An Analysis Framework for Search Sequences

Qiaozhu Mei* Kristina Klinkner Ravi Kumar Andrew Tomkins
School of Information
University of Michigan
Ann Arbor, MI 48103.
qmei@umich.edu
Yahoo! Research
701 First Avenue
Sunnyvale, CA 94089
{klinkner,ravikumar,atomkins}@yahoo-inc.com

ABSTRACT

In this paper we present a general framework to study sequences of search activities performed by a user. Our framework provides (i) a vocabulary to discuss types of features, models, and tasks, (ii) straightforward feature re-use across problems, (iii) realistic baselines for many sequence analysis tasks we study, and (iv) a simple mechanism to develop baselines for sequence analysis tasks beyond those studied in this paper. Using this framework we study a set of fourteen sequence analysis tasks with a range of features and models. While we show that most tasks benefit from features based on recent history, we also identify two categories of "sequence-resistant" tasks for which simple classes of local features perform as well as richer features and models.

Categories and Subject Descriptors. H.3.m [**Information Storage and Retrieval**]: Miscellaneous

General Terms. Algorithms, Experimentation, Measurements

Keywords. Session analysis, Sequential analysis, Query logs

1. INTRODUCTION

Consider a user who visits a search engine and queries for "mustang" then queries for "ford mustang" then queries for "nova." To a human, it is immediately clear that the user is searching for cars, and that the final query of the sequence is for the car produced by automobile manufacturer Chevrolet. However, no major search engine provides a single reference to the Chevy Nova in the first fifty results. As search engines move beyond simple navigational queries to helping users with longer-running tasks, an ability to understand the need behind a user's query, using information about the query sequence, is becoming critical.

Query processing for the "nova" query above could be improved by modeling the previous one or two queries to understand that the context of the session is automobiles. But it could also have been improved by a wider variety of sequence analysis techniques. Perhaps the user has been researching cars for the past month, so the query processing could be improved even if "nova" were the first query of the session. Perhaps the user never clicks on ads when performing automobile queries, so ads should be suppressed. Or perhaps the user often clicks on query suggestions after entering the name of a car, so "chevy nova" should be offered as a suggestion. Or perhaps other users who search for multiple cars in close proximity tend to click on results that offer model information rather than price information. Or perhaps the user entered this same query yesterday and is interested in picking up from where she left off. As these examples show, there are valuable improvements that require access only to the last one or two queries, and other improvements that may require a deeper lens into the user's history, or even an aggregate view into the query-specific sequence behavior of other users.

The body of work surrounding analysis of individual queries is both broad and mature, spanning multiple fields. The associated analysis of query sequences is at a much earlier point in its development. There have been a number of papers presenting ad hoc analysis of user sessions for understanding online behavior [12, 14, 13], improved search query processing [21], and understanding of reformulation behavior [19]. However, as yet there has been little formal work in developing sequential analysis frameworks for such problems; we undertake the development of such a framework.

Our proposed framework (Section 3) captures sequences of user behavior at multiple levels of granularity from sessions to cohesive sub-tasks, blocks of related queries, individual queries, clicks, and eye-tracking fixations. The framework supports (i) a vocabulary to discuss types of features, models, and tasks, (ii) easy feature re-use across problems, (iii) realistic baselines for the sequence analysis problems we study, and (iv) a simple mechanism to develop baselines for sequence analysis tasks beyond those studied in this paper.

We selected 14 distinct search sequence analysis tasks spanning labeling, prediction, and sequence categorization (Section 4), and mapped each one into our framework. Some of these tasks have fully-automated labels such as predicting whether the next click will be on an ad, while others require editorially generated labels such as segmenting a sequence into missions or goals undertaken by the user. We present a set of 42 task-independent base features plus an additional set of "global" features aggregated across multiple sessions by the current user or a broader population of users. These generic features are applicable to a broad range of tasks; we study them to prove the feasibility of feature re-use in such a framework — any individual task could be further improved by careful feature engineering. We compare approaches to our fourteen tasks using a log of 1.2M queries and 17K editorial judgments from a major search engine (Sections 5 and 6). Our results provide a characterization of the appropriate feature types and modeling approaches for a wide range of sequential analysis problems.

*Most of this work was done while the author was visiting Yahoo! Research.

CIKM'09, November 2-6, 2009, Hong Kong, China.

2. RELATED WORK

Search (engine) logs have been well explored to improve the web search relevance and the overall user experience. There has been a large body of work utilizing the search logs to enhance various tasks of a search engine [18, 4, 22, 1, 19, 7]. Because of its effectiveness, search log analysis has attracted even more attention recently, and is gradually becoming its own discipline, with its own set of data, problems, and techniques. Many types of search log data have been explored, including query logs [2], click logs [7], user trails in tool bar logs [20], and eye-tracking logs [8]. Many search tasks have been explored in the past: query categorization [5], query similarity analysis [3], session segmentation [13], and behavior prediction [9]. Also many learning models have been applied to the search log analysis tasks (e.g., random walks [7], Bayesian networks [17], conditional random fields [10], etc.).

There are two drawbacks in many of the current methods of search log analysis. First, they tend to treat the search log as a bag of events without exploiting the temporal relationship between individual events; e.g, for the query classification task, the information available in the user query history (or even the previous few queries issued by the user) is seldom utilized. Second is their inability to model heterogeneous search events and tasks. Many tasks are defined on top of, or involving search events at different levels: queries, clicks, sessions, goals, users, etc. Without a unified framework/model for events at different levels, search log analysis tasks are usually solved in an ad hoc way. When a new task is proposed, a custom data model is developed, a specific set of features is extracted, and a specific method is applied to solve the task. Consequently, the models, features, and methods are less reusable.

There have been some attempts to rectify these drawbacks. For example, the temporal relationship between query events is exploited in the work of Piwowarski et al. [17], who aimed to segment a sequence of queries into goal-related subsequences. Other relevant concepts include sessions [11], missions [13], and goals [13]. Boldi et al. investigated the temporal order of queries, and proposed a model of query-flow graph [6]. The long-range dependence of queries in query logs is studied in [19]. These explorations, however, focus mostly on queries and it is unclear if they can be generalized to other equally interesting search events such as clicks and pupil fixations.

Some very recent work tries to generalize the data models of different tasks in order to model search log objects at multiple levels. Downey et al. proposed an expressive language to model the searching and browsing behavior of users [9], and constructed predictive models for behavior based on such a language. Piwowarski et al. proposed a Bayesian network-based generative model for user search activities [17]. These models, however, make strong assumptions on the search behavior of users; it is hard to adapt the assumptions to a different data scheme. Our work, on the other hand, makes no assumptions about the search behavior of users. We will see that our framework can be applied to many data schemes with a set of simple projection operations.

To the best of our knowledge, there is no single framework for sequence analysis of search logs with the ability to model events at arbitrary levels, guide the development of features, and solve and characterize a gamut of search log analysis tasks in a unified way.

3. THE FRAMEWORK

In existing literature, *search sequence tasks*, or *tasks* are treated on a case-by-case basis and solutions are proposed in an ad hoc manner. Typically, for a given task, the standard recipe involves proposing a specific data model, extracting a chosen set of features from the data, and developing a specific modeling technique to solve the task. It is then difficult to reuse either the custom data model or the custom features for a related task, and there is little guidance on what features tend to be useful for which tasks. In this section we present a concrete and formal framework for tasks that incorporates two key properties of the search sequence data.

(1) Data exists at multiple levels of granularity ranging from user pupil fixations available to an eye-tracking hardware all the way up to semantic partitioning of search sessions into cohesive units. Information available at one level (e.g., clicks) may be valuable for a task defined at another level (e.g., queries). Modeling may exploit only one of these levels, or may use multiple levels simultaneously.

(2) Data is inherently temporal — a sequence of fixations, clicks, or queries evolves over time, and a model may exploit the temporal relationships between elements of the sequence. In literature, the temporal dependency is explored through notions such as "query pairs," "query sessions," or "click history," which have proven useful. We generalize and formalize them into our framework.

3.1 The nested sequence model

A *search event* (or simply, an *event*) denotes a user action with a search sequence dataset at any level of granularity: a query submitted to the search engine, or a click on a result page, or a sequence of queries issued to complete a user's goal. A *search sequence object* (or simply, an *object*) is a sequence of one or more consecutive search events. An object has many *basic features* associated with it, e.g., the timestamp of the first event in this object. Each object also has an associated search *context* that refers to all the information shown to the user during the events comprising the object, e.g., the context of a click event can be the entire list of search results shown to the user when she clicked one of them.

We model the search sequence data as a *nested sequence* of objects (Figure 1). All objects at the same level and under the same higher level object are temporally ordered and form a subsequence. For an object W, let $\langle W_1, \ldots \rangle$ denote the subsequence of objects nested immediately below W and let parent(W) be its parent. Thus,

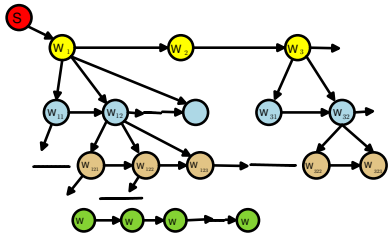

Figure 1: Nested sequence of search sequence objects.

W is completely specified by its basic features, context, and the subsequence of objects below it. Often, we do not distinguish between the basic features and the context, and refer to them together as *non-sequential* information; we adopt this terminology so that we may distinguish such information from various types of *sequential* information computed from elsewhere in the search sequence.

In many cases it may be necessary to collapse an instance of a model to a smaller instance in order to address a task. We refer to an object *projection* as any function that maps objects to objects: e.g., *folding* involves removing a layer of the tree entirely, *flattening* moves children into siblings of the parent, and *skipping* elides elements from a sequence at a particular level.

3.2 A family of tasks

Based on the nested sequence model, it is easy to give a formal definition of tasks that capture most search sequence analysis prob-

lems from the literature. Since most such tasks focus on just one level of objects, we first introduce a family of tasks focused on just one level of the model.

Let $S = \{W_1, W_2, \ldots, \}$ be a set of (possibly flattened) objects. Let $\{X_1, X_2, \ldots, \}$ be sets of features (observations) corresponding to each of the objects — we will formalize the nature of these features in the next section. Let y (resp., y_j) be the target label of S (resp., W_j). We define the following four families of tasks, and anticipate that new families may be added as necessary:

(1) *Sequence classification.* The task is to classify an entire sequence, for instance by determining if the sequence was successful according to some measure, e.g., if a session of queries were navigational or transactional. Formally, one must find y given $X_1, \ldots$.

(2) *Sequence labeling.* The task is to label the individual elements of a sequence, for instance labeling individual queries within a session as navigational, informational or transactional. One must find $y_1, y_2, \ldots$ given $X_1, X_2, \ldots$.

(3) *Sequence prediction.* This task is an online variation of sequence labeling in which the algorithm must assign a label to the current state using information from previous states, for instance by predicting if the next search will result in a click on an advertisement. Formally, one must find y_n given $X_1, \ldots, X_{n-1}$.

(4) *Sequence similarity.* In this task the goal is to compute the similarity between two sequences, for instance to cluster users based on their search behavior. Formally, the task is to compute a similarity between $\langle X_{a,1}, \ldots, X_{a,N_a} \rangle$ and $\langle X_{b,1}, \ldots, X_{b,N_b} \rangle$.

Note that in the nested sequence model, an object contains a (possibly empty) subsequence of lower level objects. Therefore, a sequence labeling task is also related to a sequence classification task, where labeling the objects at a higher level is related to the classification of the corresponding subsequence at the lower level. A fundamental difference between the labeling/classification tasks and the prediction task is that the former refer to an offline learning problem (i.e., the whole sequence, including the target object, is observed), and the latter refers to an online learning problem (i.e., the target object W_n is not fully observed).

In this work we only focus on the first three types of tasks, and leave the fourth type as future work. It is easy to see that this family of tasks generalizes many existing tasks in literature and also permits novel instantiations. In the next section we will introduce many instantiations of tasks.

3.3 Feature definition with the data model

Motivated by the allure of feature re-use across problems, and the ability to produce reasonable low-cost baselines, we now present a categorization of features that will help us understand the difficulty of a current task, and provide us guidance for solving new tasks. We break out features along three dimensions:

(1) If a feature is computed for X_i using disjoint information from that used to compute the same feature for X_j for all $j \neq i$ then we say the feature is *non-sequential*; otherwise, it is *sequential*.

(2) If a feature for X_i may be computed solely from labels y_j then the feature is *easy*; otherwise, it is *rich*.[1]

(3) If a feature uses only information from $X_1, \ldots, X_n$ then it is *local*. If it requires information from other sequences of the same user, it is *personalized*. And if it requires information from other users, it is *universal*.

We now briefly discuss the implications of these dimensions. A non-sequential feature views a sequence task as a stand alone task, and does not incorporate global consistency information. Problems that can be solved effectively using non-sequential features need not employ sequential models; these are ones for which knowledge of the current state trumps knowledge about the past and future.

Easy features are computed from the labels of other states; they capture a generalization of burstiness in the label sequence. Imagine a world in which users very rarely click on the pagination control to generate the next page of results, but once a user does so, she is almost certain to continue paginating through results for many more pages. In such a world, the easy feature capturing whether the current click was a pagination would likely be the most powerful feature to predict whether the next click will be a pagination. If the label sequence is well-modeled by a two-state Markov process, then easy features will be highly effective. Likewise, if the next label can be well-predicted by the current label, again, easy features will be highly effective; e.g., predicting behaviors in which users cycle between two types of events, or predicting behaviors where the next state is a complex function of the three prior states.

Local features employ only information from the object of study, such as the sequence being labeled, but a feature of the current object may be computed using information from other objects in the current sequence, i.e., local features may be sequential. In a search engine context, local features may be computed by a small cache that remembers the current sequence for all active users; the requirements for such a cache are manageable. However, once it becomes useful to know what the same user did when faced with a similar situation in the past, personalized features must be employed, and the cache must have access to all historical information for the user; a small in-memory cache will no longer suffice. Similarly, if it is valuable to incorporate information about how other users react in similar situations then universal features must be employed, and only carefully precomputed aggregates may realistically be loaded in real time. Our experiments show a trade-off between the modeling accuracy of personalized features against the greater coverage of universal features.

4. INSTANTIATING THE FRAMEWORK

We now investigate if our framework is indeed effective for search sequence tasks. To this end we study several instantiations of the framework, motivated by real-world tasks. We will show that the proposed framework can provide a reasonable solution for all such instantiations, and can characterize search sequence tasks by the effectiveness of different types of features. This will not only tell us how to improve the effectiveness of an existing solution, but also give us the guidance on how to approach a new task.

4.1 Data

We consider two different datasets based on a search engine's query logs; these will be our nested data sequences. In both datasets, clicks form the lowest level of the nested sequence. The first dataset, denoted RAW, consists of a sample of around 1.2M queries and corresponding clicks from a day. The second dataset, called EDIT, contains 17,355 queries with corresponding clicks, but with different types of editorial labels provided by human editors. The schemas of the two datasets are shown in Figure 2. It is easy to show that both of them are instantiated from the general nested data model with a few object projections defined in Section 3.1.

In the RAW dataset, each search sequence represents a user's queries and clicks for one day. The highest level of objects is term blocks, where a term block is a subsequence of queries with the same leading word. In the EDIT dataset, each search sequence is a session of events, or a user's events over a shorter period of time. In addition to term blocks, queries, and clicks, this dataset also has two higher levels, namely, missions, and goals, which are editor-labeled. There are also other editorial labels such as the types of

[1]This distinction does not exist for the sequence similarity family of tasks.

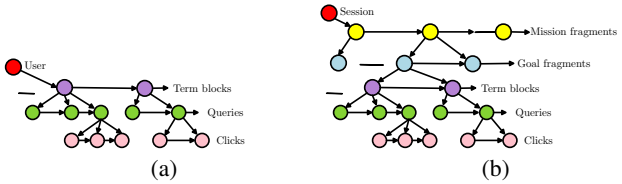

Figure 2: (a) RAW and (b) EDIT schemas.

transitions between two queries (i.e., new, zoom in, pan, zoom out, match, undef) and the query type (i.e., navigational, informational).

We partition each dataset into a training set (80%) and a test set (20%). Since no human labels are needed for sequence prediction tasks, we use the RAW dataset for these. We use the EDIT dataset to evaluate the sequence classification and sequence labeling tasks, which need human labeled judgments. To test the effectiveness of the personalized and universal features, we also collect a larger background dataset which includes all queries and clicks from all search users from the month preceding the start of the RAW dataset.

4.2 Task instantiations

We introduce 14 search sequence tasks, containing eight online prediction tasks and six offline labeling/classification tasks.

Sequential prediction tasks.

Click-level tasks include the following. ALGO: Will the next click be on one of the ten algorithmic search results? NEXTPAGE: Will the next click be on a "pagination" link to load another page of results? NEWQUERY: Is the next click on the current page, or will the user first enter a new query?

Query-level tasks include the following. TERMBLOCK: Is the next query in the same termblock (i.e., has the same first word as the previous query, which is an indicator of the situation that the current information need is not satisfied)? This is computed immediately before the next query arrives, whereas the next four tasks are computed immediately after the query arrives. FIRSTALGO: Is the first click for this query on one of the ten algorithmic search results? HASALGO: Will this query have at least one click on search results? HAS3ALGO: Will this query have at least three clicks on the ten algorithmic search results? ALSOTRY: Is the first click for this query on an "also try" query suggestion?

Sequence labeling tasks.

Query-level tasks include the following. MISSION: Segment a query sequence into missions.[2] GOAL: Segment a query sequence into goals. RESTART: Segment a query sequence into regions of new missions and restarted missions. TRANSTYPE: Label the query with the type of transition from the previous query; this is a multi-class labeling task, which includes labels as the start of a session, lexical editing, zooming in, zooming out, concept matching, new page, panning, and undefined transition.

Sequence classification tasks.

A query-level task is NAV: Classify all queries in a sequence as navigational or informational. A special case of this task is the classification of an individual query. A mission-level task is IFRESTART: Given a mission (sequence of query), whether it is a restarting mission in the current session.

The choice of the level of objects and the target labels are summarized in Table 4.2. These 14 tasks reasonably cover both tasks that have already been studied (i.e., MISSION, GOAL, NAV), and tasks that have not been studied. Although the tasks are diverse, our framework will let us treat them in a unified manner.

[2]For detailed discussion of missions and goals, see, e.g., [13].

Task	Level	Label
ALGO	click	$y_n = \delta[X_n \rightarrow \text{sr}]$
NEXTPAGE	click	$y_n = \delta[X_n \rightarrow \text{pagination}]$
NEWQUERY	click	$y_n = \delta[\text{query}(X_n) \neq \text{query}(X_{n-1})]$
TERMBLOCK	query	$y_n = \delta[\text{head}(X_n) = \text{head}(X_{n-1})]$
FIRSTALGO	query	$\delta[W_{n,1} \rightarrow \text{sr}]$
HASALGO	query	$y_n = \delta[\lvert\text{sr}(X_n)\rvert > 0]$
HAS3ALGO	query	$y_n = \delta[\lvert\text{sr}(X_n)\rvert \geq 3]$
ALSOTRY	query	$y_n = \delta[W_{n,1} \rightarrow \text{alsotry}]$
MISSION	query	$y_j = \{\text{new mission, same mission}\}$
GOAL	query	$y_j = \{\text{new goal, same goal}\}$
RESTART	query	$y_j = \{\text{new mission, same mission, old mission}\}$
TRANSTYPE	query	$y_j = \{\text{new, lexical, zoom in, pan, zoom out, match, new page, undef}\}$
NAV	query	$y = \delta[\text{navigational}]$
IFRESTART	mission	$y = \delta[\text{old mission}]$

Table 1: Task instantions.

Note that the targets (i.e., y, y_j's) of our tasks are discrete non-numeric labels. This makes maximum entropy [16] and conditional random fields [15] natural choices for the prediction and classification tasks and for the sequence labeling task, respectively.

5. EXPERIMENTS: LOCAL FEATURES

Non-sequential features. We compute the following features for the current query: length, number of words, number of search and ad results, frequency, does the query contain a stopword, does the query come from a rewriting action of the search engine, sum of frequencies of queries that are a superset of this query (treated as a bag of words), and country of the user issuing the query. The following are features for the current click: length of the url, pagination number, position (for clicks within the algorithmic search results — includes both position on the current page and position taking into account pagination), and section of the page in which url was present (search, north ads, east ads, header, footer, etc).

Easy-sequential features. For prediction tasks, in addition to non-sequential features, these include labels of the previous (one or two) queries or previous (one or two) clicks.

Rich-sequential features. In addition to the above easy-sequential features, these include the following. For a query object: number of clicks, maximum, minimum, and average positions of the clicks, maximum time between two consecutive clicks, number of "out-of-order" clicks on the ten algorithmic search results (clicks for which an earlier click occurred at a lower-ranked algorithmic search result), and time between the query and the first click. For a query-query pair, the features are: edit distance between the queries (both absolute and normalized by the length of the longest query), time elapsed between the queries, are the queries from the same term block, and is one query a superset or subset of the other in the bag of words representation. For a click-click pair, the features are: time between clicks, difference between positions, and difference in pagination. For a term block object, the features are: number of queries, clicks, and their ratio in the term block, number of out-of-order clicks, and average click position including pagination.

For prediction tasks, additional features for a query object are number and fraction of clicks with a given target label. For a term block object, we add number and fraction of prior clicks and queries with a given target label.

Based on these features, we study the performance of our framework on the tasks in Section 4.2. As a naive baseline, we employ algorithm "guess," in which we uniformly guess the majority label in the training instances.

Table 2 shows results for the online tasks evaluated using precision (pre.), recall (rec.), and accuracy (acc.) of one class of output. These tasks are learned using the MaxEnt model.

Table 3 shows results for the offline tasks. Here, we use precision/recall for binary classification, and accuracy for multi-class classification tasks. Offline tasks are learned using MaxEnt for non-sequential features and CRF for easy and rich sequential features.

Task	metric	guess	non-seq.	easy-seq.	rich-seq.
ALGO	acc.	0.722	0.742	0.757	**0.762**
	pre.	0	0.556	0.598	**0.608**
	rec.	0	0.356	0.384	**0.411**
NEXTPAGE	acc.	0.938	0.939	0.942	**0.943**
	pre.	0	0.542	0.569	**0.591**
	rec.	0	0.071	0.247	**0.259**
NEWQUERY	acc.	0.550	0.784	0.792	**0.793**
	pre.	0	0.772	0.790	**0.816**
	rec.	0	**0.738**	0.734	0.698
TERMBLOCK	acc.	0.898	0.900	0.907	**0.909**
	pre.	0	0.528	0.595	**0.600**
	rec.	0	0.158	0.271	**0.314**
FIRSTALGO	acc.	0.569	0.634	**0.663**	**0.663**
	pre.	0	0.631	0.659	**0.662**
	rec.	0	**0.862**	0.844	0.835
HASALGO	acc.	0.593	0.629	0.644	**0.724**
	pre.	0	0.637	0.653	**0.718**
	rec.	0	0.869	0.853	**0.880**
HAS3ALGO	acc.	**0.923**	0.923	0.923	0.920
	pre.	0	0	0	**0.437**
	rec.	0	0	0	**0.197**
ALSOTRY	acc.	0.939	0.939	**0.958**	**0.958**
	pre.	0	0	0.143	**0.407**
	rec.	0	0	0.0001	**0.001**

Table 2: Local sequential features on online prediction tasks.

From Table 2, we get good performance for all tasks using local sequential features. While the accuracy numbers in some cases are only a modest improvement over a uniform guess (due to the skewed nature of the class), the precision and recall improvements are non-trivial. For most tasks, while using the rich-sequential features results in the best performance, it is important to note that in almost all cases, the performance constantly improves with the sophistication of the features.

Task	metric	guess	non-seq.	easy-seq.	rich-seq.
MISSION	acc.	0.608	0.651	0.692	**0.858**
	pre.	0	0.571	0.675	**0.806**
	rec.	0	0.481	0.431	**0.846**
GOAL	acc.	0.668	0.658	0.669	**0.852**
	pre.	0.668	0.680	0.688	**0.870**
	rec.	1	0.894	0.895	**0.906**
RESTART	acc.		0.616	0.655	**0.783**
TRANSTYPE	acc.		0.406	0.508	**0.670**
NAV	acc.	0.607	0.820	0.843	**0.850**
	pre.	0	0.798	**0.854**	0.853
	rec.	0	0.704	0.724	**0.749**
IFRESTART	acc.	0.707	0.721	0.738	**0.762**
	pre.	0	0.553	**0.667**	0.627
	rec.	0	0.242	0.212	**0.460**

Table 3: Local sequential features on offline labeling tasks.

Table 3 shows more dramatic improvements for offline labeling tasks. For all six tasks, rich-sequential features outperform the rest. For the segmentation tasks such as MISSION and GOAL, rich-sequential features improve the accuracy over non-sequential features as much as 31.8% and 29.4%. Classifying query transition is an even more interesting example, where using rich sequential features improve over non-sequential features by as much as 65%. RESTART is a more difficult task than MISSION, as an additional label "restart session" is involved. In this case, rich-sequential features improve over non-sequential features by 27%. Note that guess, precision, and recall are not applicable to the multi-class labeling tasks RESTART and TRANSTYPE.

Note that MISSION and GOAL were studied in [13] with the same dataset. Although our proposed framework does not rely on custom features or models for these two tasks, its performance is still competitive (0.858 and 0.852 vs 0.844 and 0.873 in [13]).

The range of tasks we consider are sufficient to show a widely divergent set of behaviors. Within the online prediction tasks we study, the following natural clusters emerge.

HAS3ALGO**.** The task of determining whether a user will click on three or more algorithmic search results is difficult and highly skewed, and the learner is not able to beat the majority guess in accuracy. In fact, using non-sequential or easy sequential features, the learned model always predicts negative. Only with rich sequential features is the learner able to find non-trivial solutions, improving precision and recall significantly, with no increase in accuracy.

NEWQUERY**,** FIRSTALGO**.** Non-sequential features provide essentially all the value of richer feature classes for NEWQUERY (predicting a new query) and FIRSTALGO (predicting whether the first click will be on an algorithmic result). For these tasks, information about the current query dominates what our feature set reveals about the context.

ALSOTRY**,** NEXTPAGE**.** Both ALSOTRY and NEXTPAGE show a significant increase in accuracy using easy sequential features, with little additional benefit using rich sequential features. This is because in both the recent past is the best predictor of the immediate future: users searching around for the right query are likely to click on multiple query suggestion links, while users exploring beyond the first page of search results are more likely to explore further.

ALGO**,** TERMBLOCK**,** HASALGO**.** Rich sequential features are beneficial for these tasks. HASALGO shows a 22% decrease in error rate moving from easy to rich sequential features. For NEWQUERY and FIRSTALGO we see a loss of recall when employing rich sequential features. Typically high precision is more important than high recall for these types of tasks, and the more significant gain in precision at the expense of a loss in recall is acceptable; rich sequential features show an improvement in accuracy in both cases. For other tasks in which the recall is very low (e.g., NEXTPAGE, TERMBLOCK, ALSOTRY), it is interesting to see that sequential analysis obtains a huge recall gain ($> 90\%$). The story for offline tasks is more dramatic. Segmenting sequences into missions and goals show large improvements when rich sequential features are added via CRF modeling. Likewise for the multi-class problems RESTART and TRANSTYPE. Most of the gain in predicting navigational queries occurs using non-sequential features alone. And the IFRESTART task shows best precision using easy sequential features, but a doubling in recall using rich ones.

Remarks. The classes of local features we have identified seem to have good discriminating power in separating tasks at different levels of difficulty.

Additionally, while we did not introduce per-task features, the wide range of horizontal features we employ show significant improvements over random guessing, with no additional work required on a per-task basis. This supports our assertion that a framework for search sequence tasks provides a valuable baseline against which to compare more targeted algorithms.

6. EXPERIMENTS: GLOBAL FEATURES

We consider personalized and universal features computed by aggregating simple information over multiple sessions. We study two subclasses of these feature types. The first is the probability of a certain type of click ("pagination," "also try," algorithmic search, etc.) conditioned on the particular query. The second is the probability of a certain type of click conditioned on both the query and the immediately preceding click.

We aggregate these two feature types over all search sequences in the data to provide universal features, and also aggregate them on a per-user basis to provide personalized features. For example, define a local feature $F(q, c) = \delta[\text{label}(c) = \text{"pagination"}]$. Then the corresponding global feature is the average $F_{\text{agg}}(q, c) = \underset{\langle q'=q, c'\rangle \in \{S\}}{E} [F(q', c')]$. If $\{S\}$ is a set of sequences from the current user, then the resulting feature is personalized. If $\{S\}$ is a set of all search sequences, then the feature is universal.

To test the effectiveness of global sequential features, we consider five online prediction tasks: ALGO, NEXTPAGE, HASALGO, FIRSTALGO, and ALSOTRY. We perform the experiments by adding in the global features to the rich local sequential features. The comparison between global features and the performance using local features is shown in Table 4. Results are shown for universal features ("univ"), personalized features ("pers"), and the combination.

Task	metric	local	pers.	univ.	pers.+ univ.
ALGO	acc.	0.762	0.854	0.854	**0.854**
	prec.	0.608	**0.704**	0.692	0.691
	rec.	0.411	0.689	0.700	**0.711**
NEXTPAGE	acc.	0.943	0.954	0.954	**0.954**
	prec.	0.591	0.599	0.597	**0.609**
	rec.	0.259	0.315	**0.319**	0.312
HASALGO	acc.	0.724	0.733	0.758	**0.760**
	prec.	0.718	0.726	0.759	**0.762**
	rec.	**0.880**	0.884	0.868	0.864
FIRSTALGO	acc.	0.663	0.665	0.698	**0.700**
	prec.	0.662	0.668	0.712	**0.715**
	rec.	**0.835**	0.821	0.782	0.794
ALSOTRY	acc.	0.958	0.958	0.958	**0.958**
	prec.	0.407	0.409	0.406	**0.444**
	rec.	0.001	0.001	0.002	**0.002**

Table 4: Global features on online prediction tasks.

As we can see from Table 4, all five tasks gain from the inclusion of global features in terms of accuracy and precision. We do notice a loss of recall for HASALGO (1.9%) and FIRSTALGO (5.8%), but the improvement in precision is much larger (6% and 8%).

Once again, we see that an improvement in precision and accuracy for all tasks, and a loss of recall in two high-recall and low-precision tasks. It is worth mentioning that the global features significantly boost recall for the low-recall tasks: ALGO by 75%, NEXTPAGE by 20%, and ALSOTRY by 100%. For the first two tasks, the improvement from global features is more significant than the improvement from rich-sequential features.

For both ALGO and ALSOTRY, the combination of personalized and universal features performs better than either one alone, suggesting that one should employ per-user data when present, but should otherwise fall back to cross-user data.

7. CONCLUSIONS

We propose a general framework to capture sequences of user behavior at multiple levels of granularity. We demonstrate the merits of our framework by considering 14 diverse sequence analysis tasks and conducting extensive experiments on these tasks. Our framework will let us treat these diversified tasks in a unified manner. Our results provide a characterization of the appropriate feature types and modeling approaches for a wide range of sequential analysis problems. Most of our tasks benefit from features based on recent history. However, some tasks are "sequence-resistant": ALSOTRY, NEXTPAGE, NEWQUERY, FIRSTALGO, etc. It will be important to understand these.

Acknowledgments. We thank Ana-Maria Popescu, Rosie Jones, and Peter Anick for valuable discussions on this work.

8. REFERENCES

[1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR*, pages 19–26, 2006.

[2] R. A. Baeza-Yates, C. A. Hurtado, and M. Mendoza. Query recommendation using query logs in search engines. In *EDBT Workshops*, pages 588–596, 2004.

[3] E. Balfe and B. Smyth. An analysis of query similarity in collaborative web search. In *ECIR*, pages 330–344, 2005.

[4] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *KDD*, pages 407–416, 2000.

[5] S. M. Beitzel, E. C. Jensen, D. D. Lewis, A. Chowdhury, and O. Frieder. Automatic classification of web queries using very large unlabeled query logs. *ACM TOIS*, 25(2):9, 2007.

[6] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. The query flow graph: Model and applications. In *CIKM*, pages 609–618, 2008.

[7] N. Craswell and M. Szummer. Random walks on the click graph. In *SIGIR*, pages 239–246, 2007.

[8] E. Cutrell and Z. Guan. What are you looking for? An eye-tracking study of information usage in web search. In *CHI*, pages 407–416, 2007.

[9] D. Downey, S. T. Dumais, and E. Horvitz. Models of searching and browsing: Languages, studies, and application. In *IJCAI*, pages 2740–2747, 2007.

[10] J. Guo, G. Xu, H. Li, and X. Cheng. A unified and discriminative model for query refinement. In *SIGIR*, pages 379–386, 2008.

[11] D. He, A. Göker, and D. J. Harper. Combining evidence for automatic web session identification. *IPM*, 38(5):727–742, 2002.

[12] B. J. Jansen, D. L. Booth, and A. Spink. Determining the user intent of web search engine queries. In *WWW*, pages 1149–1150, 2007.

[13] R. Jones and K. Klinkner. Beyond the session timeout: Automatic hierarchical segmentation of search topics in query logs. In *CIKM*, pages 853–862, 2008.

[14] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *WWW*, pages 387–396, 2006.

[15] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

[16] K. Nigam, J. Lafferty, and A. Mccallum. Using maximum entropy for text classification. In *IJCAI Workshop*, 1999.

[17] B. Piwowarski, G. Dupret, and R. Jones. Mining user web search activity with layered Bayesian networks or how to capture a click in its context. In *WSDM*, pages 162–171, 2009.

[18] C. Silverstein, M. R. Henzinger, H. Marais, and M. Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999.

[19] J. Teevan, E. Adar, R. Jones, and M. A. S. Potts. Information re-retrieval: Repeat queries in Yahoo's logs. In *SIGIR*, pages 151–158, 2007.

[20] R. W. White, M. Bilenko, and S. Cucerzan. Studying the use of popular destinations to enhance web search interaction. In *SIGIR*, pages 159–166, 2007.

[21] R. W. White and G. Marchionini. Examining the effectiveness of real-time query expansion. *IPM*, 43(3):685–704, 2007.

[22] G.-R. Xue, H.-J. Zeng, Z. Chen, Y. Yu, W.-Y. Ma, W. Xi, and W. Fan. Optimizing web search using web click-through data. In *CIKM*, pages 118–126, 2004.