The Anatomy of a Click: Modeling User Behavior on Web Information Systems

Kunal Punera
Yahoo! Research
701 1st Avenue
Sunnyvale, CA 94089
kpunera@yahoo-inc.com

Srujana Merugu
Yahoo! Research
701 1st Avenue
Sunnyvale, CA 94089
srujana@yahoo-inc.com

ABSTRACT

The ultimate goal of information retrieval science continues to be providing relevant information to users while placing minimal cognitive load on them. The retrieval and presentation of relevant information (say, search results) as well as any dynamic system behavior (e.g., search engine re-ranking) depends acutely on estimating user intent. Hence, it is critical to use all the available information about user behavior at any stage of a search-session to accurately infer the user intent. However, the simplistic interfaces provided by search engines in order to minimize the user cognitive effort, and intrinsic limits imposed by privacy concerns, latency requirements, and other web instrumentation challenges, result in only a subset of user actions that are predictive of the search intent being captured.

In this paper, we present a dynamic Bayesian network (DBN) that models user interaction with general web information systems, taking into account both observed (clicks etc.) as well as hidden (result examinations etc.) user actions. Our model goes beyond the ranked list information access paradigm and gives a solution where arbitrary context information can be incorporated in a principled fashion. To account for heterogeneity in user behavior as well as information access tasks, we further propose a bi-clustering algorithm that partitions users and tasks, and learns separate models for each bicluster. We instantiate this general DBN model for a typical static search interface comprising of a single query box and a ranked list of search results using a set of seven common user actions and various predictive state attributes. Experimental results on real-world web search log data indicate that one can obtain superior predictive performance on various session properties (such as click positions and reformulations) compared to simpler instantiations of the DBN.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation

Keywords

Latent user intents, hidden action sequences, search sessions, user interaction modeling

CIKM'10, October 26–30, 2010, Toronto, Ontario, Canada.

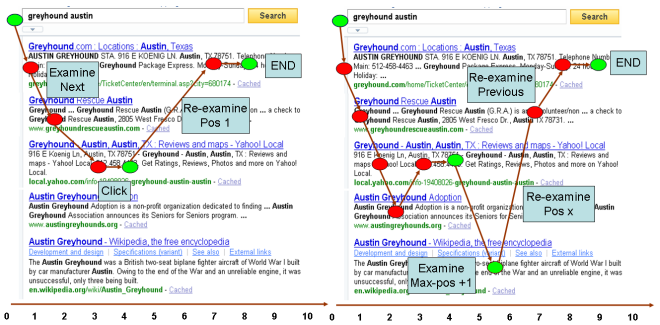

Figure 1.1: Unobserved eye movements (edges to red/dark ovals) and observed actions (edges to green/light ovals) in two search sessions for the query "greyhound austin". Note that the two search sessions result in the same observed user activity but involve very different underlying (hidden) user actions.

1. INTRODUCTION

Web systems are the main source of information for people, especially in developed countries. Many traditionally off-line activities, such as consuming news, communicating with friends, buying/selling, and even dealing with government, have moved almost entirely online. More significantly this migration of people's attention online is expected to lead to a commensurate movement of advertising dollars as well. In large part this is because online advertising lends itself to extensive measurements via implicit feedback from observing user behavior. This feedback can also be used to optimize the content and layout of these websites for the purpose of improving user experience. However, the first step in any such endeavor is to interpret the observed behavior in the context of the user's state of mind (or intent). In this paper we propose a dynamic Bayesian model to achieve this. While our proposed model is applicable to web information systems in general[1], to explain and evaluate our work in concrete terms, in this paper we focus on web search engines.

There are many benefits of learning such a model of user behavior on search engines; here we discuss just two. As a first use case, consider a user with the query "greyhound austin" (Figure 1.1). The query is inherently ambiguous in that it could refer to the dog breed or the buses. Even within the "transportation" intent the user might be interested in the Greyhound ticketing website (result 3) or finding the reviews for the service (result 5). In the absence of further knowledge of the user's intent, search engines tend to hedge their

[1]The model in its abstract form can be applied to web information systems with multi-frame, multi-tabbed views (like Facebook, Twitter, Yahoo!), and also to those with long-term offline interactions (like customer relationship management systems)

bets and show a diverse set of results. If a model could infer the user's intent from behavior, say, by first inferring which results capture the user's attention or by observing the first click, the search engine could dynamically "re-emphasize" the results to improve user experience. As a second use case, consider a situation when a user behavior model, that takes into account query terms, topics, returned results, and even demographic information, predicts identical behavior for two groups of users on a particular search task, when in reality the two groups exhibit very divergent behaviors. We might discover, for instance, that users in one group do not click on the returned results, choosing instead to reformulate their queries. This might be because those users are automated robots, or maybe are confused by the search interface. The ability of a behavior model to predict normal behavior could bring such anomalous groups of activity to the attention of researchers who could then dig deeper to debug the situation.

TECHNICAL CHALLENGES, PRIOR ART, AND MOTIVATION FOR OUR WORK.

While there is some past work on modeling the behavior of visitors to content portals [18], a large number of works target search engines. Due to paucity of space, from among these we only cite some representative works and contrast our approach with them.

As the scenarios mentioned above show we need a model that relates user behavior with user and sessions properties, like underlying intents, result relevance, demographics, etc. Moreover, this model must account for all aspects of user behavior that have been observed in experiments. A particularly important aspect is *position bias*: users tend to click more on higher ranked results no matter the ordering [5, 8]. To account for this bias, Richardson et al. [14] proposed the *examination hypothesis* that users click on a search result after examining each result independently in a biased fashion and determining its relevance. However, this model ignores sequential dependencies in the clicks: a user is unlikely to examine a relevant result later in a list if her information need is satisfied by a higher ranked relevant webpage. To handle this possibility, Craswell et al. [3] proposed the *cascade model*, which describes how the first click happens when the users scan linearly through the ranked list. The cascade model has been enhanced to handle more than one click by Guo et al. [7] and Chapelle and Zhang [2]. While, these methods have shown good results in tasks such as re-ranking search results, they all make a restrictive assumption about the top-down order of examination of results with no skips. However, this assumption does not seem valid (see Figure 1.1 for an example). In query click logs from a major search engine we found that $> 25\%$ of all clicks were *out of order*, meaning that oftentimes users click on a result after having explored and clicked on other results lower down in the ranking. Moreover, there is also evidence from eye-tracking studies done on search results lists [9] and general web pages [12] that suggest that users do not simply scan ranked lists top-down, but that they show a rich behavior of skipping and re-examining results. Unfortunately, privacy concerns, latency requirements, and other fundamental limits of search results page instrumentation, mean that behaviors like examination and skipping of results are not captured. Hence, almost all past works assume a top-down order of examination since this gives them a unique set of user actions for any observed sequences of events (clicks); this assumption, however, might hide a rich set of interaction highly predictive of user intent (see Figure 1.1). Wang et al. [16] propose an approach that allows for skipping of results but this model has many other shortcomings when compared with our approach: it allows only a very restricted set of actions, does not handle arbitrary context features, and assumes clicks on a page are independent (does not respect the *cascade model*). In our work we model users while taking into account a complex set of actions like examinations, skips, and jumps, etc. that are not observed. Moreover, in our model these actions can be dependent on each other in arbitrary ways. While the space of possible actions is huge, we abstract out a rich set of frequently occurring actions from eye-tracking studies and domain knowledge, and incorporate them into our model. As we show in experiments in Section 4, using a richer set of actions helps our system learn a more accurate user behavior model.

A second technical challenge (and criticism of past work) stems from the fact that web/search interfaces are changing dramatically and that it is vital to move away from modeling over a simple ranked list. In fact, current search engines already serve information in multiple blocks: a menu on the left, advertising to the right, and the search results in the middle. Most existing user modeling approaches do not handle this sort of a context in any principled manner. Those existing works that attempt to do so [4, 13] restrict themselves to only observed actions and limited transitions between them. Downey et al. [4] assume a coarse set of fully observed user/system actions and a deterministic user state, while Piwowarski et al. [13] employ a hierarchical HMM, where the hierarchical transitions are always observed; further comparison with HMM-based models [13, 16] is given in Section 3.1. Other simple examples of context that might be incorporated into models to make them more accurate are previous search history of the user, demographic information like gender and age, or even the time of the day. In our work we propose a general model that is not limited to the search ranked list paradigm, and can incorporate information on both observed as well as hidden predictive attributes derived from context.

Finally, a third shortcoming of most prior work is the assumption that user behavior is homogeneous enough to be modeled by a single set of parameters. A related assumption is that search strategies do not vary across tasks. Guo et al. [6] learn two click models by classifying the query into "informational" and "navigational" classes and see improved results. However, the query groups are defined apriori and not based on user behavior. In this work, we use our model of user behavior to cluster users and tasks into groups.

OUR CONTRIBUTIONS.

1. We give a general formulation of the problem of modeling user behavior in interactive web information systems. We then give a dynamic Bayesian network model for these user interactions. This model goes beyond the ranked list paradigm of information consumption and gives a principled way to incorporate domain knowledge about hidden user actions.
2. In order to deal with the heterogeneity in user behavior we give a clustering based formulation that learns a separate model for each user and task group.
3. We instantiate our general model for a search engine setting. The advantages of our model are that it can handle a richer set of user actions, even those that result in unobservable changes to the user state, and incorporate user and task properties in a principled way.
4. We evaluate our model on real-world search engine logs and show that both, the richer set of actions and the clustering capability, are important to accurately model user behavior. Our model improves performance over strong baselines on two real-world tasks: completing user sessions and predicting post-session actions.

ORGANIZATION.

In this section we motivated the problem and provided a detailed comparison with related work. In Section 2 we define the general problem of modeling user behavior on web information systems and give a dynamic Bayesian model for it in Section 3. Finally, in Section 4 we present our evaluation of an instantiation of the general model to web search engines.

2. USER BEHAVIOR MODELING

In this section, we first describe a natural way to represent user sessions on a website in terms of actions/responses. Using this representation, we provide a concrete formulation of the user behavior modeling problem. We make the discussion concrete by giving running examples from the search engine setting.

2.1 User Activity in Web Sessions

A typical single user *web session* [2] consists of a user initiating an interaction with a web system followed by an initial response from the system, and then a series of alternating *user actions* and *web system responses* that could potentially depend on each other.

Example. Figure 1.1 depicts a user's interaction with a search engine while querying "greyhound austin"; we will use this as a running example in this paper. In this case, the initial user action involves issuing the keyword query while the initial system response involves presenting a ranked list of the top-k results relevant to the query (henceforth called *SERP*). The subsequent user actions include glancing over the presented results, closely examining the associated titles and snippets, and clicking on the relevant ones. On the other hand, the system responses can include transferring control to the clicked result pages, actively reordering/grouping the search results, requesting the user to provide further clarification on the initial query, etc. □

In the current work, we take the perspective of the system designer who has complete knowledge/control of the system response policy, and focus on modeling the user behavior conditioned on this policy. Therefore, even though each session is composed of alternating series of user actions and system responses, we mainly focus on the user action sequence, and in particular, *user actions* that are predictive of the user's intent for the session. To minimize the user cognitive effort, most web systems explicitly allow and record (via web protocols) only a small number of *observed user actions* that are highly informative of the user's intent, e.g., querying, clicking, tagging, scoring. Some of these user actions, e.g, querying, tagging, can be further broken to finer-grained actions by considering the associated keywords or phrases. In addition to these observed actions, there also exist a small number of highly common *hidden user actions* that are potentially indicative of user intent, and that are typically not captured by a web system's instrumentation. Some of these, such as mouse movements and scrolling, can be recorded via slightly enhanced techniques (javascript or AJAX). Other hidden actions such as examination patterns or dwell times on content items can only be discovered using more sophisticated monitoring of users, e.g., via eye-tracking studies. Note that since the system is unaware of the hidden actions, it does not provide any response unlike in the case of observed actions.

Example. For the search scenario shown in Figure 1.1, the observed actions (marked in green ovals) include (a_1) `query`, (a_2) `click on the result currently being examined`, and (a_3) `end examination`. The hidden actions involve changing the current result position being examined by the user, and are marked by red ovals. They include (a_4) `examine the next position in result list`, (a_5) `examine the previous position in result list`, (a_6) `re-examine the first result` etc. The user's SERP activity can thus be represented as a sequence of actions; e.g., in the left panel of Figure 1.1 the user's activity is represented as $< a_1\ a_4\ a_4\ a_4\ a_2\ a_6\ a_3 >$. The full set of observed/hidden actions allowed in an instantiation of our model to a web search engine is given in Table 1(b) □

[2]In complex web systems,e.g., instant messaging, a single session might involve multiple users, but we restrict to single user session.

Example. The problem being considered applies to all kinds of web information systems. Consider the problem of designing a system response policy for recommending personalized content for users on web portals [**?**]. In this scenario a user's observed actions might be relatively straightforward, such as `click on a link` (Yahoo!), `post a message` (Twitter), `initiate a friend request` (Facebook), etc. However, the set of hidden actions that need to be modeled might be richer since the content is not just representable as a ranked list. A user might `examine the link to the above/below/left/right of current link`, or in the case of portal such as Yahoo!, `view mail` or `weather`. □

2.2 Problem Definition

Using the above representation of web sessions, our primary goal is learn a stochastic model of the web sessions (i.e., user action sequence) given the user and the system response policy, using data on a large number of sessions. For a new user session, this learnt model can then be used to dynamically infer the user's future behavior given his actions so far, which can be used to optimize the system response.

Let $\mathcal{R}_{sys}$ denote the system response policy. Let $\mathcal{A}^O, \mathcal{A}^H$ and $\mathcal{A} = \mathcal{A}^O \bigcup \mathcal{A}^H$ denote the set of observed, hidden, and all actions respectively, $\mathcal{U} = \{u_i\}_{i=1}^{N_U}$ denote the set of users. Further, let $\mathcal{W} = \{w_i\}_{i=1}^{N_W}$ denote the set of user web sessions where each session w_i is a tuple $(u_i, r_i, \bar{a}_i)$ comprising the associated user u_i, the system response r_i, and the sequence of user actions $\bar{a}_i$, with $\bar{a}_i^O$ denoting the sub-sequence of observed actions. **Example:** in web search, r_i will consist of the query as well as the top-k returned results (SERP), and $\mathcal{A}$ will contain, among others, the actions mentioned in the previous section above, and $\bar{a}_i$ will be the time-ordered sequence of observed as well as hidden actions that the user performs on the SERP. From the rest of the paper we will skip the subscript referring to the user when clear from context.

The problem of modeling user behavior in web information systems can be thought of as three different, but intertwined problems. The first problem, which we call *user behavior model learning*, involves designing a suitable parametric form for the conditional distribution $p_\theta(\bar{a}|u, r)$, and estimating the parameters θ that optimize the "model quality" with respect to the available session data consisting of the tuples $\{(u, r, \bar{a}^O)\}^{N_W}$. Note that the "training" data does not contain the hidden actions the users may have performed.

A natural way to characterize this model quality is in terms of data likelihood[3]. Since the training data consists of only the observed action sequence, the parameters are chosen to optimize the incomplete data likelihood obtained by marginalizing over the hidden action sequence, i.e.,

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \prod_{(u,r,\bar{a}^O)\in\mathcal{W}} \sum_{\bar{a}\in C(\bar{a}^O)} p_\theta(\bar{a}|u, r) \qquad (2.1)$$

where $C(\bar{a}^O)$ is the set of all the user action sequences compatible with $\bar{a}^O$. **Example:** in Figure 1.1, the two different user action sequences represented in the left and right panels are both compatible with the same observed action sub-sequence ($< a1\ a2\ a3 >$).

We refer to the second problem as *user intent inference*. Given the user behavior model $p_{\theta^*}(\bar{a}|u, r)$, a user u, system response policy $\mathcal{R}_{sys}$, and a partial sequence of observed actions $\{\bar{a}_0^O ... \bar{a}_t^O\}$ up to the t^{th} observed action, the problem is to infer the entire action sequence (including future actions and past hidden actions) for that session, i.e.,

[3]Though one might also choose to have alternate formulations based on margin-like measures

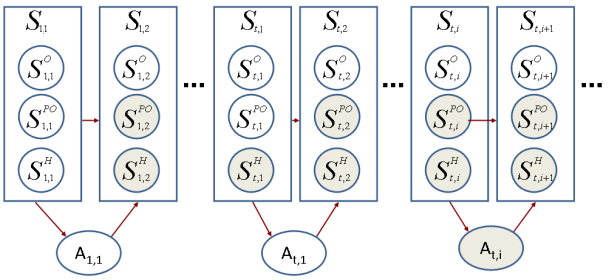

Figure 3.2: The dynamic Bayesian network model. The variables $S_{1,1}$ and $A_{t,1}$ have all attributes fully observed. States $S_{t,1}, t > 1$ only have the components $[S^O_{t,1}, S^{PO}_{t,1}]$ observed while states $S_{t,i}, t > 1, i > 1$ have only $S^O_{t,i}$ observed. Finally, the variables $A_{t,i}, i > 1$ correspond to fully hidden variables.

$$\bar{a}^* = \underset{\bar{a}}{\operatorname{argmax}}\, p_{\theta^*}(\bar{a}|u, \mathcal{R}_{sys}, \bar{a}^O_0 ... \bar{a}^O_t) = \underset{\bar{a}}{\operatorname{argmax}}\, p_{\theta^*}(\bar{a}|u, \mathcal{R}_{sys}) p_{\theta^*}(\bar{a}^O_0 ... \bar{a}^O_t|u, \bar{a}, \mathcal{R}_{sys}) \quad (2.2)$$

using Bayes rule. **Example:** in Figure 1.1, this corresponds to inferring the entire sequence of user actions (both observed and hidden) after observing only the query and, say, the initial one or two clicks.

A third related problem involves *system response optimization* wherein given the learnt user session model parameters θ^*, a partial sequence of observed actions $\{\bar{a}^O_0 ... \bar{a}^O_t\}$ up to the t^{th} observed action, the goal is to identify the future system response r_{t+} that optimizes a target optimization criteria $T(\bar{a})$; we might want to optimize for user satisfaction, number of ad clicks etc. **Example:** while current search engines respond to user clicks by transferring control to clicked pages, one could envision a scenario where after the t^{th} click, the system might decide that moving the 4^{th} result up to the 2^{nd} position improves user experience.

The three problems described above are intimately related, and optimizing the system response policy is the ultimate goal of most web systems. However, key elements of the *system response optimization* problem such as choice of appropriate target optimization criteria (user satisfaction or number of ad clicks etc.) and the space of candidate system responses vary considerably across domains, and require an in-depth exploration of multiple open issues pertaining to lifetime user behavior, mathematical representation of system user interface, etc. Hence, in the current paper, we focus on solving the first two problems: *user behavior model learning* (Equation 2.1) and *user intent inference* (Equation 2.2). In the next section we will construct our user behavior model and give the associated estimation and inference algorithms.

3. BAYESIAN MODEL

In this section, we first consider the desiderata for a user behavior model, and then present our proposed Bayesian approach. For the proposed model, we give solutions to the *user behavior model learning* and *user intent inference* problems in Section 3.3. In Section 3.4, we describe a clustering-based approach that partitions the users and tasks into homogeneous groups and estimates model parameters for each cluster. To aid clarity and concreteness we will continue using user behavior modeling on search engines as our running example.

3.1 Desiderata for a User Behavior Model

Since the primary purpose of the user behavior model is to obtain actionable knowledge of the user intentions so as to adapt the system response, it is desirable to not only figure out the various possible user actions, but the exact sequence in which these are likely to occur. Hence, a sequential model of the action sequence is appropriate.

One possible way to achieve this is by focusing only on the observed actions and modeling the hidden user behavior using a Hidden Markov Model (HMM); this is similar to the approach followed in [13, 16]. The basic assumption is that the user goes through a sequence of hidden states drawn from a discrete set and emits a signal, i.e., observed action along with associated properties, in each state, which can be used to infer the hidden state sequence. Learning such a model, however, involves some challenges. First, the appropriate number of states is not known and can often be large to accommodate observed user actions that are annotated with text (e.g., querying a phrase). Second, the learnt hidden states are difficult to interpret and validate through alternate advanced monitoring of users, e.g., via eye-tracking. This is especially important in real-world systems for guiding exploration of new potentially useful user interfaces. Finally, it is not straightforward to incorporate commonly available domain expertise on frequent hidden actions (e.g., examine next/previous result in the list) and hidden attributes predictive of user intent(e.g., number of results examined so far) into a simple HMM model.

Hence, we consider an alternate model that retains the temporal sequential nature of a Hidden Markov Model, but explicitly incorporates the hidden actions and predictive hidden attributes by considering a continuous state space consisting of both observed and hidden components and the state transitions being influenced by the intermediate actions (hidden and observed).

3.2 Bayesian Model

The fundamental assumption in our user session model is similar to that of HMM, i.e., each session involves a user traversing through a sequence of states following a first order Markov process. However, unlike in a HMM, each state is represented as a vector of attributes, which can be grouped into:
(a) *observed attributes*, whose values are known throughout the session, (e.g., number of results clicked so far)
(b) *partially observed attributes*, whose values are only known in the initial state and immediately preceding an observed action, (e.g., time since query)
(c) *hidden attributes*, whose values are only known at the start of the session, (e.g., number of results examined so far)

The user traverses from one state to another by performing one of the possible actions. These can be *observed actions* (e.g. `click`) or *hidden actions* (e.g. `examine next`). Therefore, a session comprises a fully observed initial state vector (i.e., all attributes are observed), followed by an alternating sequence of user states and actions, and is terminated by an exit state.

Following the notation introduced in Section 2.2, let w denote a web session with the associated user u and the sequence of user actions $\bar{a}$ with the terminal actions assumed to be always observed. To describe the model in further detail, we partition the action sequences into *action segments*; let $L(w)$ denote the number of action segments in session w. Each action segment is a contiguous sub-sequence of $\bar{a}$ beginning with an observed action and ending just before the next observed action. Let the t^{th} segment comprise $L(w,t)$ number of actions and be denoted by $\bar{a}_{\{t\}}$. **Example:** for the session in left panel of Figure 1.1, discussed in Section 2.1, there are two action segments, $< a_1\ a_4\ a_4\ a_4 >$ and $< a_2\ a_6 >$.

Further, let $A_{t,i}, S_{t,i}$ denote the i^{th} user action in the t^{th} segment and the user state immediately preceding this action respectively. Let $S^O_{t,i}, S^H_{t,i}$ and $S^{PO}_{t,i}$ denote the observed, hidden, and partially observed components of the state $S_{t,i}$. By definition, $A_{t,1}$ is an observed action and the partially observed attributes are known for the state $S_{t,1}$. For notational convenience, we denote the state immediately following the t^{th} action segment, as $S_{t,L(w,t)+1} = S_{t+1,1}$.

Figure 3.2 shows the dynamic Bayesian network that captures the sequential dependencies in our user behavior model.

ASSUMPTIONS.

Our model involves two key Markovian assumptions.

(A1) The user action $A_{t,i}$ following any state $S_{t,i}$ is conditionally independent of all the past session history given the state vector. In other words, we assume that the user state representation is rich enough to encode all the historical session information that determines the next user action. A larger set of hidden actions typically necessitates a richer state representation that can discriminate between the various actions.

(A2) The next user state $S_{t,i+1}$ following any state $S_{t,i}$ is conditionally independent of all the past session history given $S_{t,i}$ and the user action $A_{t,i}$ and the known system response policy. This assumption essentially ensures that the user states follow a first order Markov process, but the key difference is the observability of the user action in certain cases.[4].

Based on the above assumptions, the joint distribution of all the state and action variables $\{\{A_{t,i}, S_{t,i}\}_{i=1}^{L(w,t)}\}_{t=1}^{L(w)}$ associated with a session w can be decomposed in terms of two distributions:

1. Action Generation Probability. We capture the conditional probability of a user action given the current state via a parametric distribution $p_\beta(A_{t,i}|S_{t,i})$. Since the actions $A_{t,i}$ take values in a small finite set of actions $\mathcal{A} = \{a_q\}_{N_A}$, we pose this as a multi-class problem [5] and choose an appropriate parametric representation such as a multinomial logit or naive Bayes model. In case of multinomial logit model, we have

$$p_\beta(A_{t,i} = a_q|S_{t,i}) = \frac{\exp(\langle\beta_q, X(S_{t,i})\rangle)}{\sum_{q'=1}^{N_A}\exp(\langle\beta_{q'}, X(S_{t,i})\rangle)},$$

where $X(S_{t,i})$ is a feature vector constructed by suitable transformation of the original state vector by binning and inclusion of interaction features.

2. State Transition Probability. Similarly, we encode the dependencies between the next user state $S_{t,i+1}$ and the current state and action $(S_{t,i}, A_{t,i})$ via a parametric distribution $p_\alpha(S_{t,i+1}|S_{t,i}, A_{t,i})$. Even though there is a strong dependency on the system response policy $\mathcal{R}_{sys}$, we do not need to explicitly model it since $\mathcal{R}_{sys}$ is known and invariant in the current setting. Since $S_{t,i+1}$ is a multi-dimensional vector potentially containing attributes of different types (categorical / ordinal / real), we approximate $p_\alpha(S_{t,i+1}|S_{t,i}, A_{t,i})$ as a product of appropriately chosen multivariate conditional distributions associated with the key attribute types, e.g., Gaussian for real-valued attributes, Poisson for integer-valued attributes and Bernoulli or multinomial models for binary/categorical attributes. In most real systems, however, there is sufficient domain expertise to deterministically compute most attributes of the next state so that, in practice, the parameter α in this model has very few degrees of freedom.

To be precise, let a given session w be composed of W^O and W^H (the observed and hidden portions respectively). Then the joint distribution of the session variables is given by

$$p_\theta(W^O, W^H) = p(S_{1,1})\prod_{t=1}^{L(w)}\prod_{i=1}^{L(w,t)} p_\beta(A_{t,i}|S_{t,i})p_\alpha(S_{t,i+1}|S_{t,i}, A_{t,i}) \tag{3.3}$$

where the conditioning on user u and system $\mathcal{R}_{sys}$ in all distributions is implicit. Since only W^O is observed, the distribution $p(W^0|u, \mathcal{R}_{sys})$ is obtained by marginalizing Equation 3.3 over all the hidden action and state variables.

Example. Before proceeding to learning and inference of this model we give a simple example of how it might map to the web search setting; see Section 4.1 for the full instantiation of this model for a search engine. At any time in the session, a user could be modeled as being in a state given by a six dimensional vector: (P1) `number of results clicked`, (P2) `cumulative match of query with results clicked`, (P3) `time since query`, (P4) `current position`, (P5) `number of results examined`, and (P6) `cumulative match of query with all results examined`. Of these, values of (P1) and (P2) are known at all times (*observed*), (P3) and (P4) are known only when a click is observed (*partially observed*), and (P5) and (P6) are *hidden*. As mentioned earlier, assumption (A1) is equivalent to stating that the six dimensional state attribute vector encodes all the relevant information for determines the next action of the user (e.g., clicking, examining up/down the results). Given these attributes and some others about the initial predisposition of the user, this assumption seems fairly realistic. Furthermore, for this choice of state representation, it is clear that the all the attributes except `time since query` can be computed deterministically using the current user state and the action taken by the user. For example, if P4 has a value 2, then depending on the next action then value can be determined: P4 is 3 if action is `examine next`, or P4 is 1 if action is `examine previous` or P4 remains 3 if action is `click`. Even attributes such as `time since query` can be modeled stochastically with reasonable accuracy. Hence, the assumption (A2) that the next user state depends only on the current state and the action taken is also valid. In the user session shown in the left panel of Figure 1.1, the initial state $S_{1,1}$ after the query is given by the vector $[000000]$, the state after second action (`examine next`) $S_{1,2}$ is $[001111]$, and after the fifth action (`click`) $S_{2,1}$ is $[115333]$. Interested readers can work through this example to familiarize themselves with our user-state model. □

3.3 Learning and Inference

In this section we give algorithms to solve these two problems:

User Behavior Model Learning, i.e., estimating the model parameters $\theta = (\alpha, \beta)$ from the available web session data.

User Intent Inference, i.e., determining the most likely user action sequence (including past and future hidden actions) given the learnt user behavior model and a partially observed web session.

3.3.1 User Behavior Model Learning

Given the parametric form for the joint distribution p of all the variables associated with a web session, and a set of observations on these variables $\{(u, r, \bar{a}^O)\}^{N_w}$ in the form of training data, a natural approach for estimating the model parameters involves maximizing the likelihood of the training data (Equation 2.1). Since the session observations are incomplete, we choose to optimize the incomplete data likelihood obtained by marginalizing over the hidden action and state variables. As discussed earlier, let W^H denote the hidden state/action variables associated with the web sessions, i.e., the hidden actions $(A_{t,i}, i > 1)$ and intermediate state variables $(S_{t,i}, t > 1, i > 1)$, and, let $\tilde{p}$ denote their posterior probability

[4]Note that we define $S_{t,L(w,t)+1} = S_{t+1,1}$ so that the dependencies apply correctly even at the segment boundaries

[5]In certain case, one can have a large number of actions when naively considering actions annotated with keywords (e.g., successive queries in a query chain), but such actions can be reduced to a smaller subset of more meaningful actions (e.g., specialization, generalization) conditioned on the initial action.

function. In order to compute the likelihood, following the analysis in [11], we consider the free energy function, which is defined as the sum of the expected complete log-likelihood and the entropy of the hidden variables with respect to an arbitrary distribution $\tilde{p}$,

$$L(\tilde{p}, \theta) = \underset{\tilde{p}}{E}\left[\log \prod_{w \in \mathcal{W}} p_\theta(W^O, W^H | u, r)\right] + H(\tilde{p}).$$

Since θ consists of the action-generation model parameters (β) and the state-transition model parameters (α) the free energy function can be rewritten as

$$L(\tilde{p}, \alpha, \beta) = \underset{\tilde{p}}{E}\Bigg[\log\Bigg(\prod_{w \in \mathcal{W}} p(S_{1,1}) \prod_{t=1}^{L(w)} \prod_{i=1}^{L(w,t)} p_\beta(A_{t,i}|S_{t,i}) p_\alpha(S_{t,i+1}|S_{t,i}, A_{t,i})\Bigg)\Bigg] + H(\tilde{p}) \tag{3.4}$$

The non-convex nature of the free-energy function $L(\tilde{p}, \alpha, \beta)$ makes it intractable to obtain a global optimum. Hence, we follow a generalized EM-based alternate minimization approach that involves cyclically optimizing the free energy function over each of the arguments $(\alpha, \beta, \tilde{p})$ keeping all the other ones fixed. We now discuss each of these update steps:

Updating β: Optimizing the free energy function in Equation (3.4) with respect to β, we obtain,

$$\beta^* \leftarrow \underset{\beta}{\text{argmax}} \underset{\tilde{p}}{E}\Bigg[\sum_{w \in \mathcal{W}} \sum_{t=1}^{L(w)} \sum_{i=1}^{L(w,t)} \log p_\beta(A_{t,i}|S_{t,i})\Bigg] \tag{3.5}$$

On closer examination, we note that this maximization is equivalent to optimizing the likelihood with respect to the chosen state-action classification model (e.g., multinomial logistic regression/Naive Bayes) given a collection of labeled, weighted training samples $\{\{\{(A_{t,i}, S_{t,i})\}_{i=1}^{L(w,t)}\}_{t=1}^{L(w)}\}^{w \in \mathcal{W}}$ with the weights being determined by the posterior distribution $\tilde{p}$. The model parameters β can thus be learned in a modular fashion using the estimation techniques appropriate for the chosen classification model.

Updating α: The state transition model parameters α can also be updated in a similar fashion,

$$\alpha^* \leftarrow \underset{\alpha}{\text{argmax}} \underset{\tilde{p}}{E}\Bigg[\sum_{w \in \mathcal{W}} \sum_{t=1}^{L(w)} \sum_{i=1}^{L(w,t)} \log p_\alpha(S_{t,i+1}|S_{t,i}, A_{t,i})\Bigg] \tag{3.6}$$

As in the case of updating β case, the objective function to be maximized can be expressed as the data likelihood of a collection of weighted tuples with respect to the chosen state transition regression model. Here each tuple corresponds to a single state transition and contains the response variable (components of the next state $S_{t,i+1}$) and the covariates (components of the prior state $S_{t,i}$ and the preceding action $A_{t,i}$). The tuple weights depend on the posterior distribution $\tilde{p}$. Depending on the choice of the state-transition regression model, one can estimate the parameters α using a suitable algorithm.

Updating $\tilde{p}$. Estimating the posterior distribution of the hidden session variables $\tilde{p}(W^H)$ is much more complex than that of the parameters α, β since each W^H comprises of multiple hidden variables that span multiple session segments and exhibit sequential dependencies. A naive application of EM would involve a random initialization of the posterior distribution of all the latent variables (hidden actions/states) and cyclic optimization over each one of these distributions. This is likely to require multiple restarts and large number of iterations making it computationally infeasible. Hence, we adopt a modular sequential approach that minimizes the computational effort by a judicious propagation of information from each action-segment to the successive one.

We first partition the hidden variables associated with a session W^H into disjoint groups $\{W_t^H\}_{t=1}^{L(w)}$ associated with each segment. For each segment t, W_t^H consists of its hidden actions set $\{A_{t,i}\}_{i=2}^{L(w,t)}$, its partially observed attributes $\{S_{t,i}^{PO}\}_{i=2}^{L(w,t)}$, and the completely hidden attributes $\{S_{t,i}^{H}\}_{i=2}^{L(w,t)}$. Using this decomposition of the hidden session variables, we impose additional constraints on the posterior distribution $\tilde{p}$ by assuming that the session variables associated with the different segments are independent of each other. In other words, we restrict $\tilde{p}(W^H)$ to have the form $\prod_{t=1}^{L(w)} \tilde{p}(W_t^H)$ and separately optimize each of the component distributions.

To perform this optimization, we employ a sequential approach that involves first estimating the posterior distributions of the hidden segment variables $\tilde{p}(W_1^H)$ in the natural order of the segment. The key motivation for this choice is based on the following observation. The initial state $S_{11} = [S_{11}^0, S_{11}^{PO}, S_{11}^H]$ is completely observed. This makes it possible to infer the hidden variables associated with this segment W_1^H (including the hidden component of the state at the end of the first segment $S_{1(L(w,1)+1)}^H = S_{21}^H$) with fairly high accuracy without any additional information from other segments. However, for the later segments, there is a strong dependency between the segment hidden variables W_t^H and the hidden component of the segment initial state S_{t1}^H (which is contained in W_{t-1}^H). An accurate estimate of the distribution of S_{t1}^H is likely to result in better estimates of $\tilde{p}(W_{t-1}^H)$ and hence, it is beneficial to update the $\tilde{p}(W_{t-1}^H)$ in a sequential manner. The actual update step is given by

$$\tilde{p}(W_t^H)^* \leftarrow \underset{\tilde{p}}{\text{argmax}} \underset{\tilde{p}}{E}\Bigg[\log\Bigg(p_\alpha(S_{t+1,1}|S_{t,L(w,t)}, A_{t,L(w,t)}) \prod_{i=1}^{L(w,t)-1} p_\alpha(S_{t,i+1}|S_{t,i}, A_{t,i}) p_\beta(A_{t,i+1}|S_{t,i+1})\Bigg)\Bigg] + H(\tilde{p}) \tag{3.7}$$

Finally, since each segment consists of multiple actions and state transitions, it is also often advantageous to update the parameters (α, β) after each successive segment to obtain better convergence. Algorithm 1 provides the details of such an iterative approach and is guaranteed to converge to a local optimum.

3.3.2 User Intent Inference

Given the user behavior model parameters (α, β) and a partial user interaction sequence $\{\bar{a}_1^O ... \bar{a}_T^O\}$ up to the T^{th} segment, the inference problem consists of determining the most likely choices for the hidden state/action variables up to the T^{th} segment $\{W_t^H\}_{t=1}^T$ as well as the future state/action sequence.

Note that the action sequence up to the T^{th} segment can be easily translated to the set of observed variables in the state $\{W_t^O\}_{t=1}^T$. Hence, we observe that the inference of the most likely hidden variables up to the T^{th} segment is equivalent to estimation of the posterior distributions $\{\tilde{p}(W_t^H)\}_{t=1}^T$ assuming them to be degenerate. Therefore, we employ the same approach as in the case of model learning and sequentially infer the hidden variables as follows:

$$W_t^{H*} \leftarrow \underset{W_t^H}{\text{argmax}} \underset{\tilde{p}(S_{t,1})}{E}\Bigg[p_\alpha(S_{t+1,1}|S_{t,L(w,t)}, A_{t,L(w,t)}) \prod_{i=1}^{L(w,t)-1} p_\alpha(S_{t,i+1}|S_{t,i}, A_{t,i}) p_\beta(A_{t,i+1}|S_{t,i+1})\Bigg] \tag{3.8}$$

For future segments, i.e., when $t > T$, the update step has the same form with the main difference being that the boundary state $S_{t+1,1} = (S_{t+1,1}^O, S_{t+1,1}^{PO}, S_{t+1,1}^H)$ is entirely hidden and has to be included into W_t^H.

3.4 Clustering Users and Tasks

In the user behavior model discussed so far, we have a common action-generation and state-transition model across all user sessions. However, in practice, there is a lot of heterogeneity in user behavior, e.g., eye-tracking studies have shown that unsophisticated web users tend to browse web pages slowly and sequentially, while younger, more net-savvy, users exhibit faster and more irregular scanning behavior. Given that users on the Web are largely anonymous, this difference in behavior cannot readily be modeled by incorporating a feature in the state vector. It is also common for a single user to exhibit different behavior depending on the current task, e.g., it is well known that web users exhibit very different behaviors on navigational and informational searches. This variation across tasks can be partially addressed by incorporating the query words into the state vector. However, this solution is often inadequate for low frequency query words (a huge fraction since queries follow a heavy tail distribution) or polysemous queries (e.g., greyhound).

Before, we describe our approach to capture these behavioral variations due to latent properties of users and tasks, we need to develop the notion of *task*. We take a task q to be a domain dependent property of sessions that capture the intent behind the sessions. For example, in case of search query sessions, the task could be defined as a normalized query (with stemming, stop word removal, synonym resolution etc.). Hence, slightly different queries executed by two different users could be said to accomplish the same task. Given this augmented web session data $\{u, q, r, \bar{a}\}^{N_W}$, we consider a bi-clustering based web session model that involves simultaneously partitioning the users and tasks into disjoint groups that exhibit similar behavior in terms of the associated action-generation model and state-transition models.

Let $\mathcal{U}$, $\mathcal{Q}$ denote the set of users and tasks and $\rho : \{1, \cdots, N_U\} \mapsto \{1, \cdots, k\}$ and $\gamma : \{1, \cdots, N_T\} \mapsto \{1, \cdots, l\}$ denote the mapping from the users and tasks to the user clusters and task clusters respectively. We let the state-transition and action-generation distribution be homogeneous within each user-task bicluster, i.e., we associate with the gh^{th} bi-cluster model parameters α_{gh}, β_{gh}. The conditional likelihood of a web-session $(u, q, r, \bar{a})$, then depends on the bicluster to which user-task pair (u, q) belongs, and is given by

$$p_\theta(W) = p(S_{1,1}) \prod_{t=1}^{L(w)} \prod_{i=1}^{L(w,t)} p_{\beta_{\rho(u)\gamma(q)}}(A_{t,i}|S_{t,i}) p_{\alpha_{\rho(u)\gamma(q)}}(S_{t,i+1}|S_{t,i}, A_{t,i})$$

with implicit conditioning on u,q and R_{sys}.

Model Learning. To learn the above bi-clustering based model, we not only need to estimate the parameters α_{gh}, β_{gh}, but also identify the user and task clustering (ρ, γ) that maximizes the data likelihood, i.e.,

$$L(\tilde{p}, \alpha, \beta, \rho, \gamma) = \underset{\tilde{p}}{E}\left[\log\left(\prod_{w\in\mathcal{W}} p(S_{1,1}) \prod_{t=1}^{L(w)} \prod_{i=1}^{L(w,t)} p_{\beta_{\rho(u),\gamma(q)}}(A_{t,i}|S_{t,i}) p_{\alpha_{\rho(u),\gamma(q)}}(S_{t,i+1}|S_{t,i}, A_{t,i})\right)\right] + H(\tilde{p})$$

As in previous case, we alternately optimize over each of the arguments $(\tilde{p}, \alpha_{gh}, \beta_{gh}, \rho, \gamma)$ keeping the others fixed. The estimation of the posterior distribution $\tilde{p}$ remains identical to that of the single model with the state-transition and action-generation parameters (α, β) chosen to correspond to the bicluster $(\rho(u), \gamma(q))$. The update steps for $(\alpha_{gh}, \beta_{gh})$ also remain identical with the key difference being that only web sessions $(u, q, ., .)$ such that $\rho(u) = g$ and $\gamma(q) = h$ needs to be considered while updating the parameters associated with the gh^{th} bicluster.

To optimize the user-task clustering (ρ, γ), we note that the likelihood function is separable across users and tasks and adopt a k-means like approach. In particular, for each user U, we compute the likelihood of all the sessions of this user for each possible user cluster assignment (keeping all the other parameters including the task clustering fixed), and choose the one that results in the maximum value. The clustering of task Q is performed similarly.

$$\rho(U) \leftarrow \underset{1\le g\le k}{\text{argmax}} \underset{\tilde{p}}{E}\left[\log\left(\prod_{w\in\mathcal{W}, w=(u,q,r,a^O)|u=U} \prod_{t=1}^{L(w)} \prod_{i=1}^{L(w,t)} p_{\beta_{g\gamma(q)}}(A_{t,i}|S_{t,i}) p_{\alpha_{g\gamma(q)}}(S_{t,i+1}|S_{t,i}, A_{t,i})\right)\right] \quad (3.9)$$

$$\gamma(Q) \leftarrow \underset{1\le h\le l}{\text{argmax}} \underset{\tilde{p}}{E}\left[\log\left(\prod_{w\in\mathcal{W}, w=(u,q,r,a^O)|q=Q} \prod_{t=1}^{L(w)} \prod_{i=1}^{L(w,t)} p_{\beta_{\rho(u)h}}(A_{t,i}|S_{t,i}) p_{\alpha_{\rho(u)h}}(S_{t,i+1}|S_{t,i}, A_{t,i})\right)\right] \quad (3.10)$$

Algorithm 1 provides the overall details of the approach and is guaranteed to converge to a local optimum.

Inference. For a known user u and task q, inferring the past and future hidden session information from a partial session given the model and user-task clustering, is identical to the process described in Section 3.3.2 and depends on the model parameters $\alpha_{\rho(u),\gamma(q)}$ and $\beta_{\rho(u),\gamma(q)}$ associated with the user-task bicluster. However, in case of a session involving a new user and/or a new task, we need to first determine the best choice of user-task bi-cluster by maximizing the likelihood over the $k.l$ possible choices and then use the appropriate model parameters for further inference.

Algorithm 1 User Behavior Model Learning and Clustering

Input: Web session data $\{(u, q, r, a^O)\}^{N_W}$, # of user/task clusters (k, l)
Output: Action-generation model parameters $\{\{\beta_{gh}\}_{g=1}^{k}\}_{h=1}^{l}$,
state transition model parameters $\{\{\alpha_{gh}\}_{g=1}^{k}\}_{h=1}^{l}$
user clustering ρ and task clustering γ
Method:
Initialize parameters (α, β) and clustering (ρ, γ) randomly
repeat
Step 1: Posterior Distribution of Hidden Segment Variables
for $n = 1$ to N_W
for $t = 1$ to $L(w_n)$
Compute $\tilde{p}(W_t^H)^*$ using Equation 3.7
use α_{gh} and β_{gh} where $g = \rho(u), h = \gamma(q)$
Step 2: Action-Generation Model Parameters $(\forall [g]_1^k, [h]_1^l)$
Compute β_{gh}^* using Equation 3.5
use only sessions with $\rho(u) = g, \gamma(q) = h$
Step 3: State-Transition Model Parameters $(\forall [g]_1^k, [h]_1^l)$
Compute α_{gh}^* using Equation 3.6
use only sessions with $\rho(u) = g, \gamma(q) = h$
Step 4: User clustering $(\forall U \in \mathcal{U})$
Compute new cluster $\rho(U)$ using Equation 3.9
Step 5: Task clustering $(\forall Q \in \mathcal{Q})$
Compute new cluster $\gamma(Q)$ using Equation 3.10
until *convergence*
return $(\beta, \alpha, \rho, \gamma)$

4. EMPIRICAL EVALUATION

In this section we instantiate our proposed user behavior model to a web search setting and evaluate our learnt model against competitive baselines that use the same learning framework. The ideal evaluation would have used ground truth derived from eye-tracking studies. However, in the absence of such data we evaluate candidate methods on real-world tasks that we would expect to perform with them, completing user sessions and predicting post-session actions.

Attribute type	Attribute names
Static	QueryLength, numTotalResults, numAdsDisplayed, numSpellingCorrections, numRelatedSuggestions
Dynamic Observed	numClickedResults, lastClickedResult, maxClickedResult, cummulativeClickedResultMatch
Dynamic Part-Observed	currPosition
Dynamic Hidden	numExaminedResults, maxExaminedResults, cummulativeExaminedResultMatch

(a) State attributes

Action type	Description
Hidden	examine the next position in the list
Hidden	examine the maxExaminedResults+1 position in the list
Hidden	re-examine the previous position in the list
Hidden	re-examine the top position of the list
Hidden	re-examine a previously examined position in the list
Observed	click on the current position in the list
Observed	stop examining the results list - end session

(b) Allowed actions

Table 4.1: Instantiation of the DBN to a web search engine setting.

4.1 Instantiation to a Search Engine Setting

In our experiments, we define a *search session* as consisting of a query-task, a search results page (*SERP*), and a sequence of user actions on the *SERP*. All the information present on the *SERP* is used to define the state of the user; the attributes that make up the user state vector are listed in Table 1(a). We selected these attributes so as to comprehensively summarize the state of the search session, hence, ensuring that our Markovian assumption about the next action being conditionally independent of session-history is valid. The state attributes are modified by the user actions listed in Table 1(b). The allowable actions were selected to simulate some of the user behavior idioms discernible in eye-tracking studies. For example, before clicking users typically rapidly scan over the result set revisiting some of the already examined results. Moreover, users tend to maintain mental markers in the results-list to jump to; such as the first result, or the maximum result position visited so far.

In terms of learning, we use a multinomial logit model over the state variables for computing the action-generation probabilities. Logistic Regression with a ridge parameter of 10^{-7} was used to learn the parameters of this model. As we can see from Table 1(a) all state variables can be computed exactly given the last state and the latest action taken by the user. Hence, state-transition model is completely deterministic. Finally, in order to make marginalizing over hidden action sequences tractable, we limit the number of possible hidden actions between consecutive clicks to 10, the minimum time required for a hidden action to 1 second, and only use the most probable hidden action sequence found.

DATA USED FOR MODELING AND EVALUATION. We define a *session* to be a user issuing a single query followed by views/clicks on the set of results returned on the first search results page. A click on "next 10 results" or a "query suggestion" is regarded as the end of the current session. The session data used in these experiments comprises all search sessions of around 3500 users over a period of a month in the fall of 2008. The selected set of users was randomly picked from a larger set in which each user had a significant number (> 20) of searches within the month. The bias introduced is unavoidable since learning from fewer sessions per user is impractical and uninteresting. For each session we have the following information available: the query, the set of viewed results, the set of clicked results and the corresponding timestamps. In addition we fetch the content of the results returned by the search engine in the *SERP*. The cosine similarity of the query to the title, the important text(within <h*> tags), and the full-text of each web page is also available to the modeling methods. We selected cosine similarity since it has been shown to be meaningful on text data, however, any similarity measure could have been used. Queries that map to the same string after normalization that includes, lower case, stop word removal, porter stemming, and sorting the terms, are given the same task id. The final data used in our experiments included around 210K sessions involving 120K unique queries issued by around 3500 users. All information that can be used to identify an user was removed from the dataset. The train-test split was 50%; no validation data was needed as we did not set any parameters based on cross-validation.

4.2 Completing Partial Sessions

In this section we evaluate the performance of our approach in predicting completions of partial user sessions. In particular, we attempt to predict the remainder of clicks on a search results page after observing the first k clicks. We posed this as the *user intent inference* problem in Section 2 and gave a solution to it in Section 3.3.2.

OUR APPROACH AND BASELINES. By ALLACTSCLUSTS we refer to the model of user behavior that we learn using the instantiation of our DBN model to the search engine setting as detailed in Section 4.1. This approach models the entire set of allowable user actions and clusters the users and tasks into 4 groups each. Past work [2, 4, 13] in modeling search engine users has not taken into account a complex set of hidden user actions. In order to evaluate the performance gain from modeling these complex hidden actions, we construct a baseline (called ONLYNEXTACT) that restricts users to only the `Examine Next` action along with the two observable actions `Click` and `End Session`. In addition, this baseline does not perform any clustering of users and tasks. All other parameter settings in ONLYNEXTACT are identical to those of ALLACTSCLUSTS. This, along with the fact that ONLYNEXTACT approach uses the entire set of state attributes (in Table 1(a)) to represent users, makes it a competitive representative of past work in the area of user modeling in search engines [2, 4, 13].

In order to further quantify the benefit achieved by performing clustering, we experiment with another version of our approach (called ONECLUST) that models user session using all actions mentioned in Table 1(b), but does not perform any clustering of users and tasks. Other than setting the number of user and query clusters to 1 all other parameter settings are the same as those for ALLACTSCLUSTS.

EVALUATION MEASURES. In order to evaluate the ability of our approach in completing partial user sessions we use two metrics. NUMCLICKS denotes the absolute difference in the number of clicks predicted by the model upon observing a partial session and the actual clicks in the session. This metric scores models based on their ability to predict an aggregate property of search sessions. We use EDITDISTANCE to measure how well the model predicts the actual sequence of click positions. EDITDISTANCE is computed as the minimum number of operations (insertion, deletion, or substitution of a single position) needed to transform our predicted click position sequence into the ground truth sequence. The cost of substitution was set proportional to the absolute difference in the positions that were being substituted. We set the cost of insertion/deletions to be 5 times the cost of substitution of one position since we wanted to penalize a model less if the correct click position was found but was simply misaligned.

RESULTS. Tables 2(a) and 2(b) list the performance numbers for ALLACTSCLUSTS and ONLYNEXTACT. Note that for both NUMCLICKS and EDITDISTANCE measures lower values indicate better performance. We observe that in general the performance of ALLACTSCLUSTS is significantly better than that of ONLYNEXTACT. Moreover, this advantage in performance is especially large when very few clicks of the session have been observed. This implies that the knowledge of user behavior with a full set of hidden actions as

	All queries		Non-navigational queries	
k	AllActsClusts	OnlyNextAct	AllActsClusts	OnlyNextAct
0	**0.81**	0.83	**0.87**	0.90
1	**0.42**	0.56	**0.56**	0.66
2	**0.78**	0.82	**0.83**	0.87
3	0.88	0.90	0.92	0.93
4	0.99	0.99	1.01	1.01

(a) NumClicks measure

	All queries		Non-navigational queries	
k	AllActsClusts	OnlyNextAct	AllActsClusts	OnlyNextAct
0	**4.05**	4.19	**4.33**	4.52
1	**2.09**	2.75	**2.77**	3.26
2	**3.88**	4.08	**4.17**	4.33
3	4.39	4.45	4.57	4.63
4	4.94	4.92	5.04	5.03

(b) EditDistance measure

Table 4.2: Performance of AllActsClusts and OnlyNextAct in terms of NumClicks and EditDistance. k denotes the number of clicks that are observed before session completion is attempted. Performance numbers are not comparable across k as the test-set of queries with $> k$ clicks changes. Numbers that are bold are better with statistical significance at $\alpha < 0.01$

well as user and task cluster memberships are critical to make predictions when nothing is known about user behavior in the session. However, as more information is known after observing multiple clicks ($k \geq 3$) the need for a more complicated model is avoided. These observations hold for both the NumClicks and EditDistance measures.

We noticed that there was a huge improvement in performance after observing 1 click. We speculate that this is due to navigational queries that typically result in a single user click and are known to comprise a large fraction of search traffic (around 20% by some estimates [15]). Hence, after observing the first click, the session ends for an artificially large fraction of queries and this makes the predicting the sequence rather easy. Indeed, even a trivial policy that always ends sessions immediately scores around 0.40 after observing the first click. Hence, in order to ensure that the performance improvement of AllActsClusts in relation to OnlyNextAct is not simply due to these queries, we created a separate set of test queries from which we removed all navigational ones. The navigational queries were determined using an in-house system that uses signals/data that are unavailable to either of the approaches. From the Tables 2(a) and 2(b), we can see that while the performance of both approaches deteriorates on non-navigational queries as expected, AllActsClusts still outperforms OnlyNextAct.

The performance of the OneClust approach is very close to AllActsClusts. In fact, after observing one click ($k \geq 1$), the performance of the two approaches is statistically indistinguishable. However, the OneClust approach has a score of 0.84 and 4.17 for the NumClicks and EditDistance measures respectively when $k = 0$ averaged over all queries. As we can see from the data in Table 4.2, this is significantly worse than the performance of AllActsClusts. This shows that the knowledge of behavioral variations exposed by the user and task clustering are especially useful when zero observed information is available about a session.

4.3 Predicting Post-Session Actions

In this section, we will evaluate whether the completions of sessions performed by our approach are useful for predicting the post-session action of the user. This is a significant problem and has been been studied before [10].

Methodology. From the query logs we determine the next action performed by users after they end the session. The user can

`End`: leave the search engine for more than 30 minutes

`Next`: request another set of N results for the same query

`Reformulate`: issue another query

We then construct a classifier BaseClass that takes as input sets of features derived from the observed clicks of the session and predicts one of the above actions. The set of features used encodes the time taken for all the clicks, the highest position clicked, the number of positions skipped, time used between successive clicks, fraction of out-of-order clicks, etc. Hence, BaseClass uses the observed behavior of the user to make a prediction about what the user would do at the end of the session. We could have constructed a stronger classifier by including additional features [10], but we wanted to isolate the benefit of using unobserved behavioral features, in addition to just the features from observed behavior, on the prediction of post-session user actions.

	After first k clicks				
Approaches	0	1	2	3	4
CompClass	65.4%	63.2%	61.3%	60.6%	61%
BaseClass	33.1%	60.5%	60.5%	58.2%	63.5%

Table 4.3: Classification accuracy of CompClass and BaseClass in predicting post-session user actions.

As in the previous experiment, we used the solution we gave for the *user intent inference* problem in Section 3.3.2 to obtain the complete action sequence of users, given a partial observed one. From the predicted clicks and hidden action sequence we constructed all the features used by BaseClass as well as additional ones like number of hidden actions between clicks, total number of hidden actions in the session, fraction of `Re-examine Previous` actions, fraction of `Re-examine Position x` and `Re-examine Position 1` actions, and the fraction of `Re-examine Position max-examined+1` actions. We used Random Forest [1] to predict post-session actions from features in both BaseClass and CompClass as it has been shown to be very effective in a host of classification tasks. We used the Random Forest implementation in Weka [17] and ran it with the following parameter setting: $numTrees = 10$, $maxDepth = 0$ $(unlimited)$.

Results. The performance of CompClass and BaseClass on predicting post-session user actions via behavioral features are summarized in Table 4.3. The numbers reported are classification accuracy averaged through 10-fold cross-validation. As we can see CompClass out-performs BaseClass by a very large margin when zero clicks of the session have occurred and no observed user behavior is known. In fact, the performance of BaseClass is what a random policy would achieve for a 3-class problem with equal-sized classes. CompClass, on the other hand, uses observed and unobserved behavioral features created using automated completion of sessions by AllActsClusts. Therefore, even when no user behavior has been observed, CompClass achieves a classification accuracy of $> 65\%$. This prediction, available before a user has even interacted with the system, can be extremely useful for search engines in tailoring their search engine results pages.

As more clicks are observed in the sessions, the margin by which CompClass outperforms BaseClass drops to about 4%. Finally, when enough clicks have been observed BaseClass outperforms CompClass as the noise added by features constructed by first predicting the sequence of clicks starts overwhelming the additional benefit of knowing the complete sequence of clicks. Note, that the performance of CompClass is not comparable across different values of k since the set of queries with $> k$ clicks changes drastically.

	Task cluster TC1 *QueryLen* ≤ 2 words *NumClicks* $= 1$ *AdClicks* = no *NextQuery* = end *Navigational* = yes *Commercial* = no	**Task cluster TC2** *QueryLen* > 2 words *NumClicks* > 1 *AdClicks* = yes *NextQuery* = reform, next10 *Navigational* = no *Commercial* = yes
User cluster UC1 *Age* < 20 *Gender* = male *isWin* = no	mapquest sale craigslist	store designer shoe
User cluster UC3 *Age* 30-40 *isWin* = yes *isIE* =yes *AdClicks* = no	song school (district) national (army) us (babies r)	youtube free (download) college (enrollment) ebay

Table 4.4: User and task clusters, properties, and representative terms.

4.4 Interpreting the Discovered Biclusters

METHODOLOGY. Evaluation of clustering is a hard problem, especially when true groupings are unavailable for the test set. Hence, in this section, we present an anecdotal evaluation of one particular clustering of users and queries obtained by ALLACTSCLUSTS. The parameters are set same as before and we obtain 4x4 biclusters.

We interpret the clusters obtained in terms of properties of users and tasks that are placed in them. For the users we use as ground truth self-reported demographic properties *Age* and *Gender*, and properties like *isWin*, *isIE*, *isUS*, derived from the http-agent string. For the tasks we obtained properties like *Navigational* and *Commercial* which are obtained from third-party classifiers that predict whether a task is "navigational" or "commercial", respectively. We "tag" the clusters with user/task properties and the query-words that are over-represented in the sessions in the clusters than in the general population. The degree of over-representation is computed using p-values, which tell us the chance (by random) of seeing a word/property in a particular cluster as many times as is actually observed. We assume that word or property occurrences are governed by a hypergeometric distribution, which models samples drawn from small populations without replacement. Query words and properties that score a p-value of < 0.05 for a cluster are considered its "tags".

The results of this experiment are summarized in Table 4.4. Due to lack of space we only show the "tags" for four user-task biclusters [6]. In the table the first row lists the various session properties associated with members of each task cluster while the first column does the same for each user cluster. The meaning of these properties is evident from the names, but we will explain them as they are encountered. Note that tagging a user or task cluster with a property does not imply that all members of a cluster retain that property value. Moreover, this table also does not imply that there exists a session with the all the properties found over-represented together in a cluster. The checkerboard area of the table contains the query words found over-represented in the biclusters. Sometimes additional words are included in (...) to clarify the original word's sense.

RESULTS. Let us first consider the task clusters. The Task cluster TC1 contains queries that are typically short and often times receive exactly one click. Moreover, the user typically ends her search session after executing queries that fall into TC1. All these properties seem to point to TC1 containing navigational queries. This is further confirmed by the property *Navigational* being set to "yes". On the other hand, Task cluster TC2 contains queries that are typically longer than 2 words and often receive more than 1 click. Furthermore, users issuing queries in this cluster are likely to request more results or reformulate the query. This is a textbook definition of "information queries" [15]. Moreover, the intent behind many of these queries is also commercial as witnessed by the *Commercial* and *AdClicks* properties. Hence, it is apparent that using just the behavioral tendencies of users on tasks, ALLACTSCLUSTS has been able to clearly separate out "navigational" and "informational" tasks.

The first column of Table 4.4 shows two of the clusters found by ALLACTSCLUSTS. As we can see the users are segmented very well by their *Age*. The values for the property *Gender* did not seem to be concentrated in any user cluster expect UC1. This suggests that the behavior on the search results page is more indicative of the age of the person than the gender. Finally, we observe that the user's OS and browser tend to be correlated to their search behavior.

[6]For full results on all 16 clusters please see `www.ideal.ece.utexas.edu/~kunal/papers/cikm10-searchengineusermodel-long.pdf`

5. SUMMARY

We presented a DBN to model user behavior on web information systems. Further, we gave a clustering based formulation to handle heterogeneity due to variations in behavior of users and on tasks. Finally, we gave an instantiation of the DBM for search engines and evaluated our model against strong baselines on real-world data. Our evaluation shows that our model, due its use of a more expressive set of possible user actions and learning of different models for each user-task bicluster, outperforms baselines on two real-world tasks: completing user sessions and predicting post-session actions.

6. REFERENCES

[1] L. Breiman. Random forests. *Machine Learning*, 45(1), 2001.

[2] O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *WWW*, 2009.

[3] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *WSDM*, 2008.

[4] D. Downey, S. Dumais, and E. Horvitz. Models of searching and browsing: languages, studies, and applications. In *IJCAI*, 2007.

[5] Z. Guan and E. Cutrell. An eye tracking study of the effect of target rank on web search. In *CHI*, 2007.

[6] F. Guo, L. Li, and C. Faloutsos. Tailoring click models to user goals. In *WSCD*, 2009.

[7] F. Guo, C. Liu, A. Kannan, T. Minka, M. Taylor, Y.-M. Wang, and C. Faloutsos. Click chain model in web search. In *WWW*, 2009.

[8] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Trans. Inf. Syst.*, 2007.

[9] K. Klöckner, N. Wirschum, and A. Jameson. Depth- and breadth-first processing of search result lists. In *CHI*, 2004.

[10] J. Leskovec, S. Dumais, and E. Horvitz. Web projections: learning from contextual subgraphs of the web. In *WWW*, 2007.

[11] R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*. MIT Press, 1998.

[12] B. Pan, H. A. Hembrooke, G. K. Gay, L. A. Granka, M. K. Feusner, and J. K. Newman. The determinants of web page viewing behavior: an eye-tracking study. In *ETRA*, 2004.

[13] 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*, 2009.

[14] M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: estimating the click-through rate for new ads. In *WWW*, 2007.

[15] D. E. Rose and D. Levinson. Understanding user goals in web search. In *WWW*, 2004.

[16] K. Wang, N. Gloy, and X. Li. Inferring search behaviors using partially observable markov (pom) model. In *WSDM*, pages 211–220, 2010.

[17] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. 1999.

[18] A. Ypma and T. Heskes. Automatic Categorization of Web Pages and User Clustering with Mixtures of Hidden Markov Models. In *WEBKDD*, 2003.