Click Chain Model in Web Search

Fan Guo[1]* , Chao Liu[2], Anitha Kannan[3], Tom Minka[4], Michael Taylor[4], Yi-Min Wang[2],
Christos Faloutsos[1]

[1]Carnegie Mellon University, Pittsburgh PA 15213, USA
[2]Microsoft Research Redmond, WA 98052, USA
[3]Microsoft Research Search Labs, Mountain View CA 94043, USA
[4]Microsoft Research Cambridge, CB3 0FB, UK

fanguo@cs.cmu.edu, {chaoliu,ankannan, minka, mitaylor, ymwang}@microsoft.com,
christos@cs.cmu.edu

ABSTRACT

Given a terabyte click log, can we build an efficient and effective click model? It is commonly believed that web search click logs are a gold mine for search business, because they reflect users' preference over web documents presented by the search engine. Click models provide a principled approach to inferring user-perceived relevance of web documents, which can be leveraged in numerous applications in search businesses. Due to the huge volume of click data, scalability is a must.

We present the *click chain model* (CCM), which is based on a solid, Bayesian framework. It is both scalable and incremental, perfectly meeting the computational challenges imposed by the voluminous click logs that constantly grow. We conduct an extensive experimental study on a data set containing 8.8 million query sessions obtained in July 2008 from a commercial search engine. CCM consistently outperforms two state-of-the-art competitors in a number of metrics, with over 9.7% better log-likelihood, over 6.2% better click perplexity and much more robust (up to 30%) prediction of the first and the last clicked position.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation

1. INTRODUCTION

Billions of queries are submitted to search engines on the web every day. Important attributes of these search activities are automatically logged as implicit user feedbacks. These attributes include, for each query session, the query string, the time-stamp, the list of web documents shown in the search result and whether each document is clicked or not. Web search click logs are probably the most extensive, albeit indirect, surveys on user experience, which can be aggregated over weeks, months and even years. Extracting key statistics or patterns from these tera-byte logs is of much interest to both search engine providers, who could obtain objective measures of user experience and useful features to improve their services, and to world wide web researchers, who could better understand user behavior and calibrate their hypotheses and models. For example, the topic of utilizing click data to optimize search ranker has been well explored and evaluated by quite a few academic and industrial researchers since the beginning of this century (*e.g.* , [2, 8, 14, 15, 17]).

A number of studies have been conducted previously on analyzing user behavior in web search and their relationship to click data. Joachims et al. [9, 10] carried out eye-tracking experiments to study participants' decision process as they scan through search results, and further compared implicit click feedback against explicit relevance judgments. They found that clicks are accurate enough as relative judgement to indicate user's preferences for certain pairs of documents, but they are not reliable as absolute relevance judgement, *i.e.* ,"clicks are informative but biased". A particular example is that users tend to click more on web documents in higher positions even if the ranking is reversed [10]. Richardson et al. [16] proposed the *examination hypothesis* to explain the position-bias of clicks. Under this hypothesis, a web document must be examined before being clicked, and user-perceived document relevance is defined as the conditional probability of being clicked after being examined. Top ranked documents may have more chance to be examined than those ranked below, regardless of their relevance. Craswell et al. [4] further proposed the *cascade model* for describing mathematically how the first click arises when users linearly scan through search results. However, the cascade model assumes that users abandon the query session after the first click and hence does not provide a complete picture of how multiple clicks arise in a query session and how to estimate document relevance from such data.

Click models provide a principled way of integrating knowledge of user search behaviors to infer user-perceived relevance of web documents, which can be leveraged in a number of search-related applications, including:

- **Automated ranking alterations**: The top-part of ranking can be adjusted based on the inferred relevance so that they are aligned with users' preference.
- **Search quality metrics**: The inferred relevance and user examination probabilities can be used to compose

*Part of this work was done when the first author was on a summer internship with Microsoft Research.

WWW 2009, April 20–24, 2009, Madrid, Spain.
ACM 978-1-60558-487-4/09/04.

search quality metrics, which correlate with user satisfaction [6].

- **Adaptive search**: When the meaning of a query changes over time, so do user click patterns. Based on the inferred relevance that shifts with click data, the search engine can be adaptive.
- **Judge of the judges**: The inferred first-party relevance judgement could be contrasted/reconciled with well-trained human judges for improved quality.
- **Online advertising**: The user interaction model can be adapted to a number of sponsored search applications such as ad auctions [1, 11].

An ideal model of clicks should, in addition to enabling reliable relevance inference, have two other important properties - *scalability* and *incremental computation*; Scalability enables processing of large amounts (typically, terabytes) of clicklogs data and the incremental computation enables updating the model as new data are recorded.

Two click models are recently proposed which are based on the same examination hypothesis but with different assumptions about user behaviors. The user browsing model (UBM) proposed by Dupret and Piwowarski [5] is demonstrated to outperform the cascade model in predicting click probabilities. However, the iterative nature of the inference algorithm of UBM requires multiple scans of the data, which not only increases the computation cost but renders incremental update obscure as well. The dependent click model (DCM) which appears in our previous work [7] is naturally incremental, and is an order of magnitude more efficient than UBM, but its performance on tail queries could be further improved.

In this paper, we propose the *click chain model* (CCM), that has the following desirable properties:

- *Foundation:* It is based on a solid, Bayesian framework. A closed-form representation of the relevance posterior can be derived from the proposed approximation inference scheme.
- *Scalability:* It is fast and nimble, with excellent scalability with respect to both time and space; it can also work in an incremental fashion.
- *Effectiveness:* It performs well in practice. CCM consistently shows performance improvements over two of the state-of-the-art competitors in a number of evaluation metrics such as log-likelihood, click perplexity and click prediction robustness.

The rest of this paper is organized as follows. We first survey existing click models in Section 2, and then present CCM in Section 3. Algorithms for relevance inference, parameter estimation and incremental computation are detailed in Section 4. Section 5 is devoted to experimental studies. The paper is concluded in Section 6.

2. BACKGROUND

We first introduce definitions and notations that will be used throughout the paper. A web search user initializes a *query session* by submitting a *query* to the search engine. We regard re-submissions and reformulations of the same query as distinct query sessions. We use *document impression* to refer to the *web documents* (or URLs) presented in the first result page, and discard other page elements such as sponsored ads and related search. The document impression can be represented as $D = \{d_1, \ldots, d_M\}$ (usually $M = 10$), where d_i is an index into a set of documents for the query. A document is in a higher *position* (or rank) if it appears before those in lower positions.

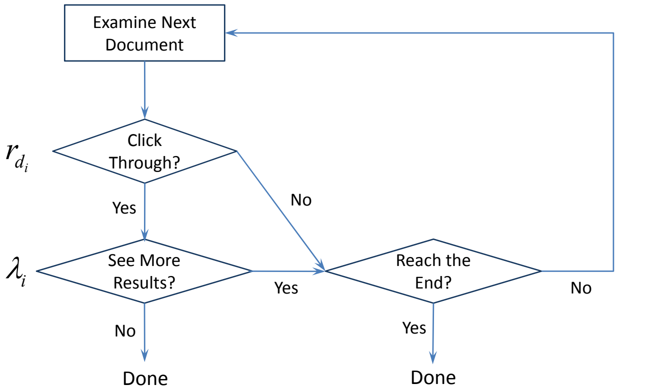


Figure 1: The user model of DCM, in which r_{d_i} is the relevance for document d_i at position i, and λ_i is the conditional probability of examining the next position after a click at position i.

Examination and clicks are treated as probabilistic events. In particular, for a given query session, we use binary random variables E_i and C_i to represent the examination and click events of the document at position i,respectively. The corresponding, examination and click probabilities for position i are denoted by $P(E_i = 1)$ and $P(C_i = 1)$, respectively.

The *examination hypothesis* [16] can be summarized as follows: for $i = 1, \ldots, M$,

$$P(C_i = 1|E_i = 0) = 0,$$
$$P(C_i = 1|E_i = 1) = r_{d_i},$$

where r_{d_i}, defined as the *document relevance*, is the conditional probability of click after examination. Given E_i, C_i is conditionally independent of previous examine/click events $E_{1:i-1}, C_{1:i-1}$. It helps to disentangle clickthroughs of various documents as being caused by position and relevance. Click models based on the examination hypothesis share this definition but differ in the specification of $P(E_i)$.

The *cascade hypothesis* in [4] states that users always start the examination at the first document. The examination is strictly linear to the position, and a document is examined only if all documents in higher positions are examined:

$$P(E_1 = 1) = 1,$$
$$P(E_{i+1} = 1|E_i = 0) = 0.$$

Given E_i, E_{i+1} is conditionally independent of all examine/click events above i, but may depend on the click C_i.

The *cascade model* [4] puts together previous two hypotheses and further constrain that

$$P(E_{i+1} = 1|E_i = 1, C_i) = 1 - C_i, \qquad (1)$$

which implies that a user keeps examining the next document until reaching the first click, after which the user simply stops the examination and abandons the query session.

We first introduce the *dependent click model* (DCM) [7]. Its user model is illustrated in Figure 1. It generalizes the cascade model to multiple clicks by putting position-dependent parameters as conditional probabilities of examining the next document after a click, *i.e.* , Eq. 1 is replaced by

$$P(E_{i+1} = 1 | E_i = 1, C_i = 1) = \lambda_i \tag{2}$$

$$P(E_{i+1} = 1 | E_i = 1, C_i = 0) = 1, \tag{3}$$

where $\{\lambda_i | 1 \leq i \leq M-1\}$ are the user behavior parameters in the model and are shared across query sessions. The probability of a query session is therefore

$$P(C_{1:M}) = \prod_{i=1}^{l} \Big((r_{d_i}\lambda_i)^{C_i} (1-r_{d_i})^{1-C_i} \Big) \cdot \Big(1 - \lambda_l + \lambda_l \prod_{j=l+1}^{M} (1-r_{d_j})\Big), \tag{4}$$

where $l = \arg\max_{1 \leq i \leq M}\{C_i = 1\}$ is the last clicked position in the query session. If there is no click, l is set to 0 in the equation above. This suggests that the user would scan the entire list of search results, which is a major limitation in the modeling assumption of DCM.

The *user browsing model* (UBM) [5] is also based on the examination hypothesis, but does not follow the cascade hypothesis. Instead, it assumes that the examination probability E_i depends only on the previous clicked position $l_i = \arg\max_{l<i}\{C_l = 1\}$ as well as the distance $i - l_i$:

$$P(E_i = 1 | C_{1:i-1}) = \gamma_{l_i, i-l_i}. \tag{5}$$

Given click observations $C_{1:i-1}$, E_i is conditionally independent of all previous examination events $E_{1:i-1}$. If there is no click before i, l_i is set to 0. Since $0 \leq l_i < i \leq M$, there are a total of $M(M+1)/2$ user behavior parameters γ, which are again shared among all query sessions. The probability of a query session under UBM is

$$P(C_{1:M}) = \prod_{i=1}^{M} (r_{d_i}\gamma_{l_i,i-l_i})^{C_i} (1 - r_{d_i}\gamma_{l_i,i-l_i})^{1-C_i}. \tag{6}$$

Maximium likelihood (ML) learning of UBM is an order of magnitude more expensive than DCM in terms of time and space cost. Under DCM, we could keep 3 sufficient statistics, or counts, for each query-document pair, and an additional $3(M-1)$ counts for global parameter estimation [6]. However, for UBM with a more complicated user behavior assumption, we should keep at least $(1+M(M+1)/2)$ counts for each query-document pair and an additional $(1+M(M+1)/2)$ counts for global parameter estimation. In addition, under our implementation (detailed in Appendix A), the algorithm usually takes dozens of iterations until convergence.

A most recent related work is a DBN click model proposed by Chapelle and Zhang [3], which appears as another paper in this proceeding. The model still satisfies the examination and cascade hypotheses as DCM does. The difference is the specification of examine-next probability $P(E_{i+1} = 1 | E_i = 1, C_i)$. Under this DBN model, (1) the only user behavior parameter in the model is denoted by γ, and $P(E_{i+1} = 1 | E_i = 1, C_i = 0) = \gamma$; (2) two values are associated with each query-document pair. For one of them, denoted by s_{d_i}, $P(E_{i+1} = 1 | E_i = 1, C_i = 1) = \gamma(1 - s_{d_i})$, whereas the other one, denoted by a_{d_i} is equivalent to document relevance under the examination hypothesis: $P(C_i = 1 | E_i = 1) = a_{d_i}$. These two values are estimated under approximate ML learning algorithms. Details about the modeling assumption and algorithmic design are available in [3].

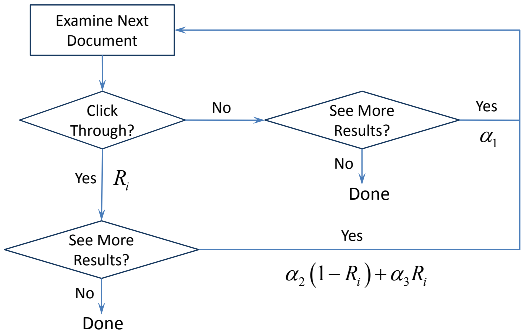


Figure 2: The user model of CCM, in which R_i is the relevance variable of d_i at position i, and α's form the set of user behavior parameters.

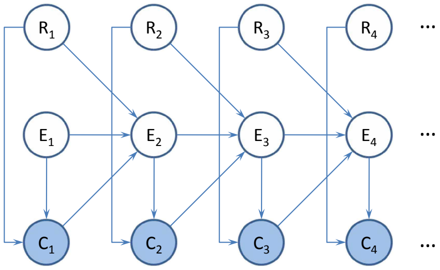


Figure 3: The graphical model representation of CCM. Shaded nodes are observed click variables.

3. CLICK CHAIN MODEL

The CCM model is based on generative process for the users interaction with the search engine results, illustrated as a flow chart in Figure 2, and as a generative model in Figure 3. User starts the examination of the search result from the top ranked document. At each position i, the user can choose to click or skip the document d_i according to the perceived relevance. Either way, the user can choose to continue the examination or abandon the current query session. The probability of continuing to examine d_{i+1} depends on her action at the current position i. Specifically, if the user skips d_i, this probability is α_1; on the other hand, if the user clicks d_i, the probability to examine d_{i+1} depends on the relevance R_i of d_i and range between α_3 and α_2.

CCM shares the following assumptions with the cascade model and DCM: (1) users are homogeneous: their information needs are similar given the same query; (2) decoupled examination and click events: click probability is solely determined by the examination probability and the document relevance at a given position; (3) cascade examination: examination is in strictly sequential order with no breaks.

CCM distinguishes itself from other click models by doing a proper Bayesian inference to infer the posterior distribution of the document relevance. In CCM, document relevance R_i is a random variable between 0 and 1. Model training of CCM therefore includes inferring the posterior distribution of R_i and estimating user-behavior parameters α's. The posterior distribution may be particularly helpful for applications such as automated ranking alterations because it is possible to derive confidence measures and other useful features using standard statistical techniques.

In the graphical model of CCM shown in Figure 3, there are three layers of random variables: for each position i, E_i and C_i denote binary examination and click events as usual, where R_i is the user-perceived relevance of d_i. The click layer is fully observed from the click log. CCM is named after the chain structure of variables representing the sequential examination and clicks through each position in the search result.

The following conditional probabilities are defined in Figure 3 according to the model assumption:

$$P(C_i = 1|E_i = 0) = 0 \quad (7)$$

$$P(C_1 = 1|E_i = 1, R_i) = R_i \quad (8)$$

$$P(E_{i+1} = 1|E_i = 0) = 0 \quad (9)$$

$$P(E_{i+1} = 1|E_i = 1, C_i = 0) = \alpha_1 \quad (10)$$

$$P(E_{i+1} = 1|E_i = 1, C_i = 1, R_i) = \alpha_2(1 - R_i) + \alpha_3 R_i \quad (11)$$

To complete the model specification we let $P(E_1 = 1) = 1$ and document relevances R_i's are *iid* and uniformly distributed within $[0, 1]$, *i.e.* , $p(R_i) = 1$ for $0 \leq R_i \leq 1$. Note that in the model specification, we did not put a limit on the length of the chain structure. Instead we allow the chain to go to infinite, with the examination probability diminishing exponentially. We will discuss, in the next section, that this choice simplifies the inference algorithm and offers more scalability. An alternative is to truncate the click chain to a finite length of M. The inference and learning algorithms could be adapted to this truncated CCM, with an order of magnitude higher space, but it still needs only one pass through the click data. Its performance of improvement over CCM would depend on the tail of the distribution of last clicked position. Details are to be presented in an extended version of this paper.

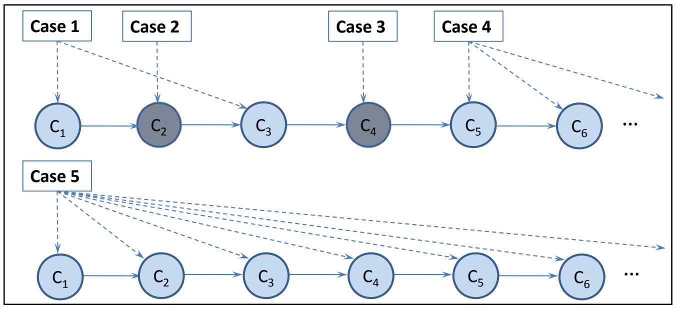
4. ALGORITHMS

This section describes algorithms for the CCM. We start with the algorithm for computing the posterior distribution over the relevance of each document. Then we describe how to estimate the α parameters. Finally, we discuss how to incrementally update the relevance posteriors and α parameters with more data.

Given the click data $C^{1:U}$ from U query sessions for a query, we want to compute the posterior distribution for the relevance of a document i, *i.e.* $P(R_i|C^{1:U})$. For a single query session, this posterior is simple to compute and store due to the chain structure of the graphical model (figure 3). However, for multiple query sessions, the graph may have loops due to sharing of relevance variables between query sessions. For example, if documents i and j both appear in two sessions, then the graph including both sessions will have a loop. This significantly complicates the posterior. One possibility is to use an iterative algorithm such as expectation propagation [13] that traverses and approximates the loops. However, for this application we propose a faster method that simply cuts the loops.

The approximation is that, when computing the posterior for R_i, we do not link the relevance variables for other documents across sessions. In other words, we approximate clicks in query sessions as conditionally independent random variables given R_i:

$$p(R_i|C^{1:U}) \approx (\text{constant}) \times p(R_i) \prod_{u=1}^{U} P(C^u|R_i). \quad (12)$$



Case	Conditions	Results
1	$i < l, C_i = 0$	$1 - R_i$
2	$i < l, C_i = 1$	$R_i(1 - (1 - \alpha_3/\alpha_2)R_i)$
3	$i = l$	$R_i\left(1 + \frac{\alpha_2 - \alpha_3}{2 - \alpha_1 - \alpha_2} R_i\right)$
4	$i > l$	$1 - \frac{2}{1 + \frac{6 - 3\alpha_1 - \alpha_2 - 2\alpha_3}{(1 - \alpha_1)(\alpha_2 + 2\alpha_3)} (2/\alpha_1)^{(i-l)-1}} R_i$
5	No Click	$1 - \frac{2}{1 + (2/\alpha_1)^{i-1}} R_i$

Figure 4: Different cases for computing $P(C|R_i)$ up to a constant where l is the last clicked position. Darker nodes in the figure above indicate clicks.

Since the prior $p(R_i)$ is already known, we only need to figure out $P\left(C^u \middle| R_i\right)$ for each session u. Once $P\left(C^u \middle| R_i\right)$ is calculated up to a constant w.r.t. R_i, Eq. 12 immediately gives an un-normalized version of the posterior. Section 4.1 will elaborate on the computation of $P\left(C^u \middle| R_i\right)$, and the superscript u is discarded in the following as we focus on a single query session.

Before diving into the details of deriving $P(C|R_i)$, we first highlight the following three properties of CCM, which will greatly simplify the variable elimination procedure as we take sum or integration over all hidden variables other than R_i:

1. If $C_j = 1$ for some j, then $\forall i \leq j, E_i = 1$.
2. If $E_j = 0$ for some j, then $\forall i \geq j, E_i = 0, C_i = 0$.
3. If $E_i = 1, C_i = 0$, then $P(E_{i+1}|E_i, C_i, R_i) = \alpha_1^{E_{i+1}}(1 - \alpha_1)^{1-E_{i+1}}$ does not depend on R_i.

Property (1) states that every document before the last click position is examined, so for these documents, we only need take care of different values of random variables within its Markov blanket in Figure 3, which is C_i, E_i and E_{i+1}. Property (2) comes from the cascade hypothesis, and it reduces the cost of eliminating E variables from exponential time to linear time using branch-and-conquer techniques. Property (3) is a corollary from Eq. 11, and it enables re-arrangement of the sum operation over E and R variables to minimize computational effort.

4.1 Deriving the Conditional Probabilities

Calculating $P(C|R_i)$ requires summing over all the hidden variables other than R_i in Figure 3, which is generally intractable. But with the above three properties, closed-form solutions can be derived.

Specifically, from property 1, we know that the last clicked position $l = \arg\max_j\{C_j = 1\}$ plays an important role. Figure 4 lists the complete results separated to two categories, depending on whether the last click position l exists or not.

Due to the space limit, we here present the derivation for cases 1∼3. The derivation of case 3 is illuminating to the other two cases.

Case 1: $i < l, C_i = 0$

By property 1 and 3, $E_i = 1, E_{i+1} = 1$ and $P(E_{i+1} = 1|E_i = 1, C_i = 0, R_i) = \alpha_1$ does not depend on R_i. Since any constant w.r.t. R_i can be ignored, we have

$$P(C|R_i) \propto P(C_i = 0|E_i = 1, R_i) = 1 - R_i \qquad (13)$$

Case 2: $i < l, C_i = 1$

By property 1, $E_i = 1, E_{i+1} = 1$, the Markov blanket of R_i consists of C_i, E_i and E_{i+1}.

$$\begin{aligned}
&P(C|R_i)\\
&\propto P(C_i = 1|E_i = 1, R_i)P(E_{i+1} = 1|E_i = 1, C_i = 1, R_i)\\
&\propto R_i(1 - (1 - \alpha_3/\alpha_2)R_i) \qquad (14)
\end{aligned}$$

Case 3: $i = l$

By property 1, the Markov blanket of R_i does not contain any variable before position i, and we also know that $C_i = 1, E_i = 1$ and $\forall j > i, C_j = 0$. We need to take sum/integration over all the $E_{>i}, R_{>i}$ variables and it can be performed as follows:

$$\begin{aligned}
&P(C|R_i)\\
&\propto P(C_i = 1|E_i = 1, R_i) \cdot \sum_{E_{j>i}} \int_{R_{j>i}} \prod_{j>i}\\
&\quad \Big(P(E_j|E_{j-1}, C_{j-1}, R_{j-1}) \cdot p(R_j) \cdot P(C_j|E_j, R_j)\Big)\\
&= R_i \cdot \Bigg\{ P(E_{i+1} = 0|E_i = 1, C_i = 1, R_i) \cdot 1\\
&\quad + P(E_{i+1} = 1|E_i = 1, C_i = 1, R_i) \cdot\\
&\quad \left(\int_0^1 p(R_{i+1})\ P(C_{i+1} = 0|E_{i+1} = 1, R_{i+1}) dR_{i+1}\right) \cdot\\
&\quad \Big\{ P(E_{i+2} = 0|E_{i+1} = 1, C_{i+1} = 0) \cdot 1\\
&\quad + P(E_{i+2} = 1|E_{i+1} = 1, C_{i+1} = 0) \cdot\\
&\quad \left(\int_0^1 p(R_{i+2})\ P(C_{i+2} = 0|E_{i+2} = 1, R_{i+2}) dR_{i+2}\right) \cdot\\
&\quad \{\ldots\}\Big\}\Bigg\}\\
&= R_i\Big((1 - \alpha_2(1 - R_i) - \alpha_3 R_i) + (\alpha_2(1 - R_i) + \alpha_3 R_i) \cdot\\
&\quad \frac{1}{2} \cdot \Big((1 - \alpha_1) + \alpha_1 \cdot \frac{1}{2} \cdot (\ldots)\Big)\Big)\\
&\propto R_i\left(1 + \frac{\alpha_2 - \alpha_3}{2 - \alpha_1 - \alpha_2} R_i\right) \qquad (15)
\end{aligned}$$

Both case 4 and case 5 need to take sum over the hidden variables after the current position i. The result of case 4 and 5 depend on the distance from the last clicked position. Therefore the total number of distinct results for computing $P(C|R_i)$ in these 5 cases when $1 \le i \le M$ are $1 + 1 + 1 + (M - 1) + M = 2M + 2$. If we impose a finite chain length M on CCM and let $P(E_{M+1}) = 0$ in Figure 3, then the number of distinct results would be $M(M + 3)/2 + 2$, which is an order of magnitude higher than the current design, and further increases the cost of subsequent step of inference and parameter estimation.

Table 1: The algorithm for computing a factorized representation of relevance posterior.

```
Alg. 1: Computing the Un-normalized Posterior
Input:   Click Data $C$ ($M \times U$ matrix);
           $C_{m,u} = 1$ if user $u$ clicks the $m$th page abstract
         Impression Data $I$ ($M \times U$ matrix);
           $I_{m,u}$ is the document index shown to user $u$
           at position $m$ which ranges between 1 and $D$
         Parameters $\alpha$
Output:  Coefficients $\beta$ (($2M + 2$)-dim vector)
         Exponents $P$ ($D \times (2M + 2)$ matrix)
Compute $\beta$ using the results in Figure 4.
Initialize $P$ by setting all the elements to 0.
for $1 \le u \le U$
    Identify the linear factors using the click data $C_{\cdot u}$,
    obtain a $M$-dim vector $b$, such that $\beta_{b_m}$ is the
    corresponding coefficient for position $m$.
    for $1 \le m \le M$
        $P_{I_{mu}, b_m} = P_{I_{mu}, b_m} + 1$
Time Complexity: $O(MU)$.
Space Complexity: $O(MD + MU)$.
($Usually\ U > D >> M = 10$.)
```

4.2 Computing the Un-normalized Posterior

All the conditional probabilities in Figure 4 for a single query session can be written in the form of $R_i^{C_i}(1 - \beta_j R_i)$ where β_j is a case-dependent coefficient dependent on the user behavior parameters α's. Let M be the number of web documents in the first result page and it usually equals 10. There are at most $2M + 3$ such β coefficients from the 5 cases as discussed above. From Eq. 12, we know that the un-normalized posterior of any web page is a polynomial, which consists of at most $(2M + 2) + 1$ distinct linear factors of R_i (including R_i itself) and $(2M + 2)$ independent exponents as sufficient statistics. The detailed algorithm for parsing through the data and updating the counting statistics is listed in Table 1. Given the output exponents P as well as coefficients β from Alg. 1, the un-normalized relevance posterior of a document d is

$$\tilde{p}_{R_d}(r) \propto r^{P_{d,2}+P_{d,3}} \prod_{m=1}^{2M+2} (1 - \beta_m r)^{P_{d,m}} \qquad (16)$$

4.3 Numerical Integration for Posteriors

Standard polynomial integration of $\tilde{p}(r)$ is straightforward but is subject to the dimensional curse: the rounding error will dominate as the order of the polynomial goes up to 100 or more. Thus we propose the numerical integration procedure for computing posterior moments in Table 2 using the midpoint rule with B equal-size bins.The jth moment for document d is therefore

$$\int_0^1 r^j \tilde{p}(r) dr \approx \frac{1}{B} \sum_{b=1}^{B} r_b^{P_{d,2}+P_{d,3}+j} \prod_{m=1}^{2M+1} (1 + \beta_m r_b)^{P_{d,m}}$$

for $j \ge 1$ divided by the normalizing constant c. Although the number of bins can be adaptively set, usually we find

Table 2: The algorithm for computing posterior moments using numerical integration.

```
Alg. 2: Numerical Integration for Posteriors
Input:  Exponents $P$ ($D \times (2M+2)$ matrix)
        Coefficients $\beta$ ($(2M+2)$-dim vector)
        Number of bins $B$
Output: Normalizing Constants $c$ ($D$-dim vector)
        Posterior Mean $\mu$ ($D$-dim vector)
        Posterior Second Moment $\xi$ ($D$-dim vector)
Create a $D \times 3$ matrix $T$ for intermediate results.
Create a $B$-dimensional vector $r$ to keep the center
    of each bin: $r_b = (b - 0.5)/B$ for $1 \le b \le B$
for $0 \le j \le 2$
    for $1 \le d \le D$
        $M_{d,j} = \sum_{b=1}^{B} \exp\big(-\log(B) + (P_{d,2} + P_{d,3} + j) \cdot$
            $\log(r_b) + \sum_{m=1}^{2M+2} P_{d,m} \log(1 + \beta_m r_b)\big)$.
for $1 \le d \le D$
    $c_d = M_{d,0}, \quad \mu_d = M_{d,1}/M_{d,0}, \quad \xi_d = M_{d,2}/M_{d,0}$
Time Complexity: $O(MDB)$.
Space Complexity: $O(MD + DB + MB)$.
```

that $B = 100$ is sufficient. To avoid numerical problems in implementation, we usually take sum of log values for the linear factors instead of multiplying them directly. In general, for posterior point estimates $\mathbb{E}_{p(r)}[f(r)]$, we can simply replace the term r_b^j in the pervious equation by $f(r_b)$ and divide the result by the normalization constant c.

4.4 Parameter Estimation

To estimate the model parameters $\boldsymbol{\alpha}$ from the data, we employ approximate maximum likelihood estimation. Ideally, we want to maximize the likelihood $p(C^{1:U}|\boldsymbol{\alpha})$, in which the relevance variables R_i are integrated out. However, as discussed earlier in section 4, the graph of relevance variables has loops, preventing exact integration. Therefore we approximate the integral by cutting all edges between query sessions. In other words, the clicks are approximated as conditionally independent given $\boldsymbol{\alpha}$: $p(C^{1:U}|\boldsymbol{\alpha}) \approx \prod_{u=1}^{U} p(C^u|\boldsymbol{\alpha})$.

For a single query session, the likelihood $p(C|\boldsymbol{\alpha})$ is computed by integrating out the R_i (with uniform priors) and the examination variables E_i. The exact derivation is similar to (15) and is omitted. Summing over query sessions, the resulting approximate log-likelihood function is

$$\begin{aligned}\ell(\alpha) = {} & N_1 \log \alpha_1 + N_2 \log \alpha_4 + N_3 \log(6 - 3\alpha_1 - \alpha_4) \\ & + N_5 \log(1 - \alpha_1) - (N_3 + N_5) \log(2 - \alpha_1) \\ & - N_1 \log 2 - (N_2 + N_3) \log 6 \qquad (17)\end{aligned}$$

where

$$\alpha_4 \triangleq \alpha_2 + 2\alpha_3$$

is an auxiliary variable and N_i is the total number of times documents fall into case i in Figure 4 in the click data. By maximizing this approximate log-likelihood w.r.t. α's, we have

$$\alpha_1 = \frac{3N_1 + N_2 + N_5 - \sqrt{(3N_1 + N_2 + N_5)^2 - 8N_1(N_1 + N_2)}}{2(N_1 + N_2)} \qquad (18)$$

and

$$\alpha_4 = \frac{3N_2(2 - \alpha_1)}{N_2 + N_3} \qquad (19)$$

Eqs. 18 and 19 leave a degree of freedom in choosing the value α_2 and α_3 introduced by applying the *iid* uniform priors over all documents. We can assign a value to α_2/α_3 according to the context of the model application (more on this in experiments). Parameter values do not depend on N_4 when the chain length is infinite.

4.5 Incremental Computation

When new click data are available, we can run Alg. 1 (Table 1) to update exponents stored in P by increasing the counts. The computational time is thus $O(MU')$, where U' is the number of users in the new data. Extra storage other than input is needed only when there are previously unseen web document of interest. If necessary, we can update the posterior relevancy estimates using Alg. 2 (Table 2). We can also update $\boldsymbol{\alpha}$ using the same counts.

5. EXPERIMENTS

In this section, we report on the experimental evaluation and comparison based on a data set with 8.8 million query sessions that are uniformly sampled from a commercial search engine. We measure the performance of three click models with a number of evaluation metrics, and the experimental results show that CCM consistently outperforms UBM and DCM: over 9.7% better log-likelihood (Section 5.2), over 6.2% improvement in click perplexity (Section 5.3) and much more robust (up to 30%) click statistics prediction (Section 5.4) As these widely used metrics measure the model quality from different aspects, the consistently better results indicate that CCM robustly captures the clicks in a number of contexts. Finally, in Section 5.5, we present the empirical examine and click distribution curves, which help illustrate the differences in modeling assumptions.

5.1 Experimental Setup

The experiment data set is uniformly sampled from a commercial search engine in July 2008. Only query sessions with at least one click are kept for performance evaluation, because those discarded sessions tend to correlate with clicks on other search elements, *e.g.* , ads and query suggestions. For each query, we sort all of its query sessions in ascending order of the time stamps, and split them evenly into the training set and the test set. The number of query sessions in the training set is 4,804,633. Moreover, in order to prevent evaluations from being biased by highly frequent queries for which even the simplest model could do well, 178 (0.16%) queries with more than $10^{3.5}$ sessions are excluded from the test set. Performances of different click models on these top queries are consistent with less frequent queries but with a much smaller margin. The final test set consists of 110,630 distinct queries and 4,028,209 query sessions. In order to investigate the performance variations across different query frequencies, queries in the test set are categorized according to the query frequency into the 6 groups as listed in Table 3.

For each query, we compute document relevance and position relevance based on each of the three models, respectively. Position relevance is computed by treating each position as a pseudo-document. The position relevance can substitute the document relevance estimates for documents that appear zero or very few times in the training set but do appear in the test set. This essentially smooths the predictive model and improves the performance on the test set.

Table 3: Summary of Test Data Set

Query Freq	# Queries	# Sessions	Avg # Clk
1∼9	59,442	216,653 (5.4%)	1.310
10∼31	30,980	543,537 (13.5%)	1.239
32∼99	13,667	731,972 (17.7%)	1.169
100∼316	4,465	759,661 (18.9%)	1.125
317∼999	1,523	811,331 (20.1%)	1.093
1000∼3162	553	965,055 (24.0%)	1.072

The cutoff is set adaptively as $\lfloor 2\log_{10}(\text{Query Frequency})\rfloor$. For CCM, the number of bins B is set to 100 which provides adequate level of accuracy.

To account for heterogeneous browsing patterns for different queries, we fit different models for navigational queries and informational queries and learn two sets of parameters for each model according to the median of click distribution over positions [6, 12]. In particular, CCM sets the ratio α_2/α_3 equal to 2.5 for navigational queries and 1.5 for informational queries, because a larger ratio implies a smaller examination probability after a click. The value of α_1 equals 1 as a result of discarding query sessions with no clicks ($N_5 = 0$ in Eq. 18). For navigational queries, $\alpha_2 = 0.10$ and $\alpha_3 = 0.04$; for informational queries, $\alpha_2 = 0.40$ and $\alpha_3 = 0.27$. Parameter estimation results for DCM and UBM will be available at the author's web site.

For performance evaluation on test data, we compute the log-likelihood and other statistics needed for each query session using the document relevance and user behavior parameter estimates learned/inferred from training data. In particular, by assuming independent document relevance priors in CCM, all the necessary statistics can be derived in close form, which is summarized in Appendix B. Finally, to avoid infinite values in log-likelihood, a lower bound of 0.01 and a upper bound of 0.99 are applied to document relevance estimates for DCM and UBM.

All of the experiments were carried out with MATLAB R2008b on a 64-bit server with 32GB RAM and eight 2.8GHz cores. The total time of model training for UBM, DCM and CCM is 333 minutes, 5.4 minutes and 9.8 minutes, respectively. For CCM, obtaining sufficient statistics (Alg. 1), parameter estimation, and posterior computation (Alg. 2) accounted for 54%, 2.0% and 44% of the total time, respectively. For UBM, the algorithm converged in 34 iterations.

5.2 Results on Log-Likelihood

Log-likelihood (LL) is widely used to measure model fitness. Given the document impression for each query session in the test data, LL is defined as the log probability of observed click events computed under the trained model. A larger LL indicates better performance, and the optimal value is 0. The improvement of LL value ℓ_1 over ℓ_2 is computed as $(\exp(\ell_1 - \ell_2) - 1) \times 100\%$. We report average LL over multiple query sessions using arithmetic mean.

Figure 5 presents LL curves for the three models over different frequencies. The average LL over all query sessions is -1.171 for CCM, which means that with 31% chance, CCM predicts the correct user clicks out of the 2^{10} possibilities on the top 10 results for a query session in the test data. The average LL is -1.264 for UBM and -1.302 for DCM, with improvement ratios to be 9.7% and 14% respectively. Thanks to the Bayesian modeling of the relevance, CCM outperforms the other models more significantly on less-frequent queries than on frequent ones, as indicated in Figure 5. Fitting different models for navigational and informational queries leads to 2.5% better LL for DCM compared with a previous implementation in [7] on the same data set (average LL = -1.327).

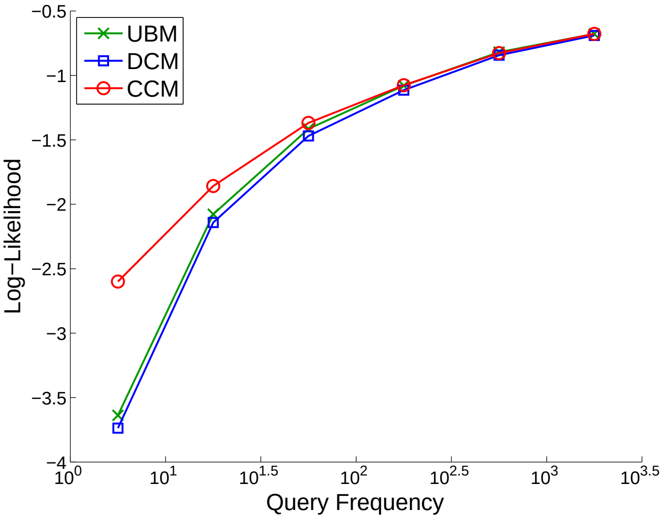


Figure 5: Log-likelihood per query session on the test data for different query frequencies. The overall average for CCM is -1.171, 9.7% better than UBM (-1.264) and 14% better than DCM (-1.302).

5.3 Results on Click Perplexity

Click perplexity was used as the evaluation metric in [5]. It is computed for binary click events at each position in a query session independently rather than the whole click sequence as in the log-likelihood computation. For a set of query sessions indexed by $n = 1, \ldots, N$, if we use q_i^n to denote the probability of click derived from the click model on position i and C_i^n to denote the corresponding binary click event, then the click perplexity

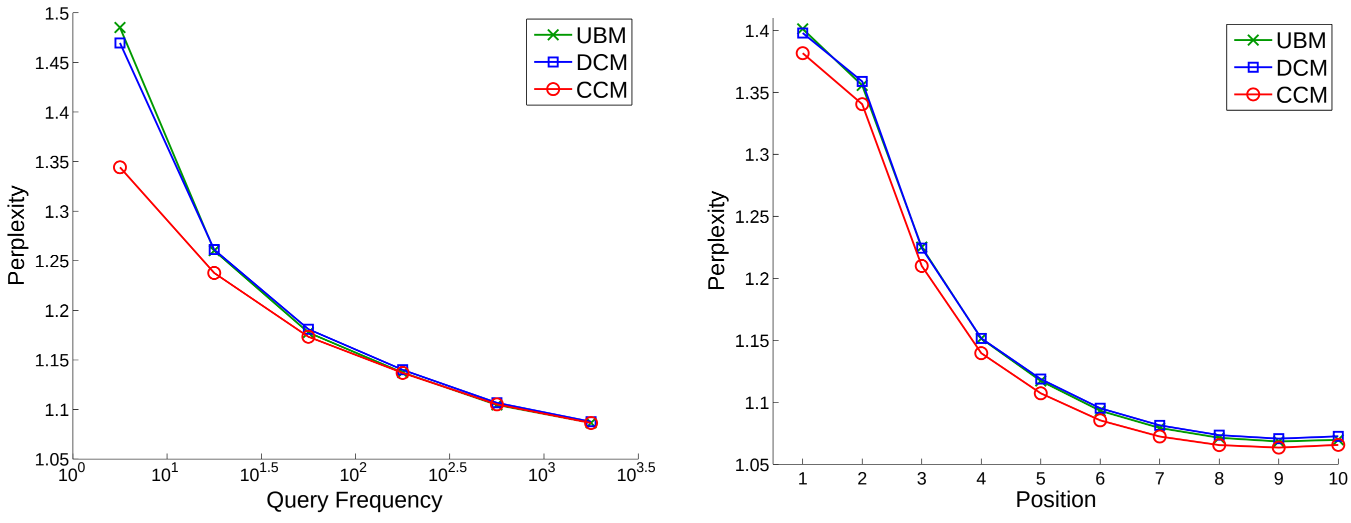
$$p_i = 2^{-\frac{1}{N}\sum_{n=1}^{N}(C_i^n \log_2 q_i^n + (1-C_i^n)\log_2(1-q_i^n))}. \tag{20}$$

A smaller perplexity value indicates higher prediction quality, and the optimal value is 1. The improvement of perplexity value p_1 over p_2 is given by $(p_2 - p_1)/(p_2 - 1) \times 100\%$.

The average click perplexity over all query sessions and positions is 1.1479 for CCM, which is a 6.2% improvement per position over UBM (1.1577) and a 7.0% improvement over DCM (1.1590). Again, CCM outperforms the other two models more significantly on less frequent queries than on more frequent queries. As shown in Figure 6(a), the improvement ratio is over 26% when compared with DCM and UBM for the least frequent query category. Figure 6(b) demonstrates that click prediction quality of CCM is the best of the three models for every position, whereas DCM and UBM are almost comparable. Perplexity value for CCM on position 1 and 10 is comparable to the perplexity of predicting the outcome of throwing a biased coin of with known $p(Head) = 0.099$ and $p(Head) = 0.0117$ respectively.

5.4 Predicting First and Last Clicks

Root-mean-square (RMS) error on click statistics prediction was used as an evaluation metric in [7]. It is calculated



(a) (b)

Figure 6: Perplexity results on the test data for (a) for different query frequencies or (b) different positions. Average click perplexity over all positions for CCM is 1.1479, 6.2% improvement over UBM (1.1577) and 7.0% over DCM (1.1590).

by comparing the observation statistics such as the first click position or the last clicked position in a query session with the model predicted position. Predicting the last click position is particularly challenging for click models while predicting the first click position is a relatively easy task. We evaluate the performance of these predictions on the test data under two settings.

First, given the impression data, we compute the distribution of the first/last clicked position in closed form, and compare the expected value with the observed statistics to compute the RMS error. The optimal RMS value under this setting is approximately the standard deviation of first/last clicked positions over all query sessions for a given query, and it is included in Figure 7 as the "optimal-exp" curve. We expect that a model that gives consistently good fit of click data would have a small margin with respect to the optimal error. The improvement of RMS error e_1 over e_2 w.r.t. the optimal value e^* is given by $(e_2 - e_1)/(e_2 - e^*) * 100\%$. We report average error by taking the RMS mean over all query sessions.

The optimal RMS error under this setting for first clicked position is 1.198, whereas the error of CCM is 1.264. Compared with the RMS value of 1.271 for DCM and 1.278 for UBM, the improvement ratios are 10% and 18%, respectively. The optimal RMS error under this setting for last clicked position is 1.443, whereas the error of CCM is 1.548, which is 9.8% improvement for DCM (1.560) but only slightly better than UBM (0.2%).

Under the second setting, we simulate query session clicks from the model, collect those samples with clicks, compare the first/last clicked position against the ground truth in the test data and compute the RMS error, in the same way as [7]. The number of samples in the simulation is set to 10. The optimal RMS error is the same as the first setting, but it is much more difficult to achieve than in the first setting, because errors from simulations reflect both biases and variances of the prediction. So we report the RMS error margin for the same model between the two settings. And we believe that a *more robust* click model should have a *smaller margin*. This margin on first click prediction is 0.366 for CCM, compared with 0.384 for DCM (4.8% larger) and 0.436 for UBM (19% larger). Similarly, on prediction of the last click position, the margin is 0.460 for CCM, compared with 0.566 for DCM (23% larger) and 0.599 for UBM (30% larger). In summary, CCM is the best of the three model in this experiment, to predict the first and the last click position effectively and robustly.

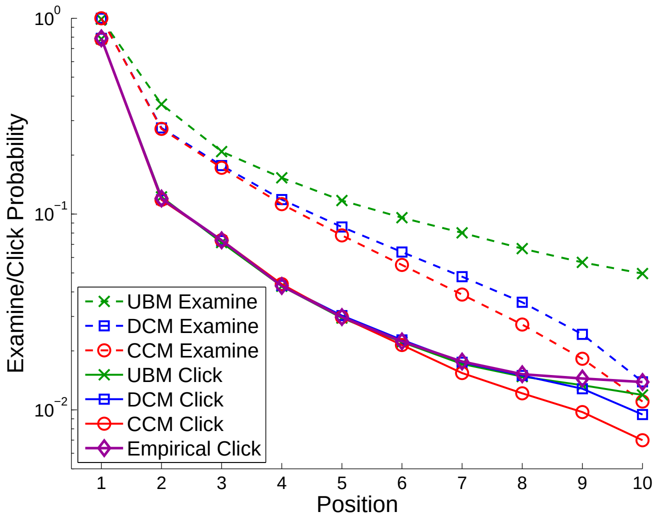
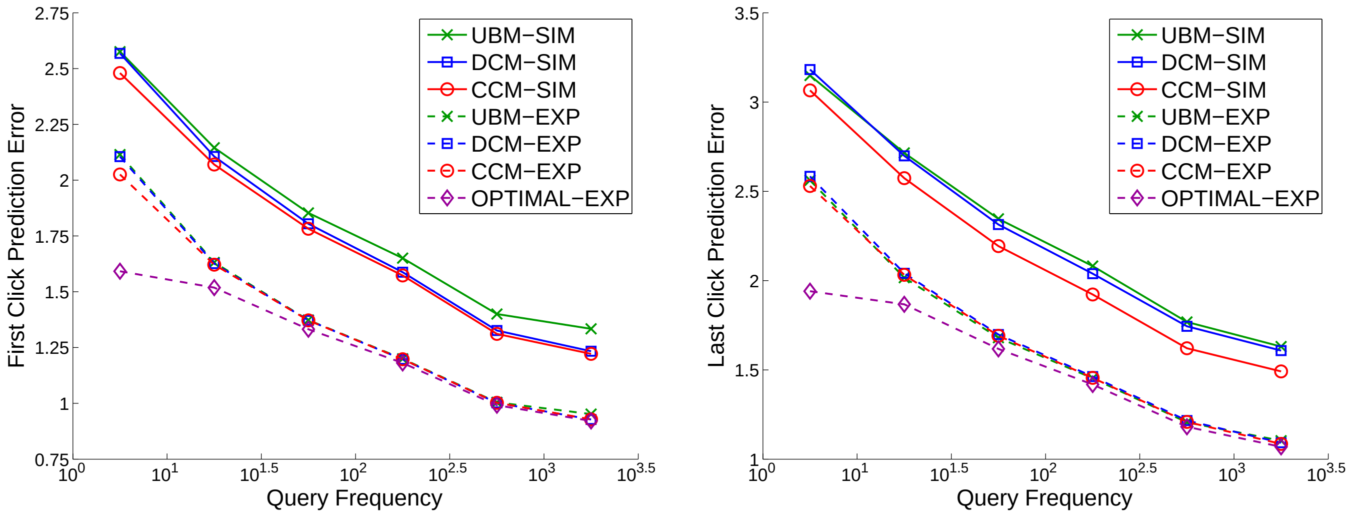


Figure 8: Examination and click probability distributions over the top 10 positions.

5.5 Position-bias of Examination and Click

Model click distribution over positions are the averaged click probabilities derived from click models based on the document impression in the test data. It reflects the position-bias implied by the click model and can be compared with



(a)

(b)

Figure 7: Root-mean-square (RMS) errors for predicting (a) first clicked position and (b) last clicked position. Prediction is based on the SIMulation for solid curves and EXPectation for dashed curves.

the objective ground truth–the *empirical click distribution* over the test set. Any deviation of model click distribution from the ground truth would suggest the existing modeling bias in clicks. Note that on the other side, a close match does not necessarily indicate excellent click prediction, for example, prediction of clicks $\{00, 11\}$ as $\{01, 10\}$ would still have a perfect empirical distribution. *Model examination distribution* over positions can be computed in a similar way to the click distribution. But there is no ground truth to be contrasted with. Under the examination hypothesis, the gap between examination and click curves of the same model reflects the average document relevance.

Figure 8 illustrates the model examination and click distribution derived from CCM, DCM and UBM as well as the empirical click distribution on the test data. All three models underestimate the click probabilities in the last few positions. CCM has a larger bias due to the approximation of infinite-length in inference and estimation. This approximation is immaterial as CCM gives the best results in the above experiments. A truncated version of CCM could be applied here for improved quality, but the time and space cost would be an order of magnitude higher, based on some ongoing experiments not reported here. Therefore, we believe that this bias is acceptable unless certain applications require very accurate modeling of the last few positions.

The examination curves of CCM and DCM decrease with a similar exponential rate. Both models are based on the cascade assumption, under which examination is a linear traverse through the rank. The examination probability derived by UBM is much larger, and this suggests that the absolute value of document relevance derived from UBM is not directly comparable to that from DCM and CCM. Relevance judgments from click models is still a relative measure.

6. CONCLUSION

In this paper, we propose the click chain model (CCM), which includes Bayesian modeling and inference of user-perceived relevance. Using an approximate closed-form representation of the relevance posterior, CCM is both scalable and incremental, perfectly meeting the challenges posed by real world practice. We carry out a set of extensive experiments with various evaluation metrics, and the results clearly evidence the advancement of the state-of-the-art. Similar Bayesian techniques could also be applied to models such as DCM and UBM.

7. ACKNOWLEDGMENTS

We would like to thank Ethan Tu and Li-Wei He for their help and effort, and anonymous reviewers for their constructive comments on this paper. Fan Guo and Christos Faloutsos are supported in part by the National Science Foundation under Grant No. DBI-0640543. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

8. REFERENCES

[1] G. Aggarwal, J. Feldman, S. Muthukrishnan, and M. Pál. Sponsored search auctions with markovian users. In *WINE '08*, pages 621–628, 2008.

[2] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR '06*, pages 19–26, 2006.

[3] O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *WWW '09*, 2009.

[4] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *WSDM '08*, pages 87–94, 2008.

[5] G. E. Dupret and B. Piwowarski. A user browsing model to predict search engine click data from past observations. In *SIGIR '08*, pages 331–338, 2008.

[6] F. Guo, L. Li, and C. Faloutsos. Tailoring click models to user goals. In *WSDM '09 workshop on Web search click data*, pages 88–92, 2009.

[7] F. Guo, C. Liu, and Y.-M. Wang. Efficient multiple-click models in web search. In *WSDM '09*,

pages 124–131, 2009.

[8] T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02*, pages 133–142, 2002.

[9] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR '05*, pages 154–161, 2005.

[10] 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.*, 25(2):7, 2007.

[11] D. Kempe and M. Mahdian. A cascade model for externalities in sponsored search. In *WINE '08*, pages 585–596, 2008.

[12] U. Lee, Z. Liu, and J. Cho. Automatic identification of user goals in web search. In *WWW '05*, pages 391–400, 2005.

[13] T. Minka. Expectation propagation for approximate bayesian inference. In *UAI '01*, pages 362–369, 2001.

[14] 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 '09*, pages 162–171, 2009.

[15] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *KDD '05*, pages 239–248, 2005.

[16] M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: estimating the click-through rate for new ads. In *WWW '07*, pages 521–530, 2007.

[17] G.-R. Xue, H.-J. Zeng, Z. Chen, Y. Yu, W.-Y. Ma, W. Xi, and W. Fan. Optimizing web search using web click-through data. In *CIKM '04*, pages 118–126, 2004.

APPENDIX

A. Algorithms for Learning UBM

For UBM, the log-likelihood for each query session is

$$\ell = \sum_{i=1}^{M} C_i(\log r_{d_i} + \log \gamma_{l_i, i-l_i}) + (1 - C_i)\log(1 - r_{d_i}\gamma_{l_i, i-l_i})$$

The coupling of relevance r and parameter γ introduced in the second term makes exact computation intractable. The algorithm could alternative be carried out in an iterative, coordinate-ascent fashion. However, we found that the fix-point equation update proposed in [5] does not have convergence guarantee and is sub-optimal in certain cases. Instead, we designed the following update scheme which usually leads to very fast convergence.

Given a query for each document d, we keep its count of click K_d and non-click $L_{d,j}$ where $1 \le j \le M(M+1)/2$, and they map one-to-one with all possible γ indices, and $1 \le d \le D$ maps one-to-one to all query-document pair. Similarly, for each γ_j, we keep its count of click K_j. Then given the initial value of $\gamma = \gamma^0$, optimization of relevance can be carried out iteratively,

$$r_d^{t+1} = \arg\max_{0<r<1} \left\{ K_d \log r + \sum_{j=1}^{M(M+1)/2} L_{d,j} \log(1 - r\gamma_j^t) \right\}$$

$$\gamma_j^{t+1} = \arg\max_{0<\gamma<1} \left\{ K_j \log \gamma + \sum_{d=1}^{D} L_{d,j} \log(1 - r_d^{t+1}\gamma) \right\}$$

for all possible d and j respectively. The "arg-max" operation can be carried out by evaluating the objective function for a sequential scan of r or γ values. And we suggest a granularity of 0.01 for both scans.

B. LL and Other Statistics of CCM

Given a query session in test data, let r_i and s_i denote the first and second moment of the document relevance posterior for the web document at position i, and $\alpha_1, \alpha_2, \alpha_3$ denote the user behavior parameter. These values are obtained during model training. We iteratively define the following constants:

$$\begin{aligned} \zeta_0 &= 1 \\ \zeta_{i+1} &= (1 - r_{M-i})(1 - \alpha_1 + \alpha_1 \zeta_i) \end{aligned}$$

Let C^i denote the actual click event at test data, then depending on the value of last clicked position l, we have the following two cases:

- $l = 0$ (no click),

$$\begin{aligned} LL = \int_S p(S_1) P(C_1 = 0 | E_1 = 1, S_1) \sum_E \prod_{j>1} \\ P(E_j | E_{j-1}, C_{j-1} = 0) p(S_j) P(C_j = 0 | E_j, S_j) = \zeta_M \end{aligned}$$

- $1 \le l \le M$

$$\begin{aligned} LL = \left(\prod_{i=1}^{l-1} (\alpha_1(1 - r_i))^{1-C_i} (\alpha_2 r_i + (\alpha_3 - \alpha_2)s_i)^{C_i} \right) \\ \cdot \Big((1 - \alpha_2(1 - \zeta_{M-l}))r_l + (\alpha_2 - \alpha_3)(1 - \zeta_{M-l})s_l \Big) \end{aligned}$$

We also define the following constants:

$$\begin{aligned} \varphi_i &= (1 - r_i)\alpha_1 + (r_i - s_i)\alpha_2 + s_i\alpha_3 \\ \xi_M &= 0 \\ \xi_{i-1} &= \phi_{i-1}\xi_i + r_i \end{aligned}$$

Then we have

$$\begin{aligned} \text{Examination Probability } P(E_i = 1) &= \prod_{j=1}^{i-1} \varphi_j \\ \text{Examination Depth } P(LEP = i) &= (1 - \varphi_i)P(E_i = 1) \\ \text{Click Probability } P(C_i = 1) &= r_i P(E_i = 1) \\ \text{First Clicked Position } P(FCP = i) &= r_i \prod_{j=1}^{i-1} \alpha_1(1 - r_j) \\ \text{Last Clicked Position } P(LCP = i) &= P(E_i = 1) \cdot \\ & \Big(r_i - \big(\varphi_i - \alpha_1(1 - r_i) \big) \xi_i \Big) \end{aligned}$$

To draw sample from CCM for performance test:

```
Initialize $\mathcal{C}_{1:M} = 0, \mathcal{E}_1 = 1, i = 1$
while $(i \le M)$ and $(\mathcal{E}_i \ne 0)$
    $\mathcal{C}_i \sim \text{Bernoulli}(r_i)$
    $\mathcal{E}_{i+1} \sim \text{Bernoulli}(\alpha_1(1 - \mathcal{C}_i) + (\alpha_2 r_i + (\alpha_3 - \alpha_2)s_i)\mathcal{C}_i)$
    $i = i + 1$
```

Detailed derivations for this subsection will be available at the author's web site and included in an extended version of this work.