

Context-Sensitive Information Retrieval Using Implicit Feedback

Xuehua Shen
Department of Computer
Science
University of Illinois at
Urbana-Champaign

Bin Tan
Department of Computer
Science
University of Illinois at
Urbana-Champaign

ChengXiang Zhai
Department of Computer
Science
University of Illinois at
Urbana-Champaign

ABSTRACT

A major limitation of most existing retrieval models and systems is that the retrieval decision is made based solely on the query and document collection; information about the actual user and search context is largely ignored. In this paper, we study how to exploit implicit feedback information, including previous queries and clickthrough information, to improve retrieval accuracy in an interactive information retrieval setting. We propose several context-sensitive retrieval algorithms based on statistical language models to combine the preceding queries and clicked document summaries with the current query for better ranking of documents. We use the TREC AP data to create a test collection with search context information, and quantitatively evaluate our models using this test set. Experiment results show that using implicit feedback, especially the clicked document summaries, can improve retrieval performance substantially.

Categories and Subject Descriptors

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

General Terms

Algorithms

Keywords

Query history, query expansion, interactive retrieval, context

1. INTRODUCTION

In most existing information retrieval models, the retrieval problem is treated as involving one *single* query and a set of documents. From a single query, however, the retrieval system can only have very limited clue about the user's information need. An optimal retrieval system thus should try to exploit as much additional context information as possible to improve retrieval accuracy, whenever it is available. Indeed, context-sensitive retrieval has been identified as a major challenge in information retrieval research[2].

There are many kinds of context that we can exploit. Relevance feedback [14] can be considered as a way for a user to provide more context of search and is known to be effective for improving retrieval accuracy. However, relevance feedback requires that a user *explicitly* provides feedback information, such as specifying the category of the information need or marking a subset of retrieved documents as relevant documents. Since it forces the user to engage additional activities while the benefits are not always obvious to the user, a user is often reluctant to provide such feedback information. Thus the effectiveness of relevance feedback may be limited in real applications.

For this reason, *implicit feedback* has attracted much attention recently [11, 13, 18, 17, 12]. In general, the retrieval results using the user's initial query may not be satisfactory; often, the user would need to revise the query to improve the retrieval/ranking accuracy [8]. For a complex or difficult information need, the user may need to modify his/her query and view ranked documents with many iterations before the information need is completely satisfied. In such an interactive retrieval scenario, the information naturally available to the retrieval system is more than just the current user query and the document collection – in general, all the interaction history can be available to the retrieval system, including past queries, information about which documents the user has chosen to view, and even how a user has read a document (e.g., which part of a document the user spends a lot of time in reading). We define implicit feedback broadly as exploiting all such naturally available interaction history to improve retrieval results.

A major advantage of implicit feedback is that we can improve the retrieval accuracy without requiring any user effort. For example, if the current query is “java”, without knowing any extra information, it would be *impossible* to know whether it is intended to mean the Java programming language or the Java island in Indonesia. As a result, the retrieved documents will likely have both kinds of documents – some may be about the programming language and some may be about the island. However, any particular user is unlikely searching for both types of documents. Such an ambiguity can be resolved by exploiting history information. For example, if we know that the previous query from the user is “cgi programming”, it would strongly suggest that it is the programming language that the user is searching for.

Implicit feedback was studied in several previous works. In [11], Joachims explored how to capture and exploit the clickthrough information and demonstrated that such implicit feedback information can indeed improve the search accuracy for a group of people. In [18], a simulation study of the effectiveness of different implicit feedback algorithms was conducted, and several retrieval models designed for exploiting clickthrough information were pro-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGIR'05, August 15–19, 2005, Salvador, Brazil.
Copyright 2005 ACM 1-59593-034-5/05/0008 ...\$5.00.

posed and evaluated. In [17], some existing retrieval algorithms are adapted to improve search results based on the browsing history of a user. Other related work on using context includes personalized search [1, 3, 4, 7, 10], query log analysis [5], context factors [12], and implicit queries [6].

While the previous work has mostly focused on using clickthrough information, in this paper, we use both clickthrough information and preceding queries, and focus on developing new context-sensitive language models for retrieval. Specifically, we develop models for using implicit feedback information such as query and clickthrough history of the current search session to improve retrieval accuracy. We use the KL-divergence retrieval model [19] as the basis and propose to treat context-sensitive retrieval as estimating a query language model based on the current query and any search context information. We propose several statistical language models to incorporate query and clickthrough history into the KL-divergence model.

One challenge in studying implicit feedback models is that there does not exist any suitable test collection for evaluation. We thus use the TREC AP data to create a test collection with implicit feedback information, which can be used to *quantitatively* evaluate implicit feedback models. To the best of our knowledge, this is the first test set for implicit feedback. We evaluate the proposed models using this data set. The experimental results show that using implicit feedback information, especially the clickthrough data, can substantially improve retrieval performance without requiring additional effort from the user.

The remaining sections are organized as follows. In Section 2, we attempt to define the problem of implicit feedback and introduce some terms that we will use later. In Section 3, we propose several implicit feedback models based on statistical language models. In Section 4, we describe how we create the data set for implicit feedback experiments. In Section 5, we evaluate different implicit feedback models on the created data set. Section 6 is our conclusions and future work.

2. PROBLEM DEFINITION

There are two kinds of context information we can use for implicit feedback. One is *short-term context*, which is the immediate surrounding information which throws light on a user’s current information need in a single *session*. A session can be considered as a period consisting of all interactions for the same information need. The category of a user’s information need (e.g., kids or sports), previous queries, and recently viewed documents are all examples of short-term context. Such information is most directly related to the current information need of the user and thus can be expected to be most useful for improving the current search. In general, short-term context is most useful for improving search in the current session, but may not be so helpful for search activities in a different session. The other kind of context is *long-term context*, which refers to information such as a user’s education level and general interest, accumulated user query history and past user clickthrough information; such information is generally stable for a long time and is often accumulated over time. Long-term context can be applicable to all sessions, but may not be as effective as the short-term context in improving search accuracy for a particular session. In this paper, we focus on the short-term context, though some of our methods can also be used to naturally incorporate some long-term context.

In a single search session, a user may interact with the search system several times. During interactions, the user would continuously modify the query. Therefore for the current query Q_k (except for the first query of a search session), there is a *query history*, $H_Q = (Q_1, \dots, Q_{k-1})$ associated with it, which consists of the pre-

ceding queries given by the same user in the current session. Note that we assume that the session boundaries are known in this paper. In practice, we need techniques to automatically discover session boundaries, which have been studied in [9, 16]. Traditionally, the retrieval system only uses the current query Q_k to do retrieval. But the short-term query history clearly may provide useful clues about the user’s current information need as seen in the “java” example given in the previous section. Indeed, our previous work [15] has shown that the short-term query history is useful for improving retrieval accuracy.

In addition to the query history, there may be other short-term context information available. For example, a user would presumably frequently click some documents to view. We refer to data associated with these actions as *clickthrough history*. The clickthrough data may include the title, summary, and perhaps also the content and location (e.g., the URL) of the clicked document. Although it is not clear whether a viewed document is actually relevant to the user’s information need, we may “safely” assume that the displayed summary/title information about the document is attractive to the user, thus conveys information about the user’s information need. Suppose we concatenate all the displayed text information about a document (usually title and summary) together, we will also have a *clicked summary* C_i in each round of retrieval. In general, we may have a history of clicked summaries C_1, \dots, C_{k-1} . We will also exploit such clickthrough history $H_C = (C_1, \dots, C_{k-1})$ to improve our search accuracy for the current query Q_k . Previous work has also shown positive results using similar clickthrough information [11, 17].

Both query history and clickthrough history are implicit feedback information, which naturally exists in interactive information retrieval, thus no additional user effort is needed to collect them. In this paper, we study how to exploit such information (H_Q and H_C), develop models to incorporate the query history and clickthrough history into a retrieval ranking function, and quantitatively evaluate these models.

3. LANGUAGE MODELS FOR CONTEXT-SENSITIVE INFORMATION RETRIEVAL

Intuitively, the query history H_Q and clickthrough history H_C are both useful for improving search accuracy for the current query Q_k . An important research question is how we can exploit such information effectively. We propose to use statistical language models to model a user’s information need and develop four specific context-sensitive language models to incorporate context information into a basic retrieval model.

3.1 Basic retrieval model

We use the Kullback-Leibler (KL) divergence method [19] as our basic retrieval method. According to this model, the retrieval task involves computing a query language model θ_Q for a given query and a document language model θ_D for a document and then computing their KL divergence $D(\theta_Q || \theta_D)$, which serves as the score of the document.

One advantage of this approach is that we can naturally incorporate the search context as additional evidence to improve our estimate of the query language model.

Formally, let $H_Q = (Q_1, \dots, Q_{k-1})$ be the query history and the current query be Q_k . Let $H_C = (C_1, \dots, C_{k-1})$ be the clickthrough history. Note that C_i is the concatenation of all clicked documents’ summaries in the i -th round of retrieval since we may reasonably treat all these summaries equally. Our task is to estimate a context query model, which we denote by $p(w|\theta_k)$, based on the

current query Q_k , as well as the query history H_Q and clickthrough history H_C . We now describe several different language models for exploiting H_Q and H_C to estimate $p(w|\theta_k)$. We will use $c(w, X)$ to denote the count of word w in text X , which could be either a query or a clicked document's summary or any other text. We will use $|X|$ to denote the length of text X or the total number of words in X .

3.2 Fixed Coefficient Interpolation (FixInt)

Our first idea is to summarize the query history H_Q with a unigram language model $p(w|H_Q)$ and the clickthrough history H_C with another unigram language model $p(w|H_C)$. Then we linearly interpolate these two history models to obtain the history model $p(w|H)$. Finally, we interpolate the history model $p(w|H)$ with the current query model $p(w|Q_k)$. These models are defined as follows.

$$\begin{aligned} p(w|Q_i) &= \frac{c(w, Q_i)}{|Q_i|} \\ p(w|H_Q) &= \frac{1}{k-1} \sum_{i=1}^{i=k-1} p(w|Q_i) \\ p(w|C_i) &= \frac{c(w, C_i)}{|C_i|} \\ p(w|H_C) &= \frac{1}{k-1} \sum_{i=1}^{i=k-1} p(w|C_i) \\ p(w|H) &= \beta p(w|H_C) + (1-\beta)p(w|H_Q) \\ p(w|\theta_k) &= \alpha p(w|Q_k) + (1-\alpha)p(w|H) \end{aligned}$$

where $\beta \in [0, 1]$ is a parameter to control the weight on each history model, and where $\alpha \in [0, 1]$ is a parameter to control the weight on the current query and the history information.

If we combine these equations, we see that

$$p(w|\theta_k) = \alpha p(w|Q_k) + (1-\alpha)[\beta p(w|H_C) + (1-\beta)p(w|H_Q)]$$

That is, the estimated context query model is just a fixed coefficient interpolation of three models $p(w|Q_k)$, $p(w|H_Q)$, and $p(w|H_C)$.

3.3 Bayesian Interpolation (BayesInt)

One possible problem with the FixInt approach is that the coefficients, especially α , are fixed across all the queries. But intuitively, if our current query Q_k is very long, we should trust the current query more, whereas if Q_k has just one word, it may be beneficial to put more weight on the history. To capture this intuition, we treat $p(w|H_Q)$ and $p(w|H_C)$ as Dirichlet priors and Q_k as the observed data to estimate a context query model using Bayesian estimator. The estimated model is given by

$$\begin{aligned} p(w|\theta_k) &= \frac{c(w, Q_k) + \mu p(w|H_Q) + \nu p(w|H_C)}{|Q_k| + \mu + \nu} \\ &= \frac{|Q_k|}{|Q_k| + \mu + \nu} p(w|Q_k) + \frac{\mu + \nu}{|Q_k| + \mu + \nu} \left[\frac{\mu}{\mu + \nu} p(w|H_Q) + \frac{\nu}{\mu + \nu} p(w|H_C) \right] \end{aligned}$$

where μ is the prior sample size for $p(w|H_Q)$ and ν is the prior sample size for $p(w|H_C)$. We see that the only difference between BayesInt and FixInt is the interpolation coefficients are now adaptive to the query length. Indeed, when viewing BayesInt as FixInt, we see that $\alpha = \frac{|Q_k|}{|Q_k| + \mu + \nu}$, $\beta = \frac{\nu}{\nu + \mu}$, thus with fixed μ and ν , we will have a query-dependent α . Later we will show that such an adaptive α empirically performs better than a fixed α .

3.4 Online Bayesian Updating (OnlineUp)

Both FixInt and BayesInt summarize the history information by *averaging* the unigram language models estimated based on previous queries or clicked summaries. This means that all previous queries are treated equally and so are all clicked summaries. However, as the user interacts with the system and acquires more knowledge about the information in the collection, presumably, the reformulated queries will become better and better. Thus assigning decaying weights to the previous queries so as to trust a recent query more than an earlier query appears to be reasonable. Interestingly, if we *incrementally* update our belief about the user's information need after seeing each query, we could naturally obtain decaying weights on the previous queries. Since such an incremental online updating strategy can be used to exploit any evidence in an interactive retrieval system, we present it in a more general way.

In a typical retrieval system, the retrieval system responds to every new query entered by the user by presenting a ranked list of documents. In order to rank documents, the system must have some model for the user's information need. In the KL divergence retrieval model, this means that the system must compute a query model whenever a user enters a (new) query. A principled way of updating the query model is to use Bayesian estimation, which we discuss below.

3.4.1 Bayesian updating

We first discuss how we apply Bayesian estimation to update a query model in general. Let $p(w|\phi)$ be our current query model and T be a new piece of text evidence observed (e.g., T can be a query or a clicked summary). To update the query model based on T , we use ϕ to define a Dirichlet prior parameterized as

$$\text{Dir}(\mu_T p(w_1|\phi), \dots, \mu_T p(w_N|\phi))$$

where μ_T is the equivalent sample size of the prior. We use Dirichlet prior because it is a conjugate prior for multinomial distributions. With such a conjugate prior, the predictive distribution of ϕ (or equivalently, the mean of the posterior distribution of ϕ is given by

$$p(w|\phi) = \frac{c(w, T) + \mu_T p(w|\phi)}{|T| + \mu_T} \quad (1)$$

where $c(w, T)$ is the count of w in T and $|T|$ is the length of T . Parameter μ_T indicates our confidence in the prior expressed in terms of an equivalent text sample comparable with T . For example, $\mu_T = 1$ indicates that the influence of the prior is equivalent to adding one extra word to T .

3.4.2 Sequential query model updating

We now discuss how we can update our query model over time during an interactive retrieval process using Bayesian estimation. In general, we assume that the retrieval system maintains a current query model ϕ_i at any moment. As soon as we obtain some implicit feedback evidence in the form of a piece of text T_i , we will update the query model.

Initially, before we see any user query, we may already have some information about the user. For example, we may have some information about what documents the user has viewed in the past. We use such information to define a prior on the query model, which is denoted by ϕ'_0 . After we observe the first query Q_1 , we can update the query model based on the new observed data Q_1 . The updated query model ϕ_1 can then be used for ranking documents in response to Q_1 . As the user views some documents, the displayed summary text for such documents C_1 (i.e., clicked summaries) can serve as some new data for us to further update the

query model to obtain ϕ'_1 . As we obtain the second query Q_2 from the user, we can update ϕ'_1 to obtain a new model ϕ_2 . In general, we may repeat such an updating process to iteratively update the query model.

Clearly, we see two types of updating: (1) updating based on a new query Q_i ; (2) updating based on a new clicked summary C_i . In both cases, we can treat the current model as a prior of the context query model and treat the new observed query or clicked summary as observed data. Thus we have the following updating equations:

$$\begin{aligned} p(w|\phi_i) &= \frac{c(w, Q_i) + \mu_i p(w|\phi'_{i-1})}{|Q_i| + \mu_i} \\ p(w|\phi'_i) &= \frac{c(w, C_i) + \nu_i p(w|\phi_i)}{|C_i| + \nu_i} \end{aligned}$$

where μ_i is the equivalent sample size for the prior when updating the model based on a query, while ν_i is the equivalent sample size for the prior when updating the model based on a clicked summary. If we set $\mu_i = 0$ (or $\nu_i = 0$) we essentially ignore the prior model, thus would start a completely new query model based on the query Q_i (or the clicked summary C_i). On the other hand, if we set $\mu_i = +\infty$ (or $\nu_i = +\infty$) we essentially ignore the observed query (or the clicked summary) and do not update our model. Thus the model remains the same as if we do not observe any new text evidence. In general, the parameters μ_i and ν_i may have different values for different i . For example, at the very beginning, we may have very sparse query history, thus we could use a smaller μ_i , but later as the query history is richer, we can consider using a larger μ_i . But in our experiments, unless otherwise stated, we set them to the same constants, i.e., $\forall i, j, \mu_i = \mu_j, \nu_i = \nu_j$.

Note that we can take either $p(w|\phi_i)$ or $p(w|\phi'_i)$ as our context query model for ranking documents. This suggests that we do not have to wait until a user enters a new query to initiate a new round of retrieval; instead, as soon as we collect clicked summary C_i , we can update the query model and use $p(w|\phi'_i)$ to immediately rerank any documents that a user has not yet seen.

To score documents after seeing query Q_k , we use $p(w|\phi_k)$, i.e.,

$$p(w|\theta_k) = p(w|\phi_k)$$

3.5 Batch Bayesian updating (BatchUp)

If we set the equivalent sample size parameters to fixed constant, the OnlineUp algorithm would introduce a decaying factor – repeated interpolation would cause the early data to have a low weight. This may be appropriate for the query history as it is reasonable to believe that the user becomes better and better at query formulation as time goes on, but it is not necessarily appropriate for the clickthrough information, especially because we use the displayed summary, rather than the actual content of a clicked document. One way to avoid applying a decaying interpolation to the clickthrough data is to do OnlineUp only for the query history $Q = (Q_1, \dots, Q_{i-1})$, but not for the clickthrough data C . We first buffer all the clickthrough data together and use the whole chunk of clickthrough data to update the model generated through running OnlineUp on previous queries. The updating equations are as follows.

$$\begin{aligned} p(w|\phi_i) &= \frac{c(w, Q_i) + \mu_i p(w|\phi_{i-1})}{|Q_i| + \mu_i} \\ p(w|\psi_i) &= \frac{\sum_{j=1}^{i-1} c(w, C_j) + \nu_i p(w|\phi_i)}{\sum_{j=1}^{i-1} |C_j| + \nu_i} \end{aligned}$$

where μ_i has the same interpretation as in OnlineUp, but ν_i now

indicates to what extent we want to trust the clicked summaries. As in OnlineUp, we set all μ_i 's and ν_i 's to the same value. And to rank documents after seeing the current query Q_k , we use

$$p(w|\theta_k) = p(w|\psi_k)$$

4. DATA COLLECTION

In order to quantitatively evaluate our models, we need a data set which includes not only a text database and testing topics, but also query history and clickthrough history for each topic. Since there is no such data set available to us, we have to create one. There are two choices. One is to extract topics and any associated query history and clickthrough history for each topic from the log of a retrieval system (e.g., search engine). But the problem is that we have no relevance judgments on such data. The other choice is to use a TREC data set, which has a text database, topic description and relevance judgment file. Unfortunately, there are no query history and clickthrough history data. We decide to augment a TREC data set by collecting query history and clickthrough history data.

We select TREC AP88, AP89 and AP90 data as our text database, because AP data has been used in several TREC tasks and has relatively complete judgments. There are altogether 242918 news articles and the average document length is 416 words. Most articles have titles. If not, we select the first sentence of the text as the title. For the preprocessing, we only do case folding and do not do stopword removal or stemming.

We select 30 relatively difficult topics from TREC topics 1-150. These 30 topics have the worst average precision performance among TREC topics 1-150 according to some baseline experiments using the KL-Divergence model with Bayesian prior smoothing [20]. The reason why we select difficult topics is that the user then would have to have several interactions with the retrieval system in order to get satisfactory results so that we can expect to collect a relatively richer query history and clickthrough history data from the user. In real applications, we may also expect our models to be most useful for such difficult topics, so our data collection strategy reflects the real world applications well.

We index the TREC AP data set and set up a search engine and web interface for TREC AP news articles. We use 3 subjects to do experiments to collect query history and clickthrough history data. Each subject is assigned 10 topics and given the topic descriptions provided by TREC. For each topic, the first query is the title of the topic given in the original TREC topic description. After the subject submits the query, the search engine will do retrieval and return a ranked list of search results to the subject. The subject will browse the results and maybe click one or more results to browse the full text of article(s). The subject may also modify the query to do another search. For each topic, the subject composes at least 4 queries. In our experiment, only the first 4 queries for each topic are used. The user needs to select the topic number from a selection menu before submitting the query to the search engine so that we can easily detect the session boundary, which is not the focus of our study. We use a relational database to store user interactions, including the submitted queries and clicked documents. For each query, we store the query terms and the associated result pages. And for each clicked document, we store the summary as shown on the search result page. The summary of the article is query dependent and is computed online using fixed-length passage retrieval (KL divergence model with Bayesian prior smoothing).

Among 120 (4 for each of 30 topics) queries which we study in the experiment, the average query length is 3.71 words. Altogether there are 91 documents clicked to view. So on average, there are around 3 clicks per topic. The average length of clicked summary

Query	FixInt ($\alpha = 0.1, \beta = 1.0$)		BayesInt ($\mu = 0.2, \nu = 5.0$)		OnlineUp ($\mu = 5.0, \nu = 15.0$)		BatchUp ($\mu = 2.0, \nu = 15.0$)	
	MAP	pr@20docs	MAP	pr@20docs	MAP	pr@20docs	MAP	pr@20docs
q_1	0.0095	0.0317	0.0095	0.0317	0.0095	0.0317	0.0095	0.0317
q_2	0.0312	0.1150	0.0312	0.1150	0.0312	0.1150	0.0312	0.1150
$q_2 + H_Q + H_C$	0.0324	0.1117	0.0345	0.1117	0.0215	0.0733	0.0342	0.1100
Improve.	3.8%	-2.9%	10.6%	-2.9%	-31.1%	-36.3%	9.6%	-4.3%
q_3	0.0421	0.1483	0.0421	0.1483	0.0421	0.1483	0.0421	0.1483
$q_3 + H_Q + H_C$	0.0726	0.1967	0.0816	0.2067	0.0706	0.1783	0.0810	0.2067
Improve	72.4%	32.6%	93.8%	39.4%	67.7%	20.2%	92.4%	39.4%
q_4	0.0536	0.1933	0.0536	0.1933	0.0536	0.1933	0.0536	0.1933
$q_4 + H_Q + H_C$	0.0891	0.2233	0.0955	0.2317	0.0792	0.2067	0.0950	0.2250
Improve	66.2%	15.5%	78.2%	19.9%	47.8%	6.9%	77.2%	16.4%

Table 1: Effect of using query history and clickthrough data for document ranking.

is 34.4 words. Among 91 clicked documents, 29 documents are judged relevant according to TREC judgment file. This data set is publicly available ¹.

5. EXPERIMENTS

5.1 Experiment design

Our major hypothesis is that using search context (i.e., query history and clickthrough information) can help improve search accuracy. In particular, the search context can provide extra information to help us estimate a better query model than using just the current query. So most of our experiments involve comparing the retrieval performance using the current query only (thus ignoring any context) with that using the current query as well as the search context.

Since we collected four versions of queries for each topic, we make such comparisons for each version of queries. We use two performance measures: (1) Mean Average Precision (MAP): This is the standard non-interpolated average precision and serves as a good measure of the overall ranking accuracy. (2) Precision at 20 documents (pr@20docs): This measure does not average well, but it is more meaningful than MAP and reflects the utility for users who only read the top 20 documents. In all cases, the reported figure is the average over all of the 30 topics.

We evaluate the four models for exploiting search context (i.e., FixInt, BayesInt, OnlineUp, and BatchUp). Each model has precisely two parameters (α and β for FixInt; μ and ν for others). Note that μ and ν may need to be interpreted differently for different methods. We vary these parameters and identify the optimal performance for each method. We also vary the parameters to study the sensitivity of our algorithms to the setting of the parameters.

5.2 Result analysis

5.2.1 Overall effect of search context

We compare the optimal performances of four models with those using the current query only in Table 1. A row labeled with q_i is the baseline performance and a row labeled with $q_i + H_Q + H_C$ is the performance of using search context. We can make several observations from this table:

1. Comparing the baseline performances indicates that on average reformulated queries are better than the previous queries with the performance of q_4 being the best. Users generally formulate better and better queries.
2. Using search context generally has positive effect, especially when the context is rich. This can be seen from the fact that the

improvement for q_4 and q_3 is generally more substantial compared with q_2 . Actually, in many cases with q_2 , using the context may hurt the performance, probably because the history at that point is sparse. When the search context is rich, the performance improvement can be quite substantial. For example, BatchUp achieves 92.4% improvement in the mean average precision over q_3 and 77.2% improvement over q_4 . (The generally low precisions also make the relative improvement deceptively high, though.)

3. Among the four models using search context, the performances of FixInt and OnlineUp are clearly worse than those of BayesInt and BatchUp. Since BayesInt performs better than FixInt and the main difference between BayesInt and FixInt is that the former uses an adaptive coefficient for interpolation, the results suggest that using adaptive coefficient is quite beneficial and a Bayesian style interpolation makes sense. The main difference between OnlineUp and BatchUp is that OnlineUp uses decaying coefficients to combine the multiple clicked summaries, while BatchUp simply concatenates all clicked summaries. Therefore the fact that BatchUp is consistently better than OnlineUp indicates that the weights for combining the clicked summaries indeed should not be decaying. While OnlineUp is theoretically appealing, its performance is inferior to BayesInt and BatchUp, likely because of the decaying coefficient. Overall, BatchUp appears to be the best method when we vary the parameter settings.

We have two different kinds of search context – query history and clickthrough data. We now look into the contribution of each kind of context.

5.2.2 Using query history only

In each of four models, we can “turn off” the clickthrough history data by setting parameters appropriately. This allows us to evaluate the effect of using query history alone. We use the same parameter setting for query history as in Table 1. The results are shown in Table 2. Here we see that in general, the benefit of using query history is very limited with mixed results. This is different from what is reported in a previous study [15], where using query history is consistently helpful. Another observation is that the context runs perform poorly at q_2 , but generally perform (slightly) better than the baselines for q_3 and q_4 . This is again likely because at the beginning the initial query, which is the title in the original TREC topic description, may not be a good query; indeed, on average, performances of these “first-generation” queries are clearly poorer than those of all other user-formulated queries in the later generations. Yet another observation is that when using query history only, the BayesInt model appears to be better than other models. Since the clickthrough data is ignored, OnlineUp and BatchUp

¹<http://sifaka.cs.uiuc.edu/ir/ucair/QCHistory.zip>

Query	FixInt ($\alpha = 0.1, \beta = 0$)		BayesInt ($\mu = 0.2, \nu = 0$)		OnlineUp ($\mu = 5.0, \nu = +\infty$)		BatchUp ($\mu = 2.0, \nu = +\infty$)	
	MAP	pr@20docs	MAP	pr@20docs	MAP	pr@20docs	MAP	pr@20docs
q_2	0.0312	0.1150	0.0312	0.1150	0.0312	0.1150	0.0312	0.1150
$q_2 + H_Q$	0.0097	0.0317	0.0311	0.1200	0.0213	0.0783	0.0287	0.0967
Improve.	-68.9%	-72.4%	-0.3%	4.3%	-31.7%	-31.9%	-8.0%	-15.9%
q_3	0.0421	0.1483	0.0421	0.1483	0.0421	0.1483	0.0421	0.1483
$q_3 + H_Q$	0.0261	0.0917	0.0451	0.1517	0.0444	0.1333	0.0455	0.1450
Improve	-38.2%	-38.2%	7.1%	2.3%	5.5%	-10.1%	8.1%	-2.2%
q_4	0.0536	0.1933	0.0536	0.1933	0.0536	0.1933	0.0536	0.1933
$q_4 + H_Q$	0.0428	0.1467	0.0537	0.1917	0.0550	0.1733	0.0552	0.1917
Improve	-20.1%	-24.1%	0.2%	-0.8%	3.0%	-10.3%	3.0%	-0.8%

Table 2: Effect of using query history only for document ranking.

μ		0	0.5	1	2	3	4	5	6	7	8	9
$q_2 + H_Q$	MAP	0.0312	0.0313	0.0308	0.0287	0.0257	0.0231	0.0213	0.0194	0.0183	0.0182	0.0164
$q_3 + H_Q$	MAP	0.0421	0.0442	0.0441	0.0455	0.0457	0.0458	0.0444	0.0439	0.0430	0.0390	0.0335
$q_4 + H_Q$	MAP	0.0536	0.0546	0.0547	0.0552	0.0544	0.0548	0.0550	0.0541	0.0534	0.0525	0.0513

Table 3: Average Precision of BatchUp using query history only

are essentially the same algorithm. The displayed results thus reflect the variation caused by parameter μ . A smaller setting of 2.0 is seen better than a larger value of 5.0. A more complete picture of the influence of the setting of μ can be seen from Table 3, where we show the performance figures for a wider range of values of μ . The value of μ can be interpreted as how many words we regard the query history is worth. A larger value thus puts more weight on the history and is seen to hurt the performance more when the history information is not rich. Thus while for q_4 the best performance tends to be achieved for $\mu \in [2, 5]$, only when $\mu = 0.5$ we see some small benefit for q_2 . As we would expect, an excessively large μ would hurt the performance in general, but q_2 is hurt most and q_4 is barely hurt, indicating that as we accumulate more and more query history information, we can put more and more weight on the history information. This also suggests that a better strategy should probably dynamically adjust parameters according to how much history information we have.

The mixed query history results suggest that the positive effect of using implicit feedback information may have largely come from the use of clickthrough history, which is indeed true as we discuss in the next subsection.

5.2.3 Using clickthrough history only

We now turn off the query history and only use the clicked summaries plus the current query. The results are shown in Table 4. We see that the benefit of using clickthrough information is much more significant than that of using query history. We see an overall positive effect, often with significant improvement over the baseline. It is also clear that the richer the context data is, the more improvement using clicked summaries can achieve. Other than some occasional degradation of precision at 20 documents, the improvement is fairly consistent and often quite substantial.

These results show that the clicked summary text is in general quite useful for inferring a user’s information need. Intuitively, using the summary text, rather than the actual content of the document, makes more sense, as it is quite possible that the document behind a seemingly relevant summary is actually non-relevant.

29 out of the 91 clicked documents are relevant. Updating the query model based on such summaries would bring up the ranks of these relevant documents, causing performance improvement.

However, such improvement is really not beneficial for the user as the user has already seen these relevant documents. To see how much improvement we have achieved on improving the ranks of the *unseen* relevant documents, we exclude these 29 relevant documents from our judgment file and recompute the performance of BayesInt and the baseline using the new judgment file. The results are shown in Table 5. Note that the performance of the baseline method is lower due to the removal of the 29 relevant documents, which would have been generally ranked high in the results. From Table 5, we see clearly that using clicked summaries also helps improve the ranks of unseen relevant documents significantly.

Query	BayesInt($\mu = 0, \nu = 5.0$)	
	MAP	pr@20docs
q_2	0.0263	0.100
$q_2 + H_C$	0.0314	0.100
Improve.	19.4%	0%
q_3	0.0331	0.125
$q_3 + H_C$	0.0661	0.178
Improve	99.7%	42.4%
q_4	0.0442	0.165
$q_4 + H_C$	0.0739	0.188
Improve	67.2%	13.9%

Table 5: BayesInt evaluated on unseen relevant documents

One remaining question is whether the clickthrough data is still helpful if none of the clicked documents is relevant. To answer this question, we took out the 29 relevant summaries from our clickthrough history data H_C to obtain a smaller set of clicked summaries H'_C , and re-evaluated the performance of the BayesInt method using H'_C with the same setting of parameters as in Table 4. The results are shown in Table 6. We see that although the improvement is not as substantial as in Table 4, the average precision is improved across all generations of queries. These results should be interpreted as very encouraging as they are based on only 62 *non-relevant* clickthroughs. In reality, a user would more likely click some relevant summaries, which would help bring up more relevant documents as we have seen in Table 4 and Table 5.

Query	FixInt ($\alpha = 0.1, \beta = 1$)		BayesInt ($\mu = 0, \nu = 5.0$)		OnlineUp ($\mu_k = 5.0, \nu = 15, \forall i < k, \mu_i = +\infty$)		BatchUp ($\mu = 0, \nu = 15$)	
	MAP	pr@20docs	MAP	pr@20docs	MAP	pr@20docs	MAP	pr@20docs
q_2	0.0312	0.1150	0.0312	0.1150	0.0312	0.1150	0.0312	0.1150
$q_2 + H_C$	0.0324	0.1117	0.0338	0.1133	0.0358	0.1300	0.0344	0.1167
Improve.	3.8%	-2.9%	8.3%	-1.5%	14.7%	13.0%	10.3%	1.5%
q_3	0.0421	0.1483	0.0421	0.1483	0.04210	0.1483	0.0420	0.1483
$q_3 + H_C$	0.0726	0.1967	0.0766	0.2033	0.0622	0.1767	0.0513	0.1650
Improve	72.4%	32.6%	81.9%	37.1%	47.7%	19.2%	21.9%	11.3%
q_4	0.0536	0.1930	0.0536	0.1930	0.0536	0.1930	0.0536	0.1930
$q_4 + H_C$	0.0891	0.2233	0.0925	0.2283	0.0772	0.2217	0.0623	0.2050
Improve	66.2%	15.5%	72.6%	18.1%	44.0%	14.7%	16.2%	6.1%

Table 4: Effect of using clickthrough data only for document ranking.

Query	BayesInt($\mu = 0, \nu = 5.0$)	
	MAP	pr@20docs
q_2	0.0312	0.1150
$q_2 + H'_C$	0.0313	0.0950
Improve.	0.3%	-17.4%
q_3	0.0421	0.1483
$q_3 + H'_C$	0.0521	0.1820
Improve	23.8%	23.0%
q_4	0.0536	0.1930
$q_4 + H'_C$	0.0620	0.1850
Improve	15.7%	-4.1%

Table 6: Effect of using only non-relevant clickthrough data

5.2.4 Additive effect of context information

By comparing the results across Table 1, Table 2 and Table 4, we can see that the benefit of the query history information and that of clickthrough information are mostly “additive”, i.e., combining them can achieve better performance than using each alone, but most improvement has clearly come from the clickthrough information. In Table 7, we show this effect for the BatchUp method.

5.2.5 Parameter sensitivity

All four models have two parameters to control the relative weights of H_Q , H_C , and Q_k , though the parameterization is different from model to model. In this subsection, we study the parameter sensitivity for BatchUp, which appears to perform relatively better than others. BatchUp has two parameters μ and ν .

We first look at μ . When μ is set to 0, the query history is not used at all, and we essentially just use the clickthrough data combined with the current query. If we increase μ , we will gradually incorporate more information from the previous queries. In Table 8, we show how the average precision of BatchUp changes as we vary μ with ν fixed to 15.0, where the best performance of BatchUp is achieved. We see that the performance is mostly insensitive to the change of μ for q_3 and q_4 , but is decreasing as μ increases for q_2 . The pattern is also similar when we set ν to other values.

In addition to the fact that q_1 is generally worse than q_2 , q_3 , and q_4 , another possible reason why the sensitivity is lower for q_3 and q_4 may be that we generally have more clickthrough data available for q_3 and q_4 than for q_2 , and the dominating influence of the clickthrough data has made the small differences caused by μ less visible for q_3 and q_4 .

The best performance is generally achieved when μ is around 2.0, which means that the past query information is as useful as about 2 words in the current query. Except for q_2 , there is clearly some tradeoff between the current query and the previous queries

Query	MAP	pr@20docs
q_2	0.0312	0.1150
$q_2 + H_Q$	0.0287	0.0967
Improve.	-8.0%	-15.9%
$q_2 + H_C$	0.0344	0.1167
Improve.	10.3%	1.5%
$q_2 + H_Q + H_C$	0.0342	0.1100
Improve.	9.6%	-4.3%
q_3	0.0421	0.1483
$q_3 + H_Q$	0.0455	0.1450
Improve	8.1%	-2.2%
$q_3 + H_C$	0.0513	0.1650
Improve	21.9%	11.3%
$q_3 + H_Q + H_C$	0.0810	0.2067
Improve	92.4%	39.4%
q_4	0.0536	0.1930
$q_4 + H_Q$	0.0552	0.1917
Improve	3.0%	-0.8%
$q_4 + H_C$	0.0623	0.2050
Improve	16.2%	6.1%
$q_4 + H_Q + H_C$	0.0950	0.2250
Improve	77.2%	16.4%

Table 7: Additive benefit of context information

and using a balanced combination of them achieves better performance than using each of them alone.

We now turn to the other parameter ν . When ν is set to 0, we only use the clickthrough data; When ν is set to $+\infty$, we only use the query history and the current query. With μ set to 2.0, where the best performance of BatchUp is achieved, we vary ν and show the results in Table 9. We see that the performance is also not very sensitive when $\nu \leq 30$, with the best performance often achieved at $\nu = 15$. This means that the combined information of query history and the current query is as useful as about 15 words in the clickthrough data, indicating that the clickthrough information is highly valuable.

Overall, these sensitivity results show that BatchUp not only performs better than other methods, but also is quite robust.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have explored how to exploit implicit feedback information, including query history and clickthrough history within the same search session, to improve information retrieval performance. Using the KL-divergence retrieval model as the basis, we proposed and studied four statistical language models for context-sensitive information retrieval, i.e., FixInt, BayesInt, OnlineUp and BatchUp. We use TREC AP Data to create a test set

μ		0	1	2	3	4	5	6	7	8	9	10
$q_2 + H_Q + H_C$	MAP	0.0386	0.0366	0.0342	0.0315	0.0290	0.0267	0.0250	0.0236	0.0229	0.0223	0.0219
	pr@20	0.1333	0.1233	0.1100	0.1033	0.1017	0.0933	0.0833	0.0767	0.0783	0.0767	0.0750
$q_3 + H_Q + H_C$	MAP	0.0805	0.0807	0.0811	0.0814	0.0813	0.0808	0.0804	0.0799	0.0795	0.0790	0.0788
	pr@20	0.210	0.2150	0.2067	0.205	0.2067	0.205	0.2067	0.2067	0.2050	0.2017	0.2000
$q_4 + H_Q + H_C$	MAP	0.0929	0.0947	0.0950	0.0940	0.0941	0.0940	0.0942	0.0937	0.0936	0.0932	0.0929
	pr@20	0.2183	0.2217	0.2250	0.2217	0.2233	0.2267	0.2283	0.2333	0.2333	0.2350	0.2333

Table 8: Sensitivity of μ in BatchUp

ν		0	1	2	5	10	15	30	100	300	500
$q_2 + H_Q + H_C$	MAP	0.0278	0.0287	0.0296	0.0315	0.0334	0.0342	0.0328	0.0311	0.0296	0.0290
	pr@20	0.0933	0.0950	0.0950	0.1000	0.1050	0.1100	0.1150	0.0983	0.0967	0.0967
$q_3 + H_Q + H_C$	MAP	0.0728	0.0739	0.0751	0.0786	0.0809	0.0811	0.0770	0.0634	0.0511	0.0491
	pr@20	0.1917	0.1933	0.1950	0.2100	0.2000	0.2067	0.2017	0.1783	0.1600	0.1550
$q_4 + H_Q + H_C$	MAP	0.0895	0.0903	0.0914	0.0932	0.0944	0.0950	0.0919	0.0761	0.0664	0.0625
	pr@20	0.2267	0.2233	0.2283	0.2317	0.2233	0.2250	0.2283	0.2200	0.2067	0.2033

Table 9: Sensitivity of ν in BatchUp

for evaluating implicit feedback models. Experiment results show that using implicit feedback, especially clickthrough history, can substantially improve retrieval performance without requiring any additional user effort.

The current work can be extended in several ways: First, we have only explored some very simple language models for incorporating implicit feedback information. It would be interesting to develop more sophisticated models to better exploit query history and clickthrough history. For example, we may treat a clicked summary differently depending on whether the current query is a generalization or refinement of the previous query. Second, the proposed models can be implemented in any practical systems. We are currently developing a client-side personalized search agent, which will incorporate some of the proposed algorithms. We will also do a user study to evaluate effectiveness of these models in the real web search. Finally, we should further study a general retrieval framework for sequential decision making in interactive information retrieval and study how to optimize some of the parameters in the context-sensitive retrieval models.

7. ACKNOWLEDGMENTS

This material is based in part upon work supported by the National Science Foundation under award numbers IIS-0347933 and IIS-0428472. We thank the anonymous reviewers for their useful comments.

8. REFERENCES

- [1] E. Adar and D. Karger. Haystack: Per-user information environments. In *Proceedings of CIKM 1999*, 1999.
- [2] J. Allan and et al. Challenges in information retrieval and language modeling. *Workshop at University of Amherst*, 2002.
- [3] K. Bharat. Searchpad: Explicit capture of search context to support web search. In *Proceeding of WWW 2000*, 2000.
- [4] W. B. Croft, S. Cronen-Townsend, and V. Larvrenko. Relevance feedback and personalization: A language modeling perspective. In *Proceedings of Second DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*, 2001.
- [5] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Probabilistic query expansion using query logs. In *Proceedings of WWW 2002*, 2002.
- [6] S. T. Dumais, E. Cutrell, R. Sarin, and E. Horvitz. Implicit queries (IQ) for contextualized search (demo description). In *Proceedings of SIGIR 2004*, page 594, 2004.
- [7] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppim. Placing search in context: The concept revisited. In *Proceedings of WWW 2002*, 2001.
- [8] C. Huang, L. Chien, and Y. Oyang. Query session based term suggestion for interactive web search. In *Proceedings of WWW 2001*, 2001.
- [9] X. Huang, F. Peng, A. An, and D. Schuurmans. Dynamic web log session identification with statistical language models. *Journal of the American Society for Information Science and Technology*, 55(14):1290–1303, 2004.
- [10] G. Jeh and J. Widom. Scaling personalized web search. In *Proceeding of WWW 2003*, 2003.
- [11] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of SIGKDD 2002*, 2002.
- [12] D. Kelly and N. J. Belkin. Display time as implicit feedback: Understanding task effects. In *Proceedings of SIGIR 2004*, 2004.
- [13] D. Kelly and J. Teevan. Implicit feedback for inferring user preference. *SIGIR Forum*, 32(2), 2003.
- [14] J. Rocchio. Relevance feedback information retrieval. In *The Smart Retrieval System-Experiments in Automatic Document Processing*, pages 313–323, Kansas City, MO, 1971. Prentice-Hall.
- [15] X. Shen and C. Zhai. Exploiting query history for document ranking in interactive information retrieval (poster). In *Proceedings of SIGIR 2003*, 2003.
- [16] S. Sriram, X. Shen, and C. Zhai. A session-based search engine (poster). In *Proceedings of SIGIR 2004*, 2004.
- [17] 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.
- [18] 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.
- [19] C. Zhai and J. Lafferty. Model-based feedback in the KL-divergence retrieval model. In *Proceedings of CIKM 2001*, 2001.
- [20] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of SIGIR 2001*, 2001.