A Two-Dimensional Click Model for Query Auto-Completion

Yanen Li[1], Anlei Dong[2], Hongning Wang[1], Hongbo Deng[2], Yi Chang[2], ChengXiang Zhai[1]

[1]Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
{yanenli2, wang296, czhai}@illinois.edu

[2]Yahoo Labs, Sunnyvale, CA 94089, USA
{anlei, hbdeng, yichang}@yahoo-inc.com

ABSTRACT

Query auto-completion (QAC) facilitates faster user query input by predicting users' intended queries. Most QAC algorithms take a learning-based approach to incorporate various signals for query relevance prediction. However, such models are trained on simulated user inputs from query log data. The lack of real user interaction data in the QAC process prevents them from further improving the QAC performance.

In this work, for the first time we have collected a high-resolution QAC query log that records every keystroke in a QAC session. Based on this data, we discover two types of user behavior, namely the horizontal skipping bias and vertical position bias which are crucial for relevance prediction in QAC. In order to better explain them, we propose a novel two-dimensional click model for modeling the QAC process with emphasis on these behaviors.

Extensive experiments on our QAC data set from both PC and mobile devices demonstrate that our proposed model can accurately explain the users' behaviors in interacting with a QAC system, and the resulting relevance model significantly improves the QAC performance over existing click models. Furthermore, the learned knowledge about the skipping behavior can be effectively incorporated into existing learning-based QAC models to further improve their performance.

Categories and Subject Descriptors

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

General Terms

Algorithms, Performance, Experimentation

Keywords

Query Auto-Completion, Two-Dimensional Click Model

SIGIR'14, July. 6-11, 2014, Golden Coast, QLD, Australia
Copyright ©2014 ACM 978-1-4503-2257-7/14/07 ...$15.00.

1. INTRODUCTION

Query auto-completion (QAC) is one of the most important components of a modern web search engine which facilitates faster user query input by predicting the users' intended queries. It is offered by most of the search engines, e-commerce portals and major browsers. With the prevalence of mobile devices, it becomes more critical because typing takes more effort in mobile devices than in PCs. Previous studies addressed the QAC problem in different perspectives, ranging from designing more efficient indexes and algorithms [5, 11, 22, 12], leveraging context in long term and short term query history [3], learning to combine more personalized signals such as gender, age and location [20], suggesting queries from a mis-spelled prefix [8].

The query auto-completion process starts when a user enters the first character into the search box. After that, she goes through a series of interactions with the QAC engine until she clicks on an intended query. Such interactions include examining the suggested results, continuing to type, and clicking on a query in the list. Although previous approaches model the relevance ranking with many features, these models are usually only trained on the final submitted queries, ignoring the entire user interactions from the first character she types to the final query she has clicked.

One difficulty for improving the QAC quality is the lack of data about fine-grain user interactions in QAC. Recent work has attempted to leverage all prefixes of a submitted query [3, 20]. However the only available data is the submitted query; while the prefixes are *simulated* from all possible prefixes of the query. Lack of associated information, such as the suggested list, user typing speed and other real user interactions, prevents such methods from further improving their performance.

For this purpose, we have collected a high-resolution QAC dataset from both PC and mobile phones, in which each keystroke of a user and the system response are recorded. As far as we know, this is the *first* dataset with this level of resolution specifically for QAC. Extensive studies have already demonstrated the importance of query log for web document retrieval [17, 13, 1, 14, 19]. Therefore, it is reasonable to believe this new kind of QAC log could potentially enable a full spectrum of researches for QAC, such as user behavior analysis, relevance ranking, interactive system design for QAC, just to name a few.

Given many possibilities for mining this new data, in this paper we focus on leveraging it for understanding users' behavior in QAC. Specifically based on our QAC log, we have observed a phenomenon that in QAC users frequently skip several suggestion lists, even though such lists contain the final submitted query. The exact reason why this happens and how frequent it happens is largely unknown. Besides, we also observed that most of the clicked queries are concentrated at top positions. Better understanding of these behaviors has a strong implication to the relevance modeling. For instance, we assume that a user does not click a suggested query due to the lack of relevance; however the skipping behavior complicates this hypothesis. If we know the positions where such skipping behavior happens, we could improve the candidate ranking in QAC by taking into account the examples in the positions where they are more likely to be examined. Despite its importance, little research has been done in explaining such behaviors.

The QAC process shares similarities with the web document retrieval: in QAC people look for intended queries with a prefix, while in document retrieval people look for relevant documents with a query. And in document retrieval, click models are widely used to model the users' examination and clicking behavior [18, 7, 9, 16, 24, 6]. Thus we could potentially adopt an existing click model to shed light on the QAC user behavior. However, there are major differences between the QAC process and document retrieval. For example, in document retrieval a user usually examines one result page before she lands on a click, while in QAC she usually types in a series of prefixes and examines multiple lists of suggestions before landing on a click. Due to this difference, most current click models are not applicable to the QAC problem without significant modification.

Therefore in this paper we propose a novel two-dimensional click model for understanding the user behaviors in QAC. This click model is consisted of a horizontal component that captures the skipping behavior, a vertical component that depicts the vertical examination bias, and a relevance model that reflects the intrinsic relevance between a prefix and a suggested query.

We have performed a set of experiments on our QAC datasets from PC and mobile phones. Results show that our proposed model can effectively model the user behavior in QAC. The resulting relevance model significant improves the QAC performance over existing click models. We also show that the learned knowledge about users' behavior, especially the probability of skipping a column of suggestion candidates, could serve as unsupervised labeling information to improve the performance of existing learning-based approaches. Furthermore, with the learned model we demonstrated some interesting insights of the user behaviors in QAC on both platforms.

We summarize our contributions as follows:

- We have collected the first set of high-resolution query log specific for the QAC process, which could enable many studies on deeper understanding of QAC.

- Based on the new QAC log, we analyze two important types of user behaviors in QAC, namely the horizontal skipping bias and vertical position bias. The horizontal skipping bias is unique to QAC and is formally introduced here for the first time.

- We propose a novel Two-Dimensional Click Model to model these user behaviors. Our model outperforms the state-of-the-art click models on relevance modeling. We also utilize our model to derive interesting insights about the QAC user behaviors on PC and mobile devices.

2. RELATED WORK

Query Auto-Completion. The query auto-completion is the search process of preferred queries given the issued prefix of a user. Most of the existing works focus on relevance ranking. For this purpose, traditional QAC engines rely on query popularity counts. However it's impossible to return queries matching a user's specific preference such as location and freshness in time *etc*. Recent QAC models employ learning-based strategy that incorporates several global and personal features [3, 20]. But there is no consensus of how to optimally train the relevance model.

The QAC process is very personal in nature, so it's almost impossible to obtain a labeled dataset by third-party annotation. Existing methods use the clicks as a relevance surrogate, and train a model trying to maximize the clicks. The straightforward way is to only utilize the data in the last prefix, and use the skip-above as well as the skip-next hypothesis to obtain a set of labels. Then we could use the learning-based algorithms to train a model that linearly combines a set of features [3]. Most recently [20] introduces a different strategy, which exploits all suggested queries for all simulated prefixes of the clicked query. However, this automatic labeling strategy might be problematic, since it may introduce many false negative examples where the user skips looking down the list. If she had to examine the list, she would have clicked a query. So there is a lot of uncertainty in the labeled examples introduced by this method. For evaluation, [4] proposes two new metrics for offline QAC.

Besides relevance modeling, there is previous work addressing different aspects of QAC. For example, [5, 11] studied the space efficiency of index for QAC. [22, 12] proposed efficient algorithms for QAC. [8] addressed the problem of suggesting query completions when the prefix is mis-spelled. And [2] studied the context-sensitive QAC for mobile search.

The QAC is a complex process where a user goes through a series of interactions with the QAC engine before clicking on a suggested query. Deciphering the user behavior in QAC is an interesting and challenging task. Despite of its importance, little research is done on this direction, mainly because of the lack of suitable QAC log. It is in this work we first collect a high-resolution QAC log and attempt to model the user behaviors.

Click Models. In the field of document retrieval, the main purpose for modeling a user's 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.* [10], stating that a document on higher rank tends to attract more clicks. Richardson et al. [18] attempted to model the true relevance of documents by imposing a multiplicative factor. Later examination hypothesis is formalized in [7], 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. Later, several extensions were proposed, such as the User Browsing Model (UBM) [9], Bayesian Browsing Model [16], General Click Model [24] and Dynamic Bayesian Network model (DBN) [6]. Despite the abundance of click models, no existing click models can be directly applied to QAC without considerable modification. The click model most similar to our work is [23], 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 structure is very different from [23]. To the best of our knowledge, our work is the first click model for modeling the QAC process.

3. DATA AND USER BEHAVIOR ANALYSIS

In this section we will introduce the high-resolution query log for QAC and the user behavior analysis based on this new kind of data.

3.1 A High-Resolution QAC Log

As mentioned above, the QAC process is under-explored because there is no appropriate dataset. Previous research relies on search query log in which only the submitted query and associated information are recorded. In order to analyze the subtle user behavior in a whole QAC process we need to record system response and user interactions for each keystroke leading to the final clicked query. With this motivation we have collected a large set of QAC sessions with real user interactions from Yahoo!. This QAC log contains millions sessions on PC and mobile phone platforms. The dataset in this study is a random sample of the original QAC log dating from Nov. 2013 to Jan. 2014.

As illustrated in Table 1, the recorded information in each QAC session includes the final clicked query, every keystroke a user has entered, timestamp of a keystroke, corresponding top 10 suggested queries to a prefix, and the anonymous user ID. It also records the user's submitted query in previous session. Table 2 lists the basic statistics of the dataset studied in this work. In PC platform each session contains 11.80 prefixes in average; while the average clicked query length is 19.68, which is substantially larger than the average prefix length, indicating the usefulness of QAC for facilitating faster user query input. We observe similar statistics on the iPhone 5 platform, with lower average prefixes in one session (9.43), suggesting that users rely even more on mobile devices where typing takes more effort.

Table 1: High-resolution QAC Log

Data Type	Example
anonymized user id	9qtfnj195p5ta
session id	rFzqRUgeurCd
time stamp	11/02/2013 23:02
prefix	oba
final submitted query	obama care
previous query	charm and charlie's
clicked URL	https://www.healthcare.gov/
top 10 suggested queries	obama care|obama|oba| obamacare|obama approval rating|...

Table 2: Dataset Basic Statistics

Platform	# Sessions	Ave Prefixes	Ave Clicked Qry Len	# Unique User IDs
PC	125,392	11.80	19.79	111,783
iPhone 5	31,227	9.43	16.98	17,331

Significance: With this QAC log, for the first time we have the opportunity to look into the real user interactions at the level of every keystroke. Such high-resolution dataset, when combined with traditional query log about user demographics and query history, could enable many new researches on QAC. For example, we could potentially utilize all lists of suggested queries to improve the QAC relevance ranking. Also, we could leverage this data to get better understanding of user behavior in the QAC process.

3.2 QAC User Behavior Analysis

Given the new QAC log, there are many possibilities to mine valuable knowledge. In this paper we aim at leveraging the data for user behavior modeling in QAC. When a user clicks on a suggested query with the help of a QAC engine, she undergoes a series of interactions with the QAC engine before she finally selects a preferred query. Such interactions are of great value for improving the quality of the QAC service. In this section we conduct two experiments to verify the need of modeling user behaviors in QAC.

The first important user behavior in the QAC process is the skipping behavior. We have observed that a user frequently skips several intermediate lists of candidates even though these lists contain her final selected query. Table 3 illustrates this skipping behavior from a real user-interaction sample. In this example, even though the query *obamacare healthcare bill* is listed within top 3 positions in each of the suggestion list, the user has skipped all of them but the last appearance. Clearly this query is preferred by the user. A plausible explanation for the skipping behavior is that the user didn't examine it due to some reasons, such as fast typing speed, too deep to look up the query, *etc*.

We performed an experiment on the dataset described in Table 2 to verify how often the skipping behavior happens. In this experiment we define that a 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. Results in Table 4 show that this behavior is frequent: it happens in 60.7% of all sessions in PC platform. Further, this behavior is consistent in all session groups with different final prefix length (57.6%, 64.8%, 59.1% and 60.2% respectively), indicating its prevalence in all queries. This result suggests a common skipping behavior in the QAC process. We observe very similar phenomenon in the iPhone 5 platform.

Table 4: Frequency of the Skipping Behavior

Category	# Sessions	% Sessions Having Skipping Behavior
All Sessions	125,392	60.7%
Sess with FPL in [1, 5]	39,405	57.6%
Sess with FPL in [6, 10]	39,882	64.8%
Sess with FPL in [11, 15]	22,892	59.1%
Sess with FPL in [16, 50]	23,213	60.2%

Note: FPL means Final Prefix Length.

In another experiment, we investigated the vertical examination bias in QAC. Using the same set of QAC sessions, we computed the distribution of clicks according to their positions in the final suggestion list and the final prefix length. Figure 1 shows the 2-dimensional click distribution on both PC and iPhone 5 platforms. Similar to the findings in the traditional click models, most of the clicks concentrate on top positions. In fact, 75.4% of clicks is located within the top 2 positions on PC and 77.5% on iPhone 5. Such

Table 3: An Example of the Skipping Behavior

Prefix	obamacar|	⇒	obamacare|	⇒	obamacare |	⇒	obamacare h|
q_1	obamacare glitches		obamacare glitches		obamacare glitches		obamacare healthcare bill √
q_2	obamacare		obamacare		obamacare healthcare bill √		obamacare healthcare insurance
q_3	obamacare healthcare bill √		obamacare healthcare bill √		obamacare facts		obamacare health plan 2014
q_4	obamacare facts		obamacare facts		obamacare rates 2014		obamacare hotline
q_5	obamacare fines		obamacare fines		obamacare fines		obamacare health exchanges

Note: query with a √ mark is the final clicked query by the user.

vertical positional bias suggests that we should boost the estimated relevance for queries which are clicked at lower ranks. Compared to PC, the clicks on iPhone 5 distribute more evenly with-in positions from 1 to 3. In addition, Figure 1 also indicates that most of the clicks are located in prefix length ranging from 3 to 12 on both platforms. Interestingly, the click probability at short prefix length (1 and 2) is very low, suggesting that users tend to skip the suggested queries at the beginning.

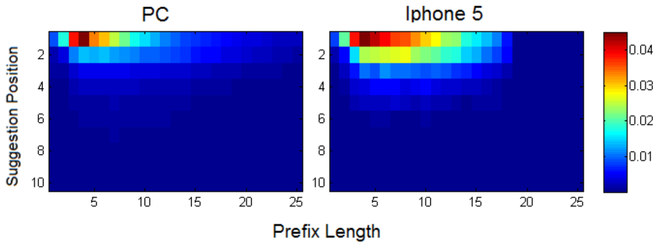

Figure 1: Distribution of the clicks. The red color corresponds to high click probability, while blue corresponds to low click probability.

The reason why we focus on these two behaviors is their important implications to relevance ranking. Recent research attempts to improve the relevance model by training on all simulated columns (lists) of suggestions [3, 20]. However, not all of columns are examined by the users in reality. As a result, it might introduce many false negative examples that hurt the performance. To validate this claim, we have conducted an experiment on our QAC dataset from PC platform (see Section 5.1 for detailed description) in which we adopt the same training strategy as [20]. For the learning-to-rank algorithm, we use the RankSVM [13]. We also adopt very similar features as [20] (see Table 6). *MostPopularCompletion* (MPC) is used as a baseline. Another baseline is to train RankSVM only by the last column (suggestion list corresponding to the last prefix). We also add the third baseline, which is also RankSVM, but trained by last 2 columns. Same as [20], we evaluate MRR across all columns where the final submitted query is within the candidates. Results in Table 5 indicate that training on all columns is inferior to the same model trained on last column. And it is even worse than the MPC baseline. Interestingly, the same model trained on only the last 2 columns achieves slightly better result than using only using last column, suggesting that adding more (useful) columns might be beneficial. We hypothesize that columns that are likely to be examined are useful for training.

Table 5: A Pilot Experiment on Relevance Training

Method	MRR@All
RankSVM - trained by all columns	0.436
RankSVM - trained by last column	0.514
RankSVM - trained by last 2 columns	0.518
MPC	0.447

4. MODELING CLICKS IN QUERY AUTO-COMPLETION

Based on the results of the above experiments, we demonstrate that the skipping behavior and vertical click position bias are prevalent and important for improving QAC quality. How to model these behaviors is a new research problem. Given the similarity between QAC and document retrieval, first we sought to apply the existing click models to this problem. But we found most of these click models are not appropriate for the following reasons: (1) most existing click models only model a single query at a time. But in QAC, a session contains a series of prefixes that are correlated. (2) traditional click models are unable to model unseen query-document pairs. However in our QAC log we observe that a large portion (67.4% in PC and 60.5% in iPhone 5) of the prefix-query pairs are unseen. Therefore we propose a new click model for QAC, with emphasis on modeling these two types of user behaviors. We first formally define the assumptions on these two types of bias; then we will describe our click model in detail. After that we will discuss the parameter estimation via Expectation Maximization.

4.1 QAC Click Bias Assumptions

Here we define two basic assumptions for the QAC problem. One is to address the click bias due to the skipping behavior, and the other is to address the click bias on vertical positions.

- **SKIPPING BIAS ASSUMPTION:** A query will not receive a click if the user didn't stop and examine the suggested list of queries, regardless of the relevance of the query.

This assumption explains why there are no clicks to intermediate prefix even though a relevant query is ranked at the top of the list.

- **VERTICAL POSITION BIAS ASSUMPTION:** A query on higher rank tends to attract more clicks regardless of its relevance to the prefix.

Similar to the click modeling for document retrieval, this as-

Table 6: Features of the H, D and R Models

Category	Feature	Feature Group	Description
H Model	CurrPosition	Prefix	Ratio of length between current prefix and the final prefix
	IsWordBoundary	Prefix	Binary indicator, whether the end of current prefix is at word boundary
	NbSuggQueries	Query	Number of suggested queries
	ContentSim	Query	Content similarity of suggested queries
	TypingSpeed	User	Typing speed at this keystroke
	QueryIntent	User	Whether the final submitted query is a navigational query
D Model	Is@Depth d	Query	Binary indicators, whether the query candidate is at depth d, $d = \{1, ..., 10\}$
R Model	MPC	Query	Candidate frequency computed based on past popularity
	TimeSense	Query	Candidate popularity measure in one day
	GeoSense	Query	Candidate popularity measure at the city where the query is issued
	QryHistFreq	User	The number of times the candidate is issued as query by the user in the past
	SameGenderFreq	Demographics	Candidate frequency over queries submitted by users of the same gender
	SameGenderLikelihood	Demographics	SameGenderFreq normalized by MPC
	SameAgeGroupFreq	Demographics	Candidate frequency over queries submitted by users of same age group
	SameAgeGroupLikelihood	Demographics	SameAgeGroupFreq normalized by MPC

sumption explains why top ranked queries receive more clicks even though they are not necessarily relevant to the given prefix.

4.2 Model Formulation

Based on the assumptions defined above, in this section we propose a Two-Dimensional Click Model (TDCM) to explain the observed clicks. This click model is consisted of a horizontal model (H Model) that explains the skipping behavior, a vertical model (D Model) that depicts the vertical examination behavior, and a relevance model (R Model) that measures the intrinsic relevance between the prefix and a suggested query. Figure 2 is a flowchart of user interactions under the TDCM model. The user interacts with the QAC engine horizontally and vertically according to the H, D and R models. Because in every QAC session, there is no click before the user leaves the process, we employ the Cascade Model assumption [7] that specifies the relations between the H, D and R models. We list the notations of TDCM in Table 7. According to the TDCM, the generative process of observing a click in a QAC session is described as follows (see Figure 2 also):

1. For a QAC session, let's assume the user has entered several characters and she is at prefix i, then she will decide whether to stop and look down to examine the list of suggested queries at ith column. This whether-to-look-down event is governed by a hidden random variable H_i, $H_i = 1$ means stop and examine, $H_i = 0$ means skip and continue to type. The task of the horizontal model (H Model) is to estimate the distribution of H: $P(H)$.

2. Once the user decides to examine vertically, following the cascade model assumption she will examine one query at a time from top to bottom. The depth of the examination is determined by another hidden random variable D_i. $D_i = j$ means the user examines the query at position j at ith column. While being equivalent to introducing a set of binary variables at each depth, this formulation is more convenient in parameter estimation. The task of the vertical model (D Model) is to estimate the distribution of D: $P(D)$.

3. If a query candidate is examined and it is relevant, according to the cascade model assumption, the user will click it. The probability a query being relevant to the given prefix is determined by the relevance model: $P(C_{ij} = 1|H_i = 1, D_i \geq j)$. The task of the relevance model (R Model) is to estimate the distribution of $P(C_{ij} = 1|H_i, D_i)$.

4. If the depth D_i is reached and no relevant queries are found, she will go back to Step 1 and continue to type another character.

5. Once a click event happens, she will end the auto-completion session, which implies there will never be more than once click observed in a session.

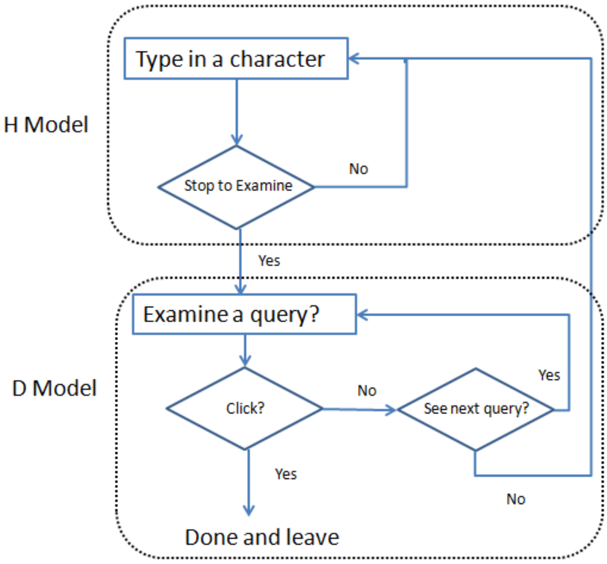

Figure 2: TDCM Flowchart

4.3 Click and Conditional Probabilities

Based on the above generative process, the probability of observing a click C in a session can be formulated as:

$$P(C) = \sum_{H,D} P(C, H, D) \tag{1}$$

Table 7: Major Notations

Symbol	Description
p_i	Prefix at ith column.
$q_{i,j}$	Query at position (i, j).
n	Number of columns in a QAC session.
H_i	Whether the user stops to examine the column i.
H	A vector of variables: $H = \{H_1..., H_n\}$.
D_i	Depth of examination at column i.
D	$D = \{D_1..., D_n\}$.
$C_{i,j}$	Whether the query at (i, j) is clicked.
C_i	A vector of variables: $C_i = \{C_{i,1}..., C_{i,M_i}\}$.
C	The click matrix: $C = \{C_1, ..., C_n\}$.
M_i	# queries in the suggestion list at column i.
w_H, w_D, w_R	Feature weights of the H, D and R model.
x_H, x_D, x_R	Features of the H, D and R model.

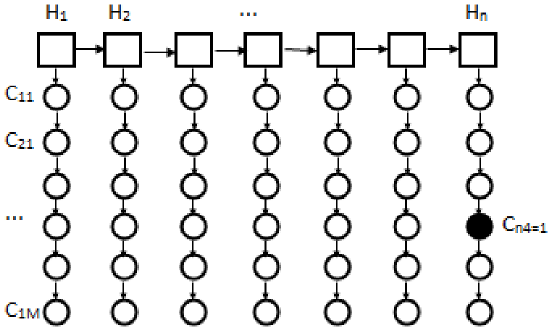

Figure 3: TDCM Model Structure

In TDCM, $H = \{H_1..., H_n\}$, $D = \{D_1..., D_n\}$ are hidden variables. $C = \{C_1, ..., C_n\}$ is the click observation matrix in which only one click is observed: $C_{n,J} = 1$, n is the number of columns in the QAC session. Figure 3 depicts the relation between the hidden and observed variables. According to the Cascade Model assumption and the real observations of a QAC session, there is always only click observed, which implies other columns don't receive any click:

$$C_{n,J} = 1 \Leftrightarrow \{C_1 = 0, ...C_{n-1} = 0, C_{n,J} = 1, C_{n,j} = 0, j \neq J\},$$

So:

$$P(C_{n,J} = 1) = P(C_1 = 0, ...C_{n-1} = 0, C_{n,J} = 1, C_{n,j} = 0, j \neq J). \tag{2}$$

Our model also follows a set of conditional probabilities:

$$P(C_{ij} = 1|H_i = 0) = 0 \tag{3}$$

$$P(C_{ij} = 1|H_i = 1, D_i < j) = 0 \tag{4}$$

$$P(C_{ij} = 0|H_i, D_i) = 1 - P(C_{ij} = 1|H_i, D_i) \tag{5}$$

$$P(D_i > d|q_d : C_{n,d} = 1) = 0. \tag{6}$$

The TDCM assumption 1 (SKIPPING BIAS ASSUMPTION) is modeled by 3 and 6. The assumption 2 (VERTICAL POSITION BIAS ASSUMPTION) is modeled by 4 and 6. Equation 6 states that if a relevant query is ranked in depth d, the examination depth at ith column must not exceed d.

4.4 The Forms of Distributions

Now we introduce the forms of distributions for H, D and R model. Different from most of the click models and similar to [21], we define the distributions via logistic functions:

$$P(H_i = 1) = \sigma({w_H}^T \cdot x_H), \tag{7}$$

where $\sigma(z)$ is a logistic function: $\sigma(z) = \frac{1}{1+e^{-z}}$.
So

$$P(H_i = 0) = 1 - \sigma({w_H}^T \cdot x_H), \tag{8}$$

Similarly, for D_i, we have:

$$P(D_i = j) = \frac{e^{{w_D}^T \cdot x_{Dj}}}{\sum_{l=1}^{M_i} e^{{w_D}^T \cdot x_{Dl}}}, \tag{9}$$

where $j \in \{1, .., M_i\}$, M_i is the number of queries in the suggestion list at ith column.

And for the R model, we have:

$$P(C_{ij} = 1|H_i = 1, D_i \geq j, \theta) = \sigma({w_R}^T \cdot x_{Ri,j}), \tag{10}$$

$$P(C_{ij} = 0|H_i, D_i) = 1 - P(C_{ij} = 1|H_i, D_i) \tag{11}$$

In the above formulations, x_H, x_D, x_R are features characterizing the H, D, R distributions. And $\theta = \{w_H, w_D, w_R\}$ are the corresponding weights for the features. As stated in [21], using this form of distribution has the advantage of incorporating more useful signals from diverse sources. And it also make it feasible for predicting the unseen prefix-query pairs.

4.5 Features

Table 6 summarizes the features used in the H, D and R models. Specifically, for the H model, we adopt these features for the following reasons. *TypingSpeed*: an expert user is less likely to use QAC than an inexperienced user. *CurrPosition*: a user tend to examine the queries at the end of typing. *IsWordBoundary*: a user is more likely to lookup queries at word boundaries. *NbSuggQueries*: it's more likely to be examined if the number of suggested queries is small. *ContentSim*: a user may be more likely to examine the list if all queries are coherent in content. *QueryIntent*: a user tends to skip the list more when searching for navigational queries.

The feature for D model are the positions a query candidate is ranked. The purpose of using this feature is that we want to use the D model to measure the pure vertical position bias. Note that the form of D model allows us to incorporate more complex features in the future.

For the R model, we have designed 8 features in total, reflecting diverse aspects of the relevance model. It includes the query popularity counts, which is widely used in the current search engines, the long term query history query counts, geo-location and time related query frequencies, and 4 other demographics features. Similar features are reported in [20], therefore comparing our model to that in [20] is meaningful.

4.6 Model Estimation via E-M Algorithm

In this section we discuss the estimation of model parameters $\theta = \{w_H, w_D, w_R\}$. A straightforward way is to take the log of Equation 1 and estimate θ by Maximum Likelihood. However since Equation 1 involves the summation of the H and D vectors, the estimation is quite complicated. Based on the form of distribution and the choice of features, we make some independent assumption of variables at different columns in order to simplify the model estimation:

$$P(H_i|H_j, i \neq j, \theta) = P(H_i|\theta) \quad (12)$$

$$P(D_i|D_j, i \neq j, \theta) = P(D_i|\theta) \quad (13)$$

$$P(C_i|H_i, D_i, H_j, D_j, i \neq j, \theta) = P(C_i|H_i, D_i, \theta) \quad (14)$$

This assumption breaks the interdependency between columns. And the likelihood of different columns are still related because they share common parameters. Under these assumptions, the log likelihood of observing a click given the model parameters θ is:

$$logP(C|\theta) = \sum_{i=1}^{n} log \sum_{H_i, D_i} P(C_i, H_i, D_i|\theta) \quad (15)$$

Model parameters $\theta = \{w_H, w_D, w_R\}$ can be estimated by maximizing Equation 15. However, direct estimation of the model parameters θ is still hard because of the summation inside the logarithm. Instead, we sought to maximize the lower bound of Equation 15:

$$\begin{aligned} logP(C|\theta) &= \sum_{i=1}^{n} log \sum_{H_i, D_i} P(C_i, H_i, D_i|\theta) \\ &\geq \sum_{i=1}^{n} \sum_{H_i, D_i} P(H_i, D_i|C_i, \theta^{old}) \cdot logP(C_i, H_i, D_i|\theta) \\ &= Q(\theta, \theta^{old}) \quad (16) \end{aligned}$$

After fully formulating the Q function, model parameters can be updated iteratively by the E-M algorithm. In the E step, we aim at calculating the posterior distribution:

$$P(H_i, D_i|C_i, \theta^{old}) = \frac{P(C_i|H_i = l, D_i = j, \theta^{old}) \cdot P(H_i = l, D_i = j|\theta^{old})}{\sum_{l=0}^{1} \sum_{j=1}^{M_i} P(C_i|H_i = l, D_i = j, \theta^{old}) \cdot P(H_i = l, D_i = j|\theta^{old})} \quad (17)$$

And in the M step, we maximize the expectation of the complete data probability in Equation 16 by gradient descent. Since the forms of distributions in H, D, R are all convex, the E-M algorithm is guaranteed to converge. And each QAC session is independent, therefore the above E-M algorithm for parameter estimation can be easily expanded to the whole set of sessions. Here we skip the detailed formulations due to the space limitation.

5. EXPERIMENTS AND RESULTS

In this section, we conduct a series of experiments to validate our major claims in the TDCM model. Firstly, due to the difference between document retrieval and QAC, we claim that most of the existing click models are not effective in modeling the QAC. We will compare our model with the state-of-the-art click models on the relevance modeling. Besides testing on PC and iPhone 5 datasets, we also experiment on a random bucket dataset which provides an unbiased evaluation of the relevance ranking. Secondly, as we have mentioned, for training a QAC relevance model, previous researches either use the last column as training data which might not have enough training cases, or use all columns as training examples [20] which might introduce too much noise. We will demonstrate that our model can be leveraged to improve existing learning-based methods by providing more appropriate training examples. Further, we will investigate the vertical position bias via our model on a side-by-side comparison of such bias on PC and iPhone 5 platforms. Finally we discuss some interesting insights about user behaviors on both platforms.

5.1 Datasets and Metrics

We use the same datasets for evaluation as in user behavior analysis (Section 3.2). The whole dataset is divided evenly into a training set and a test set. See Table 8 for detailed statistics.

Table 8: Training and Test Sets

Platform	Dataset	# Sessions	Ave Prefixes
	All	125,392	11.80
PC	Training	62,534	11.77
	Test	62,858	11.83
iPhone 5	All	31,227	9.43
	Training	15,394	9.46
	Test	15,833	9.40
Random Bucket	Test	21,154	16.15

As reported in previous research [3, 20], manual labeling of relevance for QAC is very difficult since it's hard to find consensus between individuals on the preferred queries given the same prefix. Instead a common practice of evaluating the QAC performance is to measure the prediction accuracy of the users' clicked queries [3]. In this work we adopt this evaluation strategy. In addition, because the user clicks are a biased estimate of relevance, we also set up a random bucket to collect clicks from a small portion of traffic during the same period. In this random bucket, for every prefix top-10 ranked queries are randomly shuffled and displayed to the users. By doing so, it reduces the vertical position bias and the collected user clicks can be treated as the unbiased estimation of relevance of queries [15].

For evaluation metrics, we employ the Mean Reciprocal Rank (MRR) as the main measurement of relevance. It is the standard practice in measuring QAC performance [3, 20]. Specifically, for a QAC session, the list of candidates are generated from a commercial search engine and recorded in our dataset. Columns (suggested queries associated with a prefix) in which the final submitted query does not appear among the top-10 candidates are removed from the analysis. Then we compute the average MRR across all these columns. In addition, we also report the MRR of the last column since this is the column where real user click happens. Paired t-test is adopted for validating the statistical significance with *p-value* cutoff 0.05.

For baselines, *MostPopularCompletion* (MPC) is used as a baseline. Despite its simplicity, MPC has been reported as a very competitive baseline and widely used as a main feature in the QAC engines [3, 20]. We also compare our approach to three state-of-the-art click models, including User Browsing Model (UBM)[9], Dynamic Bayesian Network model (DBN) [6] and Bayesian Sequential State model (BSS) [21]. The UBM and DBN rely on the counting of prefix-query pairs thus they are unable to predict unseen prefix-query pairs. On the other hand BSS is a content-aware method and it can predict unseen prefix-query pairs. We adopt the source code of these approaches from [21]. Since our model makes use of all columns of data, to make a fair comparison, we train these click models on the last column as well as all columns. All baselines and their description are listed in Table 9.

Table 9: Baselines for Comparison

Model	Description	Training Data
MPC	Most Popular Completion	no training is needed
UBM-last	User Browsing Model	last column
UBM-all	User Browsing Model	all columns
DBN-last	Dynamic Bayesian Network model	last column
DBN-all	Dynamic Bayesian Network model	all columns
BSS-last	Bayesian Sequential State model	last column
BSS-all	Bayesian Sequential State model	all columns
TDCM	Our model	all columns

5.2 Evaluating the Relevance Model

Normal Bucket test. We first investigate whether our click model has advantage over existing click models on improving the QAC relevance ranking. For this purpose, we compare our model to the MPC baseline and other click models on normal buckets from both PC and iPhone 5 platforms. The results are summarized in Table 10. Firstly, counting-based click models (UBM and DBN) are generally not effective for modeling the relevance in QAC. For example, the UBM-last and DBN-last methods under-perform the MPC baseline on both PC and iPhone 5 datasets. Although UBM-all DBN-all perform a little better than UBM-last and DBN-last, training on all columns of data still doesn't give an edge to these methods over MPC baseline. This is not surprising because UBM and DBN rely on counting the prefix-query pairs. However, in the dynamic environment of QAC, the percentage of unseen prefix-query pairs is large (67.4% in PC and 60.5% in iPhone 5 as observed in our datasets), which is very different from that in document retrieval. Presumably, training on all columns will add more seen prefix-query pairs, which leads to the improved MRR. However, since these models are not designed to model the whole QAC process, they are unable to capture useful signals in all columns and thus unable to improve the performance much.

In addition, the BSS model performs better then UBM and DBN. For example, when trained on last column, it achieves 0.515 on MRR@All and 0.543 on MRR@Last on the PC dataset, indicating its effectiveness of capturing relevance by content-based modeling of relevance. One advantage of content-based modeling is that it's able to interpolate the relevance model to unseen prefix-query pairs, which is critical in QAC. But training on all columns doesn't boost its performance, suggesting the importance of modeling the user behavior in the whole QAC process so as to filter out the noise. We also note that BSS is not consistent on these two platforms: for example, it doesn't work well in the iPhone 5 dataset (0.510 on MRR@All on 0.537 on MRR@Last by BSS-last).

On the other hand, our TDCM model achieves significant better results on both platforms. For example it achieves 0.525 on MRR@All and 0.573 on MRR@Last on the PC dataset. And on iPhone 5 dataset, it gets 0.580 on MRR@All and 0.668 on MRR@Last. All of these results are statistically significant better than MPC. Compared to UBM and DBN, our model overcomes their limitation by adopting the content-aware relevance model. And compared to BSS, our model takes advantage of all columns of data by properly modeling the user behavior in the whole QAC process, leading to much better and stable results on both platforms.

Table 10: Click Models Comparison on Normal Bucket

	PC MRR@All	PC MRR@Last	iPhone 5 MRR@All	iPhone 5 MRR@Last
MPC	0.447	0.534	0.542	0.646
UBM-last	0.416	0.449	0.409	0.432
UBM-all	0.445	0.452	0.431	0.432
DBN-last	0.418	0.437	0.405	0.427
DBN-all	0.454	0.442	0.435	0.423
BSS-last	**0.515**‡	0.543	0.510	0.537
BSS-all	0.495	0.523	0.480	0.479
TDCM	**0.525**‡	**0.573**‡	**0.580**‡	**0.668**‡

Note: ‡ indicates *p-value*<0.05 compared to MPC

Random Bucket test. Using normal traffic to evaluate the relevance model might be biased because a model could be optimized by chasing the clicks rather than the intrinsic relevance/utility. To make an unbiased evaluation we also test all the methods on a random bucket dataset containing 21,154 QAC sessions. We summarize the results in Table 11. Overall the MPC baseline performs worse than that in normal bucket. It's expected because as the position bias is reduced, users have more chance to click on queries that are not the most popular. Similarly, UBM and DBN models fail to outperform MPC baseline and the BSS model achieves reasonable results compared to MPC. Again, our model achieves the best results on both MRR@All (0.493) and MRR@Last (0.508) metrics, which is statistical significant compared to MPC. These results are consistent with that observed in normal traffic, confirming the superiority of our TDCM model on relevance modeling.

Table 11: Click Models Comparison on Random Bucket

	MRR@All	MRR@Last
MPC	0.429	0.485
UBM-last	0.381	0.402
UBM-all	0.397	0.393
DBN-last	0.373	0.391
DBN-all	0.388	0.391
BSS-last	**0.471**‡	0.488
BSS-all	0.460	0.469
TDCM	**0.493**‡	**0.508**‡

Note: ‡ indicates *p-value*<0.05 compared to MPC

5.3 Relevance Model Performance by Query Length

In order to investigate whether our model is robust in all sectors of queries, we break the relevance results into 5 groups according to their clicked query length. MPC results are also shown for comparison. Results in Figure 4 reveal that on both PC and iPhone 5 datasets, our model's performance decreases gradually as the submitted query length increases. There is no abrupt drop of performance in a sector of queries, indicating that our model is robust to queries with different length. In addition, the MPC baseline has similar trend as our model. This common trend suggests the importance of the query popularity count because the shorter queries generally have higher popularity counts. In the random bucket, the MRR of our model drops when the query length increase, and starts to increase again when the query length becomes larger. This trend suggests that in random bucket, MPC feature becomes less important than in normal bucket; thus longer queries will still have good MRR even though their MPC scores are smaller.

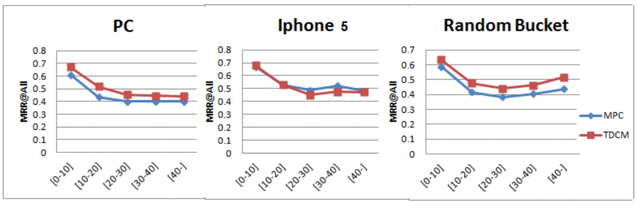

Figure 4: MRR Evaluation by Query Length. All sessions are aligned to groups based on the submitted query length. Performance is measured by MRR@All

5.4 Validating the H Model: Automatic Labeling by TDCM

Another advantage of our model over existing click models is that we can utilize the learned user behavior in QAC to enhance other learning-based methods. In the pilot experiment in Section 3.2, we have shown that even though RankSVM is a state-of-the-art ranker, when trained by all columns, its performance doesn't even beat MPC (-2.46% on MRR@All). The reason is probably because although we are sure that a user has viewed and examined the last column, it's uncertain that she has viewed other columns; so the information in previous columns is not reliable. The noise in all columns may outweigh the useful information they bring about. So by simply training on all columns it is generally not effective. In this experiment, we test whether the user examination behavior inferred by our model can be used to improve other methods. In order to achieve this, we first run TDCM on the training dataset, obtain all $P(H)$ probabilities for each session. After that, we keep the columns with high $P(H = 1)$ (>0.7). Finally we use these columns to train the corresponding models again. The labeling criteria is simple: if the candidate equals the final submitted query, we label it as positive, and other candidates are all labeled as negative. Results of this experiment are shown in Figure 5. It is indicated that using this simple automatic labeling strategy, RankSVM achieves better MRR@All across three datasets. For example, on PC dataset RankSVM achieves 0.523 on MRR@All, compared to 0.514 by training on last column only. Similar improvements are observed in iPhone 5 and Random Bucket. These results suggest that the user behavioral information inferred by our model can be applied to other models, especially the information whether a query has been examined is very useful for improving other models' performance.

5.5 Validating the D Model

Here we seek to evaluate the accuracy of the D model, that is the vertical examination distribution. Intuitively the probability of examining a position should be correlated to the clickthrough rate.

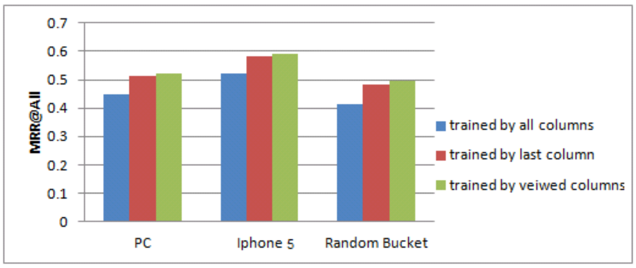

Figure 5: Model Training on Selected Columns. Viewed columns: columns whose $P(H = 1) > 0.7$. Performance is measured by MRR@All

In our feature instantiation, all features for the D model are vertical positions. So it is possible to draw the distribution of D according to the feature weights w_D, which corresponds to the probability of examining a particular position. In this experiment we run the TDCM model on both PC and iPhone 5 dataset, and draw the distribution along with the real click through rate (CTR) in Figure 6. From Figure 6 we see that the shape of the D model distribution is similar to the real CTR. Both distributions are very steep, attracting more probabilities in top positions. Our estimation of the D model is a little flatter than the real CTR. For example, on PC platform, at the top 1 position our model estimates the examination probability to be 0.397, while the real CTR is 0.500. And in the 2nd position we predict more probability (0.314) than the real CTR (0.254). Compared to PC platform, in iPhone 5 platform both the real CTR and our estimated examination distribution are flatter. This suggests an interesting conclusion that in mobile devices people tend to examine deeper down the suggestion list.

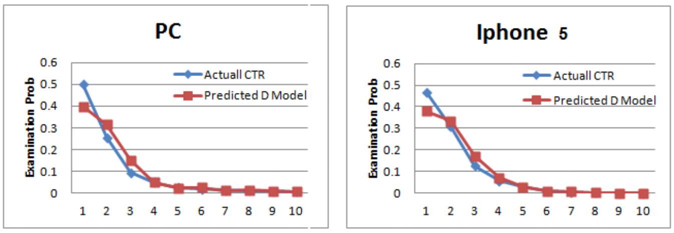

Figure 6: The D distribution VS real CTR. Positions

5.6 Understanding User Behavior via Feature Weights

An additional benefit of our proposed model is that the learned feature weights reveal the influence of different factors on users' behavior in the QAC process, which is not available in most of the existing click models. To explore this, we list a subset of learned weights in Table 12. Although the absolute values of these weights don't reflect exactly the importance of features because scales of the features are different, we can still tell their relative importance by comparing them on PC and iPhone 5 side by side.

Firstly, in the H model related features, *TypingSpeed* is the most important feature both on PC and iPhone 5. *TypingSpeed* is reversely proportional to $P(H) = 1$. Interestingly, the absolute weight of $TypingSpeed$ is larger in PC than in iPhone 5, suggesting that people tend to skip more when using QAC in PC because they type faster in PC. Another important feature is *IsWordBoundary*. Intuitively it makes sense since people tend to stop and look for query completions when they are typing at word boundaries. The *QueryIntent* feature also plays a role, indicating that people tend to skip more when looking for navigational queries; while they need more help from the QAC engine when they are seeking information and uncertain how to formulate the queries.

Secondly, the features of the D model is examination probabilities. As mentioned in the previous experiment, these probabilities are higher at the top positions. In PC, the estimated examination probabilities concentrate more on the 1st position. On iPhone 5 the 2nd and 3rd positions receive more examination probabilities than PC. This suggests that in mobile devices people will look deeper down the suggestion list.

Thirdly, for the R model, people pay more attention on long query history in iPhone 5 than in PC. This might be because typing is harder in mobile devices; so people rely on the QAC engine to store and retrieve their past queries. Another interesting finding is that geo-location related signals and time-sense signals are both important, revealing that people prefer location-relevant and fresh queries.

Table 12: Feature Weights Learned by TDCM

PC	w_H	TypingSpeed	IsWordBoundary	CurrPosition	QueryIntent
		-0.86	0.55	0.32	-0.20
	w_D	Position@1	Position@2	Position@3	
		0.397	0.314	0.152	
	w_R	MPC	QryHistFreq	GeoSense	TimeSense
		1.790	0.973	0.962	1.115
iPhone 5	w_H	TypingSpeed	IsWordBoundary	CurrPosition	QueryIntent
		-0.57	0.50	0.20	-0.28
	w_D	Position@1	Position@2	Position@3	
		0.3782	0.334	0.171	
	w_R	MPC	QryHistFreq	GeoSense	TimeSense
		4.139	3.918	0.947	1.595

6. CONCLUSION AND FUTURE WORK

The QAC problem is under-explored because of the lack of suitable query logs. In this paper we have collected a large set of QAC sessions with fine-grained user interaction information, which enables us to analyze and model the user behavior in QAC. Based on two key observations, namely the horizontal skipping bias and vertical examination bias, we have proposed a novel Two-Dimensional Click Model for modeling the QAC process. Extensive experiments on our datasets demonstrated that our TDCM model can accurately explain the user behaviors in QAC. The resulting relevance model significantly outperforms all existing click models. In addition, user behavior information learned by our model can be incorporated into other learning-based methods to further improve their performance. Using our model, we also discover some interesting user behaviors on PC and mobile devices.

As the first click model for QAC, our TDCM model could be extended in several ways in the future. For example, the independent assumption between different columns can be relaxed to capture multi-column interdependency. In addition, more complex models can replace the D model to better explain the vertical position bias.

7. ACKNOWLEDGMENTS

We would like to thank Georges Dupret from Yahoo! Labs and Xiaolong Wang from the University of Illinois at Urbana-Champaign for their discussion and advice.

8. REFERENCES

[1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR'06*

[2] M. Arias, J. M. Cantera, J. Vegas, P. de la Fuente, J. C. Alonso, G. G. Bernardo, C. Llamas, and Á. Zubizarreta. Context-based personalization for mobile web search. In *PersDB*

[3] Z. Bar-Yossef and N. Kraus. Context-sensitive query auto-completion. In *WWW'11*

[4] E. Kharitonov et al. User Model-based Metrics for Offline Query Suggestion Evaluation. In *SIGIR'13*

[5] H. Bast and I. Weber. Type less, find more: fast autocompletion search with a succinct index. In *SIGIR'06*

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

[7] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *WSDM'08*

[8] H. Duan and B.-J. P. Hsu. Online spelling correction for query completion. In *WWW'11*

[9] G. E. Dupret and B. Piwowarski. A user browsing model to predict search engine click data from past observations. In *SIGIR'08*

[10] L. A. Granka, T. Joachims, and G. Gay. Eye-tracking analysis of user behavior in www search. In *SIGIR'04*

[11] B.-J. P. Hsu and G. Ottaviano. Space-efficient data structures for top-k completion. In *WWW'13*

[12] S. Ji, G. Li, C. Li, and J. Feng. Efficient interactive fuzzy keyword search. In *WWW'09*

[13] T. Joachims. Optimizing search engines using clickthrough data. In *KDD'02*

[14] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *WWW'06*

[15] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *WSDM'11*

[16] C. Liu, F. Guo, and C. Faloutsos. Bayesian browsing model: Exact inference of document relevance from petabyte-scale data. *ACM Trans. Knowl. Discov. Data*, 2010.

[17] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *KDD'05*

[18] M. Richardson and E. Dominowska and R. Ragno. Predicting clicks: Estimating the click-through rate for new ads. In *WWW'07*

[19] M. Sahami and T. D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *WWW'06*

[20] M. Shokouhi. Learning to personalize query auto-completion. In *SIGIR'13* 2013.

[21] H. Wang, C. Zhai, A. Dong, and Y. Chang. Content-aware click modeling. In *WWW'13*

[22] R. W. White and G. Marchionini. Examining the effectiveness of real-time query expansion. *Information Processing & Management*

[23] Y. Zhang, W. Chen, D. Wang, and Q. Yang. User-click modeling for understanding and predicting search-behavior. In *KDD'11*

[24] Z. A. Zhu, W. Chen, T. Minka, C. Zhu, and Z. Chen. A novel click model and its applications to online advertising. In *WSDM'10*