UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

A context-aware time model for web search

Borisov, A.; Markov, I.; de Rijke, M.; Serdyukov, P.

DOI
10.1145/2911451.2911504
Publication date
2016
Document Version
Author accepted manuscript
Published in
SIGIR'16

Link to publication

Citation for published version (APA):
Borisov, A., Markov, I., de Rijke, M., & Serdyukov, P. (2016). A context-aware time model for web search. In *SIGIR'16: the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval: Pisa, Italy , July 17-21, 2016* (pp. 205-214). Association for Computing Machinery. https://doi.org/10.1145/2911451.2911504

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)

Download date:08 Oct 2026

A Context-aware Time Model for Web Search

Alexey Borisov[†, ‡] alborisov@yandex-team.ru
Ilya Markov[‡] i.markov@uva.nl
Maarten de Rijke[‡] derijke@uva.nl
Pavel Serdyukov[†] pavser@yandex-team.ru

[†] Yandex, Moscow, Russia
[‡] University of Amsterdam, Amsterdam, The Netherlands

ABSTRACT

In web search, information about times between user actions has been shown to be a good indicator of users' satisfaction with the search results. Existing work uses the mean values of the observed times, or fits probability distributions to the observed times. This implies a *context-independence* assumption that the time elapsed between a pair of user actions does not depend on the context, in which the first action takes place. We validate this assumption using logs of a commercial web search engine and discover that it does not always hold. For between 37% to 80% of query-result pairs, depending on the number of observations, the distributions of click dwell times have statistically significant differences in query sessions for which a given result (i) is the first item to be clicked and (ii) is not the first. To account for this *context bias* effect, we propose a *context-aware time model* (CATM). The CATM allows us(i) to predict times between user actions in contexts, in which these actions were not observed, and (ii) to compute context-independent estimates of the times by predicting them in predefined contexts. Our experimental results show that the CATM provides better means than existing methods to predict and interpret times between user actions.

Keywords

Time modeling; User behavior; Web search

1. INTRODUCTION

Search engine logs provide a rich source of information about user browsing behavior in web search. Recently, many models have been proposed to explain and predict user clicks on search engine results [13]. While click events are the most widely logged form of user interaction, there are also other behavioral signals that need to be understood, interpreted and modeled, and that can be used for ranking or prediction purposes.

We focus on behavioral signals based on times between user actions. The ability to accurately predict(i) click dwell time (i.e., time spent by a user on the landing page of a search result), and (ii) times from submission of a query to the first/last click on the results and to the next query submission (if none of the results will be clicked) allows us to optimize search engines for constructing result pages and suggesting query reformulations that minimize time it takes users to satisfy their information needs.

Existing work shows that times elapsed between user actions provide means to measure user satisfaction at the result level [18, 30], session level [18, 20] and system level [10, 39]. To interpret times elapsed between user actions, existing work uses mean values of the observed times [1–3, 8, 10, 20–22, 32, 33, 36, 39], or fits probability distributions to the observed times [30, 31]. This implies a *context-independence assumption* that the time elapsed between a pair of user actions does not depend on the context in which the first action takes place.

We validate this assumption by comparing the distributions of times associated with the same user action (i.e., submitting a given query or clicking on a given result presented for a given query), but preceded by different sequences of user interactions with the search engine. We find statistically significant differences between these distributions, which we explain by a *context bias effect*, not previously reported in the literature.

To account for the context bias effect, we propose a *context-aware time model* (CATM) that allows us to predict probability distributions of times between two user actions in a given context (which we represent by a sequence of previous user interactions with the search engine). The CATM can be used(i) to predict times between user actions in contexts, in which these user actions were not observed, and (ii) to compute context-independent estimates of the times by predicting them in predefined contexts. The *context-dependent* predictions can be used for personalized ranking and personalized query suggestion. The *context-independent* predictions can be used for predicting result relevance and in other tasks that use historical times between user actions.

We evaluate the CATM on four temporal prediction tasks (i.e., tasks in which we predict the time elapsed between two user actions), and find that the CATM outperforms the standard time modeling approach that fits probability distributions to the times observed for the first user action. We test the CATM's context-independent predictions of times between consecutive clicks on a ranking task (i.e., to rank a set of results by their relevance to a query), and find that the produced rankings result in better performance in terms of relevance than the rankings produced by the standard time modeling approaches.

In summary, we make the following contributions in our work.

C1 We introduce the notion of context bias in times elapsed between user actions, which has not previously been reported in the literature, and use logs of a commercial search engine to provide empirical evidence for it.

C2 We propose a context-aware time model. Through the use of contextual information (and, in particular, previous user in-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIGIR '16, July 17 - 21, 2016, Pisa, Italy

© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ISBN 978-1-4503-4069-4/16/07...$15.00

DOI: http://dx.doi.org/10.1145/2911451.2911504

teractions with a search engine), it provides better means for predicting and interpreting times between user actions compared to existing time modeling techniques.

The rest of the paper is organized as follows. In §2 we specify the considered temporal prediction tasks. In §3 we propose the context-aware time model. In §4 we detail our experimental setup and in §5 we present the outcomes of our experiments. We discuss related work in §6 and conclude in §7.

2. TEMPORAL PREDICTION TASKS

We describe and motivate temporal prediction tasks that we consider in our work. In each task we aim to predict the time elapsed between a pair of user actions. The user actions can be of several types:submission of a query, click on a search result, click on an ad banner, scroll through search results, zoom in on a search result (in mobile search), etc. Here, we focus on user actions of the first two types: submission of a query (Q-action) and click on a search result (C-action), because these user actions occur in all search systems.

To define a temporal prediction task, we need to specify a set of action pairs between which we aim to predict times. When choosing these pairs, we follow two principles.

1. We select pairs of user actions for which the time elapsed between them is indicative of user satisfaction. The ability to predict times, that help to anticipate user satisfaction with search results, allows us to construct result pages that maximize user satisfaction with the search.

2. We select pairs of user actions for which the times elapsed between them have a similar nature (i.e., similar human interpretations). This is important, because times that have different human interpretations are likely to be unequally useful in practical applications, and can also be used in different scenarios. Thus, we want to be able to measure the prediction performance of each group of times separately.

We define four temporal prediction tasks: *time-to-first-click*, *time-between-clicks*, *time-to-last-click*, and *time-from-abandoned-query*. The pairs of user actions used in each task are shown in Figure 1. Below, we describe them in more detail and provide motivations

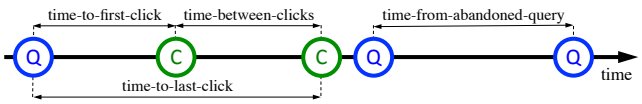

Figure 1: Times between pairs of user actions used in the considered temporal prediction tasks.

for our focus on these specific tasks.

Time-to-first-click is the time between a submission of a query and the first click on search engine results (undefined if there were no clicks on search results). Existing work shows that this time reflects the quality of result presentation: the less time it takes a user to find an attractive result, the better in terms of user experience the search engine result page (SERP) is [38]. The ability to predict time-to-first-click allows us to construct SERPs that are more intuitive to users, i.e., require less time/effort to find relevant results. Practically speaking, the predicted time can be used in any application that uses the average time-to-first-click observed in search engine logs as its more precise alternative. Existing work uses the average time-to-first-click observed for a query-result pair as a feature to predict search task difficulty [3], search goal success [20, 21], urgent information needs (e.g., how to stem a severe bleeding) [35]; and the average time-to-first-click observed for a user to cluster users based on their SERP examination strategies [7].

Time-between-clicks is the time between two consecutive clicks on search engine results (undefined for the last click on search results). We use this time as a proxy for *click dwell time* (i.e., the time spent by a user on the landing page of a search engine result), which has been shown to be a good indicator of click satisfaction [30]. The ability to predict click dwell time allows us to construct SERPs that yield more satisfied clicks, e.g., by using the predicted time-between-clicks as a feature for the search result ranker. Agichtein et al. [1, 2] use average click dwell times to improve result relevance. The Yahoo! Learning to Rank dataset [8] uses them as features for ranking. The predicted dwell times can also be utilized in most other applications that make use of the average click dwell times observed in search engine logs, which include prediction of click satisfaction [30], result usefulness [32], search task difficulty [3, 33], search goal success [20, 21], struggling vs. exploring behavior [22, 36].

Time-to-last-click is the time between a submission of a query and the last click on search engine results (undefined if there were no clicks on search results). Radlinski et al. [38] show that this time reflects the quality of search engine results: the less time it takes a user to find a relevant result, the better in terms of user experience the search engine results are. The ability to predict time-to-last-click allows us to suggest queries that require less time (effort) to satisfy user's information needs. The predicted times can be used as features for query suggestion and for other applications such as prediction of search task difficulty and search goal success (playing a similar role as the average time-to-first-click).

Time-from-abandoned-query is the time from a submission of an *abandoned query* (i.e., a query with no clicks) to the next query submission. Song et al. [41] show that this time reflects the quality of search engine results in query sessions with no clicks (thus, it is complementary to time-to-last-click, which is defined in query sessions with at least one click). A short time interval indicates user dissatisfaction (negative abandonment): the user could quickly see the problem with the presented results and it took them little time to reformulate the query. A large time interval indicates user satisfaction (positive abandonment): the user most likely could find the answer on the SERP, and it took them some time to come up with the next query. The ability to predict time-from-abandoned-query allows us to suggest queries with high chances of positive abandonment and to avoid suggesting queries that are likely to be quickly reformulated. Similar to time-to-last-click, the predicted time-from-abandoned-query can be used for query suggestion, prediction of task difficulty, search goal success, etc.

There are several other times-between-user-actions that have previously been considered in the literature, but that we do not seek to predict in this work for the reasons discussed below.

Time-between-last-click-and-next-query is the time between the last click on search engine results presented for one query and the submission of the next query. Some previous work uses both this time and the time-between-clicks as a proxy for click dwell time (called *server-side click dwell time*) [29, 30]. While both times approximate click dwell time, there are important differences between them: time-between-clicks is a sum of the actual click dwell time and the time to choose the next result to click (which is typically small and exhibits low variance), while time-between-last-click-and-next-query is a sum of the actual click dwell time and the time to formulate the next query (which is sometimes large, and exhibits high variance). Because of these differences in nature, we do not want to model time-between-clicks and time-between-

last-click-and-next-query together (see principle 2). We do not create a temporal prediction task for time-between-last-click-and-next-query, as it is very similar to the time-between-clicks task, but the observed times are likely to be more noisy.

Time-to-first-satisfied-click is the time between a submission of a query and the first click classified as satisfied. This time can be seen as a generalization of time-to-first-click and time-to-last-click: if we classify all clicks as satisfied, time-to-first-satisfied-click reduces to time-to-first-click; if we classify only the last click as satisfied, time-to-first-satisfied-click reduces to time-to-last-click. Most work defines satisfied clicks as clicks with dwell times larger than a predefined threshold (e.g., 30 seconds) [18]. Kim et al. [30] propose a method to adjust this threshold individually for each search result. We do not create temporal prediction tasks for all possible definitions of satisfied clicks, because we believe that the temporal prediction tasks for time-to-first-click and time-to-last-click capture the most important phenomena.

3. METHOD

Now that we have specified the temporal prediction tasks, we describe our approach to modeling times between user actions. In §3.1 we formalize the problem and introduce the notation. In §3.2 we describe two frameworks for modeling times between user actions. In §3.3 we describe the context-aware time model (CATM).

3.1 Problem statement

Let the sequence $(a_1, t_1), \ldots, (a_n, t_n)$ be a search history, which consists of user actions $a_1, \ldots, a_n$ and their timestamps $t_1, \ldots, t_n$. We aim to design a time model for predicting times $\tau_{i \to j} = t_j - t_i$ elapsed from the action a_i to the action a_j. We consider a probabilistic formulation of this problem, where the time $\tau_{i \to j}$ is a random variable with a probability density function $f(x; \theta)$, where θ denotes a set of parameters of the probability distribution.

To describe a probabilistic time model we need to specifya probability density function $f(x; \theta)$ and an algorithm to compute its parameters θ. Common choices for $f(x; \theta)$ are the exponential, gamma and Weibull probability density functions [30, 31]; see Table 1 for their parameterizations. Below, we discuss the frameworks for computing the function parameters θ.

Table 1: Probability density functions and their parameters.

Distribution	PDF $f(x; \theta)$	Parameters θ
Exponential	ae^{-ax}	$a > 0$
Gamma	$\frac{x^{a-1}e^{-x/b}}{b^a \Gamma(a)}$	$a, b > 0$
Weibull	$\frac{a}{b}\left(\frac{x}{b}\right)^{a-1} e^{-(x/b)^a}$	$a, b > 0$

3.2 Time modeling frameworks

We describe two frameworks for modeling times between user actions. The first framework, called *basic time modeling framework*, makes the context-independence assumption that time between a pair of user actions depends solely on the first action ID, i.e., for submitting a query (Q-action), a unique identifier of the query string, and for clicking on a search result (C-action), an identifier of the query-result pair. The second framework, called *context-aware time modeling framework*, does not make the context-independence assumption and assumes that time between a pair of user actions depends on the first action ID and on the context in which the first action takes place.

3.2.1 Basic time modeling framework

Time models operating within the basic time modeling framework make the context-independence assumption that time $\tau_{i \to j}$ depends solely on the first action ID. We formalize this as $\tau_{i \to j} \sim f(x; \theta = \Theta(a_i))$, where Θ denotes a mapping from the space of actions $\mathcal{A}$ to the space of parameters of the probability density function $f(x; \theta)$.[1] We learn the mapping Θ by maximizing the likelihood of the observed times $\tau_{i \to j}$. The corresponding optimization problem can be formalized as follows:

$$\text{DATA} = \{(a_i, \tau_i) \mid a_i \in \mathcal{A}, \tau_i \in [0; \infty)\}_{i=1}^{N} \quad (1)$$

$$\tau_i \sim f(x; \theta = \Theta(a_i)) \quad (2)$$

$$\widehat{\Theta} = \arg\max_{\Theta} \prod_{i=1}^{N} f(\tau_i; \theta = \Theta(a_i)). \quad (3)$$

Here, we write N to denote the total number of observations, $\widehat{\Theta}$ to denote the solution of this optimization problem, and we also change τ indices to simplify the notation.

Without any constraints on the mapping $\widehat{\Theta}$, this optimization problem can be further decomposed into a set of per action ID *maximum likelihood estimation* (MLE) problems (we apply the logarithm function, which is monotonic and therefore does not change the result of the arg max operator):

$$\widehat{\Theta}(a) = \arg\max_{\theta} \sum_{i=1}^{N} \mathbf{I}_a(a_i) \log f(\tau_i; \theta), \quad (4)$$

where $\mathbf{I}_a(x)$ denotes the indicator function, i.e., 1 if $x = a$ and 0 otherwise. For the exponential distribution, the MLE of its parameter (Table 1) is the inverse of the mean of the observed times. For the gamma and Weibull distributions, there is no closed form solution for the MLE problem. But it is possible to express one of their parameters as a function of the other one and $\tau_1, \ldots, \tau_N$, and to reduce the MLE problem to minimization of a scalar function.

Existing work on modeling click dwell times [30, 31] operates within the basic time modeling framework and follows the described approach to estimate their parameters.

3.2.2 Context-aware time modeling framework

We argue that considering only the ID of the first action is not enough to accurately model the time $\tau_{i \to j}$, and propose, in addition to using the ID of a_i, to use the context c_i in which the action a_i takes place. We formalize this as $\tau_i \sim f(x; \theta = \Theta(a_i, c_i))$, where Θ denotes a mapping from the Cartesian product of the space of actions $\mathcal{A}$ and the space of contexts $\mathcal{C}$ to the parameters of the probability density function $f(x; \theta)$.

Similar to Eqs. 1, 2 and 3, the optimization problem in the context-aware time modeling framework can be formalized as follows:

$$\text{DATA} = \{(a_i, c_i, \tau_i) \mid a_i \in \mathcal{A}, c_i \in \mathcal{C}, \tau_i \in [0; \infty)\}_{i=1}^{N} \quad (5)$$

$$\tau_i \sim f(x; \theta = \Theta(a_i, c_i)) \quad (6)$$

$$\widehat{\Theta} = \arg\max_{\Theta} \prod_{i=1}^{N} f(\tau_i; \theta = \Theta(a_i, c_i)). \quad (7)$$

However, without any additional constraints on the mapping $\widehat{\Theta}$, the solution of this optimization problem will have very poor generalization due to the sparsity of action-context pairs (a_i, c_i). Let us illustrate this using a few simple definitions of the context c_i:(i) number of previously clicked results, (ii) positions of previously clicked

[1] Where no confusion can arise, we write a_i to denote both the action performed by a user and the action ID.

results, (iii) positions of previously clicked results and times between previous clicks. For these definitions of the context c_i, we list in Table 2 the number of unique action-context pairs (a_i, c_i) for which we need to estimate parameters θ. We see that the number of

Table 2: Number of unique action-context pairs (a_i, c_i) for different definitions of the context c_i; N denotes the number of results on a SERP; T denotes the number of time intervals we consider.

Context c_i	# of unique (a_i, c_i)
None	$\mathcal{O}(\|\mathcal{A}\|)$
Number of previously clicked results	$\mathcal{O}(\|\mathcal{A}\| \times N)$
Positions of previously clicked results	$\mathcal{O}(\|\mathcal{A}\| \times N^N)$
Positions of previously clicked results and times between previous clicks	$\mathcal{O}(\|\mathcal{A}\| \times (N \times T)^N)$

$\widehat{\Theta}$ parameters becomes too large to be able to estimate them using the amount of log data that can be generated by search engines.

The natural way to handle this problem is to constrain the family of mappings $\widehat{\Theta}$ in Eq. 7. In §3.3, we describe the *context-aware time model* that operates within the proposed context-aware time modeling framework and provides a way to constrain the family of mappings $\widehat{\Theta}$.

3.3 Context-aware time model

In §3.2 we introduced the context-aware time modeling framework that models times between two user actions using the ID of the first action and the context in which it takes place. To construct a model that operates within this framework, we need to specify(i) the probability density function $f(x; \theta)$, (ii) representation of the context c_i, and (iii) constraints on the mapping $\widehat{\Theta}$ (Eq. 7). We also need to describe the solution of the optimization problem (5)–(7) for the chosen constraints.

3.3.1 *Probability density function* $f(x; \theta)$

We follow previous work [30, 31] and use the exponential, gamma and Weibull probability density functions.

3.3.2 *Context* c_i

We represent the context c_i, in which the action a_i takes place, by a sequence of user interactions with a search engine that preceded the action a_i. We list the attributes that we use to describe the user interactions in Table 3.

The first two attributes from the general section distinguish between Q- and C-actions. The third and fourth attributes inform us about the actual times observed between the previous actions and the average times between them. This information is useful to account for the user's reading speed, persistence, etc.

The first attribute from the Q-action section informs us about whether the query is the first in a search session or not. It is useful to distinguish between the cases(i) when a user has just started browsing, and (ii) when a user has been browsing for some time. In §5.1 we show that these cases may result in different probability distributions of times from submitting a query to the first click on the results, last click on the results and next query submission (if none of the results will be clicked). The second, third and fourth attributes from the Q-action section inform us about the query's similarity with the previous query.

The attributes from the C-action section inform us about the position of the clicked result. Existing work shows that the position of a result influences users' trust in the result's usefulness, which affects their decisions whether to click or not [13]. We allow for the possibility that the result's position might affect the time from clicking on the result to other actions.

Table 3: Attributes we use to describe user interactions with a search engine.

General	
Is Q-action	(0: no, 1: yes)
Is C-action	(0: no, 1: yes)
$\log(1 +$ observed time since previous action$)$	(0: undefined)
$\log(1 +$ average time since previous action$)$	(0: undefined)
Q-action	
Is new search session	(0: no, 1: yes)
Number of terms in issued query	(0: undefined)
BM25 (issued query, previous query)	(0: undefined)
BM25 (previous query, issued query)	(0: undefined)
C-action	
Is click on the 1st position	(0: no, 1: yes)
...	...
Is click on the 10th position	(0: no, 1: yes)

3.3.3 *Constraints on the mapping* $\widehat{\Theta}(a_i, c_i)$

Without any constraints on the mapping $\widehat{\Theta}$, the solution of the optimization problem (5)–(7) will have very poor generalization due to a very large number of action-context pairs (see §3.2.2). To deal with this problem we impose additional constraints on the family of mappings $\widehat{\Theta}$. In particular, we limit the family of possible mappings $\widehat{\Theta}(a_i, c_i)$ to the ones that can be decomposed into the mappings $\widehat{\Theta}_A(a_i)$ and $\widehat{\Theta}_C(c_i)$ defined on the space of actions $\mathcal{A}$ and the space of contexts $\mathcal{C}$, respectively. Not only does this reduce the number of effective $\widehat{\Theta}$ parameters from $\mathcal{O}(|\mathcal{A}| \times |\mathcal{C}|)$ to $\mathcal{O}(|\mathcal{A}| + |\mathcal{C}|)$, but it also separates action-specific and context-specific parameters. The latter allows us(i) to estimate times between user actions in contexts in which these user action were not observed; and (ii) to estimate context-independent parameters of times between actions, which can be used for ranking and other prediction tasks that use historical times between user actions.

Now that we have described the constraints on the mapping $\widehat{\Theta}$ at the functional level, we need to explainthe mapping $\widehat{\Theta}_A$, the mapping $\widehat{\Theta}_C$, and the decomposition of the mapping $\widehat{\Theta}$ into the mappings $\widehat{\Theta}_A$ and $\widehat{\Theta}_C$ in more detail.

Mapping $\widehat{\Theta}_A$. The mapping $\widehat{\Theta}_A$ is an equivalent of the mapping Θ in the basic time modeling framework. As in that setting, we treat it as a table of per action ID parameters, which we call *context-independent* parameters of the function $f(x; \theta)$. Thus, the number of parameters of the mapping $\widehat{\Theta}_A$ is $\mathcal{O}(|\mathcal{A}|)$.

Mapping $\widehat{\Theta}_C$. The mapping $\widehat{\Theta}_C$ describes how to adjust context-independent parameters computed by the mapping $\widehat{\Theta}_A$ to account for the context in which the first action takes place. We want this mapping to have $\mathcal{O}(|\mathcal{A}|)$ parameters so as not to increase the asymptotic number of parameters compared to time models that operate within the basic time modeling framework. Implementing the mapping $\widehat{\Theta}_C$ as a table of per context parameters will lead to a total of $\mathcal{O}(|\mathcal{C}|)$ parameters, which is likely to dominate $\mathcal{O}(|\mathcal{A}|)$. Thus, we decide to implement the mapping $\widehat{\Theta}_C$ using a machine learning algorithm. In particular, we implement it as a *recurrent neural network* (RNN) with *long short-term memory* (LSTM) cells [23]. We choose this architecture, becausewe represent the context c_i

as a sequence of numerical attributes (see §3.3.2); and for tasks that involve processing sequential data, RNNs have demostrated strong performance on a wide range of tasks. See examples in language modeling [34], speech recognition [19] and machine translation [42]. By using the RNN we reduce the number of $\widehat{\Theta}$ parameters to $\mathcal{O}(N \times M)$, where N denotes the number of attributes we use to represent user interactions with the search (18 in our work) and M denotes the maximum number of units in the RNN layers (256 in our work). As $N \times M \ll |\mathcal{A}|$, we satisfy the $\mathcal{O}(|\mathcal{A}|)$ requirement.

Decomposition of $\widehat{\Theta}$ into $\widehat{\Theta}_A$ and $\widehat{\Theta}_C$. A decomposition of $\widehat{\Theta}$ into $\widehat{\Theta}_A$ and $\widehat{\Theta}_C$ describes how to compute the parameters θ of the probility density function $f(x;\theta)$ from the action-specific parameters $\widehat{\Theta}_A(a_i)$ and the context-specific parameters $\widehat{\Theta}_C(c_i)$. We want the action-specific parameters $\widehat{\Theta}_A(a_i)$ to be context-independent estimates of parameters of the probability density function $f(x;\theta)$ for the action ID a_i (see the discussion above), and the context-specific parameters $\widehat{\Theta}_C(c_i)$ to be coefficients that inform us about how to adjust $\widehat{\Theta}_A(a_i)$ to account for the context c_i. One natural way to achieve it is(i) to have two context-specific parameters α and β for each action-specific parameter θ_i, and (ii) to apply a linear transformation $g(\theta_i) = \alpha\theta_i + \beta$ to compute the context-dependent parameters of the probability density function. We formalize it as:

$$\widehat{\Theta}(a_i, c_i) = \widehat{\Theta}_A(a_i) \circ \widehat{\Theta}^0_C(c_i) + \widehat{\Theta}^1_C(c_i). \tag{8}$$

Here, $\widehat{\Theta}^0_C(c_i)$ and $\widehat{\Theta}^1_C(c_i)$ constitute the first and the last halves of the vector $\widehat{\Theta}_C(c_i)$, which contains two times more elements than $\widehat{\Theta}_A(a_i)$, and the symbol $\circ$ denotes the element-wise product.

3.3.4 *Learning the mappings $\widehat{\Theta}_A(a_i)$ and $\widehat{\Theta}_C(c_i)$*

Now, we describe an approximate solution of the optimization problem (5)–(7) with the constraints on the mapping $\widehat{\Theta}$ defined in §3.3.3. This problem does not have closed form solutions for the mappings $\widehat{\Theta}_A$ and $\widehat{\Theta}_C$. Therefore, we propose an iterative algorithm to compute them (see Algorithm 1). We use $\widehat{\Theta}(a_i, c_i) = \mathcal{F}(\widehat{\Theta}_A(a_i), \widehat{\Theta}_C(c_i))$ as a generalized version of Eq. 8.

The algorithm initializes the mapping $\widehat{\Theta}_A$ with the solution of the optimization problem (1)–(3) in the basic time modeling framework (line 1). Then it alternates between learning $\widehat{\Theta}_C$ while fixing $\widehat{\Theta}_A$ (line 3) and learning $\widehat{\Theta}_A$ while fixing $\widehat{\Theta}_C$ (line 4) until convergence. In our experiments the process converges after 5–10 iterations (i.e., passes through the loop). It is easy to show that the likelihood does not decrease between the iterations.

Algorithm 1 Learn $\widehat{\Theta}_A(\cdot), \widehat{\Theta}_C(\cdot)$

1: $\widehat{\Theta}_A \leftarrow \arg\max_{\Theta} \prod_{i=1}^{N} f(\tau_i; \theta = \Theta(a_i))$
2: **while** not $\widehat{\Theta}_A, \widehat{\Theta}_C$ converged **do**
3: $\quad \widehat{\Theta}_C \leftarrow \arg\max_{\Theta_C} \prod_{i=1}^{N} f(\tau_i; \theta = \mathcal{F}(\widehat{\Theta}_A(a_i), \Theta_C(c_i))$
4: $\quad \widehat{\Theta}_A \leftarrow \arg\max_{\Theta_A} \prod_{i=1}^{N} f(\tau_i; \theta = \mathcal{F}(\Theta_A(a_i), \widehat{\Theta}_C(c_i))$
5: **end while**
6: **return** $\widehat{\Theta}_A, \widehat{\Theta}_C$

To find the best $\widehat{\Theta}_C$ while fixing $\widehat{\Theta}_A$ (Algorithm 1, line 3), we use the *stochastic gradient descent* (SGD) algorithm with mini-batches, because this is the most widely used algorithm for training neural networks [4]. The learning rates for each parameter are adjusted according to the ADADELTA algorithm [44] (we use the default values of $\epsilon = 10^{-6}$ and $\rho = 0.95$). We also use the gradient clipping technique [37] to alleviate the exploding gradient problem [5] (we set the value of the threshold to 1).

To find the best $\widehat{\Theta}_A$ while fixing $\widehat{\Theta}_C$ (Algorithm 1, line 4), we decompose the optimization problem into a set of per action ID MLE problems (similar to Eq. 4):

$$\widehat{\Theta}_A(a) = \arg\max_{\theta} \sum_{i=1}^{N} \mathbf{I}_a(a_i) \log f(\tau_i; \theta = \mathcal{F}(\theta, \widehat{\Theta}_C(c_i)).$$

We solve these MLE problems using the *limited memory BFGS with bound constraints (L-BFGS-B)* algorithm [46]. We use bound constraints to ensure that the distribution parameters θ take admissible values (e.g., the values of the parameters of the exponential, gamma and Weibull distributions need to be positive).

To summarize, we predict a probability distribution over the time elapsed between a pair of user actions. Unlike existing methods, which rely solely on the first action ID, CATM also considers the context in which the first action takes place. CATM represents this context as a sequence of user interactions with a search engine that precede the first action, and employs a recurrent neural network that learns how to adjust the first action ID parameters using this context representation. CATM can be used(i) to predict the time elapsed between a pair of user actions in a context, in which these actions have not been previously observed, and (ii) to obtain a context-independent estimate of the time between two user actions by predicting it in a predefined context.

4. EXPERIMENTAL SETUP

In this section we describe our experimental setup. We start with the research questions that we seek to answer (§4.1). Then we describe the datasets that we use (§4.2) and the evaluation methodology that we follow (§4.3). Finally, we describe the experiments that we conduct to answer our research questions (§4.4).

4.1 Research questions

We address the following research questions: (**RQ1**) Can we observe the context bias effect? More precisely, can we observe a difference in time probability distributions for different contexts in which the first action takes place? (**RQ2**) Do the context-aware time models, which besides the first action ID make use of its context, provide better means to explain times between user actions than the basic time models, which make the context-independence assumption and rely solely on the first action ID? (**RQ3**) Do the context-independent predictions of the CATMs, trained on the time-between-clicks task, provide better means to rank search results than existing methods based on the observed times between clicks?

4.2 Dataset

To construct datasets for the temporal prediction tasks described in §2, we collected three months of log data from a commercial web search engine. We use the first two months of the log data to train time models and the last month to evaluate their prediction performance. In each dataset, we filter out times associated with actions IDs that occur less than 25 times in the training set. For time-to-first-click and time-from-abandoned-query, we also filter out times larger than 1 minute; for time-to-last-click and time-between-clicks, we filter out times larger than 5 minutes (this complies with actual times reported in [38]). The number of observations in the resulted datasets are shown in Table 4.

Table 4: Number of observations in the datasets for each temporal prediction task.

Time between actions	Number of observations
Time-to-first-click	30,747,733
Time-between-clicks	6,317,834
Time-to-last-click	30,446,973
Time-from-abandoned-query	11,523,351

To compare the performance of ranking models based on time-between-clicks, we also collected relevance labels for 50,137 query-result pairs that occur in our time-between-clicks dataset. The relevance labels were assigned by trained judges on a scale from 0 to 4, with 0 = bad, 1 = fair, 2 = good, 3 = excellent, 4 = perfect.

4.3 Evaluation methodology

We evaluate the performance of time models using the *log-likelihood* and the *root mean squared error* (RMSE) metrics. The log-likelihood metric shows how well a time model explains the observed times between user actions. We report the logarithm of the likelihood function, averaged over all observations in the test set. Larger values of the metric indicate better performance. The RMSE metric shows the average difference between the expected values of the time probability distributions predicted by a model and the observed times-between-actions. Lower values of the metric indicate better performance. We also evaluate the time models for predicting time-between-clicks on a ranking task (i.e., to rank a set of results by their relevance to a query). We use the NDCG metric [26], and report its scores at truncation levels 1, 3, 5 and 10. Larger values of the metric indicate better performance.

We perform significance testing in terms of all metrics using a paired t-test; the differences are considered statistically significant for p-values lower than 0.05.

4.4 Experiments

Experiment 1. To answer RQ1, for each action ID we split the observed times in two *context groups*, which correspond to different sets of previous user interactions, and run the two-sample two-sided Kolmogorov-Smirnov (KS) test [14] to determine whether the observed times were drawn from the same distribution. The null hypothesis states that the observed times were drawn from the same distribution, which means that there is no context bias effect. Rejecting it, at least for some action IDs, will prove the existence of the context bias effect for these action IDs. Not being able to reject it might happen for several reasons. First, the context bias effect may not appear for certain types of queries (e.g., navigational queries) and results (e.g., irrelevant results). Second, we may not have enough data to reject the null hypothesis. Third, we may not have chosen the best context groups.

When choosing a rule to split times-between-actions in two context groups based on the context, we give preference to easily interpretable and easily reproducible ones. For the time-to-first-click, time-to-last-click and time-from-abandoned-query, we split the observed times based on whether the query is the first in the search task or not. We say that a query is the first in the search task if it does not share terms with the previous query in the search session, or if it is the first query in the search session. For the time-between-clicks, we split the observed times based on whether the clicked result is the first item to be clicked on SERP or not.

We use the KS test, because the more popular t-test is not applicable in our setting: it requires the tested samples to be drawn from a normal distribution, which is not the case for time observations. An alternative to the KS test could be the Mann-Whitney U test [14], but following previous work [31] we use the KS test.

Without a sufficient number of observations in both context groups, the KS test would be unable to detect a difference between samples even if one exists. For this reason, we apply the KS test only to actions IDs, for which there are, at least, N observations in both context groups. Using a large value of N improves test sensitivity (increases the number of action IDs for which the null hypothesis can be rejected), but reduces the number of action IDs that we use in our experiment. Thus, we run our experiment for different values of N. In particular, we use N in the range $[25, 200]$. Table 5 shows the number of action IDs with, at least, N observations in both context groups for $N = \{25, 50, 100, 200\}$.

Table 5: Number of action IDs with at least $N = \{25, 50, 100, 200\}$ observations in each context group for different times-between-actions.

	N			
Time between actions	25	50	100	200
Time-to-first-click	60,284	24,368	7894	1545
Time-between-clicks	18,954	8335	3289	1288
Time-to-last-click	59,367	23,956	7756	1500
Time-from-abandoned-query	22,190	9180	2980	569

Experiment 2. To answer RQ2, we compare the performance in terms of log-likelihood and RMSE of the context-aware time models against the basic time models on the four temporal prediction tasks described in §2.

Experiment 3. To answer RQ3, we compare the performance in terms of NDCG of rankings models based on(i) the average values of the observed times between clicks, (ii) the expected values of the probability distributions fitted to the observed times between clicks, and (iii) the expected values of the context-independent probability distributions predicted by the CATMs trained on the time-between-clicks task. To predict the context-independent probability distributions we use the following fixed context:(i) the query is the first in the search session; (ii) the result is presented on the top position and is the first item to be clicked; (iii) the time elapsed between the query submission and the click on the result is 4 seconds (the median of the times-to-first-click observed in our dataset).

5. RESULTS

In this section we present the results of the experiments described in §4.4 and provide answers to the research questions stated in §4.1.

5.1 Experiment 1

The results of Experiment 1 are given in Figure 2. The figure shows the percentage of action IDs in each temporal prediction task, for which the Kolmogorov-Smirnov test rejected the null hypothesis ($p < 0.05$) in favor of the alternative hypothesis, which states that the times-between-actions in the chosen context groups were drawn from different probability distributions. Equivalently, the alternative hypothesis states that the context bias effect exists.

RQ1. Figure 2 shows that for each time-between-actions there is more than 5% action IDs, for which the null hypothesis is rejected. This proves the existence of the context bias effect. Indeed, if times-between-actions did not depend on previous user interactions (null hypothesis was true), the number of action IDs for which the null hypothesis would be rejected had to be 5% (significance level) of the total number of the tested action IDs.

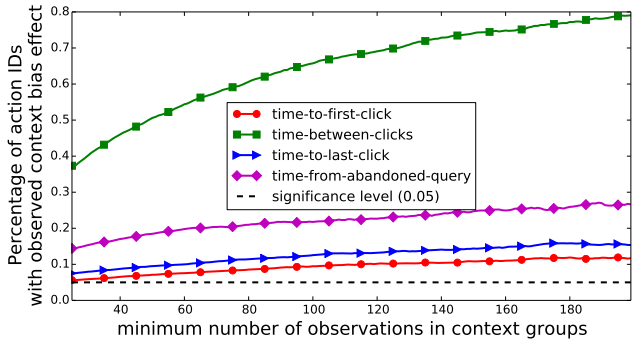

Figure 2: Percentage of actions IDs with observed context bias effect vs. minimum number of observations in context groups.

The percentage of action IDs, for which the context bias effect is detected, increases with the minimum number of observations in each context group (Figure 2). This suggests that the actual number of action IDs for which the context bias effect appears, is even larger than the number of action IDs for which we manage to detect the effect. And with more interaction data, it should be possible to detect the effect in a larger number of times-between-actions.

From the above results we conclude that there is a tangible context bias effect, which results in statistically significant differences in time-between-actions probability distributions for different contexts (in our case, for different sets of previous user interactions).

5.2 Experiment 2

The results of Experiment 2 are given in Table 6. The table shows the performance of the basic time models and the context-aware time models in terms of log-likelihood and RMSE on four temporal prediction tasks: time-to-first-click, time-between-clicks, time-to-last-click and time-from-abandoned-query (§2).

RQ2. Table 6 shows that the context-aware time models outperform the basic time models in terms of both the log-likelihood and RMSE metrics on all temporal prediction tasks. The improvements for each task and each distribution are statistically significant with $p < 0.001$. The improvements in terms of log-likelihood are comparable with the differences between the basic time models using different probability density functions, and in most cases exceed them. The improvements in terms of RMSE vary, depending on the task, between 0.98–5.32%.

To further understand the performance difference between the basic time models and the context-aware time models, we plot their performance vs. the actual times-between-actions observed in the datasets. Figures 3 and 4 show the performance on the time-between-clicks task. We start with Figure 3, which shows the performance in terms of the log-likelihood metric. Here, similar to the differences between the basic time models using different probability density functions, we observe major improvements of the context-aware time models over the basic time models for small times. In Figure 4, which shows the performance in terms of the RMSE metric, we also observe major improvements for small times.

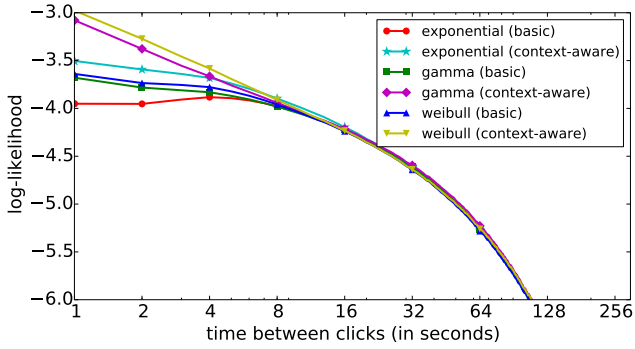

Figure 3: Time model performance in terms of the log-likelihood metric on the time-between-clicks task vs. actual times observed in the dataset.

Table 6: Performance of the basic time models and the context-aware time models on four temporal prediction tasks. Larger values of the average log-likelihood metric and lower values of the root mean squared error (RMSE) metric indicate better performance. Improvements of the context-aware time models over the basic time models using the same probability density functions (exponential, gamma, Weibull) are statistically significant ($p < 0.001$) in terms of both metrics.

Time model	Distribution	Log-likelihood	RMSE
Time-to-first-click			
Basic	exponential	−2.9050	7.52
	gamma	−2.8399	7.51
	Weibull	−2.8970	7.52
Context-aware	exponential	−2.8715	7.29
	gamma	−2.7636	7.25
	Weibull	−2.8449	7.27
Time-between-clicks			
Basic	exponential	−4.9219	60.73
	gamma	−4.9105	60.76
	Weibull	−4.9077	60.76
Context-aware	exponential	−4.8787	58.93
	gamma	−4.8556	58.98
	Weibull	−4.8504	58.94
Time-to-last-click			
Basic	exponential	−3.8360	40.85
	gamma	−3.7849	40.85
	Weibull	−3.7386	40.85
Context-aware	exponential	−3.7924	40.45
	gamma	−3.7456	40.45
	Weibull	−3.7073	40.51
Time-from-abandoned-query			
Basic	exponential	−3.5862	12.41
	gamma	−3.5422	12.41
	Weibull	−3.5560	12.41
Context-aware	exponential	−3.5210	11.75
	gamma	−3.4235	11.76
	Weibull	−3.4554	11.77

A reader might notice a drop in performance of the context-aware time models compared to the basic time models between 17 and 99 seconds in Figure 4. This is better seen in Figure 5, which plots the differences between the context-aware time models and the basic time models shown in Figure 4. Here, we see that the context-aware time models perform better for times lower than 17 seconds and larger than 99 seconds, and perform worse for times in the range of 17–99 seconds. This can be explained as follows. The average time-between-clicks in our dataset is 53.44 seconds. A naive time model that always predicts time-between-clicks to be 53.44 seconds, would have very low RMSE around 53.44 seconds and high RMSE for smaller and larger times (the overall RMSE

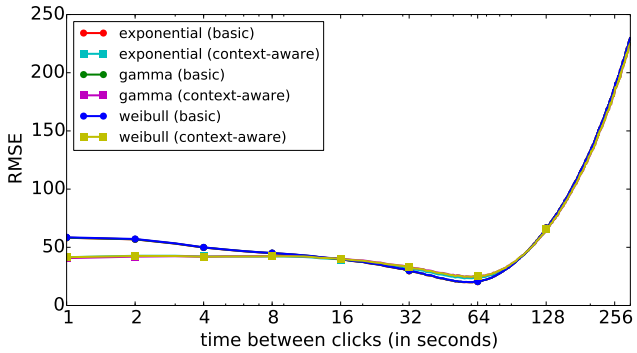

Figure 4: Time model performance in terms of the RMSE metric on the time-between-clicks task vs. actual times observed in the dataset.

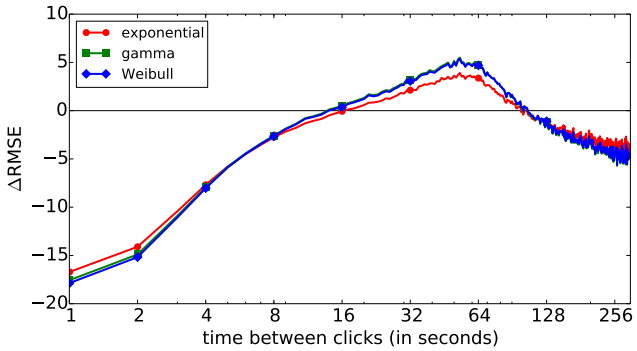

Figure 5: Performance difference in terms of the RMSE metric between the context-aware time models and the basic time models on the time-between-clicks task vs. actual times observed in the dataset.

performance of this naive approach would be low). The basic time models predict time-between-clicks across the whole time range including small and large times, and have better overall RMSE performance. However, this is achieved at the cost of having higher RMSE around the average time-between-clicks (i.e., 53.44 seconds) compared to that of the naive time model. The proposed context-aware time models, on average, predict time-between-clicks better than the basic time models (and especially so for smaller and larger times, see Figure 5). This is again achieved at the cost of having higher RMSE around the average time-between-clicks (i.e., 53.44 seconds). In fact, smaller times-between-clicks usually correspond to dissatisfied clicks, and larger times-between-clicks correspond to satisfied clicks [18]. Thus, in order to improve user satisfaction with the search, it is more important to predict small and large times-between-clicks rather than to distingush between times close to the average time-between-clicks.

From the above results we conclude that the context-aware time models, which besides the first action ID make use of its context, provide better means to explain times-between-actions that the basic time models, which rely solely on the first action ID.

5.3 Experiment 3

Table 7 shows the outcomes of Experiment 3, a comparison of the performance of ranking models based on times-between-clicks.

RQ3. Table 7 shows that the CATM-based ranking models outperform the ranking models based on the basic time models and the ranking model that scores search results by the average values of the observed times between clicks. All improvements are

Table 7: Performance of ranking models based on the time between clicks. Larger values of the NDCG metric correspond to better rankings. The improvements of the context-aware time models over the existing methods are statistically significant ($p < 0.05$).

		NDCG			
Time model	Distribution	@1	@3	@5	@10
Average	n/a	0.651	0.693	0.728	0.812
Basic	exponential	0.651	0.693	0.728	0.812
	gamma	0.646	0.693	0.728	0.812
	Weibull	0.656	0.699	0.730	0.811
Context-aware	exponential	0.668	0.710	0.743	0.820
	gamma	0.675	0.715	0.748	0.822
	Weibull	0.671	0.709	0.745	0.821

statistically significant with $p < 0.05$. Thus, we conclude that the context-independent predictions of the CATMs trained on the time-between-clicks task provide better means to rank search engine results than existing methods.

6. RELATED WORK

We discuss two types of related work:behavioral signals used to improve and evaluate web search effectiveness (§6.1); and models of user behavior used to interpret these signals (§6.2).

6.1 Behavioral information

Modern search engines log a large number of user behavioral signals to improve and evaluate their effectiveness. We classify them in two groups: behavioral signals based on user actions and behavioral signals based on times between user actions.

Behavioral signals based on user actions. User clicks provide an important source of implicit user feedback [1–3, 8–10, 20, 27, 38, 39]. They have been used(i) to infer user preferences over search results, i.e., target values for learning a ranking function [9, 27], (ii) to design features for learning a ranking function [1, 2, 8] and for other applications [3, 20], and (iii) to compare ranking functions [10, 38, 39]. Some work distinguishes different types of clicks: first click [8, 39], last click [8], long dwell time click [8], satisfied click [39] and only click [8]. Next to clicks, some recent work considers mouse cursor movements on SERPs [16, 24, 25].

Behavioral signals based on times between user actions. Click dwell time, i.e., time spent by a user on the landing page of a search result, provides valuable information about user satisfaction with the clicked result [30]. Existing work uses it as a feature for ranking [1, 2, 8] and in many tasks related to user satisfaction [3, 20–22, 30, 32, 33, 36]. Times from a submission of a query to(i) the first click [7, 10, 20, 39], (ii) the first satisfied click [39], and (iii) the last click [10] are used as features to predict user satisfaction with the clicked result [20], compare two versions of a search engine [10, 39] and to cluster users based on their SERP examination strategies [7]. Song et al. [41] use the time since a user issued an abandoned query (i.e., a query for which there were no interactions with the search results) to the next query submission for classifying the abandoned query into positively abandoned and negatively abandoned. Dupret and Lalmas [17] use times between search engine visits to compare two versions of a search engine.

6.2 Models of user behavior

Now that we have described the behavioral signals, we focus on their interpretations. To interpret a signal, we need to have a model

that explains it. We start with models for explaining behavioral signals based on user actions and then discuss models for explaining behavioral signals based on times between user actions.

Modeling user actions. The simplest way to interpret click data is to compute *click-through rates* (CTRs), i.e., the ratios of the total number of clicks and the total number of impressions observed for a group of search engine results. Unfortunately, CTRs suffer from the so-called *position bias effect*, i.e., results presented at higher positions receive more clicks than results of similar quality presented at lower positions [15, 27, 28], which leads to suboptimal performance when using CTRs for ranking. To account for position bias, a large number of *click models* have been proposed [13].

Click models make a few assumptions about user interactions with search results, which ultimately allow them to obtain per query-result scores that show better correlation with relevance than CTRs. Among the most common assumptions are(i) the *linear traversal assumption* that a user scans search results from top to bottom [15]; and (ii) the *examination hypothesis* that a user clicks on a search result if, and only if, she examined the search result and is attracted by it [15]. Recently, it has been shown that patterns of user click behaviour can be learned automatically from interaction data [6]. In addition to position bias, recent work examines other types of bias including(i) *vertical bias* driven by visually salient vertical results (e.g., image results, video results, news results) [11, 12, 43]; (ii) *query bias*, which occurs if a query does not match the user's information need [45], (iii) *duplicate bias*, which occurs if a result has been examined earlier in the search task [45]; and (iv) bias driven by individual differences between users [40].

The notion of context bias, introduced in our work for times between user actions, generalizes the idea of bias in user actions (in particular, clicks and mouse hovers), and the proposed context-aware time model should be able to account for them with appropriate representations of the context.

Modeling times between user actions. The simplest way to interpret times between user actions is to compute their mean values. The average click dwell time is frequently used as a feature for ranking [1, 2, 8] and other applications [3, 20–22, 32, 33, 36]. The average time between a submission of a query to the first click is used both as a feature [20] and as an online metric for comparing two versions of a search engine [10, 39]. The average time between a submission of a query and(i) first click classified as satisfied, (ii) last click are also used as online metrics for comparing two versions of a search engine [10, 39].

A more sophisticated way to interpret times between user actions is to fit a probability distribution [30, 31]. Liu et al. [31] find that the Weibull distribution provides better goodness-of-fit to click dwell times than the exponential distribution. The authors provide an interesting interpretation for the shape parameter of the fitted Weibull distribution, which justifies the task of modeling the full probability distribution rather than just the mean of the distribution. Our work differs from their work in that we do not make the context-independence assumption that the observed times were drawn from the same probability distribution; we predict the probability distributions separately for each click using its context, and the information about click dwell times observed for the given query-result pair in other contexts. Liu et al. [31] also show that it is possible to predict parameters of the Weibull distribution for a given result using the information about the result's landing page, such as HTML tags, frequent words that occur on the page and times to download/render/parse the page. Our approach differs from their method in that we use contextual, i.e., result-independent, information instead of result-specific information; thus, our work is complementary to that of [31].

Kim et al. [30] fit gamma distributions to click dwell times, observed in a predefined click segment and labeled as satisfied or dissatisfied. The authors use the fitted distributions to classify new clicks into satisfied and dissatisfied. In particular, they compute the following features:(i) the differences between the fitted parameters of the satisfied and dissatisfied click dwell time distributions for the clicked result's segment, (ii) their expected values and the difference between them, (iii) the absolute differences between the observed click dwell time and the expected values of the satisfied and dissatisfied distributions, (iv) the log-likelihoods of the observed click dwell time according to the satisfied and dissatisfied distributions and the difference between them. The large number of features used in their work shows the advantage of modeling the full probability distribution over modeling just the mean. Similar features, computed from the probability distributions predicted in our work can potentially be used in a broad range of applications.

Our work is the first to systematically address the problem of modeling times between user actions. We introduce the notion of context bias effect and propose a context-aware time model that predicts times elapsed between user actions using both the ID of the first action (which is what existing methods rely on) and the context in which it takes place (our novelty).

7. CONCLUSION

We introduced the notion of *context bias effect* in times between user actions (i.e., a difference in probability distributions of times associated with the same user action, but observed in different contexts); and proposed a *context-aware time model* (CATM) that allows us to estimate parameters of a probability distribution of the time elapsed between user actions in a given context. CATM's ability to account for user context can be used to predict times between user actions in a context in which these actions have not previously been observed, and to obtain context-independent estimates of times between actions by predicting them in predefined contexts.

We showed that, for 37%–80% of query-result pairs (q, r), depending on the number of observations, the distributions of times elapsed between a click on the result r and the next click on the same SERP differed in sessions, in which the result r was the first item to be clicked, and in sessions, in which there were clicks on other results before the result r was clicked. Similarly, we showed that previous user interactions in a search task influence distributions of times between(i) submission of a query and the first click on a SERP, (ii) submission of a query and the last click on a SERP, and (iii) submission of an abandoned query (i.e., a query with no clicks on a SERP) and the next query submission.

We evaluated the context-aware time model on four temporal prediction tasks and a ranking task. The results on the temporal prediction tasks show that the proposed context-aware time model, which makes use of both the ID of the first action and the context in which it takes place, provides better means to explain times between user actions than existing methods, which rely solely on the first action ID. The results on the ranking task show that the context-independent estimates of times between consecutive clicks, obtained with the context-aware time model, allow us to construct rankings that result in better performance in terms of relevance than the rankings produced by the mean values of the observed times between consecutive clicks.

As to future work, we plan to consider other representations of the context. Besides using context-independent estimates of times between consecutive clicks as features for ranking, the predictions of context-aware time models can be used in a range of other appli-

cations that use the average or predicted times between user actions. These applications include prediction of click satisfaction [30], result usefulness [32], search task difficulty [3, 33], search goal success [20, 21], urgent information needs [35], struggling vs. exploring behavior [22, 36], positive vs. negative abandonment [41] and clustering users based on their SERP examination strategies [7].

Acknowledgments. This research was supported by Ahold, Amsterdam Data Science, the Bloomberg Research Grant program, the Dutch national program COMMIT, Elsevier, the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement nr 312827 (VOX-Pol), the ESF Research Network Program ELIAS, the Royal Dutch Academy of Sciences (KNAW) under the Elite Network Shifts project, the Microsoft Research Ph.D. program, the Netherlands eScience Center under project number 027.012.105, the Netherlands Institute for Sound and Vision, the Netherlands Organisation for Scientific Research (NWO) under project nrs 727.011.005, 612.001.116, HOR-11-10, 640.006.013, 612.066.930, CI-14-25, SH-322-15, 652.002.001, 612.001.551, and the Yahoo Faculty Research and Engagement Program. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

8. REFERENCES

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

[2] E. Agichtein, E. Brill, S. Dumais, and R. Ragno. Learning user interaction models for predicting web search result preferences. In *SIGIR*, pages 3–10. ACM, 2006.

[3] J. Arguello. Predicting search task difficulty. In *Advances in Information Retrieval*, pages 88–99. Springer, 2014.

[4] Y. Bengio. Learning deep architectures for AI. *Foundations and trends in Machine Learning*, 2(1):1–127, 2009.

[5] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[6] A. Borisov, I. Markov, M. de Rijke, and P. Serdyukov. A neural click model for web search. In *WWW*, pages 531–541, 2016.

[7] G. Buscher, R. W. White, S. Dumais, and J. Huang. Large-scale analysis of individual and task differences in search result page examination strategies. In *WSDM*, pages 373–382. ACM, 2012.

[8] O. Chapelle and Y. Chang. Yahoo! learning to rank challenge overview. In *Yahoo! Learning to Rank Challenge*, pages 1–24, 2011.

[9] O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *WWW*, pages 1–10. ACM, 2009.

[10] O. Chapelle, T. Joachims, F. Radlinski, and Y. Yue. Large-scale validation and analysis of interleaved search evaluation. *TOIS*, 30(1): 6, 2012.

[11] D. Chen, W. Chen, H. Wang, Z. Chen, and Q. Yang. Beyond ten blue links: enabling user click modeling in federated web search. In *WSDM*, pages 463–472. ACM, 2012.

[12] A. Chuklin, P. Serdyukov, and M. de Rijke. Using intent information to model user behavior in diversified search. In *Advances in Information Retrieval*, pages 1–13. Springer, 2013.

[13] A. Chuklin, I. Markov, and M. de Rijke. *Click Models for Web Search*. Morgan & Claypool Publishers, August 2015.

[14] W. J. Conover. *Practical Nonparametric Statistics*. Wiley, 1980.

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

[16] F. Diaz, R. White, G. Buscher, and D. Liebling. Robust models of mouse movement on dynamic web search results pages. In *CIKM*, pages 1451–1460. ACM, 2013.

[17] G. Dupret and M. Lalmas. Absence time and user engagement: evaluating ranking functions. In *WSDM*, pages 173–182. ACM, 2013.

[18] S. Fox, K. Karnawat, M. Mydland, S. Dumais, and T. White. Evaluating implicit measures to improve web search. *TOIS*, 23(2): 147–168, 2005.

[19] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, pages 6645–6649. IEEE, 2013.

[20] A. Hassan. A semi-supervised approach to modeling web search satisfaction. In *SIGIR*, pages 275–284. ACM, 2012.

[21] A. Hassan, R. Jones, and K. L. Klinkner. Beyond DCG: user behavior as a predictor of a successful search. In *WSDM*, pages 221–230. ACM, 2010.

[22] A. Hassan, R. W. White, S. T. Dumais, and Y.-M. Wang. Struggling or exploring?: disambiguating long search sessions. In *WSDM*, pages 53–62. ACM, 2014.

[23] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[24] J. Huang, R. W. White, and S. Dumais. No clicks, no problem: using cursor movements to understand and improve search. In *SIGCHI*, pages 1225–1234. ACM, 2011.

[25] J. Huang, R. W. White, G. Buscher, and K. Wang. Improving searcher models using mouse cursor activity. In *SIGIR*, pages 195–204. ACM, 2012.

[26] K. Järvelin and J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *SIGIR*, pages 41–48. ACM, 2000.

[27] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*, pages 133–142. ACM, 2002.

[28] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR*, pages 154–161. ACM, 2005.

[29] Y. Kim, A. Hassan, R. W. White, and I. Zitouni. Comparing client and server dwell time estimates for click-level satisfaction prediction. In *SIGIR*, pages 895–898. ACM, 2014.

[30] Y. Kim, A. Hassan, R. W. White, and I. Zitouni. Modeling dwell time to predict click-level satisfaction. In *WSDM*, pages 193–202. ACM, 2014.

[31] C. Liu, R. W. White, and S. Dumais. Understanding web browsing behaviors through weibull analysis of dwell time. In *SIGIR*, pages 379–386. ACM, 2010.

[32] C. Liu, J. Liu, N. Belkin, M. Cole, and J. Gwizdka. Using dwell time as an implicit measure of usefulness in different task types. *JASIST*, 48(1):1–4, 2011.

[33] J. Liu, C. Liu, M. Cole, N. J. Belkin, and X. Zhang. Exploring and predicting search task difficulty. In *CIKM*, pages 1313–1322. ACM, 2012.

[34] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048, 2010.

[35] N. Mishra, R. W. White, S. Ieong, and E. Horvitz. Time-critical search. In *SIGIR*, pages 747–756. ACM, 2014.

[36] D. Odijk, R. W. White, A. H. Awadallah, and S. T. Dumais. Struggling and success in web search. In *CIKM*. ACM, 2015.

[37] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*, 2012.

[38] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *CIKM*, pages 43–52. ACM, 2008.

[39] A. Schuth, K. Hofmann, and F. Radlinski. Predicting search satisfaction metrics with interleaved comparisons. In *SIGIR*, pages 463–472. ACM, 2015.

[40] S. Shen, B. Hu, W. Chen, and Q. Yang. Personalized click model through collaborative filtering. In *WSDM*, pages 323–332. ACM, 2012.

[41] Y. Song, X. Shi, R. White, and A. H. Awadallah. Context-aware web search abandonment prediction. In *SIGIR*, pages 93–102. ACM, 2014.

[42] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.

[43] C. Wang, Y. Liu, M. Zhang, S. Ma, M. Zheng, J. Qian, and K. Zhang. Incorporating vertical results into search click models. In *SIGIR*, pages 503–512. ACM, 2013.

[44] M. D. Zeiler. ADADELTA: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[45] Y. Zhang, W. Chen, D. Wang, and Q. Yang. User-click modeling for understanding and predicting search-behavior. In *CIKM*, pages 1388–1396. ACM, 2011.

[46] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *TOMS*, 23(4):550–560, 1997.