Mining User Web Search Activity with Layered Bayesian Networks or How to Capture a Click in its Context

Benjamin Piwowarski *
University of Glasgow
Scotland, UK
benjamin@bpiwowar.net

Georges Dupret
Yahoo! Labs
Sunnyvale, CA. USA
gdupret@yahoo-inc.com

Rosie Jones
Yahoo! Labs
Sunnyvale, CA. USA
jonesr@yahoo-inc.com

ABSTRACT

Mining user web search activity potentially has a broad range of applications including web result pre-fetching, automatic search query reformulation, click spam detection, estimation of document relevance and prediction of user satisfaction. This analysis is difficult because the data recorded by search engines while users interact with them, although abundant, is very noisy. In this work, we explore the utility of mining search behavior of users, represented by observed variables including the time the user spends on the page, and whether the user reformulated his or her query. As a case study, we examine the contribution this data makes to predicting the relevance of a document in the absence of document content models. To this end, we first propose a method for grouping the interactions of a particular user according to the different tasks he or she undertakes. With each task corresponding to a distinct information need, we then propose a Bayesian Network to holistically model these interactions. The aim is to identify distinct patterns of search behaviors. Finally, we join these patterns to a list of custom features and we use gradient boosted decision trees to predict the relevance of a set of query document pairs for which we have relevance assessments. The experimental results confirm the potential of our model, with significant improvements in precision for predicting the relevance of documents based on a model of the user's search and click behavior, over a baseline model using only click and query features, with no Bayesian Network input.

Key References: [1, 6]

Categories and Subject Descriptors

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

General Terms

Theory, Algorithms, Experimentation

*work conducted while this author was at Yahoo! Research

Keywords

Web Retrieval, Query log analysis, User modelling, Relevance prediction, Clickthrough data

1. INTRODUCTION

There are large sources of implicit information about user web search interests in the Search Engine logs that record user actions. In particular, search engines keep records of their interaction with users in *click-through logs*, which record chiefly a temporary user id (through login or cookies), the queries issued by the user, the results returned by the engine and the resulting user clicks.

User activity models that exploit this data have been proposed in a variety of contexts. Most focus on a specific goal, such as estimating the attractiveness (the perceived relevance) of a search result snippet for a given information need [7] or on predicting the next action a user will perform [1] in order to allow, for example, page result pre-fetching if the model predicts the user will want more search results. Other authors tackle the problem of clustering user activity to present a synthesis of the observed user behavior [20]. Our work differs in that we build generic models of user activity using unsupervised Bayesian Networks, that can then be applied to solve a specific task.

Schematically, when users interact with a search engine with a specific task in mind, they issue one or more queries until they are satisfied with the results, or until they give up. They may then go on to researching a new task, or leave the search engine. We are interested in the sequence of queries related to a single query intent or information need. Following Radlinski and Joachims [15], we name these sequences *query chains*. To the best of our knowledge, there is no work that has proposed a generative *holistic* model of all user-interactions – from search to clicks – that produce chains of queries.

Query chains are an important source of information, since they span the whole search process undertaken by a user: It is a high level goal. If we wish to predict user satisfaction, and hence have a more accurate measurement of the *true* performance of a search engine, it would have to be done at this level of granularity, as the outcome of the whole process reflects the satisfaction of the user more accurately than individual clicks made on a single page of search results returned for a single query. The model we present in this paper is an attempt to build a holistic clustering model at the query chain level which is a first step towards this goal.

In order to validate our model, we must choose a task that can show that some useful information is learned by the model. We chose one which is at the core of web search, the prediction of document relevance, since modern web search engines learn to rank using relevance assessments. Predicting relevance is thus of great

importance since the amount of information that can be collected from user search interactions is orders of magnitudes larger than what can be collected by asking judges. Most approaches [15, 7] using click log data estimate the *attractiveness* of a search result and not the intrinsic document relevance. In our dataset, in 3% of the cases the user clicked on a non relevant document: This means that finding which of the clicked documents were non-relevant is a much harder task than finding the relevant ones. Even if we aggregate, we find that only 35% of the documents that were clicked *at least once* by a user were judged as not relevant in editorial assessments. This high percentage of relevant documents is due to the fact that users do not click *at random*, but most of the time click on relevant links because of the abstract and the title of the document in the result list. Using more complex user models, that take into account the full sequence of user actions, is needed to distinguish attractive from non-relevant documents.

Another advantage of using complex user models in the task of predicting document relevance is that relevance of documents that were clicked a single time for a single query can be estimated with greater accuracy than with other models, since we employ a more complete description of what the user did (how much time they spent reading a search result document, whether they clicked on another document afterwards, etc).

It is difficult to compare our results with previous work, however, since none of them try to predict document relevance from only one interaction with the user, and without using information like the document clickthrough rate. Predicting from few user interactions (ideally only one) is important since the distribution of the number of times a query appears is very skewed, with a long tail: A few queries occur many times, but a very large number occur just once. In a two month log (described next section), we observe that 73% of distinct queries (30% of the total query volume) were issued a single time. If there is just one click and one session, then it is hard to know whether the document was relevant or not, let alone if the search was successful for the user, unless the click is understood in the larger context of the query chain it belongs to. Even in cases where we have only one occurence of a query, estimating how well the user was satisfied is still an issue for evaluation purposes, or to build up a user profile. It is hence important to be able to use such data.

Our contributions include:

1. In Section 2, we present a simple methodology for grouping queries into query chains, based on search logs and in the absence of any hand-labeled data. *Query chains* are sequence of queries related to a single query intent or information need (for example, finding information about camping sites in Paris), and are different from more high level goals (like planning holidays in Paris). We define and experimentally determine a series of thresholds that are used to group together sequences of user actions.

2. To analyze this generated data, our main proposal is a tree-based layered Bayesian Network (BN) framework where latent variables are designed to explain a subset of user actions (Section 3). Each layer corresponds to a given granularity of the search process (query chain level, single query session level, search results inspection process, and document analysis process). Our model tries to predict observations (time after a click before the next action, number of queries, number of web search result pages viewed, etc.) that we extract from the logs and process as described in the above point. This model clusters user chains since it can be used to assign to a series of actions a single label (the most probable one).

3. To validate our approach (steps 1 and 2), we classify clicks as being on relevant and irrelevant documents for a given query. We use a standard machine learning technique, Gradient Boosted Trees [8], to learn from features extracted from the BN and we compare it to a baseline model (Section 4). Note that the aim of these experiments was to validate the BN model, not to obtain the best relevance prediction – more sophisticated models integrating information from various sources would be necessary to achieve this task. Another point is that this validation confirms that the whole process (building query chains and learning the BN model) makes sense.

We believe the BN model can be easily extended and used in a variety of web search settings, including predicting user satisfaction or helping to detect click spam. We compare and situate our work with respect to related ones at the end of the article (Section 5).

2. CONSTRUCTING CHAINS

In this section, we present a methodology for constructing query chains from the information associated with each user interactions with the search engine such as examining a page of search results or clicking on a document url. Previous work in the literature rely on a set of manually labeled data, at least to train a model, or on simple heuristics (30 minutes timeout). In contrast, our approach is fully automatic in the sense that it is based on parameters estimated from the data.

We make use of a sample of click-through logs from a commercial search engine over a period of 57 days. All data was treated in accordance with the search engine's privacy policy, and no attempts were made to match anonymous identifiers to real users. Queries were normalized lexically but not semantically: We removed extra spaces, and added extra quotes when missing. We ensured that all the transformations preserved the result order in the search engine.

The search logs are lists of simple events, each being a tuple made of the temporary user id, the time, the query and the action (either viewing a page of results, or clicking on a document). In this section, we will make use of Figure 1 where each point on line (1) represents one such event for the same temporary anonymous user id, and where the x-axis represents the time. We now proceed to describe each of the three steps that we took to construct the query chains.

Atomic sessions and query chains. We first construct atomic sessions, which are all the events associated with the *same user id* and the *same query string* (i.e. exactly the same sequence of characters) within a reasonable time frame: After a given amount of time we assume the user has started a new atomic session, even though the query remains the same. Line (2) of Figure 1 shows the events grouped by matching query strings. The construction of an atomic user session depends only on one parameter, the time frame span. We used a timeout of thirty minutes. This threshold is quite standard, even though others, ranging from 5 to 120 minutes, have been proposed [19].

From the 57 days logs, we removed sessions containing one or more clicks on ads and empty sessions (sessions without a click) in order to get a more homogeneous dataset. This resulted in about 65 million atomic sessions. We then built chains out of these atomic sessions. The whole process is illustrated in Figure 1.

Here we give a high-level description of our method for constructing query chains, with details in the sub-sections below. We start by concatenating all the atomic sessions for each user, generating a single initial query chain per user[1]. We compute the

[1]For simplicity, we drop overlapping sessions although more sophisticated models could take into account the various chain options to cope with overlap and/or interleaving.

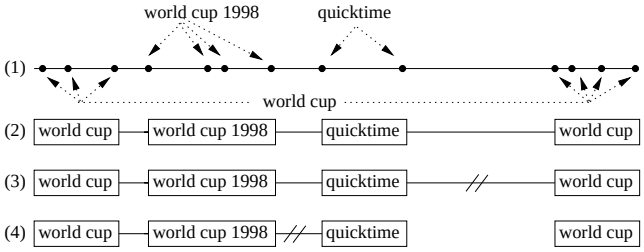

Figure 1: The four steps used to build query chains. Line (1) is a set of events for which the exact associated query string is shown. Line (2) shows the atomic sessions, composed of sets of events of matching query strings (within a 30 minute threshold), while subsequent lines shows the sequence of cuts we make to obtain query chains (time and then similarity).

time delta, i.e. the time difference between two consecutive actions (click, page-view) of an atomic session and we analyze their distribution over all users. We then decide on a global time threshold. This gives us a set of smaller chains for each user. We compute a similarity measure between any two adjacent atomic session query strings inside a chain. Again, from this data we decide on a global query similarity threshold and break up the chains into smaller ones. This process cannot detect interleaved sessions which end up as smaller, disjoint chains.

Time threshold. To set the time threshold we computed the inter-session time distribution on our extracted atomic sessions, that is the time between the last action of a session and the first action of the next session. While this was not verified, we make the intuitive hypothesis that the time between two atomic sessions will be significantly shorter if these are related to the same information need. This suggests that the observed inter-session time is the result of combining two distributions. Upon visual examination, the empirical time difference distribution seems to be composed of a log-normal[2] distribution followed by a power-law. It seems natural to associate the two distributions to the time between related sessions and unrelated sessions respectively.

We fit the observations to a mixture of these two distributions controlled by a logit function $p(Z = \text{new chain}|\Delta, a, b) = \left(1 + e^{a\Delta+b}\right)^{-1}$ that decides which distribution originated the observation, where Δ is the time difference between the two sessions, and a and b are two parameters to set:

$$p(\Delta; a, b, \alpha, \mu, \sigma) = p(z \neq \text{new chain}|\Delta, a, b)\mathcal{LN}(\Delta; \mu, \sigma) + p(z = \text{new chain}|\Delta, a, b)\mathcal{PL}(\Delta; \alpha)$$

where $\mathcal{LN}$ is the log-normal distribution and $\mathcal{PL}$ is the power law distribution. Using the EM procedure to fit the parameters, we obtain the following estimates for the alpha and log-normal distributions: $\alpha \simeq 1.11$, $\mu \simeq 3.44$, and $\sigma \simeq 1.12$.

We use a threshold that covers 99% of the log-normal distribution: $\mathrm{p}\left(\Delta \leq \text{threshold}\,|\Delta \sim \mathcal{LN}(\mu, \sigma)\right) = 0.99$. This leads to a time delta of 7 minutes and 10 seconds. These results are consistent with the findings of [11], where after a 20 minutes *query interval* (and not action interval as in our analysis), most of the searches correspond to new query intents. Line (3) of Figure 1 shows how the last atomic session "world cup" gets disconnected from the chain based on its time delta with the previous session.

[2]We also tried a Poisson distribution, but this did not fit the data well.

Similarity threshold. Using this time threshold, we got a first set of candidate chains. We observed that inside some of these chains, the queries correspond to different intents. In order to identify and segment such chains automatically, we estimated the similarity between the different query strings of adjacent atomic sessions. We computed three kinds of similarity measures, one symmetric and two asymmetric: (i) Symmetric measure: The query and its potential reformulation are transformed into two vectors of character n-grams frequencies of their query strings. The cosine between these two vectors is then used as a proximity measure. (ii) Two asymmetric measures: The degree of inclusion of the potential reformulation into the original query and its counterpart, the degree of inclusion of the original query into the reformulation. The degree of inclusion is computed as the probability that a character n-gram appearing in one query appears in the other.For simplicity and reliability, we did not perform any query processing, and we worked with character tri-grams. For example, "`abc d`" is composed of the tri-grams "`abc`", "`bc `" and "`c d`" (including spaces). This strategy is not perfect: Some chains can be incorrectly broken (eg. "computer shop" and "order laptop") and some can be incorrectly kept (e.g. "New York restaurants" and "New York hotels"). However, this would need semantic query processing and we chose to ignore this issue.

When plotting the value of the cumulative proportion of adjacent pairs of queries joined in the same chain, it can be observed that there is no precise cut point: There is an almost linear relationship between the percentage of query pairs and the maximum similarity value: The former ranges from 40% of pairs to 100% while the latter ranges from 0 to 1. We hence chose the thresholds such that 50% of atomic session pairs are included for any of the similarity thresholds. In practice, we used the following thresholds: 0.43 for the cosine, 0.36 for the n-gram inclusion of the new into the old query, and 0.43 for the n-gram inclusion of the old query into the new. We cut the chain if *all* the similarity values were below the indicated thresholds.

Generated chains. Overall our method produced 1.2 queries (standard deviation 0.6) per chain which is quite different from the 3.27 queries per chain reported in [11] and may be an artifact of our session chunking process. However, the sample used in [11] was small (4690 queries) and the logs we work with most probably contain a larger proportion of navigational or bookmark queries. Broder [3] estimates that around 50% of all queries are navigational or transactional queries that typically involve a single search followed by a few clicks (typically one), and Teevan et al. [19] found that around 24-29% of queries repeated over a long period of time could be considered navigational in that they result in a single repeated click. In order to simultaneously limit the complexity of the models and filter out spam, we filtered out chains containing more than an arbitrary threshold of 50 actions within a chain. This threshold is sufficiently high to capture most of the simple to complex user activities. This leaves us with 19,196,791 chains. In Sections 3 and 4 we will look at how useful these chains are for predicting document relevance.

3. BAYESIAN NETWORK MODEL

User interactions with a search engine seems to fit particularly well the "Test - Operate - Test - Exit" cognitive model [12]. The basic idea is that as humans have a limited short time memory, they

tend to group their actions in a hierarchy of goals: To solve a problem, they use an iterative strategy and move from a goal to one of its sub goals until their purposes are achieved. In our context, the goal of a user is to satisfy their information need. The goal hierarchy is easy to create since the process is heavily constrained by the actual search technology: A user can issue a new search string, look at another page of results or click on a document to inspect its content. At each stage, the user can either return to a higher level goal, perform another task within the same goal or perform a lower level goal. For instance, after inspecting a result page the user might issue a modified query (higher level goal), look at another page (same level) or click on a document to inspect it (lower level goal). The model assumes that a goal is completed before the user starts a new one, implying this model does not account for all possible behaviors as for example a user who opens several tabs in a browser at each stage. A wiser filtering strategy at a chain level would be better, but was not performed, as we believe the amount of data is sufficient to ignore this problem[3], which is at least the case for practical situations as predicting relevance as shown in the next section.

Ideally we would like to cluster the different types of behavior, and assign a label to each of the goals undertaken by the user, as for example "in that page, the user consulted some documents but did not find anything interesting" for a page level goal. Given such labels, it would be easy to classify documents as relevant/not relevant, or to gain insight on the user satisfaction. In practice, it is not realistic to build such categories by hand because there is a lot of variability in the search patterns [20] and because it is difficult for humans to identify patterns in data of such high dimensionality. Using readily available clusterization algorithms is not straightforward as they generally require that the distance between any two sessions be defined. This is difficult in our case as search sessions are very variable in length and structure.

The layered Bayesian Network (BN) model we propose fits naturally the hierarchical nature of the user actions, and offers a mean to clusterize sequences of various granularities corresponding to the different goal levels. Our main objective is to associate a label (a *state* of a random variable in the BN formalism) with the set of user actions at various goal levels – chain, atomic session, page of results and URL examination. Each goal label should be a good summary of the characteristics of the associated actions, like the time spent on a page, the overall relevance, or (better) the user satisfaction. These labels can in turn be used to predict these variables as explained in the next section, or to predict future actions like whether the user will rephrase his or her query. Moreover, by combining various levels of the goal hierarchy, the model provides a very natural way to capture the context around a user click at the different goal levels, from the label of the URL examination goal to the label of the chain goal.

BNs are a probabilistic framework where conditional independence relationships between random variables are exploited, in order to simplify or/and to model decision problems. These relationships are determined by an acyclic graph where a variable is independant of its non-descendants (every node that cannot be reached following the arrows) given its parent(s). The parameters of the network specify how to compute conditional probabilities of any variable given the state of its parent(s) in the graph. The structure and the parameters are enough to completly define the probability distribution over the random variables.

[3]Note that while for clustering that might not be a problem, not filtering surely has more impact on the relevance prediction (next section) task we performed.

In our framework, the structure of the BN is determined by the search session and is not learned, contrarily to the numerical relationships between the labels of the different goals, and between the goals and session observations (e.g. number of clicks) which are discovered automatically at training time. At the click level, the BN could discover that clicks on non relevant documents are associated with a shorter time delta for example. These learned parameters can in turn be used to analyze any new session, thereby enabling to use the model to predict the labels given the observations. Naturally, the labels will be harder to interpret than this example suggests, but they will nevertheless be useful and interpretable to a certain extent as underlined in Section 3.3.

3.1 Network structure: Observed and Latent variables and their relationships

Before going into model details, let us first give two examples. In a first scenario, a user issues a query, looks at the first page of results, then at the second page, clicks on a first document on the second page of results, 3 seconds later on a second document on the same page of results, 5 seconds later requests a third page of results, rewrites the query, looks at the first page of results for the second query and then abandons the search. Intuitively, the user was not satisfied and the documents which were clicked on were (probably) not relevant. Note that considering the click as relevance feedback would lead to an incorrect conclusion. In the second scenario, the user issues a search, clicks on one document and never returns. From this second session we could infer that the document was relevant and the user satisfied their information need. Readers can refer to Figure 2 for a graphical representation of the two sessions. This example is also an illustration of the Bayesian Network structure that will be described later, but for the moment the important part of the figure is the structure (in terms of goals and subgoals), the different observations, and the states (also refered to as goal type or label) associated with the different goals (boxed numbers). It is obvious that even in such a simple example, there is no certainty about the real relevance, and that we can only estimate probabilities of being relevant.

At the click level, we can first try to categorize the clicks depending on various factors like whether it was done shortly after the user saw the search results, whether it was a new click or how much time the user spent analyzing the linked document for example. The click state represents only local information, and it is difficult to draw conclusions from it alone. We could consider two classes, one for clicks occuring shortly after a page view (type 1) and those occuring after a given amount of time, in our case after the inspection of another document (type 2). The states associated to higher level goals effectively define categories of subgoals and click sequences and as such are more informative than the statistics we used to describe clicks: In Figure 2 we have three types of pages: Pages without clicks (page type 1), pages with a succession of state 1 and 2 clicks (page type 2), and pages with one click of type 1 (page type 3). We can then climb one level at a time in the hierarchy and associate three states to searches corresponding each to a different sequence of page goals, and associate two states to the chains, effectively defining two different types of atomic search sequences.

The hierarchy of states provides a context to the individual clicks and permits to distinguish the "bad" clicks of search (a) in Figure 2 from the "good" one in search (b). We expect that after evaluation of the network the most probable hierarchy of states associated with search (a) and (b) will be different.

In the BN network, we don't expect the states to be fully determined by the observations. Rather the distribution over the states

of the different latent goal variables will depend upon the network structure and the observations. Said otherwise, the different states of the latent goal random variables define as many soft clusters of the actions undertaken by the user within that goal. The next list summarizes the different types of latent random variables (i.e. the goal labels) and the observed random variables associated to them. For all but the click goal at the leaves of the hierarchy, the states also summarizes the sequence of nested sub-goals directly following them in the list: **Chain** is the root variable and represents the query chain type. The associated observation is the number of searches issued during the chain. **Search** is a sub-goal at the level of an atomic query session. The observation that we retain at this level is the number of pages of search results requested by the user for this search. **Page** represents the behavior of the user on that search result page. The observation is the number of clicks the user performed on the page. **Click** is associated with the examination of a document in the search result list. The observations associated with it are (1) the time spent examining the document that is clicked on ("delta"), (2) whether the user already clicked earlier on this document during the atomic session ("reclick") and (3) the relevance assessment of the document if available.

Other observations could be associated with each goal. For example, we could attach to a Search goal the time spent by the user looking at search result pages and clicking on documents. However, we choose to discard most of these ancillary observations to ease the model learning. Moreover, some of the time spent on a Search goal is already modeled as part of the Click goal, and this is in fact learned by the BN as explained in the next section. Exploring the best observations to associate to the different goals is future work.

Now that both the latent (goals) and observed random variables have been defined, we describe the structure of the networks. First, we have to link the observations to their corresponding latent variables: We assume that observations are fully determined by the goal they refer to (i.e. number of clicks for a page goal, number of searches for a chain goal): For example, in Figure 2 the number of different atomic searches is dependent on the chain state only. Second, we have to link the goals between themselves. As stated above, a distribution over the sequence of sub-goal states is associated with each goal state. We model this by imposing a dependence of the goal on all the subgoals (vertical arrows between two latent variables) and a first-order dependence of one of the sub-goals onto the next sub-goal (corresponding to the horizontal arrows in the depicted BN). We could use higher order dependence between different subgoals at a same hierarchical level, but as we want to analyze millions of search sessions we choose the common first-order dependence used in many other graphical models to keep the inference tractable.

Note that the particular structure we chose limits the complexity: Once the state of a latent variable associated with a goal G is known, the states of its descendants is independent of its non-descendant. This reflects our hypotheses. For instance, in Figure 2, if we know the state of the second Page variable, then the state of any other Page, of the Search variable and of the Chain variable does not influence the states of the two variables (associated to click goals) made on this page.

3.2 Conditional probabilities and learning

In the previous section, we gave a qualitative overview of our BN model – we showed the structure of a BN with respect to any possible query chain. In this section, we present it quantitatively, that is, we discuss how to compute the actual probabilities and how to learn them. We can define three sets of parameters that were learned using the Expectation-Maximization (EM) algorithm [5].

The first set are the parameters encoding the Chain prior, that is a prior probability distribution on chain states. These parameters are *distinct* for any two chains and thus encodes the membership of each chain to the different latent types.

The second are the parameters encoding the transition probability to a given goal, from a higher level goal state and from the previous same level goal (if this latter exists within the same higher level goal). One parameter is used for each possible configuration. For instance, the probability $\theta^{(search)}_{s,c,s_p} = \mathrm{p}(\text{search} = s|\text{chain} = c, \text{previous search} = s_p)$ for the given latent states s, c and s_p, is the parameter corresponding to the transition from a Search of type s_p to a Search of type s within a Chain of type c. Note that we have a distinct set of parameters for an action which is the first in the sequence (i.e. first search in a chain, first page in a search, or first click in a page). Finally, **the third set** of parameters encode the generation of observations as for example, the number of clicks given a page state. To model discrete variables with a finite number of states (click or re-click, relevant or not relevant) we use a probability table: For example, we have a parameter $\theta^{(reclic)}_{cl}$ corresponding to the probability that the user re-clicked on a document link given that the click state is cl. To model the number of clicks and the time (discretized in second units), as they are not theoretically bounded, we use a Poisson distribution where the parameter λ depends on the corresponding latent state. For example, we have a parameter $\lambda^{(delta)}_{cl}$ for the time to the next action distribution given the click state cl. We chose the Poisson distribution because it is simple and seems to match the empirical distribution of clicks and time, which can both be thought of as rare events. The two last sets of parameters are *shared* among all the constructed BNs ensuring that the state of a given goal always represents the same underlying observations and sequences of sub-goals.

The number of states for the different latent variables is not easy to set and there are computation issues when analyzing millions of sessions. The complexity of the BN grows quickly with the number of possible parent/child configurations. If there are 10 states for the Page variable and 5 for the Click variable, the complexity is $O(10 \times 5 \times 5)$ for Click related inference, and hence learning.

Experimental settings to learn parameters. We experiment with 3 models with different numbers of latent states. The two first models had 5 and 10 latent states at each level. The third one had 5 latent states for all latent variables, except for Clicks which had 20 states. We chose this last configuration because the number of observation types at the click level is larger than at the other levels, and also because we planned to validate the model on relevance prediction: We believed that a model having more Click states would be better suited, and this was indeed confirmed in the experiments. We denote the different configurations BN 5-5, BN 10-10, and BN 5-20. Note that those choices were arbitrary and made in order to have enough information so as to predict relevance while limiting the model complexity.

As discussed in Section 2, we identify 19,196,791 chains over our 57 day usage log sample. We divided the data randomly into two subsets, T_1 with 10,069,661 and T_2 with 9,127,130 chains. We also use a set of independently created manual relevance assessments for 3,627 queries[4] that we randomly divide into two roughly equal subsets (R_1 and R_2) for the experiments. Since the logs contain more queries than the dataset with relevant assessments does,

[4]This dataset is proprietary and was created by drawing a sample of real user queries from the logs

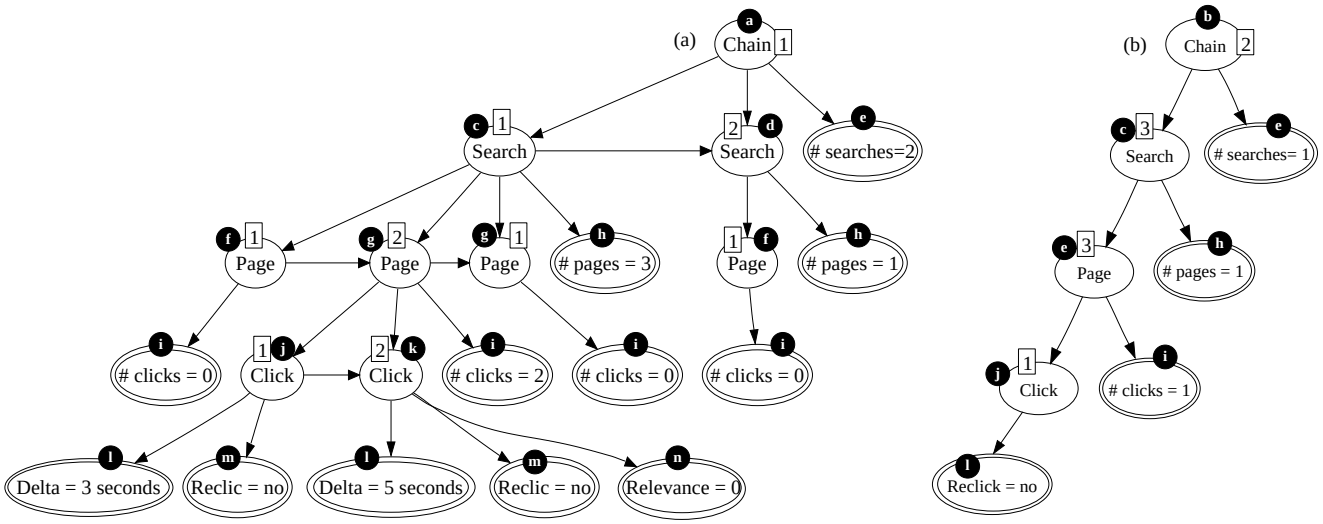

Figure 2: A Bayesian Network example, associated with one query chain. The user has issued two different queries. With respect to the first one, the user has explored three result pages and made two clicks while viewing the second one. After the first (second) click, the user has waited 3 (5) seconds before interacting another time with the search system. The boxed numbers represent one possible labeling of the different states (in the case of a deterministic version of the BN model). Double circled variables are the observation while single circled one are the latent states (goal labels) of the model. Eventually, white labels in black discs are used to distinguish different set of parameters for computing the conditional probabilities in the network.

we are left with 624 and 574 assessed queries for R_1 and R_2 respectively. Note in particular that since we are considering *only query-URL pairs which received a click*, the majority were judged relevant by human editors (97% of instances of query-URL pairs and 65% of unique query-URL pairs).

We train each of the 3 BN configurations on four different sets defined by the four possible combinations of T and R sets. The relevance of a clicked document is not always available, even for a query for which we have some relevance judgments; in those cases, we did not set the relevance variable of the corresponding document. Each training set contains one set of chains and one set of relevance assessments. We use a fixed number of iterations (500) for learning model parameters. This seems to be enough for the convergence of the EM algorithm for all the different models (log likelihood is stable after 100 iterations). Given the high number of examples, we did not check over-tuning issues. As expected, models with higher number of parameters achieve a higher average likelihood. We also observed that BN 10-10 and BN 5-20 reach a comparable data likelihood, although the latter is less complex (BN 5-20 turns out to be slightly better on average).

3.3 Analysis of learned BN

It is interesting to get insight on what the BN actually learns, and if it corresponds to the hypotheses we formulated earlier. Presenting directly the probabilities learned by the models would not mean much since the learning process is unsupervised and the labels do not represent a predefined category. Hence, we rather attempt to give an interpretation to what the three BN models learn.

We can compute expected values of observations which are modeled in the BN – number of searches, pages or clicks, and the expected total time span – for a given chain or search state. Other chain properties which are not directly modeled in the BN can also be computed. For example, the expected query length (e.g. in words) associated with a particular state C of the Chain variable will be computed by taking the sum over all atomic sessions a of the value "query-length(a)$\times p$(chain state of a is C)". This can help interpreting the kind of behavior the different Chain states summarize.

Chain latent states are distributed evenly, each accounting for approximately the same number of observations (20% with 5 latent states or 10% with 10 states). Although it could simply mean that there is no information at this level, we believe that this might signal that chain states might not yet be specialized enough so to explain the observed behavior and that adding more states might be beneficial. In future works we might learn a BN with a high number of latent states and use entropy over posteriors of latent variable states to limit the complexity of the model [2].

Whatever the chain state, three search states are enough to account for 95% of the atomic sessions (search level) for the BN 5-5 and 5-20 configurations. Four states cover 95% when the BN 10-10 configuration is used. The remaining states each account for 1% or less of the atomic sessions. This suggests that the number of search latent states is enough to cluster the atomic sessions.

Interestingly, all configurations of the BN distinguish one particular type of Search behavior quite different from the others. It is characterized by a fairly large number of clicks: Between 7 and 11 compared to 1 for other states, by a large number of page views, between 4 and 7 pages compared to around 1 for other states, and a time span of over 600 seconds while typical values for other states vary between 40 and 90 seconds. On average, 1% of the query sessions belong to this latent state. We conjecture that part of this search behavior may correspond either to click spam, or to a kind of browsing in which most of the documents returned are inspected.

Another finding is that the expected query length (in words), while nearly constant with respect to the Chain states (around 2.16), varies between 2.1 and 3.1 for the Search states for all the learned BNs. This is interesting since the query length is *not* an observation that we include while learning the BN parameters, and there was no reason a priori to observe different lengths for different search states. We might infer that there is a relationship between query search length and the corresponding user search behavior. This is

indeed confirmed by the fact that long search time and large number of browsed result pages are associated with queries of around 2.55 words, slightly over the average of 2.16 words.

Our last observation regards time span. The time spans also varies a lot between Chain states (from 71 seconds to 187 seconds) which is quite surprising given that all the other values remains approximately stable (average query length and number of searches, pages or clicks). This suggests that chains are capturing different types of user behaviors – some of which maybe involve more or less reading of the documents in the results, and that indeed, the number of chain states is not big enough to capture all the information provided by data.

4. PREDICTING RELEVANCE

As with any other clustering method, we have to evaluate the quality of both the chain construction and the clustering indirectly. The results we found show that our method is indeed capturing extra information that could be used in many classification tasks related to web search log analysis. In this section we show that the information learned by the model can be used to predict document relevance.

The basic idea is that a document belongs to a context defined – from the most general to the most specific – by the Chain, Search, Page and Click latent variables. We can also use the link we learned between a Click state membership and its associated document relevance. Typically we explore questions like "can we say something about the relevance of a document that belongs 75% to Click state 1 and 25% to Click state 2, and 60% to Search state 1 and 40% of Search state 5, etc.?" We can attempt to learn a mapping that takes as input the degrees of membership (more precisely, the probability that of a goal state given the observation) of the different latent states associated with the context of a clicked document and as output the relevance assessment for this document.

To this end we use a machine learning technique, gradient boosted trees (GBT) described in [8]. GBT produces a series of weak classifiers (shallow decision trees) whose decisions are merged through a linear combination. We chose this model since it does not necessitate feature normalization and perform very well in general. We also tried SVMs but their use was not straightforward, and did not achieve a performance as good as that of boosted trees. A drawback of GBT is that it does not capture well a continuous relationship between the predicted variable and a factor, but we did not explore other models since results were sufficiently good to validate our model. We use the implementation of [16].

4.1 Features

We first define a baseline model that takes as input a set of features unrelated to the BN, but linked with the observations the BNs were fed with: The idea is to provide *at least* the same set of features used by the BN to infer the distribution over the different latent variables, so that the difference in performance can be attributed to the BN. The more useful features we provide as a baseline, the more confident we are on the BN usefulness. We now describe the features that are used to classify a clicked document within a search session.

Word features. For each word, we use one month of held-out logs. The feature for a given query is the average value of the features over the terms of the query. We use the following term features: **q(not_last)** The number of time a term belonged to a clickless atomic session followed by an atomic session with at least one click. It is an indication of the ambiguity of a term since the user had to reformulate their query. **q(last_empty)** The proportion of time the user did not click in the the last atomic session with a query containing the term. This is an indication of how effective the term is at providing good search results. **q(misspell)** The maximum over all possible terms of the proportion of times the term was substituted in the following query. This is a good indicator of a misspell error. **q(kept)** The percentage of times the term was kept in the next atomic session of a chain. This indicates whether the keyword is useful for the query.

Other baseline features. We also use standard features that are believed to be linked to the relevance of a document: We use the time to next click (when available), the percentage of the whole session time the user spent on the document, the total search time, the position of the document in the ranking, the number of clicks and the ratio of the click number to the total number of clicks (e.g. the second click in a four click session has a 0.5 ratio).

BN features. To estimate the BN parameters for a document, the observed variables are set to the values of the chain the document belongs to, relevance assessments excluded. We then adjust the probabilities of the chain to belong to a particular chain state in order to maximize the likelihood of the observation – we use the EM algorithm, fixing all parameters but the chain prior. We only use information from the latent variables that were associated with the context of the document whose relevance we wanted to predict. For instance, in Figure 2, to predict the second click we would use the information about the relevance variable below, and all the variables appearing on the direct path from the Click variable and the Chain variable. We then compute the following features: (i) The distribution of probability of the variables "Relevance", "Click", "Page", "Search" and "Chain". For example, at the Click level, we have a feature $\mathrm{p}(\text{Click state is i}|\text{observations})$ for each possible state i of the variable. (ii) The set of latent variable states that maximize the likelihood over the whole chain session (note that we again only use the states of the variables that are "ancestors" of the click in the goal hierarchy):

$$C^* = \text{argmax}_{C \in \text{latent states}} \mathrm{p}(C \wedge \text{chain observations})$$

We also use the likelihood of the observation and the likelihood of the ML configuration, that is p(chain observations) and p(chain observations $\wedge C^*$). These two features serve as a measure of confidence over the previous BN features, and can be useful for a machine learning model based on decision trees. The interest of using only the information on the ancestors of a click goal in the goal hierarchy is that we have a *fixed set of features that characterizes the context of a click within a query chain.* This vector of features can be directly used to feed a standard machine learning algorithm.

4.2 Results

The results we report are for the following settings of GBT. The maximum tree depth in the decision trees is set to 4 for BN-based models and 8 for the baseline-based. The difference in depth is expected to counterbalance the fact that BN provides more integrated information, and some of which can be deduced from the baseline model features albeit with a deeper tree depth. We experimented with other depths, but those were sufficient to stabilize the results for both the BN and the baseline models. Each of the four different training sets is composed of an average of 630,000 examples. For practical reasons, we first subsample to 100,000 the number of examples used for training, keeping all the negative (non-relevant) examples. We use 65% of the training set for learning and the remaining 35% to control for the generalization error. The model is trained with the Bernoulli cost function, which corresponds to a

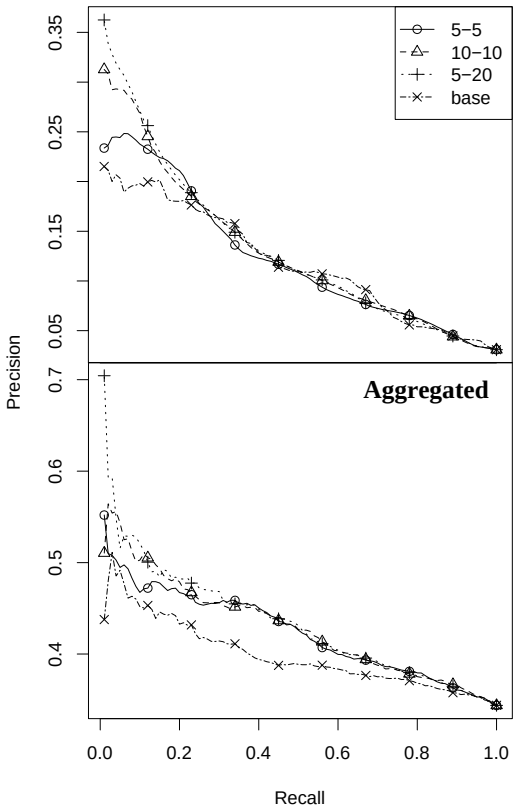

Figure 3: Precision-recall for *non relevant* documents (averaged over the four instances of each model). The proportion of non relevant document among all the documents is .02 for the top graph and .35 for the bottom one. The top graph plots the classifier performance when features associated to a *single* click are used, while the "aggregated" graph reports them when the score of a document for a query is averaged over all the classifier scores.

logistic regression. We also tried the AdaBoost cost function (to learn to rank documents according to their relevance), but it did not perform better. We use a maximum of 15,000 trees and a learning step of 0.01 in agreement with the guidelines [16]. The number of trees is sufficient, since the error on the training set was stable and the generalization error began to rise before the 15,000 tree limit.

We train each model on a dataset composed of one of the logs (say T_1) and one of the relevance assessments sets (say R_1) and we test on the complement dataset: This would be T_2 and R_2 in this example. This gives us 4 possible train/test combinations per model with distinct sessions *and* judged queries. The complete procedure is: (1) Learn the BN parameters using T_1 and R_1. (2) For each document of interest in T_1, generate the features by maximizing the likelihood of the associated chain with respect to the chain classes without using any relevance assessment. (3) Learn the GBT model that maps the baseline and BN features of a query-document pair in T_1 to their associated relevance assessments in R_1. (4) Repeat step 2, but this time using T_2. (5) Use the GBT model to predict R_2 and test the performance.

We rank over half a million observations (for each of the four test sets) of the query-URL pairs (each corresponding to one click on an assessed document) and plot the precision-recall graphs for *non relevant* documents since these documents are much more rare in our relevance assessments set (3% in average); plotting precision-recall graphs for relevant documents would not be as informative since it would produce a near perfect curve. Results reported in Figure 3-top are the interpolated precision-recall curves averaged over the four possible train/test combinations for each different model.

Since a single query-document couple can occur more than once in our data, aggregation techniques to predict relevance from more than one observation are needed so we obtain a single prediction for any query-document pair. We experimented with a basic one, where the score of a pair is given by the mean of its scores; this corresponds to a simple aggregation model where we consider predictions to come from independent and equally important predictors. More sophisticated techniques on these cases would be more adapted. On Figure 3-bottom, we rank distinct (query, URL) pairs using their mean score as computed by the different models, and plot the interpolated precision-recall curve averaged over the four datasets.

Quite intuitively, in both cases the best performing model is BN 5-20, followed by BN 10-10 and BN 5-5, and finally the baseline model. The BN 5-20 model is more stable than the others. The difference between BN 5-20 and the baseline, in terms of stability and performance, is enough to conclude that there is a real gain in using the information from the BN. Another conclusion is that the number of latent states is an important issue.

Looking at the importance of the different features is also informative. We used the relative influence extended by [8] for boosted estimates, approximated as the improvement on the square error as a result of a split in the node(s) using this feature. Figure 4 reports the values for the different models. Note that the relative influences of all the features for *one given model* are normalized and sum to one. For the baseline model, the features associated with the Bayesian networks are absent and hence have zero influence.

Among the different non BN features, the time to next click and search time are the most important. Regarding the latter, we believe it is in relation with the task difficulty and hence the relevance of documents. One feature used both by non BN and BN based models, is the misspell feature. It suggests that wrongly spelled queries are connected to worse results.

Among the different BN model configurations, we see that the influence of the chain class distribution is higher when the BN has 10 states for the Chain variable than when it has only 5 (BN Chain variable in Figure 4). The Click variable is more heavily called for contribution by models having more latent states associated with it. Overall, having more latent states seems to be beneficial for the relevance prediction task.

Eventually, we note that when the features contributed by the BN are used, they turn out to be the most important. A striking example is that the "search time" feature is barely used at all by all the BN based models, while the baseline model does use them heavily. Similarly, the "time to next action" feature is much more used by the baseline than for the BN based models. This means that they have been (somehow) successfully subsumed by the latent states of the BN: The latent states contain more reliable information than some baseline features do, which does validate our approach based on layered BN, given the difference in the tree depths when learning with GBT.

5. RELATED WORK

In this section, we review related work with respect to the three different subtopics in our work: query chains, user activity models, and document relevance prediction. We also discuss the differences between the existing approaches and ours.

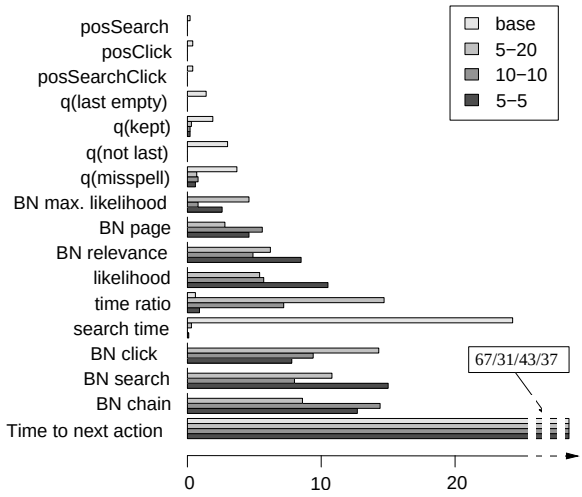

Figure 4: Average importance of the different indicators for the different models (BN 5-5, 10-10, 5-20 and baseline). We only plotted features that were selected by the learning algorithm.

Query chains. Radlinski and Joachims [15] built a Support Vector Machine (SVM) to classify pairs of queries issued within 30 minutes of one another, according to whether they belong to the same chain. They found that a simple heuristic (30 minute threshold) reached a 90% classification accuracy, while the sophisticated learning approach achieved 94% accuracy. Given the difference in algorithmic complexity, they opted for the simple heuristic. However, the general applicability of their analysis is limited since they used logs from the Cornell University library (13,500 web pages) where there is less interaction than with a general purpose web search engine.

He and al. [9] devised an algorithm for grouping queries into chains which checks for deleted or added terms in queries, as well as taking into account the inter-query time interval. They categorized query rewrites into classes such as generalization, specialization and reformulation. Machine learning techniques were then used to create a classifier for these categories. Ozmutlu [14] reported that He et al.'s results, produced from small logs, were not reproducible on their data, and conducted experiments using the same categories on a much bigger dataset using linear regression. They find that time-interval between queries, word-overlap and position in the sequence of queries from the user for that day were predictive of topic changes.

Shen et al. [17] proposed to use the content of the retrieved documents to detect query chains boundaries. While we did not consider this feature as a possible indication of boundaries for complexity reasons (computing a language model for each distinct query is not practical when dealing with large search usage logs), an adaptation of this idea through the use of a similarity matrix between words could be useful.

We also plan in future work to use a machine learning approach to build query chains. In the mean time we used simple heuristics driven by our data since we don't know how generalizable to our own dataset the techniques mentioned above are. We opted for an approach more sophisticated than Radlinski and Joachims since a visual inspection of the chains did not seem to confirm their hypothesis of a 30 minute time threshold.

User Activity Models. User activity models can be broadly divided in three categories: (1) analysis models where the aim is to gain insight into typical user behavior (2) models that try to predict the next user action and (3) models that estimate the attractiveness or perceived relevance of a document independently of layout influence .

White and Drucker [20] mine user search activity, both within the search process and within the surrounding navigation process. They distinguish two classes of users: navigators and explorers. Navigators issue a query and then browse starting from one of the results, while explorers actively use the search engine until they satisfy their user information need. White and Drucker focused on qualitative user analysis through user clustering but they also found out that "the time taken to follow a search trail is independent of the number of queries [in the chain]", which is similar to our observation that different chain states are associated to the same average number of queries but to different query chain durations.

Lau and Horvitz [11] studied the dynamics of user query refinement by trying to find the relationship between two adjacent search actions (new query, reformulation, additional results, etc.), the inter-query interval and the information goal (query category among fifteen pre-established categories). More recently, Downey et al. [6] proposed a model for web search activity. They use a list of features (at various levels, including user ones) and a model in order to compute the probability of the next action performed by the user given the features and the last user action. An example of application of their work is result pre-fetching. Their most important feature for predicting the next user action is the time since start of the atomic session. In both papers, the models are predictive and do not facilitate clustering the different parts of a search trail as our model does. On the other hand, our models would probably be bad at predicting the next user action (since the number of latent state is small) but offers a good summary of a subset of user actions.

Eventually, some models try to decouple the effect of the presentation (ranking of documents, etc.) from the perceived relevance (attractiveness) of a document. For example, the model in [7] explains a click on an URL as caused by both the document attractiveness and the position effect, i.e. the probability that a user considers ("looks at") the document snippet returned by the search engine. Assuming that an unattractive document is often not relevant, which seems intuitive, a presentation debiasing model could be fused with the BN model in order to predict relevance more accurately.

Predicting relevance. Most previous work do not predict relevance but attractiveness, i.e. whether a user is attracted by the document snippet produced by the search engine to click on it, provided he examines it. Two exceptions are Carterette and Jones [4] and Agichtein et al. [1]. Carterette and Jones use logistic regression to predict relevance of documents to a query from the click-through rate of the document at the different positions. Agichtein et al. use a set of features extracted from aggregated user behavior (over the same query-URL pair) and predict relevance with neural networks. Both models relies on a minimum number of sessions of the same query to get accurate predictions. In comparison, the BN model we built can be used to accurately predict relevance of a document, based on the outcome of one query chain only. In this article we chose not to compare to those methods, since the two possible experimental setups would have given an unfair advantage to one of the methods: In the first case, predicting from one observation, we would have a click through rate of 0% or 100% (once) for any position, thus having a very bad estimate of relevance of the document with the Carterette and Agichtien model; using all the observation for one query-document, we would have faced the problem of the

aggregation of our model's predictions, which uses a rather simplistic aggregation method.

Other methods for predicting relevance from query chains include the work of [18] who extended a language model for IR to include the feedback provided by previous interactions with the search engine. In this model, the query language model is defined as a mixture of language models, one using the query terms only and the other using the sessions of similar past queries (more precisely, to the language model of the documents clicked or not within these sessions). The purpose of this model is to include past information to answer a new user's query contrarily to our approach which is more focused on building a representation of the search process and on the relevance of a document within this session.

Compared to previous approaches, our method is the only one where a retrieved document, for the same query string, can be judged relevant or not depending on the context. This suits well the observation of various works on relevance where relevance to a user and relevance to a subject or topic are distinguished [13]. This feature could be used for user personalization.

Finally, small scale user studies performed by Kelly and al. [10] studied the relationship between the time spent on a page and the relevance of a document, but did not find any strong consistent relationship. However, in [10] the authors suggested that the time spent should be taken into account in the context of the other user actions; our own findings show that this is indeed the case.

6. CONCLUSIONS

In this paper, we proposed a fully automatic way of agglomerating atomic user sessions into chains of queries associated with a single task or information need. Our approach consists of finding two thresholds, one time related and the other based on query string similarity. We believe the thresholds we found are stable enough to be used as is in other work on general search engines, since they were stable across datasets, and correspond to basic properties of human attention, reading time, and query modification. However, we pointed out that more sophisticated approaches, capable of detecting session-interleaving and long-term information needs could improve the results.

After observing that the different user activities can be matched to a hierarchical structure where an information need task includes one or more searches that themselves include one or more pages of search results, etc., we proposed a layered Bayesian Network model where each discrete latent variables acts as a "summary" of a subset of actions like e.g. everything related to the browsing of a search results page. To validate our approach, we used features extracted from the BN to predict the relevance of documents. We showed that this set of features improves the prediction performance compared to a baseline model that doesn't use them. Our model could be improved by automatically determining the number of states for the latent variables, and by investigating what are the observations we chose at the various level of the goal hierarchy.

Similar to [6], we could use more detailed logs provided by users who sign up for a product like a search toolbar. These provide information on all the search activities undertaken by the user, in particular what the user does after having clicked on a document and before he resumes interactions with the search engine. We could also apply our model to predict user satisfaction with search results. This could be done using a BN model where satisfaction appears directly as a variable dependent on the chain type, and by learning to optimize the value of this variable from human assessments.

7. REFERENCES

[1] E. Agichtein, E. Brill, S. Dumais, and R. Ragno. Learning user interaction models for predicting web search result preferences. In *Proceedings of SIGIR 2006*, pages 3–10, New York, NY, USA, 2006. ACM Press.

[2] M. Brand. Structure learning in conditional probability models via an entropic prior and parameter extinction. *Neural Computation*, 11(5):1155–1182, 1999.

[3] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.

[4] B. Carterette and R. Jones. Evaluating search engines by modeling the relationship between relevance and clicks. In *NIPS 2007*, 2007.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via de EM algorithm. *The Journal of Royal Statistical Society*, 39:1–37, 1977.

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

[7] G. Dupret and B. Piwowarski. User behavior and search engine query logs: a generative model to predict clickthrough rate. In *Proceedings of SIGIR 2008*, 2008.

[8] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[9] D. He, A. Goker, and D. J. Harper. Combining evidence for automatic web session identification. *Information Processing & Management*, 38(5):727–742, September 2002.

[10] D. Kelly and N. J. Belkin. Display time as implicit feedback: understanding task effects. pages 377–384, New York, NY, USA. ACM.

[11] T. Lau and E. Horvitz. Patterns of search: Analyzing and modeling web query refinement. In A. Press, editor, *Proceedings of ICUM*, 1999.

[12] G. Miller, E. Galanter, and K. Pribram. *Plans and the structure of behavior*. Holt, Rhinehart, & Winston, New York, 1960.

[13] S. Mizzaro. How many relevances in information retrieval? *Interacting With Computers*, 10(3):305–322, 1998.

[14] S. Ozmutlu. Automatic new topic identification using multiple linear regression. *Information Processing & Management*, 42(4):934–950, July 2006.

[15] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *Proceeding of ACM SIGKDD 2005*, pages 239–248, New York, NY, USA, 2005. ACM Press.

[16] G. Ridgeway. Generalized boosted models: A guide to the gbm package. http://i-pensieri.com/gregr/papers/gbm-vignette.pdf, 2005.

[17] X. Shen, B. Tan, and C. Zhai. Implicit user modeling for personalized search. In *Proceedings of CIKM 2005*, pages 824–831, New York, NY, USA, 2005. ACM.

[18] B. Tan, X. Shen, and C. Zhai. Mining long-term search history to improve search accuracy. In *Proceedings of KDD 2006*, 2006.

[19] J. Teevan, E. Adar, R. Jones, and M. Potts. Information re-retrieval: Repeat queries in yahoo's logs. In *Proceedings of SIGIR 2007*. ACM, 2007.

[20] R. W. White and S. M. Drucker. Investigating behavioral variability in web search. In *Proceedings of WWW 2007*, pages 21–30, New York, NY, USA, 2007. ACM.