Mining Long-Term Search History to Improve Search Accuracy

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

ABSTRACT

Long-term search history contains rich information about a user's search preferences, which can be used as search context to improve retrieval performance. In this paper, we study statistical language modeling based methods to mine contextual information from long-term search history and exploit it for a more accurate estimate of the query language model. Experiments on real web search data show that the algorithms are effective in improving search accuracy for both fresh and recurring queries. The best performance is achieved when using clickthrough data of past searches that are related to the current query.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models

General Terms

Algorithms

Keywords

Search history, query expansion, context

1. INTRODUCTION

Most existing retrieval systems, including the web search engines, suffer from the problem of "one size fits all": the decision of which documents to return is made based only on the query, without consideration of a particular user's preferences and search context. When a query (e.g., "python") is ambiguous, the search results are inevitably mixed in content (e.g., containing documents on the snake and on the programming language), which is certainly non-optimal for the user, who is burdened by the need to sift through the mixed results. Therefore, instead of relying solely on the query, which is usually just a few keywords, retrieval systems should exploit the user's search context, which can reveal more about the user's true information need. Indeed, contextual retrieval has been identified as a major challenge in information retrieval research [1].

There are a wide variety of search contexts, from the user's background and interests, personal document collection (e.g., emails and saved web pages), to what activities the user is doing before submitting the query (e.g., reading an article on wildlife). In this paper, we focus on the user's *search history*, which is often kept in log format and records what queries the user made in the past and what results he/she chose to view. This is arguably the most important form of search context for the reasons below. First, a user's background and interests can usually be learned from his/her search history by looking at the topics covered by the past queries. For example, if there were many queries such as "debugging" and "CGI code", the user is probably interested in programming and "python" is likely to mean the programming language. Second, from the user's past (implicit) indication of document relevance we can predict his/her reaction to the current retrieved documents. For example, if the user searched with the same query "python" before and clicked on Python language website's link, we have high confidence that the user would do it again this time, and it makes good sense to list that webpage in the top. Even when there is no exact occurrence of the current query in history, we may still find similar queries like "python doc" helpful (e.g., discovering that the user prefers results from the www.python.org site). Because the relevance judgment is usually only inferred from user activities (e.g., clicking on a link, viewing/saving/bookmarking a page), this belongs to the category of implicit feedback, which has been studied in [3, 4, 9, 5] and shown to effectively improve retrieval performance. Finally, search history is readily available without extra user efforts. In the web search domain, user search history can be obtained by a proxy from web logs, or by a search engine using HTTP redirects. If privacy is a concern, we can use browser plug-ins and perform result reranking at the client-side [7].

Search history can be divided into *short-term* and *long-term* types. Short-term search history is limited to a single *search session*, which contains a (normally consecutive) sequence of searches with a coherent information need and usually spans a short period of time. Often, a user composes an initial query, views the returned documents, and if unsatisfied, modifies the query and repeats the search process. All these activities, which form the short-term search history, shed light on the current information need and make useful search context. As shown in [5], queries and clickthrough data in the short-term search history provide implicit feedback that can be used to estimate a more accurate query language model and improve retrieval performance.

Long-term search history is, in contrast, unlimited in time scope and may include *all* search activities in the past. Compared with short-term search history, it has several advantages. There is no need to detect session boundaries (determining whether a previous search shares the same information need as the current one), which is often a difficult task. Nor do we need to limit the context to the contiguous chain of searches in a session; any search in the past that

KDD'06, August 20–23, 2006, Philadelphia, Pennsylvania, USA.

is related to the current one should be leveraged. This also means that we may find context for the very first query in a chain, which is impossible if the search history is constrained to be short-term.

Although the extension from short-term to long-term seems natural and promising, the full potential of long-term search history cannot be reached easily. This is because long-term history inevitably involves a lot of noisy information that is irrelevant (sometimes even distractive) to the current search; only those searches that are related to the current one should be considered as useful context. For example, searches like "world cup" that are irrelevant to the current query "python" would not be helpful, and such noise can overwhelm the signal of related past searches. For this reason, when exploiting short-term search history we need to detect session boundaries first, so that only those searches with the same information need are used. Unfortunately, most existing studies on long-term search context[1] fail to address this problem, even though they still tend to get positive results; they often use all available context as a whole (or divide it into chunks by time), without distinguishing between relevant and irrelevant parts. Such work includes [9], which interpolates the current query with different chunks (time periods) of history (browsed web pages) for personalized search, and [10], which constructs user profiles from indexed desktop documents for search result reranking.

In this paper, we systematically study how to exploit a user's long-term search history to improve retrieval accuracy. We propose mixture models to represent a user's information need and apply statistical language modeling techniques to discover relevant context from the search history, and exploit it to obtain improved estimates of the query model. We then evaluate the methods on a test set of Web search histories collected from some real users. We find that mined search history information, can substantially improve retrieval performance for both recurring and fresh queries, and works best when clickthrough data is used with a discriminative weighting scheme for past searches. We also find that although recent history tends to be much more useful than remote history (especially for fresh queries), all of the entire history is helpful for improving the search accuracy of recurring queries.

The rest of the paper is organized as follows. In Section 2, we introduce a context-sensitive information retrieval approach that allows us to incorporate contextual information mined from search history. In Section 3, we define the search history mining task and cast it as a query language model estimation problem. In Section 4, we develop several algorithms based on statistical language models to mine long-term search history. We describe how we build a test set by collecting users' web search history in Section 5 and present our experiment results in Section 6. Section 7 concludes our work.

2. CONTEXT-SENSITIVE INFORMATION RETRIEVAL

Traditional retrieval models take the retrieval problem as matching a query with a set of documents [8], and thus are inadequate for modeling personalized and contextual search. Previous work [5, 6] has proposed to use statistical language models for context-sensitive search. We follow this approach in this paper. As the background, we briefly describe here how context-sensitive search can be achieved through statistical language modeling.

A language model is a probabilistic model of text. In retrieval, we often use the simplest unigram language models (i.e., word distributions) to model both queries and documents. In the Kullback-Leibler (KL) divergence retrieval method [11], the retrieval task is taken as computing a query language model θ_q for the query q and a document language model θ_d for each document d and then scoring the document according to their KL divergence $D(\theta_q||\theta_d)$, which is defined as

$$D(\theta_q||\theta_d) = \sum_{w \in V} p(w|\theta_q) \log \frac{p(w|\theta_q)}{p(w|\theta_d)} \quad (1)$$

where w is a word in the vocabulary set V.

Clearly, with this method, our main task is to estimate θ_d and θ_q. We estimate the document model θ_d using Bayesian smoothing with Dirichlet prior [12]:

$$p(w|\theta_d) = \frac{c(w;d) + \mu p(w|\theta_C)}{|d| + \mu} \quad (2)$$

where $c(w;d)$ is the count of w in d, $|d|$ is the document length, $p(w|\theta_C)$ is the collection language model and μ is the Dirichlet prior. In our study, we use the TREC web corpus WT10g to estimate the collection language model $p(w|\theta_C)$ and set μ to 10, which is optimized for the contextless baseline and suitable for short snippet texts.

When no other evidence is available (i.e., being contextless), the query model θ_q can be estimated solely based on the query text using the Maximum Likelihood Estimator (MLE):

$$p(w|\theta_q) = \frac{c(w;q)}{|q|} \quad (3)$$

where $c(w;q)$ is the count of w in q and $|q|$ is the query length. But when there is contextual information mined from search history, we can incorporate it as additional evidence to improve our estimate of the query language model. This natural incorporation of extra information is why language models are particularly suitable for modeling context-sensitive search.

Specifically, given search history H, we can estimate the context-sensitive query model $p(w|\theta_{q,H})$ as

$$p(w|\theta_{q,H}) = \lambda p(w|\theta_q) + (1-\lambda) p(w|\theta_H) \quad (4)$$

where $p(w|\theta_q)$ is the context-independent language model estimated using query text only, $p(w|\theta_H)$ is a history language model learned from search history, and $\lambda \in [0,1]$ is the interpolation weight.

In the following sections, we will describe how to compute the context-sensitive query model $p(w|\theta_{q,H})$ based on long-term search history, and how to estimate λ.

3. SEARCH HISTORY MINING

In a typical scenario of information retrieval, after a query is submitted, the retrieval system will return a set of result documents with titles and summaries displayed. The user can then select to view the full texts of some results (usually by clicking on them). Thus the search history generally includes three components: past queries, their search results, and the information on which results were clicked/viewed (also known as clickthrough data).

Formally, let q_i be a query, $\mathcal{D}_i$ be the set of its result documents, $\mathcal{C}_i (\mathcal{C}_i \subseteq \mathcal{D}_i)$ be the set of clicked ones, and t_i be q_i's submission time. If q_k is the current query, its search history H_k consists of all previous queries $q_1, q_2, \cdots, q_{k-1}$ ordered by time, and their corresponding $\mathcal{D}_i$'s and $\mathcal{C}_i$'s.

As described previously, we can use the following interpolation formula to compute the context-sensitive query model for the current query q_k:

$$p(w|\theta_{q_k,H_k}) = \lambda_{q_k} p(w|\theta_{q_k}) + (1-\lambda_{q_k}) p(w|\theta_{H_k}) \quad (5)$$

[1]They are often not strictly what we define as search history (queries and viewed results), but being instead all documents (web pages and/or emails) the user has viewed over a long period of time.

where λ_{q_k} is the weight on the original query, and θ_{H_k} is the history model for q_k.

The goal of search history mining is to estimate the best history model θ_{H_k} from q_k's history H_k, the one that is most informative of the user's search context and thus can bring greatest increase in retrieval accuracy. There are several challenges in this task: First, a past search can contain different components (query, results and clickthrough). We should find the best way to combine these pieces of contextual information. Second, as we have discussed, not all past queries are equally important. We need to identify queries related to the current one and weight them appropriately. Third, when the search history has hundreds or thousands of entries, efficiency may become a concern. These issues will be addressed in the next section.

4. HISTORY LANGUAGE MODELS

In this section, we discuss how to compute the history language model θ_{H_k}, which is regarded as the search context of the current query q_k and to be interpolated with θ_{q_k}. Our strategy is to first generate a *unit history model* for each history query, and then combine them to get the overall history model.

4.1 Unit History Model

For each past query $q_i \in H_k$, we will estimate a language model θ_i that captures the user's information need at that particular moment. We call this a unit history model, because it represents a basic unit of search history that can be integrated to produce the overall history model. We use the following formula to compute it:

$$p(w|\theta_i) = \lambda_q p(w|\theta_{q_i}) + (1-\lambda_q)\cdot \frac{\sigma_C \sum_{d_j \in \mathcal{C}_i} p(w|\theta_{d_j}) + \sigma_{NC} \sum_{d_j \in \mathcal{NC}_i} p(w|\theta_{d_j})}{\sigma_C |\mathcal{C}_i| + \sigma_{NC} |\mathcal{NC}_i|} \quad (6)$$

where λ_q is the interpolation weight on the original query model, θ_{q_i} and θ_{d_j} are the query and document language models, and $\mathcal{NC}_i = \mathcal{D}_i - \mathcal{C}_i$ is the set of non-clicked results. The fraction in the above formula is essentially a weighted average of result document models, with σ_C and σ_{NC} being the weights on clicked and non-clicked results, and $|\mathcal{C}|$ and $|\mathcal{NC}|$ being the number of clicked and non-clicked results.

Below we discuss three special degenerated cases for setting the parameters, each involving a single component of search history (namely, query, documents and clickthrough):

- $\lambda_q = 1$: (6) simplifies to

$$p(w|\theta_i) = p(w|\theta_{q_i}) \quad (7)$$

  The unit history model is based on query text only.

- $\lambda_q = 0, \sigma_C = \sigma_{NC}$: (6) simplifies to

$$p(w|\theta_i) = \frac{\sum_{d_j \in \mathcal{D}_i} p(w|\theta_{d_j})}{|\mathcal{D}_i|} \quad (8)$$

  The unit history model makes use of result documents only by averaging document models. Because query texts are usually short, the user's information need can often be better inferred from these result documents. This resembles pseudo feedback, and we expect the formula to give higher performance than the previous one.

- $\lambda_q = 0, \sigma_C \neq 0, \sigma_{NC} = 0$: (6) simplifies to

$$p(w|\theta_i) = \frac{\sum_{d_j \in \mathcal{C}_i} p(w|\theta_{d_j})}{|\mathcal{C}_i|} \quad (9)$$

  The unit history model is generated from clicked result documents only. Because clicked documents reflect a user's implicit feedback, the constructed language model should be more accurate than the one in (8), where clickthrough information is not used. If there are no clicks ($\mathcal{C}_i = \emptyset$), the query (and its unit history model) is ignored when this formula is used.

The general form of (6) combines different components of search history. Typically, we set $\sigma_C > \sigma_{NC} > 0$, so that clicked results receive more weight than non-clicked ones. On our data set, we find that the setting $\lambda_q = 0, \sigma_C = 20, \sigma_{NC} = 1$ achieves good performance.

4.2 Overall History Model

We use a weighted average of the unit history models of past queries as the overall history model:

$$p(w|\theta_{H_k}) = \frac{\sum_{q_i \in H_k} \lambda_i p(w|\theta_i)}{\sum_{q_i \in H_k} \lambda_i} \quad (10)$$

We discuss two general weighting schemes below.

4.3 Equal Weighting

With equal weighting, unit history models of different past queries are assigned equal weights:

$$\lambda_i = 1, \forall q_i \in H_k \quad (11)$$

If the unit history models only rely on query texts (7), and queries are assumed to be of equal length, the probability of a term in the overall history model is proportional to its global frequency in all queries of the history. Similar things can be said about (8) and (9) for search results and those clicked ones.

This simple weighting scheme suffers from the problem that, as it tries to assign equal weights to every piece of search history, none of them obtains much weight to be influential. It produces a global but weak description of the user's long-term interests.

4.4 Discriminative Weighting

As we have discussed in Section 1, out of all past searches, only those that are related to the current query are important as its context. We should therefore concentrate the weight mass on them, and ignore other random, noisy parts of the search history. We call this approach discriminative weighting, as we are selective about which parts of search history to use according to the current query.

Generally, the more similar to the current query a previous query is, the more weight it should have in the computation of the overall history model. Below we describe several methods for calculating similarity scores between two queries, which can be used as interpolation weights in (10).

4.4.1 Cosine Similarity

For each query q_i, we compute a TF-IDF vector v_i that corresponds to the concatenation of all its result documents:

$$v_i[w] = \sum_{d_j \in \mathcal{D}_i} c(w; d_j) \log \frac{N+1}{DF(w)+0.5}, \quad (12)$$

where $v_i[w]$ is the element in the TF-IDF vector that corresponds to term w, N is the number of documents in the background corpus (WT10g), and $DF(w)$ is w's document frequency. We choose to use concatenation of result documents rather than query text because query text is usually very short, so there may not be enough overlapping between two queries, even if they are related.

The cosine similarity between two vectors is defined as

$$cos(v_i, v_j) = \frac{v_i \cdot v_j}{|v_i||v_j|} \quad (13)$$

and is always in $[0, 1]$.

Since $cos(v_i, v_k)$ measures how close q_i is related to q_k, we can naturally use it for λ_i in (10).

4.4.2 EM Estimation

Here we present a more principled approach, in which λ_i is derived from mixture weights in a generative model.

Suppose there is a mixture model θ_{mix}:

$$p(w|\theta_{\text{mix}}) = \mu_C p(w|\theta_C) + \mu_q p(w|\theta_{q_k}) + \sum_{i=1}^{k-1} \mu_i p(w|\phi_i), \quad (14)$$

where θ_C is the background language model estimated from the corpus (WT10g), θ_{q_k} is MLE from the current query q_k's text, and ϕ_i is MLE from the concatenation of result documents of q_i, a past query in H_k:

$$p(w|\phi_i) = \frac{\sum_{d_j \in \mathcal{D}_i} c(w; d_j)}{\sum_{d_j \in \mathcal{D}_i} |d_j|} \quad (15)$$

μ_C, μ_q and μ_i are mixture weights and constrained by

$$\mu_C + \mu_q + \sum_{i=1}^{k-1} \mu_i = 1 \quad (16)$$

Let Λ denote the set of mixture weights (μ_C, μ_q, μ_i). We want to choose Λ^* to maximize the log likelihood for the mixture model to generate the result documents of q_k:

$$\begin{aligned} \Lambda^* &= \text{argmax}_\Lambda \log p(\mathcal{D}_k|\Lambda) \\ &= \text{argmax}_\Lambda \sum_{d_j \in \mathcal{D}_k} \sum_{w \in d_j} c(w; d_j) \log p(w|\theta_{\text{mix}}) \end{aligned} \quad (17)$$

From (14) and (17), it can be easily seen that, the closer q_i is related to q_k, the larger mixture weight (μ_i) ϕ_i will have in the mixture model (because it fits $\mathcal{D}_k$ better). Indeed, μ_i reaches its maximum when $\mathcal{D}_i$ is identical to $\mathcal{D}_k$. Therefore, we can use μ_i for λ_i in (10).

To estimate these mixture weights, we use the EM algorithm. Let w_j be the j-th word in the concatenation of all result documents in $\mathcal{D}_k$. The Q-function is

$$\begin{aligned} \sum_{j=1}^{L} \Big(& p(Z_{Cj}|\Lambda^{(n)}) \log \mu_C p(w_j|\theta_C) \\ & + p(Z_{qj}|\Lambda^{(n)}) \log \mu_q p(w_j|\theta_{q_k}) \\ & + \sum_{i=1}^{k-1} p(Z_{ij}|\Lambda^{(n)}) \log \mu_i p(w_j|\phi_i) \Big) \end{aligned} \quad (18)$$

where $L = \sum_{d_j \in \mathcal{D}_i} |d_j|$ is the sum of q_k's result document lengths, $\Lambda^{(n)}$ is the set of parameters at the n-th iteration, and Z_{Cj}, Z_{qj}, Z_{ij} are the hidden variables, indicating the events of w_j being generated by θ_C, θ_{q_k}, ϕ_i respectively.

In the E-step, we have

$$p(Z_{Cj}|\Lambda^{(n)}) = \frac{\mu_C^{(n)} p(w_j|\theta_C)}{\mu_C^{(n)} p(w_j|\theta_C) + \mu_q^{(n)} p(w_j|\theta_{q_k}) + \sum_{i=1}^{k-1} \mu_i^{(n)} p(w_j|\phi_i)}$$

$$p(Z_{qj}|\Lambda^{(n)}) = \frac{\mu_q^{(n)} p(w_j|\theta_{q_k})}{\mu_C^{(n)} p(w_j|\theta_C) + \mu_q^{(n)} p(w_j|\theta_{q_k}) + \sum_{i=1}^{k-1} \mu_i^{(n)} p(w_j|\phi_i)}$$

$$p(Z_{ij}|\Lambda^{(n)}) = \frac{\mu_i^{(n)} p(w_j|\phi_i)}{\mu_C^{(n)} p(w_j|\theta_C) + \mu_q^{(n)} p(w_j|\theta_{q_k}) + \sum_{l=1}^{k-1} \mu_l^{(n)} p(w_j|\phi_l)}$$

In the M-step, we have

$$\mu_C^{(n+1)} = \frac{\sum_{j=1}^{L} p(Z_{Cj}|\Lambda^{(n)})}{L}$$

$$\mu_q^{(n+1)} = \frac{\sum_{j=1}^{L} p(Z_{qj}|\Lambda^{(n)})}{L}$$

$$\mu_i^{(n+1)} = \frac{\sum_{j=1}^{L} p(Z_{ij}|\Lambda^{(n)})}{L}$$

Because we have computed μ_q, the mixture weight on q_k, we may estimate λ_{q_k} in (5) based on it, instead of using a fixed value:

$$\lambda_{q_k} = \frac{\mu_q}{\mu_q + \sum_{i=1}^{k-1} \mu_i} \quad (19)$$

This way, the weighting in the final contextual model θ_k is very flexible: when there is a rich amount of relevant search history (reflected by a large value of $\sum_{i=1}^{k-1} \mu_i$ compared to μ_q), there will be significant weight on the history model θ_{H_k}; on the other hand, when the search history is mostly irrelevant, the MLE model θ_{q_k} from query text will dominate. Moreover, all the weighting parameters (i.e., λ_{q_k} and λ_i's) will be estimated rather than manually set.

4.4.3 Hybrid Method

The EM estimation method, although shown to produce more accurate weights, runs much slower than the cosine similarity method, due to the fact that the EM algorithm usually needs many iterations to converge, and each iteration is generally more complex than just computing a cosine similarity value. This will be a big concern for longer search history, when there are hundreds or thousands of queries.

We observe on our data set that, with discriminative weighting, only a small number of previous queries are most related to the current query and receive non-insignificant weights (which is exactly what we intend to see). Motivated by this, we first run the cosine similarity method, identify the queries with highest similarity scores, and keep them in a *working set*. We then run the EM estimation method only on the queries in this working set, and assign zero weights to other queries in the search history. We find this approach yields similar retrieval accuracy as the original EM method, yet runs almost as fast as the cosine method.

5. DATA COLLECTION

To our knowledge, there is no publicly available collection of search logs that contain reasonably long period of users' search history with implicit feedback information. Therefore, we chose to create our own data set in the web search domain by making a plug-in for the Firefox browser to record a user's long-term search history. Specifically, the plug-in saves to a log file all user search activities that are captured from the browser, including queries issued to the Google search engine, search results (with titles, summaries and URLs) returned, and the information of which results are clicked on. The plug-in collects search history in the background and is intentionally kept transparent from the user so that it will not interfere with her normal search activities.

Four computer science students kept the plug-ins installed on their personal computers for over a month and then submitted their individual search logs to us (they were free to delete any sensitive queries that they do not want to disclose). Next the users were asked to pick at least 15 queries from their own search logs, starting from the back (the most recent history). The queries selected from

the search logs would be evaluated to create a test data set. They must satisfy the following conditions, so that there is room for potential improvement of retrieval accuracy with long-term context.

1. A selected query should have at least one relevant document. Thus misspelled queries and queries issued just to check for the existence of something are excluded.

2. A selected query should either match the person's interests and background (e.g., computer science, pop music, football) or belong to a search session (a chain of queries for the same information need), being a reformulation of some previous query.

For each query, we chose the top 20 results retrieved from Google as the collection of documents (with Google's ranking information removed) to be scored by our retrieval methods. We only use their snippet texts (title + summary). To evaluate these result documents, the users were presented with the set of top 20 results retrieved from Google and asked to judge whether each document was relevant or not. If a query was a known-item search, i.e., if the user knew exactly what the needed result would look like, then only that result should be deemed relevant. Otherwise, if the user was exploring some topic, then he/she should mark all results matching that topic as relevant. For example, if the user has visited Python language's website before and is searching for it again, only this result should be considered relevant; if the user does not know Python and searches to get some ideas about it, then a tutorial on Python is also relevant.

We distinguish two types of queries due to their different nature and retrieval performance. If a query has occurred before in the search history (in exact form or with keywords' order changed) and there are clicks associated with its earlier occurrence(s), it belongs to the category of *recurring* queries. Otherwise, we call the query *fresh*. Usually, the purposes of recurring queries are navigational rather than informational or transactional [2]. Recurring queries are also more likely to reflect the user's long-term interests. It tends to be easier to improve the retrieval performance of recurring queries, as the user is very likely to choose exactly those results clicked on in an earlier search.

Table 1 shows some statistics of the collected log data. The large difference in the number of queries is due to some users not using Firefox for all of their web searches.

Table 1: Statistics of search log data

	user1	user2	user3	user4
# days in search history	65	44	69	64
# queries	1255	355	376	136
# queries with ≥ 1 clicks	607	238	207	79
avg. # clicks for query with ≥ 1 clicks	1.26	1.48	1.56	1.37
# testing queries	71	63	19	17
# fresh/recurring queries	54/17	59/4	12/7	13/4
avg. # rel. results per query	2.09	4.14	3.58	6.59

6. EXPERIMENT RESULTS

In this section, we empirically evaluate the performance of the proposed methods on our data set of personal web search logs. We will also study the influence on retrieval accuracy of individual components and different time cutoffs in search history.

We use the standard TREC mean average precision (MAP) and precision at top 5 documents (Pr@5) as our evaluation metrics, which respectively measure the system's overall retrieval accuracy and its performance for those documents that are most viewed. We pool together the queries and judgments of all four users, so that the evaluation result will be a weighted average over these users, with the number of testing queries of each user as weights. We also report the performance for fresh and recurring queries separately, because they display very different behaviors.

6.1 Effects of Using Different Search History Components

We first study how useful each search history component (i.e., query, documents and clickthrough) are as search context. Table 2 shows the performance of using only certain components with the equal weighting and EM estimation methods. The rows "Contextless", "Equal", "EM" correspond respectively to the baseline method of using only query text, equal weighting and discriminative weighting with EM estimation. The text in parentheses indicate which component is used. For equal weighting, we set λ_{q_k} to 0.1 for fresh queries and 0.02 for recurring queries, which perform better than other values.

Table 2: Effects of using different search history components

	Fresh		Recurring	
	MAP	**pr@5**	**MAP**	**pr@5**
Contextless	0.371	0.233	0.265	0.138
Equal (query)	0.355	0.228	0.309	0.206
Equal (docs)	0.387	0.264	0.396	0.231
Equal (clickthrough)	0.394	0.258	0.470	0.250
Equal (combination)	0.391	0.261	0.460	0.244
EM (query)	0.368	0.237	0.257	0.138
EM (docs)	0.404	0.275	0.390	0.244
EM (clickthrough)	0.425	0.274	0.772	0.331
EM (combination)	0.430	0.271	0.766	0.325

We find that from the search history, queries alone usually does not help (except in the case of recurring queries with equal weighting), probably because query texts are too short to make useful search context. In contrast, result documents are able to improve retrieval performance, especially for recurring queries. Finally, clicked results yield the highest increase in search accuracy, suggesting the usefulness of clickthrough as implicit feedback.

The fact that both result documents and clickthrough bring improvement in retrieval performance prompts us to combine them by setting $\sigma_C = 20\sigma_{NC}$ in (6), so that both result documents and clickthrough are used in the computation of the history model. However, we do not observe performance gain over using only clickthrough.

6.2 Comparison of Contextual Models

Table 3 shows the retrieval accuracy of different methods. The rows "Contextless", "Equal", "Cosine", "EM", "Hybrid" correspond, respectively, to the baseline method of using query text only, equal weighting, discriminative weighting with cosine similarity, discriminative weighting with EM estimation and the hybrid method. In the methods that use search history, we combine result documents and clickthrough by setting $\sigma_C = 20\sigma_{NC}$ as before.

We observe that all contextual methods perform better than the contextless one, indicating that long-term history indeed provides helpful search context. We also find that recurring queries get a

Table 3: Retrieval accuracy of different methods

	Fresh		Recurring	
	MAP	**pr@5**	**MAP**	**pr@5**
Contextless	0.371	0.233	0.265	0.138
Equal	0.391	0.261	0.460	0.244
Cosine	0.409	0.265	0.705	0.325
EM	0.430	0.271	0.766	0.325
Hybrid	0.420	0.268	0.802	0.331

lot more improvement from the use of search context than fresh queries, because by nature recurring queries have more relevant search history available as implicit feedback. As we have expected, the discriminative weighting methods outperform the equal weighting one, proving the advantage of selective use of search history. Finally, we note that the EM estimation method achieves higher retrieval accuracy than the cosine similarity method, and the hybrid method has comparable performance.

6.3 Effects of Using Different Search History Lengths

To find out whether recent search history is most useful and whether remote search history helps, we truncate the search history to different lengths (e.g., one day back from the current query), and plot the change of MAP (of the EM weighting method) with respect to time cutoff in Figure 1.

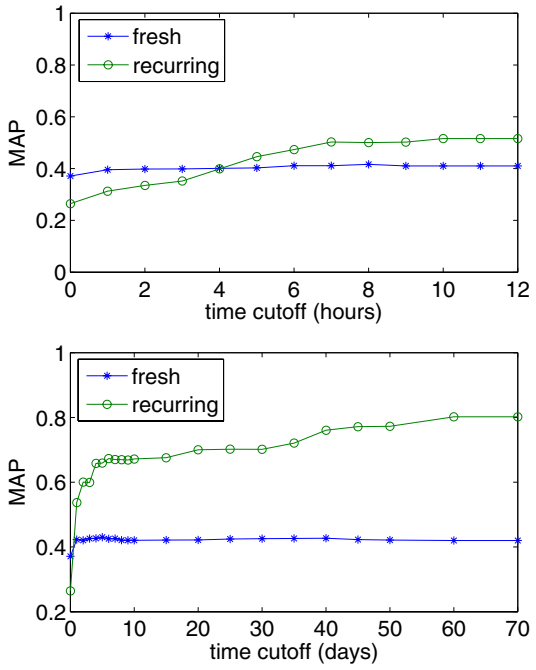

Figure 1: Effects of different search history lengths: within 12 hours (top) & within 70 days (bottom)

We see that for fresh queries, the dominant increase in MAP comes from the most recent history (especially within one hour), while for recurring queries, although recent history is clearly more important, remote history also contributes to the improvement in retrieval accuracy. We believe the difference is because recurring queries are more likely to reflect a user's long-term interests, and thus have more relevant history to leverage.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we systematically explored how to exploit long-term search history, which consists of past queries, result documents and clickthrough, as useful search context that can improve retrieval performance. We emphasized the importance of discriminative use of search history, by concentrating on the most relevant past queries. We cast the search history mining problem as estimating a more accurate query model from evidence in the search history, and developed methods based on statistical language modeling for this task. We collected real web search data as our test set and shown in our experiments that the contextual methods can effectively improve search accuracy over the traditional, contextless method for both fresh and recurring queries, with the EM-based discriminative weighting scheme achieving best performance. We also found through our study of different cutoffs in search history that although recent history is more important, remote history is also useful, especially for recurring queries..

The current work can be extended in several ways: First, the mixture model used in this paper is quite simple. For example, we just concatenate the result documents and treat them equally, ignoring their internal relationship (e.g., they may be clustered). A more appropriate generative model should take these issues into account. Second, in the web search domain the result documents are actually structured (e.g. they have URLs) and it would be interesting to explore how these structural elements could be used. Third, we plan to implement our algorithms inside a web browser search plug-in (UCAIR Toolbar [7]) to provide contextual search on the client-side, which would greatly benefit people's daily search.

8. ACKNOWLEDGMENT

This work is supported in part by the National Science Foundation grants IIS-0347933 and IIS-0428472 and by a Google Research Grant. We thank the graduate students who have helped us collect the search history data and make the relevance judgments.

9. REFERENCES

[1] J. Allan et al. Challenges in information retrieval. In *SIGIR Forum*, volume 37, 2003.
[2] A. Z. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.
[3] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of SIGKDD 2002*, pages 133–142, 2002.
[4] D. Kelly and J. Teevan. Implicit feedback for inferring user preference: A bibliography. *SIGIR Forum*, 37(2):18–28, 2003.
[5] X. Shen, B. Tan, and C. Zhai. Context-sensitive information retrieval using implicit feedback. In *Proceedings of SIGIR 2005*, pages 43–50, 2005.
[6] X. Shen, B. Tan, and C. Zhai. Implicit user modeling for personalized search. In *Proceedings of CIKM 2005*, 2005.
[7] X. Shen, B. Tan, and C. Zhai. Ucair toolbar: A personalized search toolbar (demo). In *Proceedings of SIGIR 2005*, page 681, 2005.
[8] K. Sparck Jones and P. Willett, editors. *Readings in Information Retrieval*. Morgan Kaufmann Publishers, 1997.
[9] 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*, pages 675–684, 2004.
[10] J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activites. In *Proceedings of SIGIR 2005*, 2005.
[11] C. Zhai and J. Lafferty. Model-based feedback in KL divergence retrieval model. In *Proceedings of the CIKM 2001*, pages 403–410, 2001.
[12] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of ACM SIGIR'01*, pages 334–342, Sept 2001.