Analyzing User's Sequential Behavior in Query Auto-Completion via Markov Processes

Liangda Li[¶†], Hongbo Deng[‡], Anlei Dong[‡], Yi Chang[‡], Hongyuan Zha[¶†], and Ricardo Baeza-Yates[‡]

[†]College of Computing
Georgia Institute of Technology
Atlanta, GA 30032

[‡]Yahoo Labs
701 First Avenue
Sunnyvale, CA 94089

[¶]Software Engineering Institute
East China Normal University
Shanghai, China 200062

ldli@cc.gatech.edu, {hbdeng, anlei, yichang}@yahoo-inc.com, zha@cc.gatech.edu, rbaeza@acm.org

ABSTRACT

Query auto-completion (QAC) plays an important role in assisting users typing less while submitting a query. The QAC engine generally offers a list of suggested queries that start with a user's input as a prefix, and the list of suggestions is changed to match the updated input after the user types each keystroke. Therefore rich user interactions can be observed along with each keystroke until a user clicks a suggestion or types the entire query manually. It becomes increasingly important to analyze and understand users' interactions with the QAC engine, to improve its performance. Existing works on QAC either ignored users' interaction data, or assumed that their interactions at each keystroke are independent from others. Our paper pays high attention to users' sequential interactions with a QAC engine in and across QAC sessions, rather than users' interactions at each keystroke of each QAC session separately. Analyzing the dependencies in users' sequential interactions improves our understanding of the following three questions: 1) how is a user's skipping/viewing move at the current keystroke influenced by that at the previous keystroke? 2) how to improve search engines' query suggestions at short keystrokes based on those at latter long keystrokes? and 3) facing a targeted query shown in the suggestion list, why does a user decide to continue typing rather than click the intended suggestion? We propose a probabilistic model that addresses those three questions in a unified way, and illustrate how the model determines users' final click decisions. By comparing with state-of-the-art methods, our proposed model does suggest queries that better satisfy users' intents.

Categories and Subject Descriptors:
I.2.6 [**Artificial Intelligence**]: Learning; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; J.4 [**Computer Applications**]: Social and Behavioral Sciences

General Terms: Algorithm, Experimentation, Performance

Keywords: hidden Markov model, variational Inference, query auto-completion

SIGIR'15, August 09 - 13, 2015, Santiago, Chile.

DOI: http://dx.doi.org/10.1145/2766462.2767723.

1. INTRODUCTION

Query auto-completion (QAC) has been widely used in modern search engines to reduce users' effort to submit a query by predicting the users' intended queries. The QAC engine generally offers a list of suggested queries that start with a user's input as a prefix, and the list of suggestions is changed to match the updated input after the user types each character. Let us suppose that a user is going to submit a query q to the search engine, and the user types the prefix of the query q of length i as $q[1..i]$ sequentially. The QAC engine will return the corresponding suggestion list after the user types each character, until the user clicks the suggestion q from the list or presses return, ending the interaction with the QAC engine. Usually, even for submitting the same query q, different users may have different interactions with the QAC engine, which are shown from their different sequential behaviors. For example, user u_a chooses the suggestion q at position 5 after typing 3 characters, while user u_b chooses the suggestion q at position 1 after typing 5 characters. In order to better improve users' search experience, it becomes increasingly important to analyze users' sequential behavior with the QAC engine, to understand users' real preferences and then improve the performance of QAC.

Recently, many studies have been proposed to address the QAC problem in different perspectives, including designing more efficient indexes and algorithms [2, 20, 11], leveraging context in long term and short term query history [1], investigating the time-sensitive aspect of QAC [19, 21], learning to combine more personalized signals [18], etc. Despite of those numerous works on QAC, most of them only utilize the information of submitted queries and associated prefixes, thus lose details of how users' interact with the QAC engine, such as the suggested query lists of each prefix before query submission, users' query typing speed, and so on. Recently, a high-resolution QAC dataset was collected from PC (personal computer) and mobile phones [15], where each keystroke of users and clicks were recorded. A two-dimensional click model was trained on this high-resolution QAC dataset, revealing users' behaviors such as horizontal skipping bias and vertical position bias. However, this work assumed that users' behaviors at different keystrokes are independent in order to simplify the model estimation, which results in information loss.

Our work, on the other hand, attempts to capture three types of relationship between users' behaviors at different keystrokes that are ignored or failed to model until now: 1) **State transitions between skipping and viewing.** The study on high-resolution query log data revealed that a user may choose to either view or skip the suggestion list at each keystroke in a QAC session. It already

explored how users' interactions with QAC engine at the current keystroke, such as typing speed and whether the end of current prefix is at word boundary, influence users' decisions on skipping or viewing. However, besides those factors, we believe that such decisions should also be influenced by their decisions on skipping or viewing at the previous keystroke. For instance, imagine a user u has 5 sequential skipping moves in one QAC session and 2 sequential skipping moves in another QAC session, the chance becomes higher for the same user to stop and view the suggestion list at the current keystroke after 5 sequential skipping moves. On the other hands, if the same user has already viewed too many keystrokes continuously but finds no intended query, it becomes more likely that he/she may skip the next one; 2) **Users' real preference of suggestions**. For each keystroke, the associated users' real preference is hard to be detected from the current suggested query list alone. On the other hand, we need to utilize the rankings of suggested query lists of latter keystrokes together with users' final click choices to re-rank the suggested queries in the list of the current keystroke. Intuitively, a clicked query, i.e. the user's intended query, should get a higher rank not only at the keystroke he/she makes the click, but also at previous keystrokes where this query appears, despite that it is not clicked at that time; and 3) **User-specific cost between position clicking and typing**. Some users prefer typing than viewing and clicking, while others don't. Consequently, users' click choices are not only affected by their intent, but also by the position where the intended query is shown and their preference of clicking that position over typing the remaining keystrokes. For instance, a user that prefers clicking will probably click an intended query the first time it is shown to him/her, despite that it may be shown in a low position; while another user focuses on typing his/her intended query despite that the query already appears in the suggestion list, until it is ranked at the top position, or even worse, he/she will type the entire query manually without any intent to click the suggestions.

To model these three aspects, we propose a probabilistic model, which is a combination of three parts that address each separately. The hidden Markov model part takes the skipping and viewing choices as two different states, and assumes the transition between keystrokes is influenced by users' interactions with the QAC engine at that keystroke. The logistic regression part weighs a set of our designed user-specific relevance features that imply users' own preference on each prefix-query pair, which is expected to capture users' real preference. The Dirichlet prior part estimates the ratio between position-biased clicking and typing costs. Those three parts together determine the probability that a user clicks a certain suggested query located at a certain position of the suggested query list of a certain keystroke in a QAC session. We develop a mean-field variational inference algorithm to learn the parameters that optimize the data likelihood.

We evaluate our method on high-resolution QAC logs collected from a commercial search engine on both PCs and mobile phones. We compare the performance of our model with some alternative probabilistic models and state-of-the-art QAC algorithms. Experimental results show that the proposed method can predict queries that better satisfy users' search intent. Moreover, the learned model provides us insights into the relationship between users' behaviors at different keystrokes.

In a nutshell, the major contributions of this paper include:

1. We explore the relationship between users' behaviors at different keystrokes in each QAC session, which existing works failed to model;
2. We propose a new probabilistic model that combines three parts to model each aspect of the relationship between users' behaviors at different keystrokes, and make them together effectively predict users' click actions on high-resolution QAC logs; and
3. We propose a generic model whose parameters are shared by all users, instead of using user-specific parameters, which sharply reduces the number of parameters to learn, and is suitable for new users.

The rest of the paper is organized as follows. We first introduce related work in Section 2, after that we formally define our problem and introduce the proposed model in Section 3. In Section 4, we develop a fast mean-field variational inference algorithm to optimize the solution. We then describe and report the experimental evaluations in Section 5, and finally present our conclusions and future work in Section 6.

2. RELATED WORK

Query Auto-Completion (QAC). The main objective of QAC is to predict users' intended queries and assist them formulate a query while typing. The most popular QAC algorithm is to suggest completions according to their past popularity. Generally, a popularity score is assigned to each query based on the frequency of the query in the query log from which the query database was built. This simple QAC algorithm is called MostPopularCompletion (MPC), which can be regarded as an approximate maximum likelihood estimator [1].

Several QAC methods [1, 19, 18, 21] were proposed to extend MPC from various aspects. Bar-Yossef and Kraus [1] introduced the context-sensitive QAC method by treating users' recent queries as context and taking into account the similarity of QAC candidates with this context for ranking. But there is no consensus of how to optimally train the relevance model. Shokouhi [18] employed learning-based strategy to incorporate several global and personal features into the QAC model. However, these methods only exploit the final submitted query or simulate the prefixes of the clicked query, which do not investigate the users' interactions with the QAC engine.

In addition the above models, there are several studies addressing different aspects of QAC. For example, [19, 21] focused on the time-sensitive aspect of QAC. Other methods studied the space efficiency of index for QAC [2, 11]. Duan and Hsu [7] addressed the problem of suggesting query completions when the prefix is misspelled. Kharitonov et al. [13] proposed two new metrics for offline QAC evaluation, and [12] investigated user reformation behavior for QAC.

The QAC is a complex process where a user goes through a series of interactions with the QAC engine before clicking on a suggestion. As can be seen from the related work, little attention has been paid to understand the interactions with the QAC engine. Until recently, Li et al. [15] created a two-dimensional click model to combine users' behaviors with the existing learning-based QAC model. The study assumed users' behaviors at different keystrokes, even for the consecutive two keystrokes, are independent in order to simplify the model estimation, which results in information lose. In this paper, we attempt to directly model and leverage the relationship between users' behaviors, so as to improve the performance of QAC.

Click Models. This work is related to click models. In the field of document retrieval, the main purpose for modeling users' clicks is to infer the intrinsic relevance between the query and document by explaining the positional bias. The position bias assumption was first introduced by Granka et al. [9], stating that a document on higher rank tends to attract more clicks. Richardson et al. [17]

attempted to model the true relevance of documents by imposing a multiplicative factor. Later examination hypothesis is formalized in [5], with a key assumption (Cascade Assumption) that a user will click on a document if and only if that document has been examined and it is relevant to the query. In addition, several extensions were proposed, such as the User Browsing Model (UBM) [8], the Bayesian Browsing Model [16], the General Click Model [23] and the Dynamic Bayesian Network model (DBN) [4]. Despite the abundance of click models, these existing click models cannot be directly applied to QAC without considerable modification. The click model most similar to our work is [22], which models users' clicks on a series of queries in a session. However, because of the main difference between QAC and document retrieval, our model is very different from [22].

3. PROBLEM DEFINITION

In this section, we first introduce the concept of the high-resolution QAC log, and then propose appropriate models to predict how likely a user will click a certain query at a certain location in a QAC session.

3.1 A High-Resolution QAC Log

Traditionally, the search query log only includes the submitted query and its associated search results, while it does not contain the sequential keystrokes (prefixes) users typed in the search box, as well as their corresponding QAC suggestions. In order to better analyze and understand real users' behaviors, a high-resolution QAC log is introduced and analyzed in [15], which records users' interactions with a QAC engine at each keystroke and associated system respond in an entire QAC process. For each submitted query, there is only one record in a traditional search query log. However, in the high-resolution QAC log, each submitted query is associated with a **QAC session**, which is defined to begin with the first keystroke a user typed in the search box towards the final submitted query. The recorded information in each QAC session includes each keystroke a user has entered, the timestamp of a keystroke, the corresponding top 10 suggested queries to a prefix, the anonymous user ID, and the final clicked query.

Let us take a toy example to briefly introduce how a user interacts with a QAC engine and makes the final click in an entire QAC session. As shown in Figure 1, a **QAC session** contains S keystrokes and each keystroke has a suggested query list of length D.[1] A QAC session ends at the keystroke where the user clicks a query in the suggested query list, or when the prefix at that keystroke is exactly the query the user enters into the search engine. Among the $S \times D$ *slot*s in each QAC session, where each slot q_{ij} is indexed by the i-th position at the j-th keystroke, a user clicks at most one of them, although the user intended query may appear in many slots. Since users' clicked queries are usually their intended queries, appropriate modeling of users' click actions can be a good solution of the QAC problem. The ideal QAC engine should be able to rank the user intended query higher with less keystrokes or short prefixes. In this work, we leverage such a QAC log data to get a better understanding of user's sequential behavior in the QAC process.

3.2 Assumptions on QAC User Behavior

We view the QAC problem, which predicts a user's intended query, as the problem of predicting the query the user will click. Unlike existing works on query auto-completion, which paid no attention to those non-clicked suggested queries at the keystrokes before users' make clicks, or failed to straightforwardly analyze and reveal the difference between click cases and non-click cases, our paper proposes a model which predicts the most likely slot a user will click in each QAC session by capturing the relationship between users' behaviors at different keystrokes.

To predict how likely a user will click a certain slot, there are mainly three issues we need to solve:

- Whether the user has viewed the slot;
- Whether the query shown on the slot satisfies the user's intent; and
- Whether the user is willing to click the slot.

We use Figure 2 to illustrate how the above three issues together determine users' click choices among all potential slots. Figure 2 shows a QAC session which contains S(=4) columns/keystrokes, and each keystroke contains D(=4) positions, thus makes a total number of 16 potential slots to click. Among those 16 potential slots, the queries in 12 of them do not satisfy the user's intent, thus the user will not click it anyway. On the other hand, the user's intended query appears in the other four slots, where each keystroke contains one appearance. The user viewed the suggested query lists at both the first and second keystrokes, however, the intended query is ranked at a relatively low position, which the user failed to pay attention to, or thought this position costs him/her too much effort to click, thus he/she didn't click it. At the third keystroke, although the intended query is ranked at the top position, the user didn't view this suggestion, thus he/she missed the information and failed to click. Finally, at the last keystroke, the user viewed it, found his/her intended query lies at the top position, and then he/she clicked that query.

3.3 Modeling Clicks in QAC

In the following, we show how to better address the above issues by taking the relationship between users' behaviors at different keystrokes into account. We also introduce some analysis conducted on the real-world high-resolution QAC log, to show why our proposed ideas are reasonable.

Skipping or viewing a keystroke? Existing works [2, 1] mostly failed to address the first (and the third) issue since they lack the information of users' interactions with a QAC engine before making a click, i.e., the high-resolution QAC log. Generally, it is assumed that a user makes a decision to either viewing or skipping at each keystroke, which depends only upon the user's interaction with the QAC engine at that keystroke. Based on its definition that "skipping behavior happens when the final clicked query is ranked within top 3 in the suggestion list of any of the prefixes except the final prefix" as introduced in [15], we count the sequential appearance of skipping and viewing states statistically on a real-world high-resolution QAC log collected from a commercial search engine, and find that 78.4% of the states are followed the same type of states. Thus the transitions between skipping and viewing states are not random, and the error of inferring state type based only upon user behavior at that keystroke can be significant. Thus, our paper, on the other hand, assumes that the user's decision will also be influenced by his/her decision at the previous keystroke besides his/her interaction. As shown in Figure 1, we use a hidden Markov model to capture such influence, and take viewing and skipping as $K = 2$ states. We assume that T state transition rules exist in the observed QAC logs, where each rule is actually a probability matrix corresponding to a certain type of viewing/skipping behavior. Users' interactions with the QAC engine, such as typing speed and reaching word boundary, at the current keystroke influence their choice of transition rules at this keystroke. For instance, for a fast

[1]We experiment with real-world QAC logs where $D = 10$.

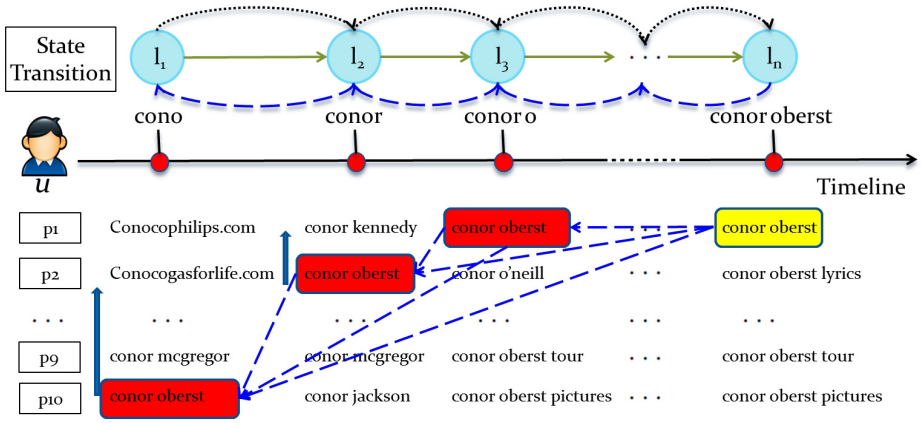


Figure 1: A Toy Example of a QAC Session in High-resolution QAC logs. Yellow tag highlights the query a user finally clicks, red tag highlights the user's intended query he/she doesn't click. Black dot line represents the dependency between users' skipping/viewing states captured by Markov process, and blue line denotes the influence of suggested query lists of latter keystrokes together with users' final click choices to the raise of the ranking of intended queries in the list of the current keystroke.

typing user, the probability of transition from skipping to skipping can be very small. When a user reaches a word boundary, he is very likely to follow the transition rule where the probability of transition from skipping to viewing is significant. Assuming T user interaction patterns lie in the entire QAC log, and each user interaction pattern ω_t' corresponds to one state transition rule δ_t, at each keystroke, we analyze a user's interaction with the QAC engine and find which interaction pattern it belongs to, then choose accordingly the corresponding move transition rules. We use features introduced in [15] to describe users' interactions with the QAC engine, and denote those interaction features as X'.

How to capture users' intent? We adopt logistic regression (LR) to model how likely a query satisfies a user's intent under the current prefix. Existing works [15] already demonstrated a set of relevance features, which characterize the relevance between a certain prefix-query pair, to be effective in predicting the user's intent given certain prefixes. However, we find those features not enough due to two drawbacks: 1) those features mainly reflect the general interest of majority of users, and care little about users' personal history and preference; and 2) those features rarely imply the relationship between different prefixes, which makes them difficult to utilize users' preference of queries in the suggested lists of other keystrokes to improve the ranking at the current keystroke. As shown in Figure 1, a user's intended query, should get a higher rank at previous keystrokes where this query appears, despite that it is not clicked at that time. Actually, the real-world QAC log shows that 29.4% of users' submitted queries (this number counts redundant appearances) have been submitted more than 3 times by a user, while among those queries, only 18.4% of them has been submitted multiple times by more than 25% of users, i.e. different users favorite different query sets. Thus, besides existing features, we also employ a set of user-specific relevance features, which are designed to capture users' personal preference of queries and their corresponding relationship with keystrokes.

We summarize these search behavior features in Table 2. Our features generally originate from statistical counting of users' interactions with the QAC engine in their own historical sessions. For each user, given a certain query, we measure the number of times the same query has been clicked by that user in the past (denoted as Query Clicks). For users who have some queries daily issued, such as "facebook" or "youtube", this feature is capable of predicting

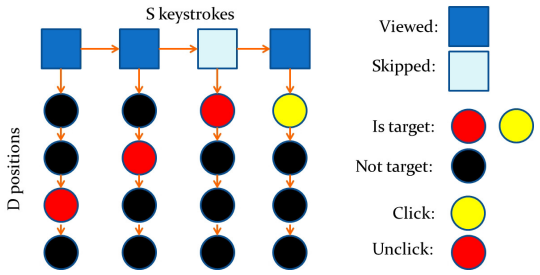


Figure 2: How Users Choose to Click Suggested Queries.

Table 1: User-Specific Relevance Features

Feature x	Description
Query Length	The average length of queries a user clicked.
Query Word Number	The average number of words in queries that a user clicked.
Query Clicks	The number of clicks a user makes on the current query.
Prefix/Query Ratio	The percentage of the appearance of *PQLR* of the current prefix-query pair in the history.

his intent at the first few keystrokes. We also measure the average length of queries the user has clicked in the past (denoted as Query Length), the average number of words in queries that a user clicked (denoted as Query Word Number). In addition, we define the ratio between the length of a prefix and that of a query as *Prefix/Query Length Ratio (PQLR)*, and calculate the distribution of the associated PQLR of queries the user clicked in the past. For each new coming query, we estimate the percentage of the appearance of the associated PQLR in the user's history (denoted as Prefix/Query Ratio). PQLR is supposed to capture a user's preference of typing when taking the query length into consideration.

Clicking a position or continue typing? In addressing the third issue, we find it very necessary to take a user's tendency of viewing and clicking a certain position or continuing typing into consid-

Table 2: Major Notations

Symbol	Description
S	The number of keystrokes in a QAC session.
D	The number of suggested queries at a keystroke.
x	The relevance features.
x'	The interaction features.
β	The bias of viewing and clicking each position.
z	Whether a query is clicked.
ω	Feature weights of the relevance features.
δ_t, ω'_t	The t-th move transition pattern and the associated user interaction pattern.

eration. On the real-world QAC log, we find that when a user's intended query (the click this user finally clicks in a QAC session) is ranked within the top 2 positions, 37.6% of them will be clicked by users. On the other hand, if this intended query is ranked out of the top 2 positions, only 13.4% of them will be clicked by users. Furthermore, such tendencies can be very different for different groups of users. For mobile users, 42.9% of intended queries will be clicked if they are ranked within the top 2 positions, otherwise, 23.0% of them will be clicked. While for PC users, 35.1% of intended queries will be clicked if they are ranked within the top 2 positions, otherwise, 11.7% of them will be clicked. Here we can clearly find that on mobile, users are more likely to make clicks when their intended queries appear on the suggestion lists, even if they are ranked at low positions, while PC users prefer typing, since typing on PC is much more convenient than on mobile.

We use a D-dimensional Dirichlet prior α to learn the relative cost ratio of viewing and clicking a certain position against typing. Such a prior allows each user m to have distinguished position bias β_m in the preference of viewing and clicking than typing. Users who prefer clicking than typing will have a higher average *clicking/typing cost ratio (CTCR)* than users who prefer typing than clicking. The gaps of CTCR between high position and low position of users who rarely click suggested queries in low positions are generally larger than those of users who tend to click their intended queries the first time they are suggested no matter how lower their positions are.

Let us consider a typical scenario where M users issue M corresponding query sequences, where for each user m, the QAC log records N_m QAC sessions. The n-th QAC session of user m contains $S_{m,n}$ keystrokes, where each keystroke contains D suggested queries. In total, each QAC session has $S \times D$ potential slots where a user can click, and we use $z_{m,n,s,d} = 1$ to denote that a user clicks the slot ranked d at the s-th keystroke, and $z_{m,n,s,d} = 0$ otherwise. We also use $x_{m,n,s,d}$ to denote the relevance feature of a prefix-query pair, where the prefix is at the s-th keystroke and the query is ranked d at that keystroke. Then given the weights ω of those relevance features, we can derive the relevance score of each suggestion as $\omega x_{m,n,s,d}$. In addition, we use $x'_{m,n,s}$ to denote a user's interaction with the QAC engine at the s-th keystroke.

Based on our proposed solution of the listed three issues and above definitions, finally, we present our generative model as follows:

- For each user m, draw a D dimensional membership vector $\beta_m \sim \text{Dirichlet}(\alpha)$.
- For each move transition pattern t, draw a move transition matrix $\delta_t \sim \text{Dirichlet}(\alpha')$, where each pattern is associated with a user interaction pattern ω'_t.

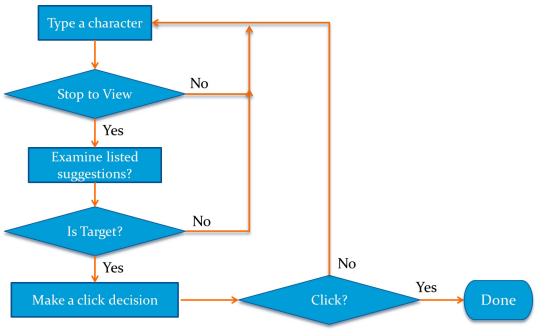


Figure 3: RBCM Flowchart.

- For the n-th query issued by user m, and for the prefix at step s,
  - Draw the user's move transition pattern membership $\pi_{m,n,s} \sim \text{Multinomial}(\theta)$, where θ is the prior distribution of move transition pattern membership.
  - Draw the user's interaction with the QAC engine through $x'_{m,n,s} \sim \text{Gaussian}(\omega'_{\pi_{m,n,s}}, \sigma)$.
  - Draw the user's next move, which is either type or view, $Y_{m,n,s} \sim \text{Multinomial}(\delta_{\pi_{m,n,s-1},Y_{m,n,s-1}})$. If we have $Y_{m,n,s} = 1$, continue to type; otherwise, stop and view the results.
  - For each position d in the suggestion list of the current prefix, draw the user's clicked suggestion through $z_{m,n,s,d} \sim \text{Multinomial}(\text{LR}(\beta_{m,d}\omega x_{m,n,s,d}))$. If we have $z_{m,n,s,d} = 1$, select the suggestion in position d to click, then go to the $n+1$-th query; otherwise, continue to type, i.e., go to the next prefix.

Notice that the proposed model is a combination of three parts that address the listed issues respectively. We name this probabilistic model as relationship-based click model (RBCM). To better illustrate the generative process of the proposed RBCM model, we show the flowchart of user behaviors in Figure 3.

Under our RBCM model, the joint probability of the click information $Z = \{\{z_{m,n}\}_{n=1}^{N_m}\}$, the relevance features $X = \{\{x_{m,n}\}_{n=1}^{N_m}\}$, the user interaction features $X' = \{\{x'_{m,n}\}_{n=1}^{N_m}\}$ and latent variables $\{\beta_{1:M}, \pi, Y\}$ can be written as follows:

$$
\begin{aligned}
&p(Z, X, X', \beta_{1:M}, Y, \pi, \delta | \alpha, \alpha', \omega, \omega', \theta) \\
&= \prod_{m,n,s,d} P(z_{m,n,s,d} | x_{m,n,s,d}, \beta_{m,d}, \omega, Y_{m,n,s}) \\
&\quad \times \prod_{m,n,s} P(Y_{m,n,s} | Y_{m,n,s-1}, \delta, \pi_{m,n,s-1}) P(x'_{m,n,s} | \pi_{m,n,s-1}, \omega') \\
&\quad \times \prod_{m,n,s} P(\pi_{m,n,s} | \theta) \prod_t P(\delta_t | \alpha'_t) \prod_m P(\beta_m | \alpha).
\end{aligned}
$$

4. INFERENCE

In this section, we derive a mean-field variational Bayesian inference algorithm for our proposed RBCM model.

4.1 Variational Inference

Under the RBCM model, given observations of both click information $Z = \{Z_m\} = \{\{z_{m,n}\}_{n=1}^{N_m}\}$, the relevance features X, and the user interaction features X', the log-likelihood for the complete data is given by $\log P(Z, X, X' | \alpha, \alpha', \omega, \omega', \theta)$. Since

this true posterior is hard to infer directly, we turn to variational methods [3, 14], whose main idea is to posit a distribution over the latent variables with variational parameters, and find the settings of the parameters so as to make the distribution close to the true posterior in Kullback-Leibler (KL) divergence. Our paper chooses to introduce a distribution of latent variables q specified as the mean-field fully factorized family as follows:

$$q(Y, \pi, \beta_{1:M}, \delta|\rho, \phi, \gamma_{1:M}, \eta) = \prod_m q_2(\beta_m|\gamma_m) \prod_t q_2(\delta_t|\eta_t) \times \prod_m \prod_n \prod_s q_1(Y_{m,n,s}|\rho_{m,n,s}) q_1(\pi_{m,n,s}|\phi_{m,n,s})$$

where q_1 is a multinomial, q_2 is a Dirichlet, and $\{\rho, \phi, \gamma_{1:M}\}$ are the set of variational parameters. We optimize those free parameters to tight the following lower bound $\mathcal{L}'$ for our likelihood:

$$\log p(Z, X, X'|\alpha, \alpha', \omega, \omega', \theta) \geq -E_q\left[\log q(Y, \pi, \beta, \delta|\rho, \phi, \gamma_{1:M}, \eta)\right] + E_q\left[\log p(Z, X, X', \beta, Y, \pi, \delta|\alpha, \alpha', \omega, \omega', \theta)\right]. \quad (1)$$

Under a coordinate descent framework, we optimize the lower bound as in Eqn (1) against each variational latent variable and the model hyper-parameter. For variational latent variables, we have the following process

- update rules for γ's as:

$$\gamma_{m,d} = \alpha_d + \sum_n \sum_s \log\left(1 + \exp\left(z_{m,n,s,d}\omega_m x_{m,n,s,d}\right)\right);$$

- update rules for ρ's as:

$$\rho_{m,n,s,k=1} \propto \exp\left(-\sum_d z_{m,n,s,d} - b_{m,n,s,1}\right);$$

$$\rho_{m,n,s,k=2} \propto \exp\left(b_{m,n,s,2} - \sum_d \log\left(1 + \exp\left(z_{m,n,s,d} - \omega_m x_{m,n,s,d}\left[\Phi(\gamma_{m,d}) - \Phi\left(\sum_d \gamma_{m,d}\right)\right]\right)\right)\right).$$

where

$$b_{m,n,s,k} = \sum_{k'} \sum_t \phi_{m,n,s,t}\left[\Phi(\eta_{t,k',k}) - \Phi\left(\sum_{k''} \eta_{t,k',k''}\right)\right] + \sum_{k'} \sum_t \phi_{m,n,s+1,t}\left[\Phi(\eta_{t,k,k'}) - \Phi\left(\sum_{k''} \eta_{t,k.k''}\right)\right]$$

- update rules for ϕ's as:

$$\phi_{m,n,s,t} \propto \exp\left(\sum_{k,k'} \rho_{m,n,s,k}\rho_{m,n,s+1,k'}\left[\Phi(\eta_{t,k,k'}) - \Phi\left(\sum_{k''} \eta_{t,k,k''}\right)\right] - \frac{1}{2\sigma^2}\left\|x'_{m,n,s} - \omega'_t\right\|_2^2\right)$$

- update rules for η's as:

$$\eta_{t,k,k'} = \alpha'_t + \sum_m \sum_n \sum_s \phi_{m,n,s,t} \sum_{k,k'} \rho_{m,n,s,k}\rho_{m,n,s-1,k'}$$

4.2 Learning

We use a variational expectation-maximization (EM) algorithm [6] to compute the empirical Bayes estimates of the Dirichlet hyper-parameters α and α' in our RBCM model. This variational EM algorithm optimizes the lower bound as in Eqn (1) instead of the real likelihood, and iteratively approximates the posterior by fitting the variational distribution q and optimizes the corresponding bound against the parameters.

In updating α, we use a Newton-Raphson method, since the approximate maximum likelihood estimate of α doesn't have a closed form solution. The Newton-Raphson method is conducted with a gradient and Hessian as follows:

$$\frac{\partial \mathcal{L}'}{\partial \alpha_d} = N\left(\Psi\left(\sum_d \alpha_d\right) - \Psi(\alpha_d)\right) + \sum_m \left(\Psi(\gamma_{m,d}) - \Psi\left(\sum_d \gamma_{m,d}\right)\right),$$

$$\frac{\partial \mathcal{L}'}{\partial \alpha_{d_1}\alpha_{d_2}} = N\left(\mathbb{I}_{(d_1=d_2)}\Psi'(\alpha_{d_1}) - \Psi'\left(\sum_d \alpha_d\right)\right).$$

Similar update rules can be derived for α'.

On the other hand, to obtain the approximate maximum likelihood estimation of ω, we employ the stochastic gradient descent to update ω in each interaction based on the observed click data z, relevance features x, and the inferred latent variables ρ and γ. On the other hand, the approximate maximum likelihood estimation of ω' will lead to the following update rule,

$$\omega'_t = \frac{\sum_m \sum_n \sum_s \phi_{m,n,s,t} x'_{m,n,s}}{\sum_m \sum_n \sum_s \phi_{m,n,s,t}}.$$

Our variational inference algorithm, named RBCM, can be interpreted intuitively in the following way. The CTCR distribution γ of each user is determined by both the topic prior and the accuracy that the learned weights of relevance features predict users' intended queries. Users' viewing/skipping states ρ at each keystroke is determined by the influence from users' states at the previous keystroke and that from users' states at the next keystroke. The state transition at each keystroke is determined by the transition prior, users' interaction patterns at that keystroke, and users' states at that keystroke and the keystroke before that. The probability of the interaction of a user m at the s-th keystroke in the n-th QAC session belonging to interaction pattern k is jointly determined by the state transition between the current and the next keystroke and the user's interaction at the current keystroke.

In our mean-field variation inference algorithm, the computational cost of inferring variational variables is $O(N*\bar{S}*T+M*D)$, where $\bar{S}$ is the average number of keystrokes in a QAC session, and $N = \sum_m N_m$ is the total number of QAC sessions in the entire high-resolution QAC log. The computational cost of the estimation of Dirichlet hyper-parameters is $O(M * D)$. The computational cost of the estimation of weights of relevance features is $O(N * \bar{S} * D)$, while the cost of the estimation of user interaction patterns is $O(N * \bar{S} * T)$. Thus the total computational cost of our algorithm is $O(N * \bar{S} * (T + D))$. Since T and D are both small constants, we can view the computational cost as linear in the total number of keystrokes in all sessions in the QAC log.

5. EXPERIMENTS

We evaluated our RBCM model on real-world data sets, and compared the performance with the following baselines: three alternative probabilistic models that only use two parts of the proposed model and four state-of-the-art QAC algorithms:

Alternative-A: This model does not use the hidden Markov model to capture the state transition of skipping/viewing moves. The state of skipping/viewing is determined by users' interactions with the QAC engine only.

Alternative-B: This model avoids using user-dependent features in the logistic regression part to capture users' real preference of suggested queries. It only utilizes user-independent features in the logistic regression part.

Alternative-C: This model avoids using a Dirichlet prior to

model users' CTCR. Instead, it assumes users have no preference in clicking different positions as well as typing, i.e., the probabilities of clicking any positions and typing are all equal, and user's clicked suggestion is drawn via $z_{m,n,s,d} \sim \text{Multinomial}(\text{LR}(\omega x_{m,n,s,d}))$.

MPC [1, 18]: This method, named MostPopularCompletion, is a widely used baseline in Query Auto-Completion, and employed as one main feature in many QAC engines.

UBM [8]: This User Browsing Model proposes a number of assumptions on user browsing behavior that allows the estimation of the probability of observing a document. It depends on statistical counting of prefix-query pairs, thus unable to predict unseen prefix-query pairs.

BSS [10]: This Bayesian Sequential State model uses a probabilistic graphical model to characterize the document content and dependencies among the sequential click events within a query with a set of descriptive features. This is a content-aware model, which is able to predict unobserved prefix-query pairs.

TDCM [15]: This is a two-dimensional click model which emphasizes two kinds of user behaviors. It consists of a horizontal model, which explains the skipping behavior, and a vertical model that depicts the vertical examination behavior.

5.1 Real-world Data

We conducted extensive experiments on two real-world high-resolution QAC logs that collected from a commercial search engine. The first data set, which we name *LargeQAC*, contains high-resolution QAC logs from May 2014 to July 2014. The collection consists of a sample of 7.4 million QAC session from about 40,000 users over a 3-month period. We randomly selected a subset of active users who submitted over 500 QAC sessions during this period, and collected their corresponding search activities, including the anonymized user ID, query string, timestamp, and the clicked URL. As a result, we collected 3,954 users with 2.6 million queries, and their activities span from 22 days to 3 months. According to different platforms (PC or mobile phones), we split the entire dataset into two subsets. One is PC, which contains 1.6 million QAC sessions, while the other is mobile phones, which contains 1.0 million QAC sessions.

The second data set is also collected from a commercial search engine. We name this data *SmallQAC* to distinguish it from the previous one. This data set is constituted of random sampled high-resolution QAC logs dating from Nov 2013 to Jan 2014. The log contains 125 thousand QAC sessions from PCs. Since existing QAC algorithms utilizing high-resolution QAC logs have already shown rich results on the QAC log, we utilize both data sets to evaluate our proposed model and compare with the state-of-the-art methods in the following section.

5.2 Experimental Results

Model Fitness. This section evaluates the fitness of our proposed model on real-world data, and compares our model with probabilistic model based methods. We split the data based on the time information: the QAC sessions occurred in the first 90% of the time period are used as the training data, while the remaining 10% used as the test data. Table 3 shows the log predictive likelihood on sessions falling in the final 10% of the total time of QAC log data. According to Table 3, RBCM fits the real-world data better than the three alternative probabilistic models and TDCM. This illustrates that in the proposed RBCM model, all three parts play an important role in capturing the relationship between users' behaviors at different keystrokes. Alter-A performs the worst among all three alternative probabilistic models, which shows the importance of using the Markov process to model the state transition between skipping and viewing. Alter-B performs the best among three alternative probabilistic models, which shows that user-specific features do not fully utilize the relationship between users' behaviors. The reason may be that, even when the relevance features and their associated weights fail to reflect users' real preference of suggested queries, the other two parts of the proposed model can reduce the harm of those mispredictions by inferring reasonable skipping/viewing states and user-specific CTCR. TDCM performs the worst, since it fails to utilize the relationship between users' behaviors from any aspects.

Query Auto-Completion. To evaluate the effectiveness of the proposed model in suggesting users intended queries in each QAC session, we compare the proposed model with both alternative probabilistic models and state-of-the-art QAC algorithms. All compared methods re-rank the suggested queries at each keystroke and compete to rank the intended query as high as possible. In the proposed RBCM model, we use the relevance model part $\text{LR}(\beta\omega x)$ to re-rank the suggested queries. We employ the Mean Reciprocal Rank (MRR) as the relevance measurement, which is a widely used evaluation metric in measuring QAC performance [1, 18, 15],

$$MRR = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{\text{rank}_q},$$

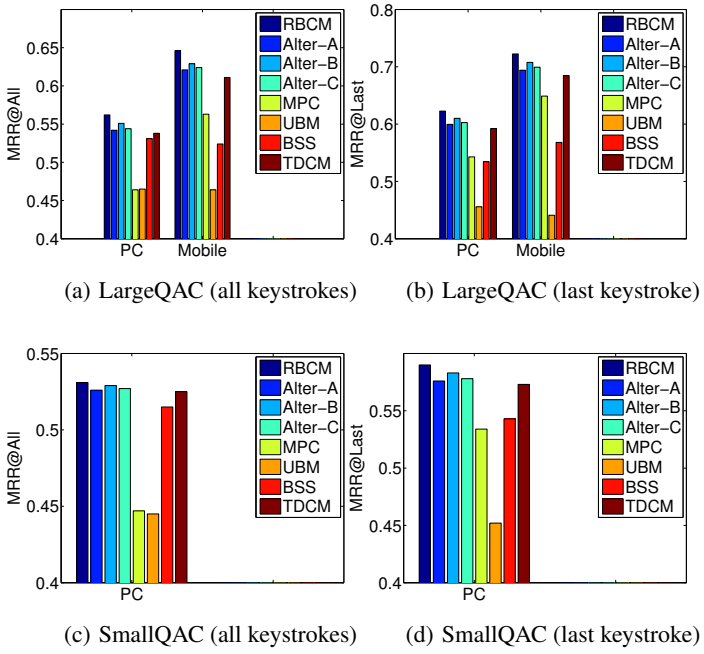
where Q is the set of queries a user finally submitted, and rank_q denotes the rank of the query q in the suggested query list.

Notice that among the suggested query lists of all keystrokes, those lists that do not contain users' finally submitted queries are removed from our experimental analysis. Since our experiments are conducted on high-resolution QAC data, we report both the average MRR score of all keystrokes, and the average MRR of the last keystroke only, since this is the keystroke where the user's click occurs. Notice that existing works which didn't make use of high-resolution QAC logs usually used the MRR of the last keystroke to measure their performance. In the following experiments, the whole dataset is divided evenly into a training set and a test set for different settings.

Figure 4 compares the proposed model with alternative probabilistic models and state-of-the-art QAC algorithms by MRR. We can observe that RBCM performs the best among all compared approaches, and outperforms existing QAC algorithms by over 6%, for all the data sets and different settings. The advantage of RBCM over Alter-A illustrates the necessity of modeling the transition between skipping and viewing. The learned transition rules work for both existing users and new coming users, thus can provide better query suggestions for new coming users than TDCM, which models no relationship between users' behaviors at different keystrokes. In addition, the three alternative probabilistic models generally perform better than existing QAC algorithms. Such phenomenon demonstrates the effectiveness of making use of the relationship between users' interactions in different keystrokes in solving the QAC task. Essentially, appropriate modeling of such relationships together makes the proposed model much better than those alternative baselines. In specific, the advantage of Alter-B over TDCM illustrates that for those unseen prefix/query pairs, the proposed model can also achieve a sound performance, which attributes to the skipping/viewing transition and the user-specific preference of position clicking and typing captured by the proposed model. Besides our proposed model and its alternatives, TDCM performs better than the rest of existing QAC algorithms, which we attribute to

Table 3: Log Predictive Likelihood on Real-world Data

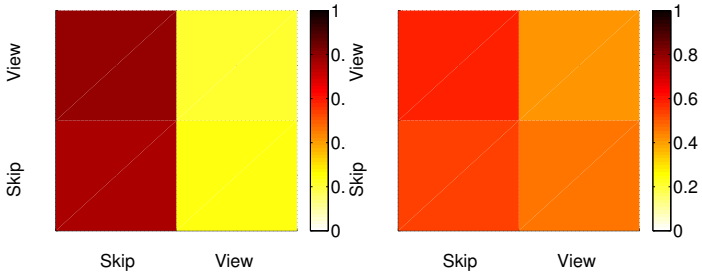
Model/Data set	Platform	RBCM	Alter-A	Alter-B	Alter-C	TDCM
LargeQAC	PC	-185.37	-192.83	-189.43	-191.84	-208.65
LargeQAC	Mobile	-177.95	-187.64	-184.91	-185.50	-195.04
SmallQAC	PC	-206.32	-218.14	-214.98	-216.69	-234.83



(a) LargeQAC (all keystrokes) (b) LargeQAC (last keystroke)

(c) SmallQAC (all keystrokes) (d) SmallQAC (last keystroke)

Figure 4: Performance Comparison of QAC Methods.



(a) Skipping User (b) Viewing User

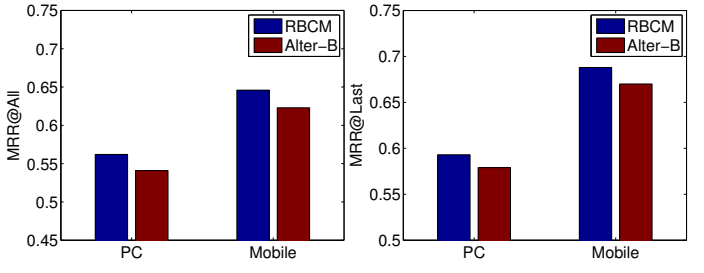
Figure 5: State Transition of Skipping/Viewing.

the usage of high-resolution QAC logs. BBS outperforms UBM since it adopts the content-aware relevance model. MPC performs the worse, since it pays little attention to users' behaviors in QAC logs. By comparing with the performance using all the keystrokes and last keystroke only, we find that the advantages of the proposed model are ever more significant when measured by MRR@All. It indicates that the proposed model can recommend user intended queries higher when only a few keystrokes have been typed.

State Transition of Skipping/Viewing. Based on the state transition matrices and the corresponding user interaction patterns that learned by the proposed RBCM model from real world QAC logs, we provide a detailed analysis on the difference of transition rules between skipping/viewing and the associated user interaction patterns. Among all learned state transition rules, we pick two of them which differ the most in probability. Figure 5 shows the state transition rules and their corresponding user interaction patterns, where each block is the transition probability from the previous state (vertical) to the current state (horizontal). Intuitively, we find Figure 5(a) represents *skipping user*s, i.e., users who prefer skipping than viewing, as these users have a much higher probability to skip at the current keystroke no matter the previous state is skipping or viewing, while Figure 5(b) represents *viewing user*s, i.e., users who prefer viewing than skipping, because these users are more likely to switch to the viewing state from previous skipping or viewing status. Notice that *skipping* and *viewing user*s only refer to the tendency of users' at each keystroke. A user who always skips suggested query lists will behave like *skipping user* consistently, while a user who has no habit in querying may alternatively switch between *skipping* and *viewing user*s from time to time.

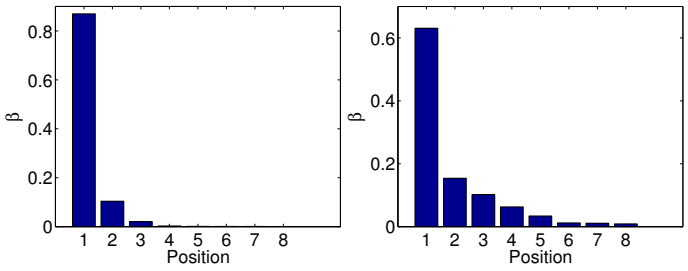
From Figure 5, we find that no matter what the state of the current keystroke is, *skipping user*s are more likely to skip the suggested query list of next keystroke than *viewing user*s. On the contrary, *viewing user*s are more likely to view the suggested query list of next keystroke than *skipping user*s under all circumstances. We also find that no matter which type a user belongs to, if he/she already viewed the suggested query list of the current keystroke, he/she will be more likely to skip the next keystroke than that under the situation where he/she skipped the current keystroke. Moreover, the corresponding user interaction patterns of different state transition rules appear quite different. *Skipping user*s generally have faster typing speed, and come across less word boundaries, and enter more navigational queries.

Users' Real Preference of Suggested Queries. We use this series of experiments to discuss how our designed user-specific features enable the proposed model to understand users' real preference of suggested queries. We compare the proposed model with Alter-B on the QAC task measured by MRR, so as to illustrate the importance of using user-specific features. From Figure 6, we find that the proposed model performs better than Alter-B in recommending users' queries that satisfy their intent. In addition, we list a subset of learned weights of those designed user-specific features in Figure 7. From here, we can find that the history of queries that a user clicked plays a very important role in predicting the future queries he/she will click, especially when certain prefixes are given. Meanwhile, the length of the queries a user used to click is a significant signal for learning users' real preference of queries in the suggested query lists. The reason can be that such a signal implies users' clicking habit from some aspects. A skipping user who always clicks long queries will probably ignore his/her intended queries shown to him/her, before he/she enters enough number of keystrokes. Under such situation, the signal of query length will be capable of capturing users' real preference at keystrokes with short prefixes, which enables the proposed model to rank the intended queries at higher positions than those shorter queries. On the other hand, when a user has no preference in clicking long queries,



(a) LargeQAC (all keystrokes) (b) LargeQAC (last keystroke)

Figure 6: Comparison of RBCM with Alter-B.



(a) Typing User (b) Clicking User

Figure 8: User-Specific Clicking/Typing Cost Ratio.

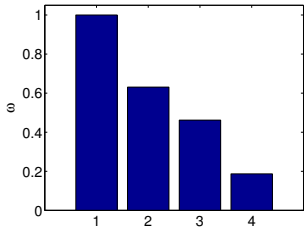


Figure 7: Weights of Relevance Features Learned by RBCM. Indice of selected user-specific relevance features: 1-Query Clicks, 2-Query Length, 3-Prefix/Query Ratio, 4-Query Word Number. The values of weights are scaled to the range of $[0, 1]$ to clarify the comparison of relative importance of different features.

this signal will not take effect, and the proposed model will recommend popular queries according to the frequency of the occurrence of prefix-query pairs. Such suggested queries are usually not that much longer than the given prefixes.

User-specific Cost Between Position Clicking and Typing. Based on the latent variable γ learned by the proposed model, we analyze the specific cost balance between position clicking and typing of different users. We select two subsets of users: one named *typing user*, which is formed by the top 20% users with the highest average click positions, and another, *clicking user*, is formed by the bottom 20% users with the lowest average click positions. We calculate the averaged *CTCR* of both two subsets of users, separately, and plot the results in Figure 8. From here, we find that the learned CTCR γ's of *typing user*s and *clicking user*s have very different distributions. Although users from both subsets are most likely to click the top query of a suggested query list, *clicking user*s also occasionally click queries located at the middle positions of a suggested query list, while *typing user*s rarely click those positions.

Furthermore, we try to distinguish the difference between *typing user*&*clicking user* and *viewing user*&*skipping user*. We select the top 20% users with the largest percentage of viewing states , and the bottom 20% users with the smallest percentage of skipping states. We compared the selected subsets of *typing user*s with *skipping user*s, and the selected subset of *clicking user*s with *viewing user*s. Table 4 shows the overlaps of the two pairs of compared subsets. According to Table 4, we find that there exists overlap between *typing user*s and *skipping user*s, *clicking user*s and *viewing user*s. For users sharing the same tendency of skipping/viewing

Table 4: Overlap between *typing user-skipping user* pair and *clicking user-viewing user* pair

Overlap	Percentage
typing user∩*skipping user*	34.2%
clicking user∩*viewing user*	22.7%

state choices, the difference of users in choosing clicking and typing becomes smaller. Skipping and typing are two reasons that a user does not click his/her intended query when it appears. Skipping users are more likely to click upper position queries, but this is not always true. A user who owns a high typing speed may not only prefer skipping than viewing but also prefer typing than clicking. However, he/she may also click the intended query the first time he/she views it.

Case Study of Query Auto-Completion. Now we show a few examples that illustrate how RBCM recommends users queries that better satisfy user intent by capturing the relationship between users' behaviors at different keystrokes. Figure 9 shows a QAC session where a user finally submitted "star wars", which is the user's intended query in this session. From Figure 9, we can find that the proposed model generally ranks users' intended queries higher than TDCM, especially at keystrokes with shorter prefixes. For example, with the prefix "st", RBCM ranks the intended query at the position 3 while TDCM ranks it at the position 10. The reason is that the proposed model utilizes the user's preference of the clicked queries at the last keystroke to improve its ranking at shorter keystrokes in the future by modeling the relationship between users' behaviors at different keystrokes. Our designed relevance features show that the query "star wars" has been issued many times by this user. Thus, although for the entire user collection, the generally frequencies of the appearances of "star wars" are relatively low given short prefixes, such as "st" or "sta", the proposed model ranks this query at the higher position for the specific user than other users who rarely search "star wars". We also notice that queries of similar intent, such as "star wars the old republic" and "star wars episode 7" are also ranked higher by our proposed model than by TDCM, which emphasizes that our model can better capture users' personal interests. Actually, through the analysis of the historical log of this particular user, we can find that he/she is a science fiction fan, which explains why our model also ranks "star trek" higher than TDCM. Thus we can conclude that appropriate modeling of such relationships is critical for predicting users' intended queries given short prefixes.

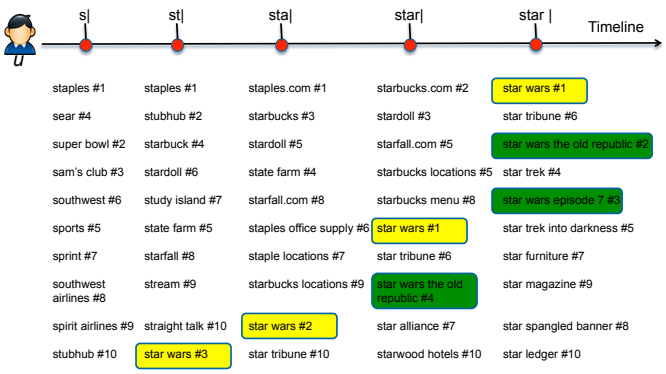


Figure 9: Case Study: The position of list queries from top to down shows the ranking of suggested queries predicted by TDCM, while the number tagged with # behind each query shows its ranking given by the proposed model. The yellow box highlights the user's intended query, and the green box highlights queries satisfy similar user intent. Notice that "|" is the cursor.

6. CONCLUSION AND FUTURE WORK

In this paper, we have presented a probabilistic model to solve the query auto-completion (QAC) task by capturing the relationship between users' behaviors at different keystrokes in high-resolution QAC logs. The proposed model integrates three parts, each addressing a single aspect of the above relationship, and illustrates how the three parts together determine users' final click decisions. We have applied the proposed model to predict users' intended queries on real world high-resolution QAC logs collected from a commercial search engine, and compare it with several alternative approaches. Experimental results show that the improvements of our proposed model are consistent, and our model achieves the best performance. In future work, it would be interesting to consider more complex relationships between users' behaviors at different keystrokes. For instance, we could increase the number of states from simply skipping or viewing to viewing to a certain position d, which enables a more precise modeling of users' behaviors in QAC.

7. ACKNOWLEDGMENT

This work is supported in part by NSF grant IIS-1116886.

8. REFERENCES

[1] Z. Bar-Yossef and N. Kraus. Context-sensitive query auto-completion. In *Proceedings of the 20th international conference on World wide web*, pages 107–116. ACM, 2011.

[2] H. Bast and I. Weber. Type less, find more: fast autocompletion search with a succinct index. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 364–371. ACM, 2006.

[3] D. Blei and M. Jordan. Variational inference for dirichlet process mixtures. In *Bayesian Analysis*, volume 1, pages 121–144, 2005.

[4] O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *Proceedings of the 18th international conference on World wide web*, pages 1–10. ACM, 2009.

[5] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 87–94. ACM, 2008.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[7] H. Duan and B.-J. P. Hsu. Online spelling correction for query completion. In *Proceedings of the 20th international conference on World wide web*, pages 117–126. ACM, 2011.

[8] G. E. Dupret and B. Piwowarski. A user browsing model to predict search engine click data from past observations. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 331–338. ACM, 2008.

[9] L. A. Granka, T. Joachims, and G. Gay. Eye-tracking analysis of user behavior in www search. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 478–479. ACM, 2004.

[10] A. D. H. Wang, C. Zhai and Y. Chang. Content-aware click modeling. In *WWW*, 2013.

[11] B.-J. P. Hsu and G. Ottaviano. Space-efficient data structures for top-k completion. In *Proceedings of the 22nd international conference on World Wide Web*, pages 583–594. International World Wide Web Conferences Steering Committee, 2013.

[12] J.-Y. Jiang, Y.-Y. Ke, P.-Y. Chien, and P.-J. Cheng. Learning user reformulation behavior for query auto-completion. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 445–454. ACM, 2014.

[13] E. Kharitonov, C. Macdonald, P. Serdyukov, and I. Ounis. User model-based metrics for offline query suggestion evaluation. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 633–642. ACM, 2013.

[14] L. Li, H. Deng, A. Dong, Y. Chang, and H. Zha. Identifying and labeling search tasks via query-based hawkes processes. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 731–740, New York, NY, USA, 2014. ACM.

[15] Y. Li, A. Dong, H. Wang, H. Deng, Y. Chang, and C. Zhai. A two-dimensional click model for query auto-completion. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '14, pages 455–464, New York, NY, USA, 2014. ACM.

[16] C. Liu, F. Guo, and C. Faloutsos. Bayesian browsing model: Exact inference of document relevance from petabyte-scale data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(4):19, 2010.

[17] M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web*, pages 521–530. ACM, 2007.

[18] M. Shokouhi. Learning to personalize query auto-completion. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 103–112. ACM, 2013.

[19] M. Shokouhi and K. Radinsky. Time-sensitive query auto-completion. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 601–610. ACM, 2012.

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

[21] S. Whiting and J. M. Jose. Recent and robust query auto-completion. In *Proceedings of the 23rd international conference on World wide web*, pages 971–982. International World Wide Web Conferences, 2014.

[22] Y. Zhang, W. Chen, D. Wang, and Q. Yang. User-click modeling for understanding and predicting search-behavior. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1388–1396. ACM, 2011.

[23] Z. A. Zhu, W. Chen, T. Minka, C. Zhu, and Z. Chen. A novel click model and its applications to online advertising. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 321–330. ACM, 2010.