognizing the search context, it would be again impossible to recognize the correct sense.

In order to optimize retrieval accuracy, we clearly need to model the user appropriately and personalize search according to each individual user. The major goal of user modeling for information retrieval is to accurately model a user's information need, which is, unfortunately, a very difficult task. Indeed, it is even hard for a user to precisely describe what exactly is his/her information need.

What information is available for a system to infer a user's information need? Obviously, the user's query (often a few keywords) provides the most direct evidence. Indeed, most existing retrieval systems rely solely on the query to model a user's information need. However, since a query is often extremely short, the user model constructed based on a keyword query is inevitably impoverished . An effective way to improve user modeling in information retrieval is to ask the user to *explicitly* specify which documents are relevant (i.e., useful for satisfying his/her information need), and then to improve user modeling based on such examples of relevant documents. This is called *relevance feedback*, which has been proved to be quite effective for improving retrieval accuracy [19, 20]. Unfortunately, in real world applications, users are usually reluctant to make the extra effort to provide relevant examples for feedback [11].

It is thus very interesting to study how to infer a user's information need based on any *implicit* feedback information, which naturally exists through user interactions and thus does not require any extra user effort. Indeed, several previous studies have shown that implicit user modeling can improve retrieval accuracy. In [3], a web browser *(Curious Browser)* is developed to record a user's explicit relevance ratings of web pages (relevance feedback) and browsing behavior when viewing a page, such as dwelling time, mouse click, mouse movement and scrolling (implicit feedback). It is shown that the dwelling time on a page, amount of scrolling on a page and the combination of time and scrolling have a strong correlation with explicit relevance ratings, which suggests that implicit feedback may be helpful for inferring user information need. In [10], user clickthrough data is collected as training data to learn a retrieval function, which is used to produce a customized ranking of search results that suits a group of users' preferences. In [25], the clickthrough data collected over a long time period is exploited through query expansion to improve retrieval accuracy.

While a user may have general long term interests and preferences for information, often he/she is searching for documents to satisfy an "ad hoc" information need, which only lasts for a short period of time; once the information need is satisfied, the user would generally no longer be interested in such information. For example, a user may be looking for information about used cars

in order to buy one, but once the user has bought a car, he/she is generally no longer interested in such information. In such cases, implicit feedback information collected over a long period of time is unlikely very useful, but the immediate search context and feedback information, such as which of the search results for the current information need are viewed, can be expected to be much more useful. Consider the query "Java" again. Any of the following immediate feedback information about the user could potentially help determine the intended meaning of "Java" in the query: (1) The previous query submitted by the user is "hashtable" (as opposed to, e.g., "travel Indonesia"). (2) In the search results, the user viewed a page where words such as "programming", "software", and "applet" occur many times.

To the best of our knowledge, how to exploit such immediate and short-term search context to improve search has so far not been well addressed in the previous work. In this paper, we study how to construct and update a user model based on the *immediate* search context and implicit feedback information and use the model to improve the accuracy of *ad hoc* retrieval. In order to *maximally* benefit the user of a retrieval system through implicit user modeling, we propose to perform "eager implicit feedback". That is, as soon as we observe any new piece of evidence from the user, we would update the system's belief about the user's information need and respond with improved retrieval results based on the updated user model. We present a decision-theoretic framework for optimizing interactive information retrieval based on eager user model updating, in which the system responds to every action of the user by choosing a system action to optimize a utility function. In a traditional retrieval paradigm, the retrieval problem is often to match a query with documents and rank documents according to their relevance values. As a result, the whole retrieval process is a simple independent cycle of "query" and "result display". In the proposed new retrieval paradigm, the user's search context plays an important role and the inferred implicit user model is exploited immediately to benefit the user, even for browsing the results. The new retrieval paradigm is thus fundamentally different from the traditional paradigm, and is inherently more general.

We further propose specific techniques to capture and exploit two types of implicit feedback information: (1) identifying related immediately preceding query and using the query and the corresponding search results to select appropriate terms to expand the current query, and (2) exploiting the viewed document summaries to immediately rerank any documents that have not yet been seen by the user. Using these techniques, we develop a client-side web search agent UCAIR (User-Centered Adaptive Information Retrieval) on top of a popular search engine (Google). Experiments on web search show that our search agent can improve search accuracy over Google. Since the implicit information we exploit already naturally exists through user interactions, the user does not need to make any extra effort. Thus the developed search agent can improve existing web search performance without additional effort from the user.

The remaining sections are organized as follows. In Section 2, we discuss the related work. In Section 3, we present a decision-theoretic interactive retrieval framework for implicit user modeling. In Section 4, we present the design and implementation of an intelligent client-side web search agent (UCAIR) that performs eager implicit feedback. In Section 5, we report our experiment results using the search agent. Section 6 concludes our work.

2. RELATED WORK

Implicit user modeling for personalized search has been studied in previous work, but our work differs from all previous work in several aspects: (1) We emphasize the exploitation of *immediate* search context such as the related immediately preceding query and the viewed documents in the same query session, while most previous work relies on long-term collection of implicit feedback information [25]. (2) We perform eager feedback and bring the benefit of implicit user modeling as soon as any new implicit feedback information is available, while the previous work mostly exploits long-term implicit feedback [10]. (3) We propose a retrieval framework to integrate implicit user modeling with the interactive retrieval process, while the previous work either studies implicit user modeling separately from retrieval [3] or only studies specific retrieval models for exploiting implicit feedback to better match a query with documents [23, 27, 22]. (4) We develop and evaluate a personalized Web search agent with online user studies, while most existing work evaluates algorithms offline without real user interactions.

Currently some search engines provide rudimentary personalization, such as Google Personalized web search [6], which allows users to explicitly describe their interests by selecting from predefined topics, so that those results that match their interests are brought to the top, and My Yahoo! search [16], which gives users the option to save web sites they like and block those they dislike. In contrast, UCAIR personalizes web search through implicit user modeling without any additional user efforts. Furthermore, the personalization of UCAIR is provided on the client side. There are two remarkable advantages on this. First, the user does not need to worry about the privacy infringement, which is a big concern for personalized search [26]. Second, both the computation of personalization and the storage of the user profile are done at the client side so that the server load is reduced dramatically [9].

There have been many works studying user query logs [1] or query dynamics [13]. UCAIR makes direct use of a user's query history to benefit the same user *immediately* in the same search session. UCAIR first judges whether two neighboring queries belong to the same information session and if so, it selects terms from the previous query to perform query expansion.

Our query expansion approach is similar to automatic query expansion [28, 15, 5], but instead of using pseudo feedback to expand the query, we use user's implicit feedback information to expand the current query. These two techniques may be combined.

3. OPTIMIZATION IN INTERACTIVE IR

In interactive IR, a user interacts with the retrieval system through an "action dialogue", in which the system responds to each user action with some system action. For example, the user's action may be submitting a query and the system's response may be returning a list of 10 document summaries. In general, the space of user actions and system responses and their granularities would depend on the interface of a particular retrieval system.

In principle, every action of the user can potentially provide new evidence to help the system better infer the user's information need. Thus in order to respond optimally, the system should use *all* the evidence collected so far about the user when choosing a response. When viewed in this way, most existing search engines are clearly non-optimal. For example, if a user has viewed some documents on the first page of search results, when the user clicks on the "Next" link to fetch more results, an existing retrieval system would still return the next page of results retrieved based on the original query without considering the new evidence that a particular result has been viewed by the user.

We propose to optimize retrieval performance by adapting system responses based on *every* action that a user has taken, and cast the optimization problem as a decision task. Specifically, at any time, the system would attempt to do two tasks: (1) User model

updating: Monitor any useful evidence from the user regarding his/her information need and update the user model as soon as such evidence is available; (2) Improving search results: Rerank immediately all the documents that the user has not yet seen, as soon as the user model is updated. We emphasize eager updating and reranking, which makes our work quite different from any existing work. Below we present a formal decision theoretic framework for optimizing retrieval performance through implicit user modeling in interactive information retrieval.

3.1 A decision-theoretic framework

Let $\mathcal{A}$ be the set of all user actions and $\mathcal{R}(a)$ be the set of all possible system responses to a user action $a \in \mathcal{A}$. At any time, let $A_t = (a_1, ..., a_t)$ be the observed sequence of user actions so far (up to time point t) and $R_{t-1} = (r_1, ..., r_{t-1})$ be the responses that the system has made responding to the user actions. The system's goal is to choose an optimal response $r_t \in \mathcal{R}(a_t)$ for the current user action a_t.

In a retrieval system, the most important factor affecting the optimality of the system's response is naturally how well the response addresses the user's information need. Indeed, at any time, we may assume that the system has some "belief" about what the user is interested in, which we model through a term vector $\vec{x} = (x_1, ..., x_{|V|})$, where $V = \{w_1, ..., w_{|V|}\}$ is the set of all terms (i.e., vocabulary) and x_i is the weight of term w_i. Such a term vector is commonly used in information retrieval to represent both queries and documents. For example, the vector-space model, assumes that both the query and the documents are represented as term vectors and the score of a document with respect to a query is computed based on the similarity between the query vector and the document vector [21]. In a language modeling approach, we may also regard the query unigram language model [12, 29] or the relevance model [14] as a term vector representation of the user's information need. Intuitively, $\vec{x}$ would assign high weights to terms that characterize the topics which the user is interested in.

Another component in our user model is the documents that the user has already viewed. Obviously, even if a document is relevant, if the user has already seen the document, it would not be useful to present the same document again. We thus introduce another variable $S \subset \mathcal{D}$ ($\mathcal{D}$ is the whole set of documents in the collection) to denote the subset of documents in the search results that the user has already seen/viewed. In general, at time t, we may represent a user model as $\mathcal{M} = (S, \vec{x}, A_t, R_{t-1})$, where S is the seen documents and $\vec{x}$ is the system's "understanding" of the user's information need.

To case choosing an optimal system response for any user action a as a statistical decision problem, we introduce a loss function $L(a, r, \mathcal{M}) \in \Re$, defined on the space of responses (i.e., $r \in \mathcal{R}(a)$) and user models. The loss function encodes our decision preferences and assesses the optimality of responding with r when the current user model is $\mathcal{M}$ (in particular, the current belief of the user's information need is $\vec{x}$) and the current user action is a. Since we can never be sure about the user's information need $\vec{x}$, we treat $\vec{x}$ and $\mathcal{M}$ both as random variables. With such a set up, according to Bayesian decision theory, the optimal decision at time t is to choose a response that minimizes the Bayes risk, i.e.,

$$r_t^* = argmin_{r \in \mathcal{R}(a_t)} \int_{\mathcal{M}} L(a_t, r, \mathcal{M}) P(\mathcal{M}|U, \mathcal{D}, A_t, R_{t-1}) \mathrm{d}\mathcal{M} \quad (1)$$

where $P(\mathcal{M}|U, \mathcal{D}, A_t, R_{t-1})$ is the posterior probability of the user model given all the observations about the user U we have made up to time t. The variable U refers to any user factors that we want to model (e.g., readability).

Leaving aside how to define and estimate these probabilistic models and the loss function, we can see that such a decision-theoretic formulation suggests that an optimal retrieval system should update its belief about the user model (i.e., $P(\mathcal{M}|U, \mathcal{D}, A_t, R_{t-1})$) in response to *every* user action and always choose a response based on the most current belief. For example, when a user clicks on a document link to view its content, the action should trigger an update on the system's model about the user's information need, based on the assumption that the displayed information about this document is attractive to the user and thus is indicative of the user's information need.

To simplify the computation of Equation 1, let us assume that the posterior probability mass $P(\mathcal{M}|U, \mathcal{D}, A_t, R_{t-1})$ is mostly concentrated on the mode $\mathcal{M}^* = argmax_{\mathcal{M}} P(\mathcal{M}|U, \mathcal{D}, A_t, R_{t-1})$. We can then approximate the integral with the value of the loss function at $\mathcal{M}^*$. That is,

$$r_t^* \approx argmin_{r \in \mathcal{R}(a_t)} L(a_t, r, \mathcal{M}^*) \quad (2)$$

$$= argmin_{r \in \mathcal{R}(a_t)} L(a_t, r, S, \vec{x}^*, A_t, R_{t-1}) \quad (3)$$

where $\vec{x}^* = argmax_{\vec{x}} P(\vec{x}|U, \mathcal{D}, A_t, R_{t-1})$.

Thus, the decision theoretic framework suggests that, in order to choose the optimal response to a_t, the system performs two tasks: (1) compute the current user information model and obtain $\vec{x}^*$ based on all the useful information. (2) choose a response r_t to minimize the loss function value $L(a_t, r_t, S, \vec{x}^*, A_t, R_{t-1})$. Note that our framework is quite general since we can potentially model any kind of user actions and system responses. In most cases, as we may expect, the system's response is some ranking of documents, i.e., for most actions a, $\mathcal{R}(a)$ consists of all the possible rankings of the unseen documents, and the decision problem boils down to choosing the best ranking of unseen documents based on the most current user model. When a is the action of submitting a keyword query, such a response is exactly what a current retrieval system would do. However, we can easily imagine that a more intelligent web search engine would respond to a user's clicking of the "Next" link (to fetch more unseen results) with a more optimized ranking of documents based on any viewed documents in the current page of results. In fact, according to our eager updating strategy, we may even allow a system to respond to a user's clicking of browser's "Back" button after viewing a document in the same way, so that the user can maximally benefit from implicit feedback. These are exactly what our UCAIR system does.

3.2 Loss functions

The exact definition of loss function L depends on the responses, thus is inevitably application-specific. We now briefly discuss some possibilities when the response is to rank all the unseen documents and present the top k of them. Let $r = (d_1, ..., d_k)$ be the top k documents, S be the set of seen documents by the user, and $\vec{x}^*$ be the system's best guess of the user's information need. We may simply define the loss associated with r as the negative sum of the probability that each of the d_i is relevant, i.e., $L(a, r, \mathcal{M}) = -\sum_{i=1}^{k} P(relevant|d_i)$. Clearly, in order to minimize this loss function, the optimal response r would contain the k documents with the highest probability of relevance, which is intuitively reasonable.

One deficiency of this "top-k loss function" is that it is not sensitive to the internal order of the selected top k documents, so switching the ranking order of a non-relevant document and a relevant ones would not affect the loss, which is unreasonable. To model ranking, we can introduce a factor of the user model – the probability of each of the k documents being viewed by the user,

$P(view|d_i)$, and define the following "ranking loss function":

$$L(a, r, \mathcal{M}) = -\sum_{i=1}^{k} P(view|d_i)P(relevant|d_i)$$

Since in general, if d_i is ranked above d_j (i.e., $i < j$), $P(view|d_i) > P(view|d_j)$, this loss function would favor a decision to rank relevant documents above non-relevant ones, as otherwise, we could always switch d_i with d_j to reduce the loss value. Thus the system should simply perform a regular retrieval and rank documents according to the probability of relevance [18].

Depending on the user's retrieval preferences, there can be many other possibilities. For example, if the user does not want to see redundant documents, the loss function should include some redundancy measure on r based on the already seen documents S.

Of course, when the response is not to choose a ranked list of documents, we would need a different loss function. We discuss one such example that is relevant to the search agent that we implement. When a user enters a query q_t (current action), our search agent would attempt to expand the query (i.e., adding new words to the query) based on the preceding query q_{t-1} in case q_{t-1} and q_t are related. In this case, a response r can be any subset of terms T in our vocabulary V. Note that the ultimate system response would be documents retrieved using the expanded query, but if our search system relies on some standard algorithm/search engine to actually carry out search, the system's major decision would be really on the choice of terms for query expansion.

Our loss function thus should be defined on T. One possibility is

$$\begin{aligned} L(a, r, \mathcal{M}) &= L(q_t, T, S, \vec{x}^*, A_t, R_{t-1}) \\ &= -\delta(related(q_{t-1}, q_t)) \times \\ &\quad \sum_{i=1}^{|V|} \delta(freq(t_i, r_{t-1}) * freq(t_i, r'_t) > \theta) \\ &\quad +(1 - \delta(related(q_{t-1}, q_t)))|T| \end{aligned}$$

where $\delta(x)$ is an indicator function, which is equal to 1 if x is true and 0 otherwise. $freq(t_i, r_{t-1})$ is the frequency of term t_i in the results for the previous query and $freq(t_i, r'_t)$ is the frequency of term t_i in the "tentative" retrieval results for the current query q_t *without* query expansion. θ is a frequency threshold. This loss function basically says that if the two queries are related, then the loss is smaller if we add to T a term t_i which has a high frequency in both the previous query results and the tentative results for the current query, whereas if the two queries are unrelated, then the loss is the smallest if we let T be the set of terms in q_t . It is not hard to see that according to this loss function, the optimal decision rule of choosing the subset of terms to be added to q_t would be to first decide whether the two queries are related. If they are, add the overlapping terms in the retrieval results of the two queries to T, but if they are not, do not add any term to T. Whether the two queries are related can be decided based on some standard retrieval formula, which essentially matches each query's retrieval results to see if their similarity is sufficiently high.

While this loss function is fairly heuristic, it shows the possibility of using our framework to model different kinds of responses.

3.3 Implicit user modeling

Implicit user modeling is captured in our framework through the computation of $\vec{x}^* = argmax_{\vec{x}} P(\vec{x}|U, \mathcal{D}, A_t, R_{t-1})$, i.e., the system's current belief of what the user's information need is. Here again there may be many possibilities, leading to different algorithms for implicit user modeling. We now discuss a few of them.

First, when two consecutive queries are related, the previous query can be exploited to enrich the current query and provide more search context to help disambiguation. For this purpose, instead of performing query expansion as we did in the previous section, we could also compute an updated $\vec{x}^*$ based on the previous query and retrieval results. The computed new user model can then be used to rank the documents with a standard information retrieval model.

Second, we can also infer a user's interest based on the summaries of the viewed documents. When a user is presented with a list of summaries of top ranked documents, if the user chooses to skip the first n documents and to view the $(n+1)$-th document, we may infer that the user is not interested in the displayed summaries for the first n documents, but is attracted by the displayed summary of the $(n+1)$-th document. We can thus use these summaries as negative and positive examples to learn a more accurate user model $\vec{x}^*$. Here many standard relevance feedback techniques can be exploited [19, 20]. Note that we should use the displayed summaries, as opposed to the actual contents of those documents, since it is possible that the displayed summary of the viewed document is relevant, but the document content is actually not. Similarly, a displayed summary may mislead a user to skip a relevant document. Inferring user models based on such displayed information, rather than the actual content of a document is an important difference between UCAIR and some other similar systems.

In UCAIR, both of these strategies for inferring an implicit user model are implemented.

4. UCAIR: A PERSONALIZED SEARCH AGENT

4.1 Design

In this section, we present a client-side web search agent called UCAIR, in which we implement some of the methods we discussed in the previous section for performing personalized search through implicit user modeling. UCAIR is a web browser plug-in that acts as a proxy for web search engines. Currently, it is only implemented for Internet Explorer and Google, but it is a matter of engineering to make it running on other web browsers and interact with other search engines.

The issue of privacy is a primary obstacle for deploying any real world applications involving serious user modeling, such as personalized search. For this reason, UCAIR is strictly running as a client-side search agent, as opposed to server-side software. This way, the captured user information always resides on the computer that the user is using, thus the user does not need to release any information to the outside. Client-side personalization also allows the system to easily observe a lot of user information that may not be easily available to a server. Furthermore, performing personalized search on the client-side is more scalable than on the server-side, since the overhead of computation and storage is distributed among clients.

As shown in Figure 1, the UCAIR toolbar has 3 major components: (1) The (implicit) user modeling module captures a user's search context and history information, including the submitted queries and any clicked search results and infers search session boundaries. (2) The query modification module selectively improves the query formulation according to the current user model. (3) The result re-ranking module re-ranks any unseen search results immediately whenever the user model is updated.

In UCAIR, we consider four basic user actions: (1) submitting a keyword query; (2) viewing a document; (3) clicking the "Back" button; (4) clicking the "Next" link on a result page. For each

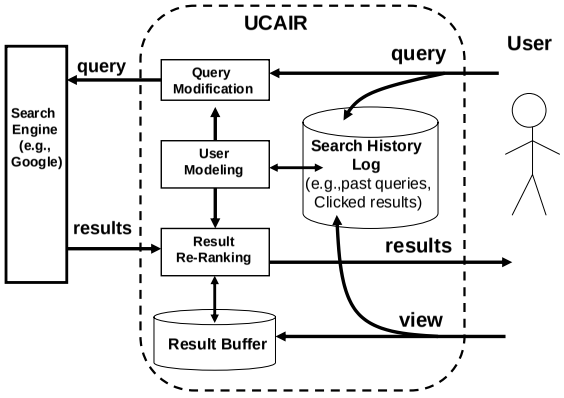

Figure 1: UCAIR architecture

of these four actions, the system responds with, respectively, (1) generating a ranked list of results by sending a possibly expanded query to a search engine; (2) updating the information need model $\vec{x}$; (3) reranking the unseen results on the current result page based on the current model $\vec{x}$; and (4) reranking the unseen pages and generating the next page of results based on the current model $\vec{x}$.

Behind these responses, there are three basic tasks: (1) Decide whether the previous query is related to the current query and if so expand the current query with useful terms from the previous query or the results of the previous query. (2) Update the information need model $\vec{x}$ based on a newly clicked document summary. (3) Rerank a set of unseen documents based on the current model $\vec{x}$. Below we describe our algorithms for each of them.

4.2 Session Boundary Detection and Query Expansion

To effectively exploit previous queries and their corresponding clickthrough information, UCAIR needs to judge whether two adjacent queries belong to the same search session (i.e., detect session boundaries). Existing work on session boundary detection is mostly in the context of web log analysis (e.g., [8]), and uses statistical information rather than textual features. Since our client-side agent does not have access to server query logs, we make session boundary desions based on textual similarity between two queries. Because related queries do not necessarily share the same words (e.g., "java island" and "travel Indonesia"), it is insufficient to use only query text. Therefore we use the search results of the two queries to help decide whether they are topically related. For example, for the above queries "java island" and "travel Indonesia"', the words "java", "bali", "island", "indonesia" and "travel" may occur frequently in both queries' search results, yielding a high similarity score.

We only use the titles and summaries of the search results to calculate the similarity since they are available in a retrieved search result page and fetching the full text of every result page would significantly slow down the process. To compensate for the terseness of titles and summaries, we retrieve more results than a user would normally view for the purpose of detecting session boundaries (typically 50 results).

The similarity between the previous query q' and the current query q is computed as follows. Let $\{s'_1, s'_2, \ldots, s'_{n'}\}$ and $\{s_1, s_2, \ldots, s_n\}$ be the result sets for the two queries. We use the pivoted normalization TF-IDF weighting formula [24] to compute a term weight vector $\vec{s}_i$ for each result s_i. We define the *average result* $\vec{s}_{avg}$ to be the centroid of all the result vectors, i.e., $(\vec{s}_1 + \vec{s}_2 + \ldots + \vec{s}_n)/n$. The cosine similarity between the two average results is calculated as

$$\vec{s'}_{avg} \cdot \vec{s}_{avg} / \sqrt{\vec{s'}^2_{avg} \cdot \vec{s}^2_{avg}}$$

If the similarity value exceeds a predefined threshold, the two queries will be considered to be in the same information session.

If the previous query and the current query are found to belong to the same search session, UCAIR would attemp to expand the current query with terms from the previous query and its search results. Specifically, for each term in the previous query or the corresponding search results, if its frequency in the results of the current query is greater than a preset threshold (e.g. 5 results out of 50), then the term would be added to the current query to form an expanded query. In this case, UCAIR would send this expanded query rather than the original one to the search engine and return the results corresponding to the expanded query. Currently, UCAIR only uses the immediate preceding query for query expansion; in principle, we could exploit all related past queries.

4.3 Information need model updating

Suppose at time t, we have observed that the user has viewed k documents whose summaries are $s_1, ..., s_k$. We update our user model by computing a new information need vector with a standard feedback method in information retrieval (i.e., Rocchio [19]). According to the vector space retrieval model, each clicked summary c_i can be represented by a term weight vector $\vec{c}_i$ with each term weighted by a TF-IDF weighting formula [21]. Rocchio computes the centroid vector of all the summaries and interpolates it with the original query vector to obtain an updated term vector. That is,

$$\vec{x} = \alpha\vec{q} + (1-\alpha)\frac{1}{k}\sum_{i=1}^{k}\vec{c}_i$$

where $\vec{q}$ is the query vector and α is a parameter that controls the influence of the clicked summaries on the inferred information need model. In our experiments, α is set to 0.5. Note that we update the information need model whenever the user views a document.

4.4 Result Reranking

In general, we want to rerank all the unseen results as soon as the user model is updated. Currently, UCAIR implements reranking in two cases, corresponding to when a user clicks on the "Back" and "Next" buttons in the Internet Explorer. In both cases, the current (updated) user model would be used to rerank the unseen results so that the user would see improved search results immediately.

To rerank any unseen document summaries, UCAIR uses the standard vector space retrieval model and scores each summary based on the similarity of the result and the current user information need vector $\vec{x}$ [21]. Since implicit feedback is not completely reliable, we bring up only a small number (e.g. 5) of highest reranked results to be followed by any originally high ranked results.

5. EVALUATION OF UCAIR

We now present some results on evaluating the two major UCAIR functions: selective query expansion and result reranking based on user clickthrough data.

5.1 Sample results

The query expansion strategy implemented in UCAIR is intentionally conservative to avoid misinterpretation of implicit user models. In practice, whenever it chooses to expand the query, the expansion usually makes sense. In Table 1, we show how UCAIR can successfully distinguish two different search contexts for the query

	Google result (user query = "java map")	UCAIR result (user query ="java map")	
		previous query = "travel Indonesia" expanded user query = "java map Indonesia"	previous query = "hashtable" expanded user query = "java map class"
1	Java map projections of the world ... www.btinternet.com/ se16/js/mapproj.htm	Lonely Planet - Indonesia Map www.lonelyplanet.com/mapshells/...	Map (Java 2 Platform SE v1.4.2) java.sun.com/j2se/1.4.2/docs/...
2	Java map projections of the world ... www.btinternet.com/ se16/js/oldmapproj.htm	INDONESIA TOURISM : CENTRAL JAVA - MAP www.indonesia-tourism.com/...	Java 2 Platform SE v1.3.1: Interface Map java.sun.com/j2se/1.3/docs/api/java/...
3	Java Map java.sun.com/developer/...	INDONESIA TOURISM : WEST JAVA - MAP www.indonesia-tourism.com/ ...	An Introduction to Java Map Collection Classes www.oracle.com/technology/...
4	Java Technology Concept Map java.sun.com/developer/onlineTraining/...	IndoStreets - Java Map www.indostreets.com/maps/java/	An Introduction to Java Map Collection Classes www.theserverside.com/news/...
5	Science@NASA Home science.nasa.gov/Realtime/...	Indonesia Regions and Islands Maps, Bali, Java, ... www.maps2anywhere.com/Maps/...	Koders - Mappings.java www.koders.com/java/
6	An Introduction to Java Map Collection Classes www.oracle.com/technology/...	Indonesia City Street Map,... www.maps2anywhere.com/Maps/...	Hibernate simplifies inheritance mapping www.ibm.com/developerworks/java/...
7	Lonely Planet - Java Map www.lonelyplanet.com/mapshells/	Maps Of Indonesia www.embassyworld.com/maps/...	tmap_30.map Class Hierarchy tmap.pmel.noaa.gov/...
8	ONJava.com: Java API Map www.onjava.com/pub/a/onjava/api_map/	Maps of Indonesia by Peter Loud users.powernet.co.uk/...	Class Scope jalbum.net/api/se/datadosen/util/Scope.html
9	GTA San Andreas : Sam www.gtasanandreas.net/sam/	Maps of Indonesia by Peter Loud users.powernet.co.uk/mkmarina/indonesia/	Class PrintSafeHashMap jalbum.net/api/se/datadosen/...
10	INDONESIA TOURISM : WEST JAVA - MAP www.indonesia-tourism.com/...	indonesiaphoto.com www.indonesiaphoto.com/...	Java Pro - Union and Vertical Mapping of Classes www.fawcette.com/javapro/...

Table 1: Sample results of query expansion

"java map", corresponding to two different previous queries (i.e., "travel Indonesia" vs. "hashtable"). Due to implicit user modeling, UCAIR intelligently figures out to add "Indonesia" and "class", respectively, to the user's query "java map", which would otherwise be ambiguous as shown in the original results from Google on March 21, 2005. UCAIR's results are much more accurate than Google's results and reflect personalization in search.

The eager implicit feedback component is designed to immediately respond to a user's activity such as viewing a document. In Figure 2, we show how UCAIR can successfully disambiguate an ambiguous query "jaguar" by exploiting a viewed document summary. In this case, the initial retrieval results using "jaguar" (shown on the left side) contain two results about the Jaguar cars followed by two results about the Jaguar software. However, after the user views the web page content of the second result (about "Jaguar car") and returns to the search result page by clicking "back" button, UCAIR automatically nominates two new search results about Jaguar cars (shown on the right side), while the original two results about Jaguar software are pushed down on the list (unseen from the picture).

5.2 Quantitative Evaluation

To further evaluate UCAIR quantitatively, we conduct some user studies on the effectiveness of the eager implicit feedback component. It is a challenge to quantitatively evaluate the potential performance improvement of our proposed model and UCAIR over Google in an unbiased way [7]. Here, we design a user study, in which participants would do normal web search and judge a randomly and anonymously mixed set of results from Google and UCAIR at the end of the search session; participants do not know whether a result comes from Google or UCAIR.

We recruited 6 graduate students for this user study, who have different backgrounds (3 computer science, 2 biology, and 1 chemistry). We use query topics from TREC [1] 2004 Terabyte track [2] and TREC 2003 Web track [4] topic distillation task in the way to be described below.

An example topic from TREC 2004 Terabyte track appears in Figure 3. The title is a short phrase and may be used as a query to the retrieval system. The description field provides a slightly longer statement of the topic requirement, usually expressed as a single complete sentence or question. Finally the narrative supplies additional information necessary to fully specify the requirement, expressed in the form of a short paragraph.

Initially, each participant would browse 50 topics either from

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

<top>

<num> Number: 716

<title> Spammer arrest sue

<desc> Description: Have any spammers been arrested or sued for sending unsolicited e-mail?

<narr> Narrative: Instances of arrests, prosecutions, convictions, and punishments of spammers, and lawsuits against them are relevant. Documents which describe laws to limit spam without giving details of lawsuits or criminal trials are not relevant.

</top>

Figure 3: An example of TREC query topic, expressed in a form which might be given to a human assistant or librarian

Terabyte track or Web track and pick 5 or 7 most interesting topics. For each picked topic, the participant would essentially do the normal web search using UCAIR to find many relevant web pages by using the title of the query topic as the initial keyword query. During this process, the participant may view the search results and possibly click on some interesting ones to view the web pages, just as in a normal web search. There is no requirement or restriction on how many queries the participant must submit or when the participant should stop the search for one topic. When the participant plans to change the search topic, he/she will simply press a button to evaluate the search results before actually switching to the next topic.

At the time of evaluation, up to 30 highly ranked results from Google and UCAIR (some are overlapping) are randomly mixed together so that the participant would not know whether a result comes from Google or UCAIR. These are the results that the user would have seen had he/she decided to continue the search with Google and UCAIR, respectively. The participant would then judge the relevance of these results. We measure precision at top n ($n = 5, 10, 20, 30$) documents of Google and UCAIR. We also evaluate precisions at different recall levels.

Altogether, 368 documents judged as relevant from Google search

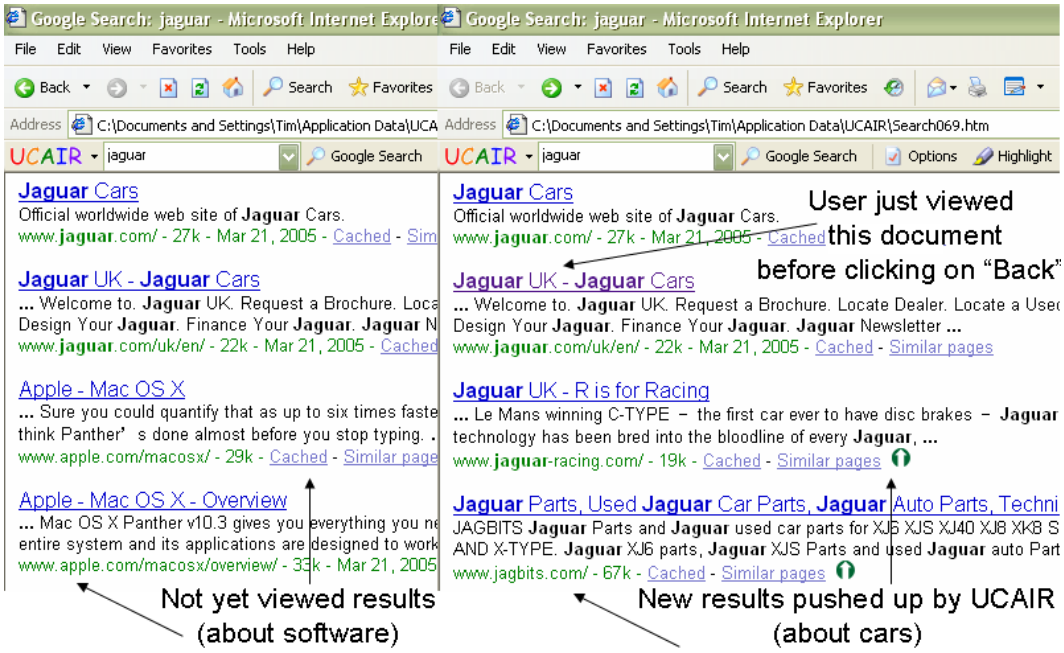

Figure 2: Screen shots for result reranking

results and 429 documents judged as relevant from UCAIR by participants. Scatter plots of precision at top 10 and top 20 documents are shown in Figure 4 and Figure 5 respectively (The scatter plot of precision at top 30 documents is very similar to precision at top 20 documents). Each point of the scatter plots represents the precisions of Google and UCAIR on one query topic.

Table 2 shows the average precision at top n documents among 32 topics. From Figure 4, Figure 5 and Table 2, We see that the search results from UCAIR are consistently better than those from Google by all the measures. Moreover, the performance improvement is more dramatic for precision at top 20 documents than that at precision at top 10 documents, indicating that implicit feedback with clickthrough data helps recall more than the precision at top ranks, which makes sense intuitively, and is similar to what happens in pseudo feedback.

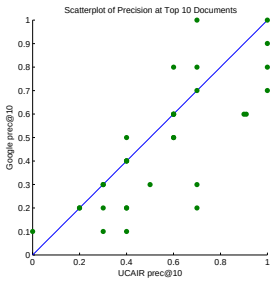

Figure 4: Precision at top 10 documents of UCAIR and Google

The plot in Figure 6 shows the precision-recall curves for UCAIR and Google, where it is clearly seen that the performance of UCAIR is consistently and considerably better than that of Google at all levels of recall.

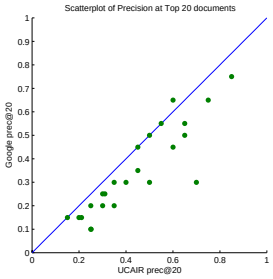

Figure 5: Precision at top 20 documents of UCAIR and Google

Ranking Method	prec@5	prec@10	prec@20	prec@30
Google	0.538	0.472	0.377	0.308
UCAIR	0.581	0.556	0.453	0.375
Improvement	**8.0%**	**17.8%**	**20.2%**	**21.8%**

Table 2: Table of average precision at top n documents for 32 query topics

6. CONCLUSIONS

In this paper, we studied how to exploit implicit user modeling to intelligently personalize information retrieval and improve search accuracy. Unlike most previous work, we emphasize the use of *immediate* search context and implicit feedback information as well as *eager* updating of search results to maximally benefit a user. We presented a decision-theoretic framework for optimizing interactive information retrieval based on eager user model updating, in which the system responds to every action of the user by choosing a system action to optimize a utility function. We further propose specific techniques to capture and exploit two types of implicit feedback information: (1) identifying related immediately preceding query and using the query and the corresponding search results to select appropriate terms to expand the current query, and (2)

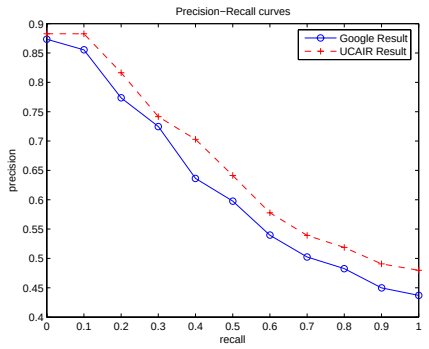

Figure 6: Precision at top 20 result of UCAIR and Google

exploiting the viewed document summaries to immediately rerank any documents that have not yet been seen by the user. Using these techniques, we develop a client-side web search agent (UCAIR) on top of a popular search engine (Google). Experiments on web search show that our search agent can improve search accuracy over Google. Since the implicit information we exploit already naturally exists through user interactions, the user does not need to make any extra effort. The developed search agent thus can improve existing web search performance without any additional effort from the user. We plan to distribute UCAIR for public use in the summer of 2005.

7. ACKNOWLEDGEMENT

We thank the 6 participants of our evaluation experiments. This work was supported in part by the National Science Foundation grants IIS-0347933 and IIS-0428472.

8. REFERENCES

[1] S. M. Beitzel, E. C. Jensen, A. Chowdhury, D. Grossman, and O. Frieder. Hourly analysis of a very large topically categorized web query log. In *Proceedings of SIGIR 2004*, 2004.

[2] C. Clarke, N. Craswell, and I. Soboroff. Overview of the TREC 2004 terabyte track. In *In Proceedings of The Thirteenth Text Retrieval Conference(TREC2004)*, 2004.

[3] M. Claypool, P. Le, M. Waseda, and D. Brown. Implicit interest indicators. In *Proceedings of Intelligent User Interfaces 2001*, pages 33–40, 2001.

[4] N. Craswell, D. Hawking, R. Wilkinson, and M. Wu. Overview of the TREC 2003 web track. In *In Proceedings of The Twelfth Text Retrieval Conference(TREC2003)*, 2003.

[5] W. B. Croft, S. Cronen-Townsend, and V. Larvrenko. Relevance feedback and personalization: A language modeling perspective. In *Proeedings of Second DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*, 2001.

[6] Google Personalized. http://labs.google.com/personalized.

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

[8] X. Huang, F. Peng, A. An, and D. Schuurmans. A new dynamic web log session boundary detection based on statistical language modeling. *Journal of the American Society for Information Science and Technology*, 55(14):1290–1303, 2004.

[9] G. Jeh and J. Widom. Scaling personalized web search. In *Proceedings of WWW 2003*, 2003.

[10] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of SIGKDD 2002*, 2002.

[11] D. Kelly and J. Teevan. Implicit feedback for inferring user preference: A bibliography. *SIGIR Forum*, 2003.

[12] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of SIGIR'01*, pages 111–119, Sept 2001.

[13] T. Lau and E. Horvitz. Patterns of search: Analyzing and modeling web query refinement. In *Proceedings of the Seventh International Conference on User Modeling*, 1999.

[14] V. Lavrenko and B. Croft. Relevance-based language models. In *Proceedings of SIGIR'01*, pages 120–127, Sept 2001.

[15] M. Mitra, A. Singhal, and C. Buckley. Improving automatic query expansion. In *Proceedings of SIGIR 1998*, 1998.

[16] My Yahoo! http://mysearch.yahoo.com.

[17] G. Nunberg. As google goes, so goes the nation. *New York Times*, May 2003.

[18] S. E. Robertson. The probability ranking principle in ir. *Journal of Documentation*, 33(4):294–304, Dec. 1977.

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

[20] G. Salton and C. Buckley. Improving retrieval performance by retrieval feedback. *Journal of the American Society for Information Science*, 41(4), 1990.

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

[22] X. Shen, B. Tan, and C. Zhai. Context-sensitive information retrieval using implicit feedback. In *Proceedings of SIGIR 2005*, 2005.

[23] X. Shen and C. Zhai. Exploiting query history for document ranking in interactive information retrieval (poster). In *Proceedings of SIGIR 2003*, 2003.

[24] A. Singhal. Modern information retrieval: A brief overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 24(4):35–43, 2001.

[25] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *Proceedings of WWW 2004*, 2004.

[26] E. Volokh. Personalization and privacy. *Communications of the ACM*, 43(8):84–88, 2000.

[27] R. W. White, J. M. Jose, C. J. van Rijsbergen, and I. Ruthven. A simulated study of implicit feedback models. In *Proceedings of ECIR 2004*, pages 311–326, 2004.

[28] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *Proceedings of SIGIR 1996*, 1996.

[29] C. Zhai and J. Lafferty. Model-based feedback in KL divergence retrieval model. In *Proceedings of the CIKM 2001*, 2001.