A Novel Click Model and Its Applications to Online Advertising

Zeyuan Allen Zhu [1,2,*], Weizhu Chen [2], Tom Minka [3], Chenguang Zhu [2,4], Zheng Chen [2]

[1]Fundamental Science Class, Department of Physics, Tsinghua University
Beijing, China, 100084
zhuzeyuan@hotmail.com

[2]Microsoft Research Asia
Beijing, China, 100080
{v-zezhu, wzchen, v-chezhu, zhengc}@microsoft.com

[3]Microsoft Research Cambridge
Cambridge, CB3 0FB, UK
minka@microsoft.com

[4]Department of Computer Science and Technology, Tsinghua University
Beijing, China, 100084
zcg.cs60@gmail.com

ABSTRACT

Recent advances in *click model* have positioned it as an attractive method for representing user preferences in web search and online advertising. Yet, most of the existing works focus on training the click model for individual queries, and cannot accurately model the tail queries due to the lack of training data. Simultaneously, most of the existing works consider the query, url and position, neglecting some other important attributes in click log data, such as the local time. Obviously, the click through rate is different between daytime and midnight. In this paper, we propose a novel click model based on Bayesian network, which is capable of modeling the tail queries because it builds the click model on attribute values, with those values being shared across queries. We called our work *General Click Model (GCM)* as we found that most of the existing works can be special cases of GCM by assigning different parameters. Experimental results on a large-scale commercial advertisement dataset show that GCM can significantly and consistently lead to better results as compared to the state-of-the-art works.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval Models; H.3.5 [**Information Storage and Retrieval**]: Online Information Services; G.3 [**Probability and Statistics**].

General Terms

Algorithms, Design, Experimentation, Human Factors.

Keywords

Attribute, Search engine, Bayesian, Gaussian, Advertisement

1. INTRODUCTION

Utilizing *implicit feedback* is one of the most essential techniques for a search engine to better entertain its millions or billions of users. Implicit feedback can be regarded as a vector of *attribute values*, including the query text, timestamps, localities, the click-or-not flag, etc. Given a query, whether user clicks a url is strongly correlated with the user's opinions on the url. Besides, implicit feedback is readily available. Terabytes of such data is produced every day, with which a search engine can automatically adapt to the needs of users by putting the most relevant search results and advertisements in the most conspicuous places.

Following T. Joachims [13] in 2002, implicit feedback such as click data has been used in various ways: towards the optimization of search engine ranking functions (e.g. [4] [3] [6]), towards the evaluation of different ranking functions (e.g. [5] [12] [14]), and even towards the display of advertisements or news [1] [19]. Most of the works above rely on a core method: to learn a *click model*. Basically, the search engine logs a large number of real-time query sessions, along with the user's click-or-not flags. This data is regarded as the training data for the click model, which will be used for predicting the click through rate (CTR) of future query sessions. The CTR can help improve the normalized discounted cumulative gain (NDCG) of the search results (e.g. [6]), and play an essential role in search auctions (e.g. [9] [2]).

However, clicks are *biased* with respect to presenting order, reputation of sites, user-side configuration (e.g. display resolution, web browser), etc. The most substantial evidence is given by the eye-tracking experiment carried out by T. Joachims [14] [15], in which it was observed that users tend to click web documents at the top even if the search results are shown in reverse order. All of these show that it is desirable to build an unbiased click model.

Recently, a number of studies [19] [7] [8] [6] [10] have attempted to explain the position-biased click data. In 2007, M. Richardson *et al.* [19] suggested that the relevance of a document at position i should be further multiplied by a term x_i, and this idea was later formalized as the *examination hypothesis* [7] or the *position model* [6]. In 2008, N. Craswell *et al.* [7] compared this hypothesis to their new *cascade model*, which describes a user's behavior by assuming she scans from top to bottom. As the cascade model takes into account the relevance of urls above the clicked one, it outperforms the examination hypothesis [7].

In 2008, G. Dupret *et al.* [8] extended the examination hypothesis by considering the dependency on the positional distance to the previous click in the same query session. In 2009, F. Guo *et al.* [10] and O. Chapelle *et al.* [6] proposed two similar Bayesian network click models, that generalized the cascade model by

WSDM'10, February 4–6, 2010, New York City, New York, USA.

[*] This work was done when the first author was visiting Microsoft Research Asia.

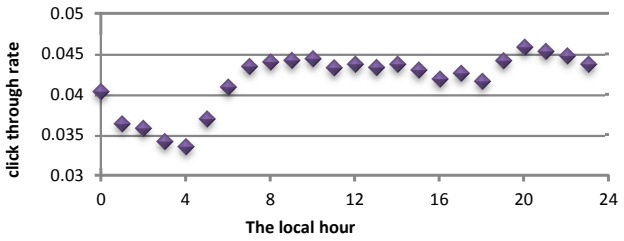

Figure 1. The empirical CTR with respect to the local hour.

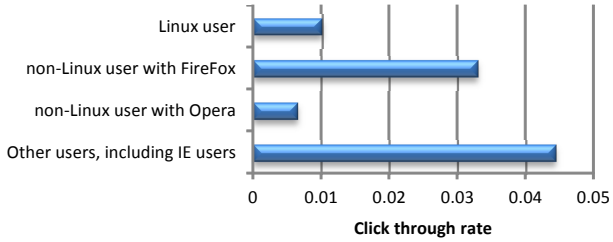

Figure 2. The empirical CTR with respect to the user agent.

analyzing user behavior in a chain-style network, within which the probability of examining the next result depends on the position and the identity of the current document.

Nevertheless, despite their successes, previous works have some limitations. First, they focus on training the click model for each individual query, and cannot accurately predict tail queries (lowfrequency queries) due to the inadequate training data. Second, the aforementioned models only considered the position-bias, neglecting other bias such as the local time (Figure 1) and the user agent (Figure 2), which are parts of the session-specific attributes in the log. We remark that the clicks through rate in these two graphs are averaged over all the advertisements from Jul. 29th to Jul. 31st, 2009 in a commercial search engine. This type of phenomenon has also been observed by D. Agarwal *et al.* [1] in predicting clicks for front page news of Yahoo!.

Based on these observations, it is fairly straightforward to build a click model on multiple attribute values shared across queries, instead of building models for individual queries. This may bring us the generalization to predict tail queries despite the lack of training data for a single query. Furthermore, we believe some other attributes are very important for click prediction and can be further incorporated to improve the accuracy. But how to accurately model the impact of different attribute values on the final prediction is a challenging problem.

In this paper, we propose a *General Click Model* built upon a Bayesian network and we employ the Expectation Propagation method [16] to perform approximate Bayesian inference. Our model assumes that users browse urls from top to bottom, and defines the transition probabilities between urls based on a list of attribute values, including the traditional ones such as "position" and "relevance" and our newly-designed ones such as the "local hour" and the "user agent". In summary, we highlight GCM with the following distinguishing features:

- *Multi-bias aware*. The transition probabilities between variables depend jointly on a list of attributes. This enables our model to explain bias terms other than the position-bias.
- *Across-query learning*. The model learns queries altogether and thus can predict clicks for one query – even a new query – using the learned data from other queries.
- *Extensibility*: The user may actively add or remove attributes applied in our GCM model. In fact, all the prior works mentioned above can reduce to our GCM as special cases when only one or two attributes are incorporated.
- *One-pass*. Our click model is an on-line algorithm. The posterior distributions will be regarded as the prior knowledge for the next query session.
- *Application to ads*. We have demonstrated our click model in the CTR prediction of advertisements. Experimental results show that our click model outperforms the prior works.

The rest of the paper is organized as follows: we first introduce some definitions and comment on prior works in Section 2, in which the discoveries motivate us to establish our *General Click Model* proposed in Section 3. Next in Section 4 we conduct experiments upon advertisement data and web search data, and compare the results in a number of metrics. We then discuss the extensions in Section 5 and conclude in Section 6.

2. BACKGROUND

We first clarify some definitions that will be used throughout this paper. When a user submits a *query* to the search engine, a *query session* is initiated. Specially, if a user re-submits the same query, a different query session is initiated. In our model, we only process the first page on a query session (we will discuss the usage of other pages in Section 5).

In each query session, there is a sequence of *urls*, denoted by $U = \{u_1, \dots u_M\}$ where the smaller subscript represents a higher rank, i.e. closer to the top. For regular search results, M is usually set to 10; while for ads data, M varies for different queries.

In the query session, each display of a url is called a *url impression*, which is associated with a list of *attribute values*, such as the user's IP address, the user's local time and the category of the url.

Now, we will introduce some prior works concerning click models, which fall into two categories: *examination hypothesis* and *cascade model*.

2.1 Examination Hypothesis

The *examination hypothesis* assumes that if a displayed url is clicked, it must be both *examined* and *relevant*. This is based on the eye-tracking studies which testify that users are less likely to click urls in lower ranks. In other words, the higher a url is ranked, the more likely it will be examined. On the other hand, the relevance of a url is a query-document based parameter which directly measures the probability that a user intends to click this url if she examines it. More precisely, given a query q and the url u at position i, the examination hypothesis assumes the probability of the binary click event C as follows:

$$P(C = 1|q, u, i) = \underbrace{P(C = 1|u, q, E = 1)}_{r_{u,q}} \cdot \underbrace{P(E = 1|i)}_{x_i} \quad (1)$$

Notice that (1) applies a hidden random variable E which denotes whether the user has examined this url.

In general, the examination hypothesis makes the following assumptions: if the user clicks it, then the url must have been

examined; if the user has examined the url, the click probability only depends on the relevance $r_{u,q}$; and the examination probability x_i depends solely on the position i.

Based on the examination hypothesis, three simple models studying $r_{u,q}$ and x_i have been introduced: the *Clicks Over Expected Clicks (COEC) model* [20], the *Examination model* [6], and the *Logistic model* [7]. They have been compared in [6] and experimentally proved to be outperformed by the cascade model.

An important extension of the examination hypothesis is the *user browsing model (UBM)* proposed by G. Dupret et al. [8]. It assumes the examination E depends not only on the position i but also on the previous clicked position l in the same query session ($l = 0$ if not existed).

$$P(C = 1|q, u, i, l) = \underbrace{P(C = 1|u, q, E = 1)}_{r_{u,q}} \cdot \underbrace{P(E = 1|i, l)}_{x_{i,l}} \quad (2)$$

2.2 Cascade Model

The *cascade model* [7] differs from the examination hypothesis above in that it aggregate the clicks and skips in a single query session into a single model. It assumes the user examines urls from the first one to the last one, and the click depends on the relevance of all the urls shown above.

Let E_i, C_i be the probablistic events indicating whether the ith url ($1 \le i \le M$) is examined and clicked respectively. The cascade model makes the following assumptions:

- $P(E_1) = 1$
- $P(E_{i+1} = 1|E_i = 0) = 0$
- $P(E_{i+1} = 1|E_i = 1, C_i) = 1 - C_i$
- $P(C_i = 1|E_i = 1) = r_{u_i,q}$ where u_i is the ith url

in which the third assumption implies that if a user finds her desired url, she immediately closes the session; otherwise she always continues the examination. The cascade model assumes that there is no more than one click in each query session, and if examined, a url is clicked with probability $r_{u,q}$ and skipped with probability $1 - r_{u,q}$. Thus,

$$P(C_i = 1) = r_{u_i,q} \prod_{j=1}^{i-1} \left(1 - r_{u_j,q}\right) \quad (3)$$

Based on the cascade model, two Bayesian network models have been proposed in 2009, both aiming at modifying the third assumption $P(E_{i+1} = 1|E_i = 1, C_i) = 1 - C_i$, and allowing multiple clicks in a single session. We will separately introduce these two models in the following subsections.

2.2.1 Click Chain Model (CCM)

The *click chain model* is introduced by F. Guo *et al.* [10]. It differs from the original cascade model in defining the transition probability from the ith url to the $(i+1)$th. CCM replaces the third assumption in cascade model with the following:

- $P(E_{i+1} = 1|E_i = 1, C_i = 0) = \alpha_1$ (4)
- $P(E_{i+1} = 1|E_i = 1, C_i = 1) = \alpha_2\left(1 - r_{u_i,q}\right) + \alpha_3 r_{u_i,q}$ (5)

where $\alpha_1, \alpha_2, \alpha_3$ are three global parameters independent of the users and the urls. CCM assumes that if the url at position i has been examined, the user clicks it according to the relevance $r_{u_i,q}$ as usual; if the user chooses not to click, the probability of continuing is α_1; if the user clicks, the probability to continue ranges between α_2 and α_3, depending on the relevance $r_{u_i,q}$.

CCM assumes that $\alpha_1, \alpha_2, \alpha_3$ are given and fixed, and then leverages the Bayesian inference to infer the posterior distribution of the document relevance r. Under the infinite-chain assumption the authors derived a simple method in computing the posterior, which enables CCM to run very efficiently.

2.2.2 Dynamic Bayesian Network (DBN)

The DBN model proposed by O. Chapelle and Y. Zhang [6] is very similar to CCM, but differs in the transition probability:

- $P(E_{i+1} = 1|E_i = 1, C_i = 0) = \gamma$ (6)
- $P(E_{i+1} = 1|E_i = 1, C_i = 1) = \gamma\left(1 - s_{u_i,q}\right)$ (7)

γ is a pre-defined parameter, and $s_{u_i,q}$, in place of $r_{u_i,q}$, is the measurement of the user's satisfaction on the actual content of u_i given query q. It is emphasized in [6] that a click does not necessarily imply the user's satisfaction on the content, instead, the user may have been attracted by some misleading abstracts. Therefore, the introduction of $s_{u_i,q}$ is to depict the actual relevance, rather than the perceived relevance $r_{u_i,q}$. Both values are estimated by applying the expectation-maximization algorithm in their paper, while there exists a Bayesian inference version to the model that is very similar to DBN on [17].

3. GENERAL CLICK MODEL

We now introduce our *General Click Model (GCM)*. Basically, it is a nested structure. The outer model in this nested structure is a Bayesian network, in which we assume users scan urls from top to bottom. The transition probabilities in this network are controlled by our inner model. Specifically, each individual probability is defined as summation of parameters, each corresponding to a single attribute value. This nested structure enables GCM to overcome not only the position-bias, but also other kinds of bias in learning the relevance and predicting the clicks.

3.1 The Outer Model

The outer Bayesian network of the General Click Model is illustrated in Figure 4, and the flow chart of the user behavior is given in Figure 3. The subscript goes from 1 to M, where M is the total number of urls on this page. As before, we define two binary random variables C_i and E_i that indicate whether the user clicks or examines the url on the ith position. In addition, we employ three continuous random variables at each position, A_i, B_i and R_i. The continuous behavior of A_i, B_i and R_i will enable GCM to handle not only the position-bias but other kinds of session-specific bias, to be explained later.

We assume the user examines the displayed urls from position $i = 1$ to $i = M$. After examining url u_i ($E_i = 1$), the user chooses to click it according to the relevance R_i. The click event will occur if and only if $R_i > 0$. Either way, the user will continue to examine the next url u_{i+1} with some probability: if u_i has been clicked ($C_i = 1$), the user will examine u_{i+1} if and only if $A_i > 0$; if u_i has not been clicked ($C_i = 0$), the user will examine u_{i+1} if

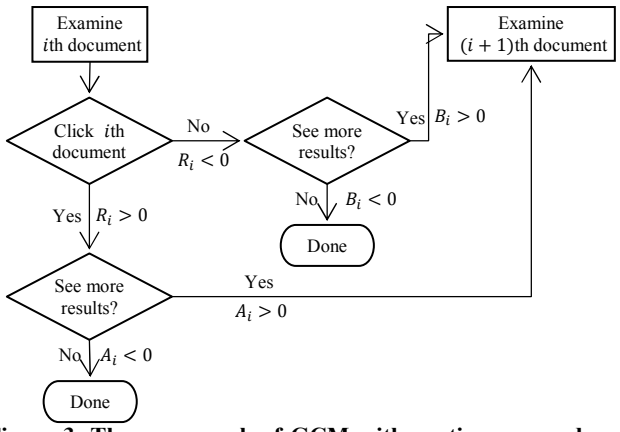

Figure 3. The user graph of GCM with continuous random variables R_i, A_i, B_i.

and only if $B_i > 0$. The following equation sets precisely describe the model:

- $P(E_1) = 1$ (8)
- $P(E_{i+1} = 1 | E_i = 0) = 0$ (9)
- $P(E_{i+1} = 1 | E_i = 1, C_i = 0, B_i) = \mathbb{I}(B_i > 0)$ (10)
- $P(E_{i+1} = 1 | E_i = 1, C_i = 1, A_i) = \mathbb{I}(A_i > 0)$ (11)
- $P(C_i = 1 | E_i = 1, R_i) = \mathbb{I}(R_i > 0)$ (12)

where $\mathbb{I}(.)$ is the characteristic function, and we define $\Phi = \{A_i, B_i, R_i | i = 1 \dots M\}$.

This model differs from DBM or CCM in that the transition probability depends on continuous random variables in Φ. Next, we will show that those variables are further modeled as the summation of a list of parameters, each corresponding to an *attribute value*.

3.2 The Inner Model

When a query session is initiated with query q and urls $U = \{u_1, \dots u_M\}$, the attributes the search engine holds are far beyond the url and the query themselves. We separate the session-specific attributes into two categories:

- The *user-specific attributes*: the query, the location, the browser type, the local hour, the IP address, the query length, etc. We denote their values by $f_1^{user}, \dots f_s^{user}$.
- The *url-specific attributes*: the url, the displayed position (=i), the classification of the url, the matched keyword, the length of the url, etc. For a specific url u_i, we denote these attribute values by $f_{i,1}^{url}, \dots f_{i,t}^{url}$.

As an illustration, if we take five attributes into account: the query, the browser type, the local hour, the url, and the position, we have $s = 3, t = 2$. For a specific url impression on position 2, we may have the following values:

- f_1^{user}="Microsoft Research"; f_2^{user}=IE; f_3^{user}=7pm;
- $f_{2,1}^{url}$="research.microsoft.com", $f_{2,2}^{url}$=2.

At this point, we assume that attributes are all of discrete values[1]. Furthermore, we assume each value f is associated with three

[1] We will consider the continuous feature values in Section 5.

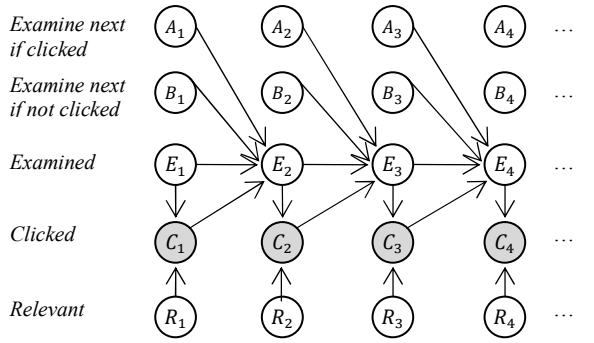

Figure 4. The Bayesian network of GCM. A_i, B_i, E_i, R_i are hidden variables, while C_i is the observed clicks. $1 \le i \le M$

parameters θ_f^A, θ_f^B and θ_f^R, each of which is a continuous random variable. We define:

$$A_i = \sum_{j=1}^{s} \theta_{f_j^{user}}^A + \sum_{j=1}^{t} \theta_{f_{i,j}^{url}}^A + err$$
$$B_i = \sum_{j=1}^{s} \theta_{f_j^{user}}^B + \sum_{j=1}^{t} \theta_{f_{i,j}^{url}}^B + err \quad (13)$$
$$R_i = \sum_{j=1}^{s} \theta_{f_j^{user}}^R + \sum_{j=1}^{t} \theta_{f_{i,j}^{url}}^R + err$$

where err is an error term satisfying $N(0,1)$ distribution. For simplicity of explanation, we define $\Theta = \{\theta_f^A, \theta_f^B, \theta_f^R | \forall f\}$ where f enumerates from all distinct attribute values. Besides error terms, we treat A_i, B_i and R_i as summations of $s + t$ parameters in Θ, and those parameters satisfy independent Gaussian distributions. We emphasize that variables in Φ are defined for a specific query session, while parameters in Θ are defined across sessions. We will next use the Bayesian inference to learn the distributions for those parameters.

3.3 The Algorithm: On-line Inference

Our algorithm is built upon the *Expectation Propagation* method [16]. Given the structure of a Bayesian network with hidden variables, EP takes the observation values as input, and is capable of calculating the inference of any variable. We simply assume an EP calculator G exists while the detailed algorithmic design can be found in [16] [18].

Algorithm: The General Click Model

1. Initiate $\Theta = \{\theta_f^A, \theta_f^B, \theta_f^R | \forall f\}$ and let each parameter in Θ satisfy a prior $N(0, 1/(s+t))$.
2. Construct a Bayesian inference calculator G using *Expectation Propagation*.
3. For each session s
4. $M \leftarrow$ number of urls in s
5. Obtain the attribute values
$$F = \{f_1^{user}, \dots f_s^{user}\} \cup \{f_{i,1}^{url}, \dots f_{i,t}^{url}\}_{i=1}^{M}$$
6. Input $\{\theta_f^A, \theta_f^B, \theta_f^R | f \in F\} \subset \Theta$ to G as the prior Gaussian distributions.
7. Input the user's clicks to G as observations.
8. Execute G to measure the posterior distributions for $\{\theta_f^A, \theta_f^B, \theta_f^R | f \in F\}$, and update them in Θ
9. End For

As stated previously, we assign each parameter in Θ a Gaussian distribution, and update this distribution in the assumed-density filtering mechanism [16]: for each query session, the calculated posterior distributions will be used as prior distributions for the next query session.

At the beginning of the algorithm, we assume all parameters in Θ satisfy a default prior Gaussian distribution, say $N(0,1/(s+t))$, for all distinct values f. We assume the nested Bayesian network (described in section 3.1 and 3.2) has been constructed and the Bayesian inference calculator G is properly set.

We will process the query sessions one by one. For each coming session we obtain its attribute value list

$$F = \{f_1^{user}, \dots f_s^{user}\} \cup \{f_{i,1}^{url}, \dots f_{i,t}^{url}\}_{i=1}^{M}$$

and retrieve their corresponding parameters in Θ as prior Gaussian distributions for G, along with the click-or-not flags. G will calculate the posterior Gaussian distributions of θ_f^A, θ_f^B and θ_f^R for each related attribute value $f \in F$. The inferred posterior Gaussians are saved for the next iteration.

Note that if M is fixed, the Bayesian network structure stays fixed throughout the algorithm. Though values f_j^{user} and $f_{i,j}^{url}$ vary from session to session, however, the structure of the Bayesian *factor graph* remains unique. In some other words, for example, A_i is always the summation of $s+t$ Gaussians and an err term. This behavior enables us to pre-calculate the Bayesian inference formula for G and speed up the on-line inference calculation, for example using software Infer.NET [18]. If M varies, such as for advertisement data, we may build M different calculators $G_1, G_2, ..$ and classify the query session according to the corresponding value M.

3.4 Reduction from Prior Works

In this section we will see that all prior models we mentioned in Section 2 can be regarded as special cases of GCM, and this is why our model is named *General Click Model*.

As shown above, prior models give the transition probabilities explicitly, such as $r_{u_i,q}$. Instead, we model them as continuous random variables A_i, B_i and R_i and defined each of them as the summation of a list of parameters in Eq. (13). The following lemma connects prior works to our continuous-random-variable definition.

Lemma: If we define an attribute value f to be the pair of query and url $f = (u_i, q)$, the traditional transition probability

$$P(C_i = 1 | E_i = 1) = r_{u_i,q}$$

can reduce to

$$P(C_i = 1 | E_i = 1, R_i) = \mathbb{I}(R_i > 0)$$

if we set $R_i = \theta_f^R + err$ and θ_f^R is a point mass Gaussian (also known as the *Dirac delta distribution*) centered at $F^{-1}(r_{u_i,q})$, where F is the cumulative distribution function of $N(0,1)$.

Proof. Assume the probability density function of $N(0,1)$ is $p(x)$, we make the following calculation:

$$P(C_i = 1 | E_i = 1) = \int p(x) \cdot \mathbb{I}\left(x + F^{-1}(r_{u_i,q}) > 0\right) \mathrm{d}x = \int_{-F^{-1}(r_{u_i,q})}^{\infty} p(x) \mathrm{d}x = r_{u_i,q} \blacksquare$$

Similarly, this lemma can be extended to A_i and B_i. We will next adopt the lemma and re-write Eq. (13) in the form that prior models can reduce to our GCM, with the restriction that all Gaussian distributions degenerate to point mass Gaussians.

3.4.1 Examination hypothesis

The traditional *examination hypothesis* assumes that the click probability is the multiplication of a position-based examination rate x_{i+1} and a relevance-based click rate $r_{u_i,q}$ (see Eq.(1)). In GCM, if

$$P(B_i > 0) = P(A_i > 0) = x_{i+1};\ P(R_i > 0) = r_{u_i,q} \quad (14)$$

we immediately arrive at the examination hypothesis Eq.(1) according to Eq. (8) ~ (12). To achieve this we define two attributes $f_1 = i+1$ and $f_2 = (u_i, q)$. According to the lemma, we fix parameters $\theta_{f_1}^A, \theta_{f_1}^B$ and $\theta_{f_2}^R$ to the point mass Gaussians centered at $F^{-1}(x_{i+1})$, $F^{-1}(x_{i+1})$ and $F^{-1}(r_{u_i,q})$ respectively. Then, Eq. (14) will be achieved if we define the following in Eq. (13):[2]

$$A_i = \theta_{f_1}^A + err;\ B_i = \theta_{f_1}^B + err;\ R_i = \theta_{f_2}^R + err \quad (15)$$

Its extension, the *user browsing model* (UBM) [8] can similarly reduce to GCM, by letting $P(A_i > 0) = x_{i+1,l}$ and $P(R_i > 0) = r_{u_i,q}$, where l is the distance to the previous click. The only modification we need is to set $f_1 = (i+1, l)$, and $\theta_{f_1}^A, \theta_{f_1}^B$ be the point mass Gaussians centered at $F^{-1}(x_{i+1,l})$.

3.4.2 Cascade models

In the traditional *cascade model*, it is just a special case of GCM where $B_i > 0$ and $A_i < 0$, meaning that the user always examines the next url if not clicked, and immediately stops if clicked (see Eq. (10) and (11)). This can be approximated if we define a dummy attribute f_1, and let $\theta_{f_1}^A, \theta_{f_1}^B$ be point mass Gaussians at -10 and $+10$ respectively. Again, we let $f_2 = (u_i, q)$ and set Eq. (13) to

$$A_i = \theta_{f_1}^A + err;\ B_i = \theta_{f_1}^B + err;\ R_i = \theta_{f_2}^R + err \quad (16)$$

In the *click chain model* (CCM), α_1, α_2 and α_3 are global constants. We define a dummy attribute f_1 and let $\theta_{f_1}^B$ be fixed to point mass centered at $F^{-1}(\alpha_1)$, while $f_2 = (u_i, q)$ and $\theta_{f_2}^R$ are as before. Then, we add a new parameter $\theta_{f_2}^A$, a point mass Gaussian centered at $F^{-1}(\alpha_2(1 - r_{u_i,q}) + \alpha_3 r_{u_i,q})$. Under such configuration, we arrive at Eq. (4) and Eq. (5) with the following:

$$A_i = \theta_{f_2}^A + err;\ B_i = \theta_{f_1}^B + err;\ R_i = \theta_{f_2}^R + err \quad (17)$$

In the *dynamic Bayesian network* (DBN) model, γ is a global constant. We again define a dummy attribute f_1 and let $\theta_{f_1}^B$ satisfy the point mass Gaussian at $F^{-1}(\gamma)$. Then we define $\theta_{f_2}^A$ to be point mass Gaussian at $F^{-1}(\gamma(1 - s_{u_i,q}))$, in which $f_2 = (u_i, q)$ and $\theta_{f_2}^R$ are as before. Now we arrive at Eq. (6) and (7) in changing Eq. (13) to Eq. (17).

[2] In consistence with Eq. (13), we tacitly assume that $\theta_{f_1}^R = \theta_{f_2}^A = \theta_{f_2}^B = 0$, similarly hereinafter.

Table 1. Advertisement dataset

Set	Query Freq	#Queries	Train set		Test set	
			#Sessions	#Urls	#Sessions	#Urls
1	1~10	141	866	5,698	177	1,057
2	10~30	1,211	24,928	1,664,403	2,122	13,664
3	30~100	5,058	308,203	1,810,009	18,629	105,716
4	100~300	3,988	674,654	3,148,826	40,304	180,532
5	300~1000	1,651	847,722	3,011,482	54,098	184,606
6	1,000~3,000	481	792,422	2,470,665	48,449	147,561
7	3,000~10,000	132	660,645	1,508,985	42,067	92,122
8	10,000~30,000	22	315,832	769,786	19,338	48,808
9	30,000+	7	642,835	1,046,948	37,796	64,236
All	All of above	12,691	4,267,241	15,431,104	262,803	837,245

Table 2. Search dataset

Set	Query Freq	#Queries	Train set		Test set	
			#Sessions	#Urls	#Sessions	#Urls
1	1~10	2,238	10,847	100,144	5,686	50,651
2	10~30	2,379	43,254	392,736	19,923	175,832
3	30~100	2,035	106,962	973,685	52,313	466,811
4	100~300	587	97,984	868,812	49,355	425,080
5	300~1000	219	111,431	960,250	57,902	488,684
6	1,000~3,000	79	128,270	1,114,753	64,696	549,797
7	3,000~10,000	24	115,082	1,045,407	51,827	459,584
8	10,000~30,000	5	101,584	943,057	53,805	493,313
All	All of above	7,566	715,414	6,398,844	355,507	3,109,752

4. EXPERIMENTS

In this section, we conduct experiments on the advertisement data of a commercial search engine. Four different metrics have been employed to verify and compare the accuracy for different click models. At the same time, we had an additional test on the web search data in the last sub-section.

4.1 Experimental Setup

We implemented the *Cascade model* [7], the *Click Chain Model* [10] and the *Dynamic Bayesian Model* [6] under the Bayesian inference framework Infer.NET 2.3 [18]. The global parameters, $\alpha_1, \alpha_2, \alpha_3$ in CCM and γ in DBM are automatically studied using Bayesian inference, and the details of which can be found in the Appendix. For the cascade model, we ignored all the sessions with more than one clicks in the training data. Those three algorithms are employed as the baseline models and we ignored the examination-hypothesis-based ones such as UBM [8], because the most recent works have clearly suggested that the examination-hypothesis-based models are worse than the cascade-based ones [10] [6]. All the programs, including our *General Click Model*, are implemented in MS Visual C# 2008, and the experiments are carried out on a 64-bit server with 47.8 GB of RAM and eight 2.40 GHz AMD cores.

Next, we will introduce two datasets sampled from a commercial search engine that will be used in our experiment.

4.1.1 Advertisement Dataset

We collect three day's click through data with ads clicks and url impressions and 12,691 queries are sampled. As stated before, we restrict ourselves to the results on the first page. If multiple clicks exist on a single page, we ignore the click order and assume the user clicks from top to bottom. We retrieve 4,530,044 query sessions and 16,268,349 url impressions from the log from Jul. 29th to Jul. 31st. The number of url impressions on a single page vary from 1 to 9 on this search engine, with an average of 3.6 url impressions in each query session. We use the first 68 hours of data to train and predict on the last 4 hours.

Following [10], we divide the queries according to the *query frequency* – the number of sessions in each query (see Table 1). We conduct experiments not only for the whole data set (Set "All" in Table 1), but also for individual frequency intervals Set 1 ~ Set 8. We discard Set 9 because the number of sessions for each query is so large that a simple model can predict it accurately. We employ 21 different attributes in our GCM for this dataset, including the user IP, the user country, the user agent, the local hour, the ad id, the ad category, the matched keyword, etc.

4.1.2 Search Dataset

Similar to the advertisement data, we retrieve a three-day log of web search results from Jun. 28th to Jun. 30th, and sample 7,568 queries with 959,148 query sessions and 8,813,048 url impressions. The first two days of data are used as the training set while the third day is used for testing. We classify the queries according to their frequency in Table 2. We ignore two 30,000+ frequency queries "google" and "facebook", because nearly all the users simply click on the first result and close the session.

Regarding the lack of data, we have very limited attributes for this search dataset. Except for the query, url and position, we employ in GCM the following attributes: the user country, the user agent, the global hour and the domain of the url. A total of 7 attributes.

4.2 Evaluation on Log-Likelihood

A very common measurement of the accuracy for click models is the *Log-Likelihood (LL)*, also known as the *empirical cross entropy*. For a single url impression, if the predicted click rate is c, the LL value is $\log c$ if this url is clicked, and is $\log(1-c)$ if not clicked. The LL of a dataset with a number of query sessions is measured as the average LL on individual url impressions. A perfect click prediction has an LL of 0 and the larger this number indicates the better the prediction. Based on [10], the improvement of LL value ℓ_1 over ℓ_2 is computed as $\left(e^{\ell_1-\ell_2}-1\right)\times 100\%$.

We demonstrate our LL test result for the advertisement dataset in Figure 5. The baseline algorithm equally predicts all url impressions with the same probability – the average probability over the entire test set. Being aware that the click probability for advertisement data is significantly smaller than for web search data, one may find that even the baseline algorithm's LL value is very close to 0.

From Figure 5 we clearly see the superiority of our proposed GCM in the click prediction of advertisement data. We emphasize that GCM overwhelms the most recent click models CCM and DBN especially for tail queries – less frequent queries. This is expected because our model trains queries altogether, while prior works train the data by query, thus lacking the training data for tail queries. Our experiment also confirmed the result in [6] that DBN should perform better than the cascade model. On the entire dataset our improvement is 1.2% over CCM and DBN, and 1.5% over the Cascade model. We remark that this percentage is significant because ads data has a rather low click rate.

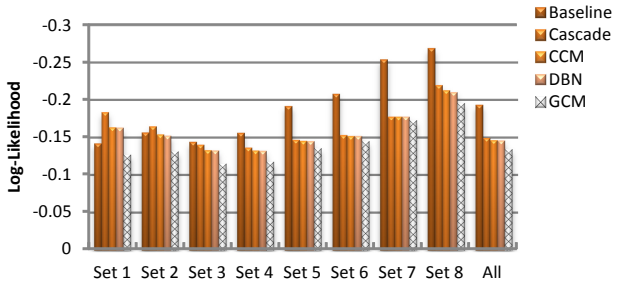

Figure 5. The log-likelihood of different models on the advertisement dataset, for different query frequencies.

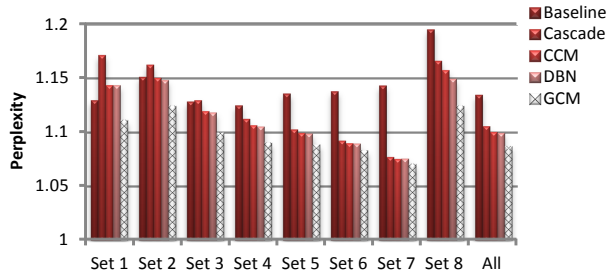

Figure 6. The perplexity of different models on the advertisement dataset, for different query frequencies.

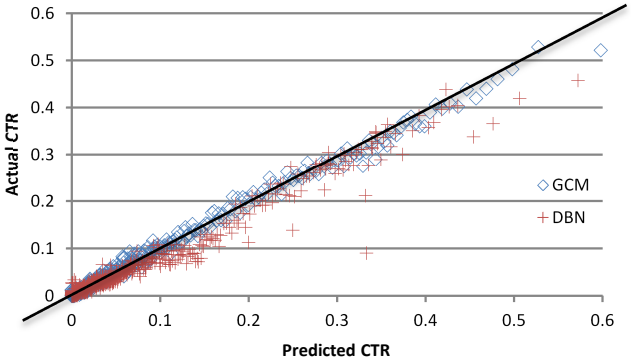

Figure 7. Actual vs predicted CTR for GCM and DBN

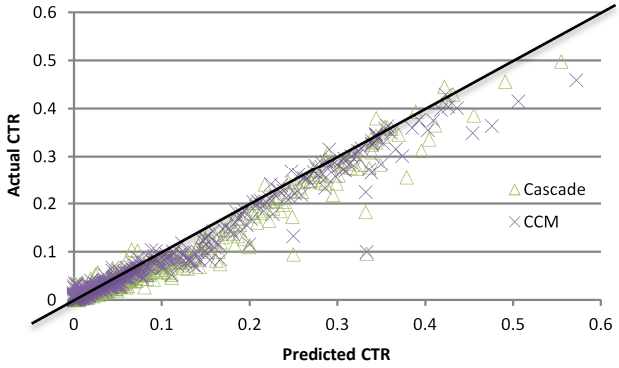

Figure 8. Actual vs predicted CTR for Cascade and CCM

4.3 Evaluation on Perplexity

We also incorporate *click perplexity* [11] [10] as the evaluation metric for our model. This value measures the accuracy for individual positions separately and will penalize a model that performs poorly even in a single position. For a given position i, and a set of query sessions $s_1, \dots s_N$. We assume that all sessions have more than i url impressions, and use $c_1, \dots c_N$ to denote the binary click events of the ith url for $s_1, \dots s_N$ respectively. Let $q_1, \dots q_N$ indicate the corresponding predicted click rates. The click perplexity at position i is:

$$p_i = 2^{\frac{1}{N}\sum_{i=1}^{N} c_i \log_2 q_i + (1-c_i)\log_2(1-q_i)}$$

The perplexity of the entire dataset is the average of p_i over all positions. A perfect click prediction will have a perplexity of 1 and the smaller this number indicates better prediction accuracy. The improvement of perplexity value p_1 over p_2 is calculated as $(p_2 - p_1)/(p_2 - 1) \times 100\%$ [10].

In Figure 6 we have compared the perplexity for different models. Our proposed CCM outperforms the cascade model with a 17.4% improvement, CCM with 12.9% and DBN with 12.1%. Again, the superiority is highlighted when the query frequency is low. We will illustrate the positional perplexity in the Section 4.5.

4.4 Evaluation on R2

In this experiment, we sort url impressions according to their predicted CTR, and then divide them into blocks of 1,000 url impressions each. In block i, we define x_i the predicted CTR that is averaged over 1,000 individual impressions, and define y_i the actual CTR that is the number of empirical clicks divided by 1,000. We then draw the x-y scatter graph for all the four models in Figure 7 and Figure 8. One may see that the points (x_i, y_i) of GCM are the closest to $y = x$, and thus it has the highest click prediction accuracy. More precisely, we use the R^2 value to measure the prediction.

The *coefficient of determination*, also known as R^2, has been widely used in statistics to measure the linear relationship between two sets of data. For $\{x_i\}_{i=1}^N$ and $\{y_i\}_{i=1}^N$, R^2 is calculated as the following (assuming $a = 1, b = 0$ in our case):

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - ax_i - b)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

The larger R^2 indicates the more correlated $\{y_i\}_{i=1}^N$ to $\{x_i\}_{i=1}^N$, and thus the better performance of the model. An optimal value of R^2 is 1. Among the four models GCM does the most outstanding job, with an R^2 of 0.993, while Cascade, CCM and DBN receive 0.956, 0.939 and 0.958 respectively.

4.5 Evaluation on Bias

To distinguish between models on how well the position-bias is explained, we separately compare the prediction accuracies for different positions, on the basis of click probability (Figure 9) and position perplexity (Figure 10).

In Figure 9, we averaged the click rates for all 9 positions, and compare them with the actual click rates. Results show that all the models accurately predict the click rate on the first two positions, while GCM is undoubtedly the best model to explain the click rate for the last four url impressions. Something worth noting is that

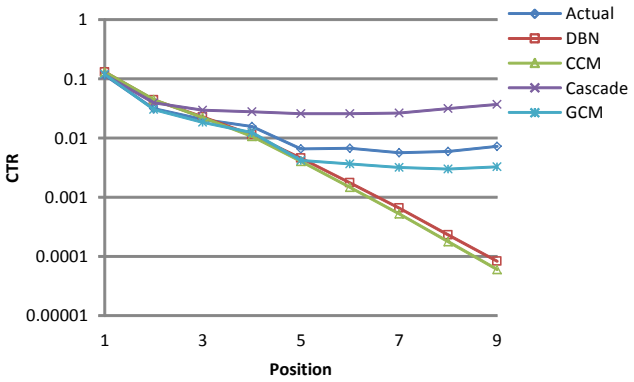

Figure 9. Positional CTR for the advertisement data

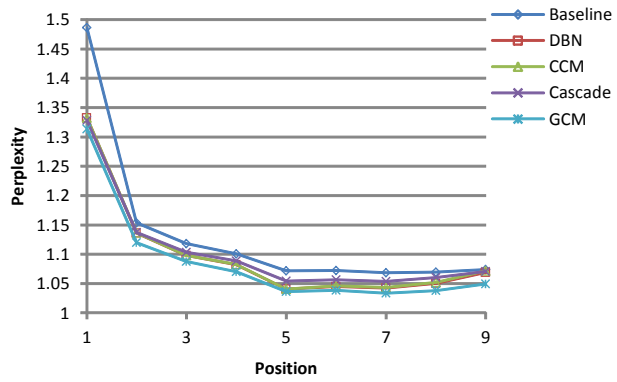

Figure 10. Positional perplexity for the advertisement data.

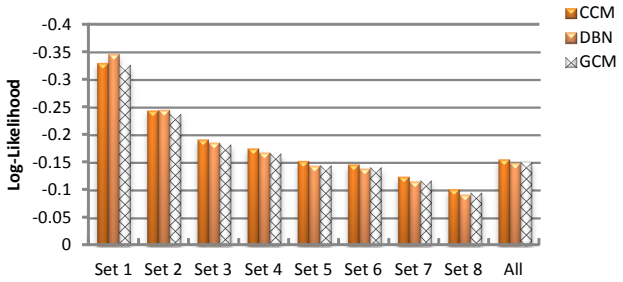

Figure 11. The log-likelihood of different models on the search dataset, for different query frequencies.

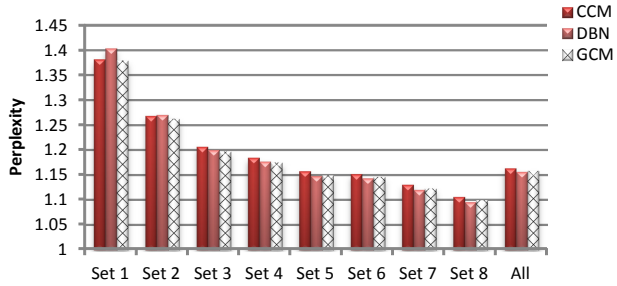

Figure 12. The perplexity of different models on the search dataset, for different query frequencies.

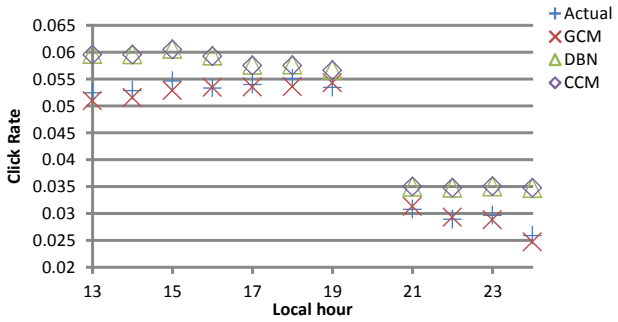

Figure 13. Comparisons of the estimated and actual click rates for different local hours on the advertisement dataset.

those global constants, $\alpha_1, \alpha_2, \alpha_3$ in CCM and γ in DBN, are not associated with the position i. These variables force the click rate to decrease exponentially with i, while for advertisement data, this assumption is not necessarily true.

In Figure 10, we can also see that GCM is the best among the five. It has an improvement of 5.6% on the first position, and around 30% on the last position over CCM and DBN.

To work in concert with our discovery in Figure 1, we examined how well our GCM predicts the query sessions for different local hours. Though the test set we employ has a span of only 4 hours in the server time, the local hour of global users varies from 13:00 to 24:00. The results in Figure 13 show that our GCM successfully explained the local hour bias, while in contrast, DBN and CCM fail to explain the CTR drop for the midnight users.

At last, we compare the influences of all attributes incorporated in GCM and see which of them are the most important. Fix an attribute, we enumerate from all of its discrete values and retrieve a set of Gaussian distributions. Then, the standard deviation of those Gaussians' mean values are calculated. According to our model, the larger this deviation, the more influential this attribute is. Results show that the three most influential attributes are the position, the match type (strongly related to the relevance) and the user agent (recall Figure 2).

4.6 Additional Test on Search Data

We have shown the overwhelming performance of our proposed GCM on ads data. As an addition to this paper, we hope to know how well it predicts the clicks in web search results.

In Figure 11 and Figure 12, we compared our proposed GCM with the most recent model CCM and DBN on the search data. The log-likelihood result and the perplexity result both illustrate that GCM does comparably well with the state-of-the-art models, and with a slight improvement on the low frequency query set, as expected. One thing worth noting is that DBN and CCM separately show their competence on high and low frequency data sets respectively, while GCM does well on all kinds of frequencies.

We regard the reasons for the insignificant improvement for search data as the follows, and we will do more investigation for this additional work in the future:

- Prior works focused on the search data and reasonably simplified the model. This enables the model to do well on the search data, but may fail in explaining the ads.

- Based on the search data available to us, most of the important attributes we employed for the ads data are missing. We incorporate in GCM only 7 attributes for the web search data, in comparison with the 21 attributes for the advertisement data.

As mentioned by an anonymous reviewer, the problem of click mode in advertising is considered significantly harder than in web search. This is because the volume of the low amount of data available as well as the low CTR rates. So it is not surprising that a more complex model such as GCM does better than competitors on ad data and does not show improvements on search data.

5. DISCUSSIONS & FURTHER WORKS

We have seen that the prior works can theoretically reduce to our GCM, and at the same time, our model outperforms prior works in advertisement data. In this section we discuss some pros and cons and potential extensions.

To learn CTR@1. One of the most important by-products of the cascade click model is an unbiased CTR measurement assuming the url is placed at position 1. This value can be further used to improve NDCG [6], or build the auction for ads [9]. In our GCM, we can predict CTR@1 in this way.

CTR@1 can be learned in a similar way in our model. For a given url impression, assuming its position to be 1, some of the user-specific attributes are missing during the prediction, such as the local hour and user location. Under these circumstances, we may calculate the expected distributions $\mathbb{E}[A_i], \mathbb{E}[B_i]$ and $\mathbb{E}[R_i]$ over the distributions of all missing attributes. Practically, these distributions can be approximated by the empirical data. At last, $P(R_1 > 0)$ is the probability of clicks if this url is put at position 1.

Using the variance. One important feature of GCM is that each attribute value is associated with a variance, attached to its Gaussian distribution. This value measures the significance of this attribute so we no longer need an extra confidence calculation such as the Appendix of [6]. If GCM is applied to the real-time search engine, this variance could be enlarged periodically, maybe once a day, because the web data keeps changing as time goes by.

Continuous attribute values. Our model assumes the attribute values to be discrete, however, there might exist some continuous attributes, e.g. the widely used *BM25* score in ranking. One way to incorporate such an attribute is to divide continuous values into discrete bins, such as 1,000 equal-width intervals. A more straight-forward way is to modify Eq. (13) by adding multiplication terms such as $\theta^R_{BM25} \cdot x$, where x is the BM25 value and θ^R_{BM25} is a global parameter that is independent of the value x. θ^R_{BM25} behaves as a dynamic weight associated to this attribute and can be learned by Bayesian inference.

Make use of the page structure. As stated in [6], we can make use of the pagination links on the search page in designing a more accurate Bayesian network. More importantly, in the ads distribution of our commercial search engine, url impressions are allocated in two different areas – the main line and the side bar. In our experiment, the actual CTR of the former is significantly larger than the latter. We will in our further work separate our Bayesian network into two parts which might better explain the area-bias of the CTR estimation in the advertisement data.

Running time. Our GCM achieved the result on the entire ads dataset in 10.3 hours and the entire search dataset in 2.7 hours. Under our implementation, CCM needs 2.1h/1.6h and DBN needs 1.2h/0.8h. We will investigate if any approximate Bayesian inference calculation exists that can help improve GCM's efficiency. Meanwhile, the search engine can classify the queries and initiate a bunch of GCMs that work in parallel for different query sets, and thus makes GCM capable of handling billion-scale real-time data.

6. CONCLUSION

In this paper, we proposed a novel click model called General Click Model (GCM) to learn and predict user click behavior towards display urls on a search engine. The contribution of this paper is three-fold. Firstly, different from previous approaches that learn the model based on each individual query, GCM learns the click model based on multiple attributes and the influence of different attribute values can be measured by Bayesian inference. This advantage in learning helps GCM to achieve a better generalization and can lead to better results, especially for the tail queries. Secondly, most of the existing works only consider the position and the identity of url when learning the model. GCM considers more session-specific attributes and we demonstrate the importance of these attributes to the final prediction. Finally, we found most of the existing click models can be reduced to GCM by assigning different parameters.

We conducted extensive experiments on a large-scale commercial advertisement dataset to compare the performance between GCM and three state-of-the-art works. Experimental results show that GCM can consistently outperform all the baseline models on four metrics.

7. ACKNOWLEDGMENTS

The authors want to thank Haixun Wang, Gang Wang and Dakan Wang from MSRA for useful discussions, and acknowledge Matt Callcut and all three anonymous reviewers for their comments.

Zeyuan Allen Zhu thanks Fan Guo from CMU for his suggestions on this topic. Zeyuan is also partially supported by the National Innovation Research Project for Undergraduates (NIRPU).

8. REFERENCES

[1] Agarwal, D., Chen, B-C., and Elango, P. Spatio-Temporal Models for Estimating Click-through Rate. In *WWW* 2009.

[2] Aggarwal, G., Muthukrishnan, S., Pál, D., and Pál, M. General Auction Mechanism for Search Advertising. In *WWW* 2009.

[3] Agichtein, E., Brill, E., and Dumais, S. Improving web search ranking by incorporating user behavior information. In *SIGIR* 2006.

[4] Agichtein, E., Brill, E., Dumais, S., and Ragno, R. Learning User Interaction Models for Predicting Web Search Result Preferences. In *SIGIR* 2006.

[5] Carterette, B. and Jones, R. Evaluating search engines by modeling the relationship between relevance and clicks. In *NIPS* 2008.

[6] Chapelle, O. and Zhang, Y. A Dynamic Bayesian Network Click Model for Web Search Ranking. In *WWW* 2009.

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

[8] Dupret, G. and Piwowarski, B. A User Browsing Model to Predict Search Engine Click Data from Past Observations. In *SIGIR* 2008.

[9] Goel, A. and Munagala, K. Hybrid Keyword Search Auctions. In *WWW* 2009.

[10] Guo, F., Liu, C., Kannan, A., Minka, T., Taylor, M., and Wang, Y-M. Click Chain Model in Web Search. In *WWW* 2009.

[11] Guo, F., Liu, C., and Wang, Y-M. Efficient Multiple-Click Models in Web Search. In *WSDM* 2009.

[12] Joachims, T. Evaluating retrieval performance using clickthrough data. In *SIGIR Workshop on Mathematical/Formal Methods in Information Retrieval* 2002.

[13] Joachims, T. Optimizing search engines using clickthrough data. In *SIGKDD* 2002.

[14] Joachims, T., Granka, L., Pan, B., Hembrooke, H., and Gay, G. Accurately Interpreting Clickthrough Data as Implicit Feedback. In *SIGIR* 2005.

[15] Joachims, T., Granka, L., Pan, B., Hembrooke, H., Radlinski, F., and Gay, G. Evaluating the accuracy of implicit feedback from clicks and query reformulations in Web search. In *ACM Transactions on Information Systems* 2007.

[16] Minka, T. *A family of algorithms for approximate Bayesian inference*. MIT, 2001. PhD thesis.

[17] Minka, T., Winn, J., Guiver, J., and Kannan, A. *Click through model - sample code*. Microsoft Research Cambridge, 2009. http://research.microsoft.com/en-us/um/cambridge/projects/infernet/docs/Click%20through%20model%20sample.aspx.

[18] Minka, T., Winn, J., Guiver, J., and Kannan, A. *Infer.NET 2.3*. Microsoft Research Cambridge. http://research.microsoft.com/infernet, 2009.

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

[20] Zhang, W. V. and Jones, R. Comparing Click Logs and Editorial Labels for Training Query Rewriting. In *WWW* 2007.

9. APPENDIX

9.1 Our Implementation to Prior Work

For better comparisons between models, we equally employ the Infer.NET framework for all baseline programs. Inspired by the code to a very similar model of DBN [17], we implemented Cascade, CCM and DBN in the way that the probabilities are assumed to obey Beta distributions. At the beginning of the program, all those distributions are set to a uniform $Beta(1,1)$, and they will be updated according to Bayesian inference [16].

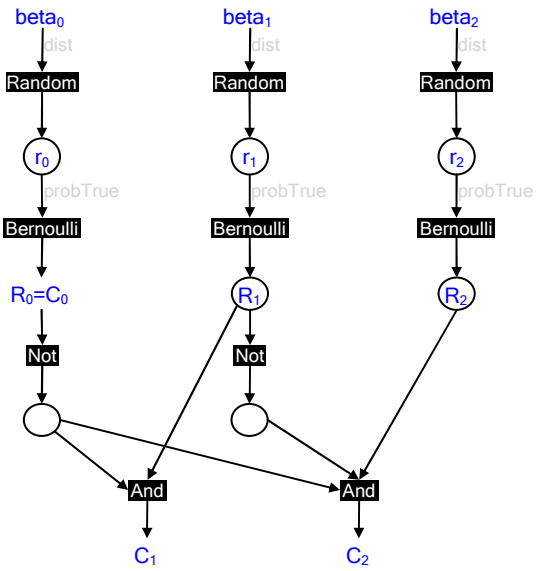

Figure 14. The factor graph of Cascade model under Infer.NET when $M = 3$. Circles are hidden variables and $beta_0..beta_2$, $C_0..C_2$ are observed values

We first look at the Cascade model, we draw the factor graph of the Bayesian network assuming $M = 3$.

In Figure 14 we see that the relevance r_0, r_1, r_2 obey the given Beta distributions $beta_0$, $beta_1$ and $beta_2$ respectively, and the binary events $R_0, .. R_2$ are defined according to the Bernoulli distribution $P(R_i = 1) = r_i$. Based on Expectation Propagation, posterior distributions of r_0, r_1, r_2 can be approximated by new Beta distributions using Infer.NET, and will be used as the prior distribution for the next query session.

For the sake of simplicity, we ignore the factor graph for CCM and DBN here. The basic ideas are the same: the transition probability, for example

$$P(E_{i+1} = 1 | E_i = 1, C_i = 1) = \gamma(1 - s_{u_i,q})$$

can be written in the language of Infer.NET, through two Boolean variables Γ and S, satisfying $P(\Gamma = 1) = \gamma$ and $P(S = 1) = s_{u_i,q}$, where γ and $s_{u_i,q}$ follow some Beta distributions of their own. Under the conditions of $E_i = 1$ and $C_i = 1$, E_{i+1} will happen if $\Gamma = 1$ and $S = 0$.

In our implementation, we not only require the relevance $r_{u_i,q}$ and the satisfaction rate $s_{u_i,q}$ to follow Beta distributions, at the same time, we let $\alpha_1, \alpha_2, \alpha_3$ in CCM and γ in DBN satisfy their corresponding Beta distributions. They will be inferred by the Expectation Propagation process and automatically adjusted during the experiment.

Notice that our implementation of CCM discards the infinite-chain assumption, and thus runs slower than it has been reported in [10]. For DBN, we have not followed the EM steps according to [6], and used Bayesian inference instead. This is because we want to compare the models rather than the optimization methods.