A Click Sequence Model for Web Search

Alexey Borisov
Yandex & University of Amsterdam
Moscow, Russia
alborisov@yandex-team.ru

Martijn Wardenaar
University of Amsterdam
Amsterdam, The Netherlands
martijnwardenaar@gmail.com

Ilya Markov
University of Amsterdam
Amsterdam, The Netherlands
i.markov@uva.nl

Maarten de Rijke
University of Amsterdam
Amsterdam, The Netherlands
derijke@uva.nl

ABSTRACT

Getting a better understanding of user behavior is important for advancing information retrieval systems. Existing work focuses on modeling and predicting single interaction events, such as clicks. In this paper, we for the first time focus on modeling and predicting sequences of interaction events. And in particular, sequences of clicks.

We formulate the problem of click sequence prediction and propose a click sequence model (CSM) that aims to predict the order in which a user will interact with search engine results. CSM is based on a neural network that follows the encoder-decoder architecture. The encoder computes contextual embeddings of the results. The decoder predicts the sequence of positions of the clicked results. It uses an attention mechanism to extract necessary information about the results at each timestep. We optimize the parameters of CSM by maximizing the likelihood of observed click sequences.

We test the effectiveness of CSM on three new tasks: (i) predicting click sequences, (ii) predicting the number of clicks, and (iii) predicting whether or not a user will interact with the results in the order these results are presented on a search engine result page (SERP). Also, we show that CSM achieves state-of-the-art results on a standard click prediction task, where the goal is to predict an unordered set of results a user will click on.

CCS CONCEPTS

• **Information systems → Users and interactive retrieval**;

KEYWORDS

Click model, User behavior, Web search

ACM Reference Format:
Alexey Borisov, Martijn Wardenaar, Ilya Markov, and Maarten de Rijke. 2018. A Click Sequence Model for Web Search. In *SIGIR'18: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, July 8–12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209978.3210004

SIGIR'18, July 8–12, 2018, Ann Arbor, MI, USA

ACM 978-1-4503-5657-2/18/07…$15.00
https://doi.org/10.1145/3209978.3210004

1 INTRODUCTION

Search engines play an important role in our everyday lives. One way to improve them is by getting a better understanding of user search behavior, such as clicks, dwell times, mouse movements, etc. So far, models of user behavior have focused on modeling and predicting single events, e.g., clicks [9] and mouse movements [22], and properties of these events, e.g., time between clicks [4]. In this paper for the first time we focus on modeling and predicting sequences of information interaction events and, in particular, sequences of clicks.

Although people tend to make only one (or sometimes no) click on a search engine result page (SERP), multi-click query sessions constitute a significant part of search traffic. For example, about 23% of the query sessions in the Yandex relevance prediction challenge dataset contain multiple clicks (see §4.1 for details). It is commonly assumed that users traverse search results from top to bottom, which leads to the assumption that clicks are ordered by the position of search results. However, it was shown that in practice this assumption does not always hold, and that up to 27.9%–30.4% of multi-click sequences, depending on the dataset, are not ordered by position [43].

We aim to create tools that help us understand, model and predict sequences of clicks on search engine results, which is important because it provides an opportunity for improving the user search experience. For example, knowing that a user is likely to click on many results or that there are high chances that the user will interact with the results in an order other than the one in which the results are presented on a SERP can be used by a search engine to proactively show an advice or make a change in the ranking.

We propose a *click sequence model* (CSM) that predicts a probability distribution over click sequences. At the core of our model is a neural network with encoder-decoder architecture. We implement the encoder as a *bidirectional recurrent neural network* (bi-RNN) that goes over the search engine result page from top to bottom and from bottom to top and outputs contextual embeddings of the results. We implement the decoder as a recurrent neural network (RNN) with an attention mechanism. The decoder is initialized with the final states of the forward and backward RNNs of the encoder. It is used to predict the sequence of positions of the clicked results. The whole network is trained by maximizing the likelihood of the observed click sequences.

We evaluate our proposed CSM using a publicly available click log and show that CSM provides good means to generate a short list of K click sequences that contains the observed click sequence with a high probability. We present an analysis of the performance of CSM for query sessions with different numbers of clicks and query sessions in

which clicks are ordered/not ordered by position. We measure the performance of CSM on two new tasks: predicting the number of clicks and predicting ordered/unordered sequences of clicks. Additionally, we show that CSM achieves state-of-the-art results on the standard click prediction task, which allows us to compare CSM to traditional click models that model and predict single events, namely clicks.

Overall, we make the following contributions:

- We formulate a novel problem of predicting click sequences.
- To solve this problem, we propose a click sequence model (CSM) based on neural networks.
- We evaluate CSM on a range of prediction tasks, namely predicting click sequences, predicting the number of clicks, predicting ordered/unordered sequences of clicks and, finally, predicting clicks themselves.

As to the potential impact of the proposed CSM model, we believe it can be used to predict that (i) a user will click on more than one result, which may indicate that a user has a complex information need [30]; or that (ii) a user will interact with the results not in the order in which these results are presented on the SERP, which may indicate that a user is struggling and there are problems in the ranking of the results [36]. CSM can help us identify queries for which there is a room for improvement (in terms of user experience) and it can serve as a quick analysis tool to interpret how a particular change in the ranking of the results will influence user click behavior.

The rest of the paper is structured as follows. In Section 2 we provide a precise statement of the click sequence modeling and prediction problems that we are tackling. Section 3 introduces our neural network based model for predicting click sequences. In Section 4 we describe the setup of our experiments and Section 5 presents the results of those experiments. We describe related work in Section 6 and conclude in Section 7.

2 PROBLEM STATEMENT

In this section, we formulate the problem of click sequence prediction (§2.1) and propose three prediction tasks that can be solved by a model capable of predicting click sequences (§2.2).

2.1 Problem

Since the number and order of clicks may vary even for the same query and ranking of results (e.g., due to different users and contexts), there exists no unique *correct* click sequence, but a (possibly infinite) set of *probably correct* sequences does exist. Therefore, the main goal of this paper is to build a model that, given a query and a ranking of results, describes these probably correct click sequences.

To achieve this goal, we define the *click sequence prediction problem* as follows. First, we learn a probability distribution $\mathcal{M}$ over all possible click sequences. Second, we use this learned distribution to obtain the K most probable click sequences. These K sequences are then used to reason about the properties of the set of probably correct sequences mentioned above, e.g., predicting the expected number of clicks, the expected order of clicks, etc.

More formally, we define a click sequence model $\mathcal{M}$ as follows:

$$\mathcal{M}\colon \quad P(s \mid q, r_1, \ldots, r_N), \tag{1}$$

where q is a query, $r_1,\ldots,r_N$ is an ordered list of results and s is a sequence of positions of the clicked results $(p_1,\ldots,p_S)$.

2.2 Prediction tasks

There are many possible applications for a model $\mathcal{M}$ that is capable of (i) predicting a probability distribution over click sequences (Eq. 1), and (ii) retrieving the K most probable click sequences. It can be used to simulate user behavior, which is important in *online learning to rank* research [21, 42], or as a tool for analyzing how a particular change in the ranking of results will influence user click behavior. However, we do not investigate these applications in this work. Instead, we address three tasks that are both practically useful and help to evaluate the performance of the model $\mathcal{M}$.

Task 1 (predicting the number of clicks). The goal of this task is to predict on how many results a user will click. Clicking on more than one result might indicate that a user has a complex information need. Clicking on more than three or four results might indicate that a user is struggling or doing an exhaustive search [19, 36]. Both signals can be used by a search system to proactively show an advice or make a change in the ranking. Thus, we formally define Task 1 as predicting whether a user will click on $\leq L$ results.

To estimate the probability of clicking on $\leq L$ results, we generate a large number K (e.g., $K = 1024$) most probable click sequences $s_1,\ldots,s_K$ and marginalize over those sequences that have $\leq L$ clicks:

$$P(|s| \leq L) = \sum_{s \in \{s_1, \ldots, s_K\}} P(s) \mathbb{1}[|s| \leq L]. \tag{2}$$

Task 2 (predicting non-consecutive click sequences). The goal of this task is to predict whether a user will interact with results in the order these results are presented on a SERP or in a different order, which we refer to as a *non-consecutive* order. Interacting with results in a non-consecutive order might indicate that a user is struggling [40]. As mentioned in Task 1, such a signal can be used by a search engine to proactively show an advice or make a change in the ranking.

Similarly to Task 1, we estimate the probability of clicking on results in a non-consecutive order by summing probabilities of the K most probable click sequences $s_1,\ldots,s_K$ according to $\mathcal{M}$ in which a user clicks on a result r_i after clicking on a result r_j located below r_i $(i < j)$:

$$P(\downarrow\uparrow) = \sum_{s \in \{s_1, \ldots, s_K\}} P(s) \mathbb{1}[s \text{ is non-consecutive}]. \tag{3}$$

Task 3 (predicting clicks). The last task is actually a standard task solved by click models [9]. The goal is to predict a subset of the presented results $r_1,\ldots,r_N$ on which a user will click. Being able to predict that a user will not interact with a subset of results, opens the door for reranking [52].

Similarly to Task 1 and Task 2, we estimate the click probability for position p by summing probabilities of the K most probable click sequences $s_1,\ldots,s_K$ according to $\mathcal{M}$ in which a user clicks on that position:

$$P_{\text{click}}(p \mid q, r_1, \ldots, r_N) = \sum_{s \in \{s_1, \ldots, s_K\}} P(s) \mathbb{1}[p \in s]. \tag{4}$$

In practice, it is probably better to use simpler models to predict clicks. So we use this task mainly to compare with existing work. We expect the results for a good model $\mathcal{M}$ to be not much worse compared to the results for click models specifically developed for

this task [9]. In fact, as we show in §5, CSM achieves state-of-the-art performance on this task.

3 METHOD

In this section we propose the *click sequence model* (CSM), a model for predicting click sequences. We use s to denote a sequence of positions of the clicked results with a special *end of sequence* (EOS) token appended to it, i.e., $s=(p_1,\ldots,p_k,\text{EOS})$.

CSM is a neural network that is trained to maximize the likelihood of observed click sequences:

$$\mathcal{L}(s_1,\ldots,s_{|S|}) \rightarrow \max_{\Theta}, \tag{5}$$

where Θ denotes the network parameters and $S=(s_1,\ldots,s_{|S|})$ denotes click sequences used for training.

The network consists of two parts, called *encoder* and *decoder*. The encoder takes a user's query q and a list of search engine results $r_1,\ldots,r_N$ as input and computes embeddings of the results, $\mathbf{r}_1,\ldots,\mathbf{r}_N$. The embedded results $\mathbf{r}_1,\ldots,\mathbf{r}_N$ are passed to the decoder, which at each timestep $t=0,1,\ldots$ outputs a probability distribution over $N+1$ positions. Positions $1,\ldots,N$ correspond to clicking on the results $r_1,\ldots,r_N$. The $(N+1)$-th position corresponds to predicting that there will be no clicks (EOS). Upon observing a click at timestep t, the decoder updates its current state using the position p_t of the clicked result. Figure 1 illustrates the workflow in the form of a UML sequence diagram.

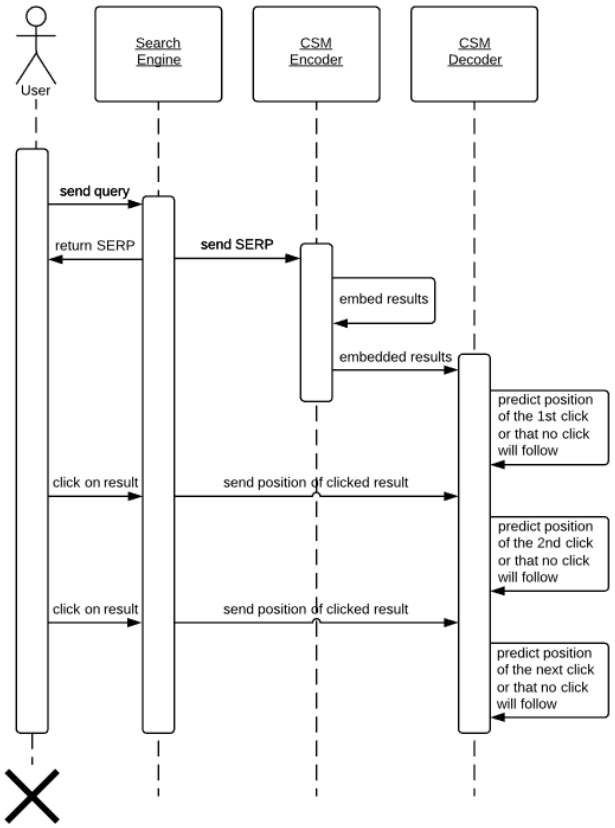

Figure 1: Modeling click sequences with CSM.

In §3.1 we discuss the implementation of the encoder and decoder. Then, in §3.2 we explain how to achieve the main goal of this study, i.e., predict K most probable click sequences using CSM. Finally, in §3.3 we specify training details.

3.1 Network architecture

Encoder. The aim of the encoder is to obtain information from a user's query q and results $r_1,\ldots,r_N$ presented on a SERP, and pass this information to the decoder. We represent the encoded information as a list of embeddings $\mathbf{q},\mathbf{r}_1,\ldots,\mathbf{r}_N$, where each result embedding $\mathbf{r}_i$ should contain:(i) information about the result r_i, (ii) the results surrounding r_i and (iii) the query q. Below we sometimes use $\mathbf{r}_0$ instead of $\mathbf{q}$ to simplify the notation.

We propose to implement the encoder as a bidirectional recurrent neural network, which goes over the SERP in the top down order, i.e., $q,r_1,\ldots,r_N$, and in the reverse order, i.e., $r_N,\ldots,r_1,q$. The first, *forward RNN*, produces embeddings $\overrightarrow{\mathbf{q}}(=\overrightarrow{\mathbf{r}_{0:0}}),\overrightarrow{\mathbf{r}_{0:1}},\ldots,\overrightarrow{\mathbf{r}_{0:N}}$. The second, *backward RNN*, produces embeddings $\overleftarrow{\mathbf{r}_{N:N}}$, $\ldots$, $\overleftarrow{\mathbf{r}_{N:1}}$, $\overleftarrow{\mathbf{q}}(=\overleftarrow{\mathbf{r}_{N:0}})$. These embeddings are concatenated to form the final embeddings $\mathbf{q}(=\mathbf{r}_0),\mathbf{r}_1,\ldots,\mathbf{r}_N$ produced by the encoder.

We represent $q,r_1,\ldots,r_N$ using the best performing behavioral features proposed in [5]. These features count the number of times a particular click pattern, i.e., a set of positions of the clicked results, was observed on a SERP. A query q is represented as a 2^N dimensional vector, where each component counts the number of times a click pattern was observed in query sessions generated by q. The representation of a search result r consists of two parts, both of size $N2^N$. The components of the first part count, for each position $p=1,\ldots,N$, the number of times a click pattern was observed in query sessions in which r appears on position p. The components of the second part are similar, but include only query sessions generated by q. We apply a linear transformation to these sparse behavioral features to obtain the embeddings $\mathbf{x}_0,\ldots,\mathbf{x}_N$ of $q,r_1,\ldots,r_N$, which are passed to the RNNs.

We describe the encoder formally using Eqs. 6–9:

$$\mathbf{x}_i = \begin{cases}\text{Embed}(q) & i=0\\ \text{Embed}(r_i) & i=1,\ldots,N\end{cases} \tag{6}$$

$$\overrightarrow{\mathbf{r}_{0:0}},\ldots,\overrightarrow{\mathbf{r}_{0:N}} = \text{RNN}_{\text{forward}}(\mathbf{x}_0,\ldots,\mathbf{x}_N) \tag{7}$$

$$\overleftarrow{\mathbf{r}_{N:N}},\ldots,\overleftarrow{\mathbf{r}_{N:0}} = \text{RNN}_{\text{backward}}(\mathbf{x}_0,\ldots,\mathbf{x}_N) \tag{8}$$

$$\mathbf{r}_i = [\overrightarrow{\mathbf{r}_{0:i}},\overleftarrow{\mathbf{r}_{N:i}}] \quad (i=0,\ldots,N) \tag{9}$$

Decoder. The aim of the decoder is to predict a probability distribution over $(N+1)$ positions at each timestep. (As mentioned at the start of §3, the $(N+1)$-th position corresponds to predicting that there will be no clicks.) To make a good prediction at timestep $(t+1)$, we need to incorporate into the decoder the information about the position p_t of the result clicked at timestep t.

We propose to implement the decoder as an RNN that at each timestep $t=0,1,\ldots$ outputs a vector $\mathbf{o}_t$ used to predict the probability distribution, and updates its hidden state using the position p_t of the observed click at timestep t. We also use an attention mechanism [2] to help the decoder extract the most relevant information from the list of embeddings $\mathbf{r}_0,\ldots,\mathbf{r}_N$ at each timestep.

We initialize the hidden state of the decoder RNN using the concatenation of the final states of the forward and backward RNNs of the encoder, $[\overrightarrow{\mathbf{r}_{0:N}},\overleftarrow{\mathbf{r}_{N:0}}]$, passed through a linear transformation; we

use W_{init} to denote the transformation matrix. To obtain the probability distribution over $(N+1)$ positions at timestep t, we concatenate the vector $\mathbf{o}_t$ predicted by the decoder RNN and the attention vector $\mathbf{a}_t$ computed at timestep t, and pass the result through a linear transformation W_{output} followed by softmax.[1] We represent the position p_t of the observed click at timestep t as a one-hot vector $\mathbf{p}_t$ of size N. And apply a linear transformation W_{pos} to it before passing to the decoder RNN.

We describe the decoder formally using Eqs. 10–13.

$$\mathbf{s}_0 = W_{\text{init}}[\overrightarrow{\mathbf{r}_{0:N}}, \overleftarrow{\mathbf{r}_{N:0}}] \quad (10)$$

$$\mathbf{a}_{t+1} = \text{Attention}(\mathbf{s}_t, [\mathbf{r}_0, \ldots, \mathbf{r}_N]) \quad (11)$$

$$\mathbf{s}_{t+1}, \mathbf{o}_{t+1} = \text{RNN}_{\text{step}}(\mathbf{s}_t, \mathbf{a}_t, W_{pos}\mathbf{p}_t) \quad (12)$$

$$P(p_{t+1} \mid \ldots) = \text{Softmax}\big(W_{\text{output}}[\mathbf{o}_{t+1}, \mathbf{a}_{t+1}]\big) \quad (13)$$

To alleviate the *exploding gradient problem* [3], we use *gated recurrent units* (GRUs) in both the forward and backward RNNs of the encoder, and in the decoder RNN.

3.2 Beam search

As stated in §2.1, our main goal is to predict K most probable click sequences for a given query and search results. These K sequences are then used to reason about actual user click behavior, i.e., sequences of clicks that a user could actually perform for the given query and search results (we call them *probably correct* sequences, see §2.1).

CSM defines a probability distribution over infinitely many click sequences. Extracting K most probable sequences in this case is not straightforward (since we cannot simply go over all sequences and pick K best ones). We need a means of generating K most probable sequences without having to calculate the probability for every possible click sequence. To do that, we suggest to use beam search [14].

In our experiments, we use $K \leq 1024$ and *beam size* $= K$. Setting the beam size to K guarantees that the K sequences generated by beam search have the highest probabilities according to CSM, i.e., they are indeed most probable click sequences according to CSM. Using a smaller beam size allows us to generate K sequences faster, but does not guarantee that these sequences are the most probable ones.

3.3 Training

We learn the parameters Θ of the CSM network (both the encoder and decoder parts) by maximizing the log-likelihood of observed click sequences. We optimize these parameters using *stochastic gradient descent* (SGD). The learning rates for each parameter are adjusted according to the Adam [28] algorithm using the default values of $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$. We also use *gradient clipping* [39] with the norm set to 1 to alleviate the *exploding gradient problem* [3], which, as mentioned earlier, GRUs also try to mitigate.

4 EXPERIMENTAL SETUP

In this section we describe our experimental setup. We start by describing the data we use to conduct our experiments (§4.1). Then we discuss our evaluation methodology (§4.2), formulate research questions (§4.3) and list the experiments we run to answer these research questions (§4.4).

[1] Using the output of an RNN together with an attention vector has been shown to improve prediction performance [2, 38]. In the literature, this idea is known as *deep output* [38].

Table 1: The number of query sessions in the test set split by the number and order of clicks. By ordered click sequences we mean those where a user clicks on results in the order they appear on a SERP. The total number of click sequences in the test set is 100,000.

Number of clicks	Ordered sequences	Unordered sequences
0	30,466	0
1	46,550	0
2	8851	2143
3	3437	1856
4	1564	1290
5	751	814
6	407	512
7	244	305
8	156	195
9	85	137
10+	73	164

4.1 Data

We use Yandex Relevance Prediction dataset[2] released in 2011 by Yandex, the major search engine in Russia. The dataset consists of 146,278,823 query sessions ordered by time. We use the first half of the dataset for training CSM, and 100,000 randomly selected query sessions from the second half of the dataset for evaluation. Borisov et al. [5] also use the first half of the dataset for training, which allows a direct comparison with their work.

The statistics about the number of query sessions in the test set split by the number and order of clicks is given in Table 1.

4.2 Evaluation methodology

To properly evaluate the proposed CSM model, we would need to know a set of all (or at least a sample of) probably correct click sequences for each test query and search results. Then we could measure how well the K most probable sequences predicted by CSM describe the properties of the known probably correct sequences.

In practice, however, we observe only one (or a few) of all probably correct click sequences and, therefore, we cannot even argue about their properties. The best we can do is to check whether the observed click sequence appears in the list of K sequences predicted by CSM. In particular, we measure recall@K, i.e., the fraction of query sessions for which CSM includes the observed click sequence in the list of K most probable sequences.

Since CSM is the first model for predicting click sequences, there are no baselines to compare against. However, we can use as a reference level the percentage of query sessions in which users interact with results in the order these results are presented on a SERP. In our test data, this percentage equals 92.73%. This is an upper bound under the assumption that a user scans search results sequentially from top to bottom. This means that a model, that predicts only click sequences ordered by position, will contain the observed click sequence in the list of K most probable sequences for $\leq 92.73\%$ of query sessions.

[2] https://academy.yandex.ru/events/data_analysis/relpred2011/ (last visited May 5, 2018)

Task 1. Predicting whether a user will click on $\leq L$ results is a new task and, hence, there are no standard metrics to evaluate performance on this task and no existing baselines to compare to.

We propose to evaluate the performance on this task using perplexity and, because this is a classification problem for a fixed L, *area under curve* (AUC). Perplexity measures how "surprised" a model is upon observing $\leq L$ clicks. AUC measures the model's discriminative power.

We use a naive baseline which predicts that a user will make $\leq L$ clicks with a constant probability calculated on the training set. AUC of such method is 0.5.

Task 2. Predicting whether a click sequence will be ordered by position is also a new task and, similarly to Task 1, there are no standard metrics to evaluate the performance on this task and no existing baselines to compare to.

Similarly to Task 1, we use perplexity and AUC. Our naive baseline predicts that a click sequence will be ordered by position with a constant probability calculated on the training set. AUC of such method is also 0.5, as in Task 1.

Task 3. Following [5, 12, 15, 16, 43], we evaluate the performance on the standard click prediction task using perplexity, which measures how "surprised" a model is upon observing an unordered set of clicks on search engine results.

We use the following click models as our baselines: *dynamic Bayesian network* (DBN) [8], *dependent click model* (DCM) [17], *click-chain model* (CCM) [17], *user browsing model* (UBM) [12] and *neural click model* (NCM) [5]. Borisov et al. [5] use the same data for training these click models, which allows us to compare with the result reported in their work.

4.3 Research questions

We aim to answer the following research questions:

- **RQ1** How well does CSM, described in §3, predict probably correct click sequences?
  - (a) For how many query sessions, the observed click sequence occurs in the list of K most probable click sequences predicted by CSM? How fast does this number increase with K?
  - (b) How well does CSM perform for query sessions in which clicks (i) follow the order in which results are presented on a SERP, and (ii) do not follow the order in which results are presented on a SERP.
  - (c) How well does CSM perform for query sessions with different number of clicks?
  - (d) Do K most probable click sequences provide good means to reason about the probability distribution over click sequences predicted by CSM?
- **RQ2** How well does CSM predict the number of clicks on search results (see Task 1 in §2.2)?
- **RQ3** How well does CSM predict whether or not a user will click on results in the order they are presented on a SERP (see Task 2 in §2.2)?
- **RQ4** How well does CSM predict clicks on search results (see Task 3 in §2.2)? Does it reach the performance of the state-of-the-art click models?

4.4 Experiments

We design our experiments to answer our research questions.

E1(a). To answer RQ1(a), we measure the percentage of query sessions for which the observed click sequence occurs in the list of K most probable click sequences according to CSM. We use $K = \{1,2,3,\ldots,1024\}$.

E1(b). To answer RQ1(b), we measure the percentage of query sessions for which the observed click sequence occurs in the list of K most probable click sequences according to CSM separately (i) for query sessions in which clicks are ordered by position, and (ii) for query sessions in which clicks are not ordered by position. We use $K = \{1,2,3,\ldots,1024\}$.

E1(c). To answer RQ1(c), we measure the percentage of query sessions for which the observed click sequence occurs in the list of K most probable click sequences according to CSM for query sessions with $\leq L$ clicks. We use $K = \{1,2,3,\ldots,1024\}$ and $L = \{1,2,3,4,5\}$.

E1(d). To answer RQ1(d), we compute the total probability of K most probable click sequences according to CSM. If this probability is close to 1, we conclude that using K most probable click sequences is enough to form a representative empirical distribution over click sequences. And, thus, K most probable click sequences provide good means to reason about the properties of the probability distribution over click sequences predicted by CSM. If the total probability mass of K most probable click sequences is small, we conclude that using these sequences is not enough to reason about the probability distribution over click sequences predicted by CSM. We use $K = \{1,2,3,\ldots,1024\}$.

E2. To answer RQ2, we compute probabilities of clicking on $\leq L$ results by marginalizing over the K most probable click sequences according to CSM (see Eq. 2). We use these probabilities to compute perplexity and AUC. We use $K = 1024$ and $L = \{1,2,3,4,5\}$.

E3. To answer RQ3, we compute the probability that a user will click on results in the order these results are presented on a SERP by marginalizing over the K most probable click sequences according to CSM (see Eq. 3). We use this probability to compute perplexity and AUC, $K = 1024$.

E4. To answer RQ4, we compute probabilities of clicking on each result by marginalizing over the K most probable click sequences according to CSM (see Eq. 4). We use these probabilities to compute perplexities for each position and average these perplexity values over positions to obtain the final score. We use $K = 1024$.

In our experiments, we use embeddings of size 256, and the same number of GRUs in all RNNs. We train CSM using *stochastic gradient descent* with mini-batches of 64 query sessions and the parameters specified in §3.3.

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 §4.3.

5.1 Experiment 1(a)

Figure 2 shows recall at different values of K (i.e., the percentage of query sessions for which the observed click sequence occurs in the list of K most probable sequences predicted by CSM) in linear and logarithmic scales. The percentage of query sessions in which clicks

are ordered by position equals 92.73%. We show it on the plots as a reference level.

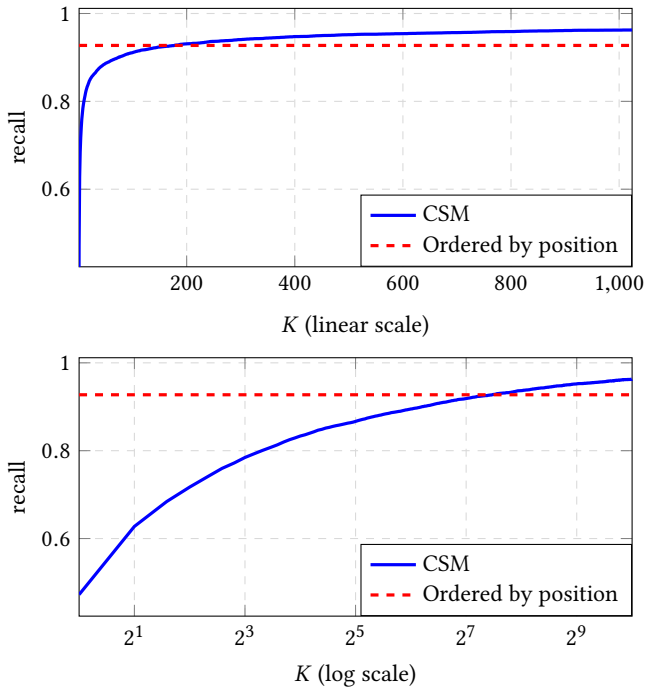

Figure 2: Percentage of query sessions for which the observed click sequence occurs in the list of K most probable click sequences predicted by CSM (blue, solid) and percentage of click sequences ordered by position (red, dashed).

We find that for 47.24% of query sessions, CSM assigns the highest probability to the observed click sequence. For 62.76% of query sessions, the observed sequence appears in the list of two sequences with the highest probabilities according to CSM. Since the curve on the logarithmic scale is concave (see Figure 2, bottom plot), we conclude that recall of CSM increases slower than logarithmically with K. The percentage of query sessions in which clicks are ordered by position can be seen as an upper bound under the assumption that a user scans search results sequentially from top to bottom. CSM does not make this assumption, and, as a result, is able to reach and surpass this upper bound, achieving 96.26% recall at $K = 1024$.

Answering RQ1(a), we conclude that recall of CSM increases slower than logarithmically with K, starting from 47.24% at $K = 1$ and reaching 96.26% at $K = 1024$, which is higher than recall under the sequential assumption (92.73%).

5.2 Experiment 1(b)

Figure 3 shows recall at different values of K (in linear and logarithmic scales) for (i) all query sessions (black, solid), (ii) query sessions in which clicks are ordered by position (blue, solid), and (iii) query sessions in which clicks are not ordered by position (red, solid). Dashed lines show percentages of query sessions in the corresponding groups, in which clicks on results happen in the order these results are presented on a SERP. Obviously, the second group has 100% of query sessions with clicks ordered by position (and, hence, the blue dotted line denotes recall of 1), while the third group has no such sessions (and, hence, the red dotted line denotes recall of 0).

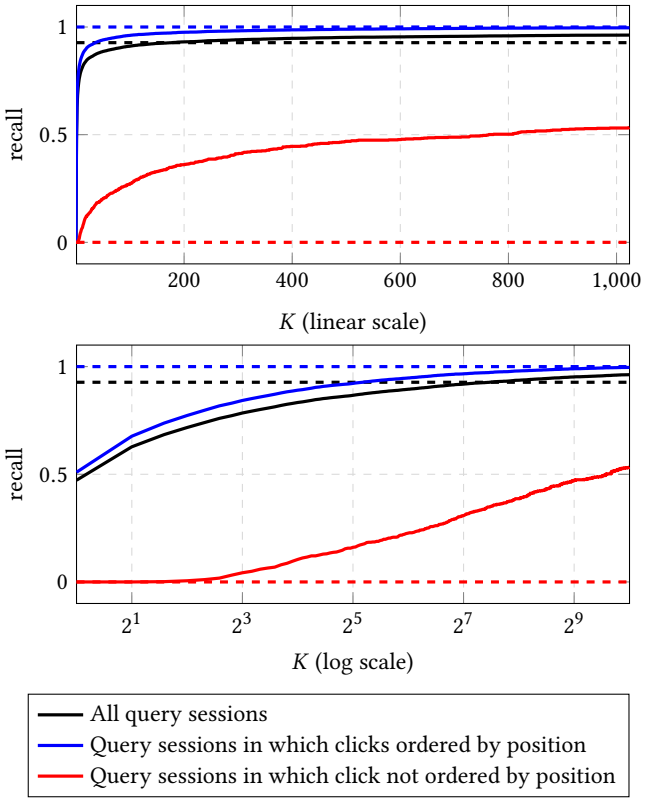

Figure 3: Recall at different values of K for (i) all query sessions (black, solid) (ii) query sessions in which clicks are ordered by position (blue, solid), and (iii) query sessions in which clicks are not ordered by position (red, solid). Dashed lines show percentages of query sessions in the corresponding groups, in which clicks on results happen in the order these results are presented on a SERP.

As we find in §5.1, CSM assigns the highest probability to the observed click sequence for 47.24% of query sessions. If we consider only query sessions in which clicks are ordered by position, this percentage goes up to 50.94% and then increases slower than logarithmically with K (see Figure 3, bottom plot) achieving 99.62% at $K = 1024$. However, if we consider only query sessions in which clicks are not ordered by position, this percentage goes down to 0 and then increases logarithmically with K for $K \geq 5$ (see Figure 3, bottom plot), achieving 53.37% at $K = 1024$.

It is to be expected that predicting click sequences, where clicks are not ordered by position, is a much more difficult task than predicting ordered clicks. First, in our training data the number of ordered click sequences is greater than the number of unordered click sequences (see Table 1 for the statistic computed on the test set; the training set shares the same distribution). Second, the number of possible ordered click sequences is less than the number of possible unordered click sequences. For click sequences of length L, there are

$\binom{N}{L} = \frac{N!}{L!(N-L)!}$ possible ordered click sequences and N^L possible unordered click sequences.

And this is where CSM makes a difference: in more than 50% of cases, the observed click sequence appears in the top $K = 1024$ sequences predicted by CSM. Note that under the assumption that a user scans search results sequentially from top to bottom such sequences cannot be predicted at all (see the red dotted line at zero recall). Even in a simple case of predicting ordered click sequences, CSM almost reaches the perfect recall of 1 for $K = 1024$.

Answering RQ1(b), we conclude that recall of CSM is much higher in query sessions in which clicks follow the presentation order than in those in which users click on higher ranked results after clicking on a lower ranked result.

5.3 Experiment 1(c)

Figure 4 shows recall at different values of K for (i) all query sessions and (ii) query sessions with L clicks. Dashed lines show percentages of query sessions in the corresponding groups, in which clicks on results happen in the order these results are presented on a SERP.

For click sequences of length $L = 0$ and $L = 1$, recall of CSM approaches 1 (already for small values of K). Recall of CSM for sequences of length $L = 2,3,4$ at $K = 1024$ is higher than the percentages of query sessions in which click sequences of length L are ordered by position. For $L = 5$ and $K = 1024$, recall of CSM approaches the percentage of query sessions in which click sequences are of length 5 and are ordered by position. For sequences of length ≥ 2, recall of CSM first increases logarithmically with K (for $K \geq K_0(L)$), and then might increase both faster and slower then logarithmically with K depending on L and the range of K.

We can see that the longer a click sequence is, the more difficult it is to predict such a sequence. This is intuitive and can be explained similarly to §5.2. First, we have more training data for shorter click sequences (see Table 1). Second, the number of possible click sequences of length L increases exponentially with L, making the prediction task more difficult.

Answering RQ1(c), we conclude that recall of CSM is very high in query sessions with a small number of clicks, and lower in query sessions with a larger number of clicks.

5.4 Experiment 1(d)

Figure 5 plots, for different values of K, the total probability of K most probable click sequences predicted by CSM averaged over query sessions in the test set. We write $\sum_{i=1}^{K} P_{\text{CSM}}(s_i)$ to denote this probability.

We find that the total probability of K most probable click sequences grows fast with K, starting from 46.29% at $K = 1$ and 71.69% at $K = 4$, and achieving 91.81% at $K = 128$ and 96.47% at $K = 1024$. Thus, even for modest values of K, the total probability of the K most probable click sequences is not significantly less than 1.

Answering RQ1(d), we conclude that the K most probable click sequences according to CSM provide good means to reason about the whole probability distribution over click sequences predicted by CSM.

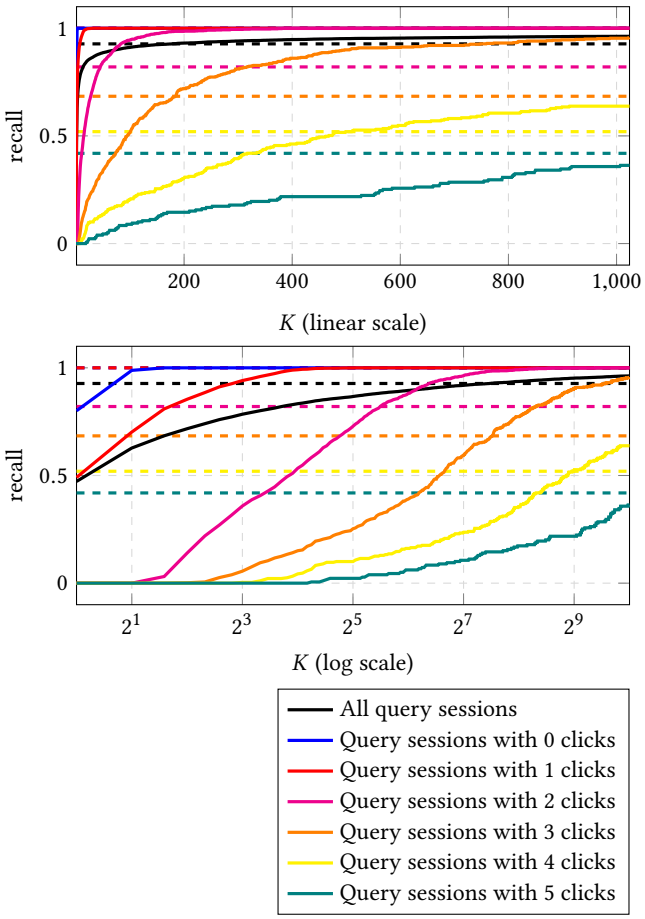

Figure 4: Recall at different values of K for (i) all query sessions and (ii) query sessions with L clicks. Dashed lines show percentages of query sessions in the corresponding groups, in which clicks on results happen in the order these results are presented on a SERP.

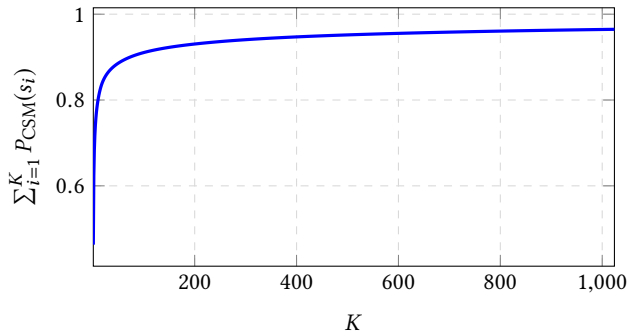

Figure 5: Total probability of K most probable sequences predicted by CSM for different values of K.

5.5 Experiment 2

The results on Task 1 (§2.2) are given in Tables 2 and 3. Table 2 shows perplexity of CSM upon observing a sequence of $\leq L$ clicks. Table 3 shows AUC of CSM on the same task. Recall, that the naive baseline

predicts that a user will make $\leq L$ clicks with a constant probability optimized on the training set.

Table 2: Perplexity of observing a sequence of $\leq L$ clicks. Lower values correspond to better prediction performance.

	$L=0$	$L\leq 1$	$L\leq 2$	$L\leq 3$	$L\leq 4$	$L\leq 5$
Baseline	1.8512	1.7169	1.4450	1.2779	1.1784	1.1068
CSM	1.7155	1.6153	1.3852	1.2438	1.1602	1.1029

Table 3: AUC for the task of predicting whether a user will click on $\leq L$ results.

	$L=0$	$L\leq 1$	$L\leq 2$	$L\leq 3$	$L\leq 4$	$L\leq 5$
Baseline	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
CSM	0.7362	0.7278	0.7353	0.7535	0.7566	0.7795

Both Tables 2 and 3 show that CSM predicts the number of clicks better than the baseline and its performance increases with L (lower perplexity and higher AUC). The latter result is intuitive as more sequences have $\leq L$ clicks for larger L and, thus, the prediction task becomes easier as L grows.

Answering RQ3, we conclude that CSM provides good means to predict the number of clicked results.

5.6 Experiment 3

The results on Task 2 (§2.2) are given in Table 4. The table shows perplexity and AUC of CSM when predicting a sequence of clicks ordered by position.

Table 4: Performance for the task of predicting whether a user will click on results in the order these results are presented on a SERP. Lower values of perplexity and larger values of AUC correspond to better prediction performance.

	Perplexity	AUC
Baseline	1.2984	0.5000
CSM	1.2788	0.6826

Table 4 shows that CSM outperforms the baseline in terms of both perplexity and AUC. Thus, answering RQ3, we conclude that CSM provides good means to predict whether a user will interact with results in the order these results are presented on a SERP.

5.7 Experiment 4

The results on Task 3 (§2.2), i.e., the click prediction task, are given in Table 5. The results for DBN, DCM, CCM, UBM and NCM are according to [5].

Table 5 shows that CSM outperforms DBN, DCM, CCM and UBM by a large margin and matches the performance of NCM, which is reported to be the state-of-the-art click model [5]. Answering RQ4, we conclude that CSM provides good means to predict clicks on search engine results, achieving the state-of-the-art performance.

Table 5: Perplexity for the click prediction task. Lower values correspond to better prediction performance. The results for DBN, DCM, CCM, UBM and NCM are according to [5].

Click model	Perplexity
DBN	1.3510
DCM	1.3627
CCM	1.3692
UBM	1.3431
NCM	1.3318
CSM	1.3312

6 RELATED WORK

We describe two types of related work: user interactions in search and user modeling.

6.1 User interactions in search

Log data from interactive systems such as search engines is one of the most ubiquitous forms of data available, as it can be recorded at little cost [46]. These data have become a popular source for improving the performance of search engines. In particular, logged interactions have been successfully adopted to improve various aspects of search, including document ranking [1, 26, 37], query auto-completion [25, 31] and query suggestion [6, 48], to improve recommender systems [35], optimizing presentations [45], and evaluation [20].

In the context of web search, many types of implicit information interaction behavior have been studied over the years. Early work, e.g., by Craswell et al. [10], focuses on single clicks and, in particular, on the *first* click. And assumes that a user abandons examination of web results upon the first click. Guo et al. [17] expand on this by studying sessions with *multiple* clicks, looking not just at the first click but also at follow-up clicks, the last click and dependencies between clicks, reflecting more complex information behavior.

There is a very broad spectrum of research that studies and tries to interpret information interaction behavior that involves multiple clicks, either by also taking additional signals into consideration or by zooming in on specific aspects of sequences of clicks. Examples of the former include work by Huurnink et al. [23] who examine click signals, download behavior, and purchase signals in vertical search and find high degrees of correlation between the three. Time, such as dwell time or time between user actions such as clicks, has been found to be another important source of implicit signals [4]: times elapsed between user actions provide means to measure user satisfaction at the result level [13, 27], session level [13, 18] and system level [7, 41]. And beyond that, on mobile or screen-less devices there is range of interaction signals that are different from signals familiar from desktop environment – due to the context of use and due to gesture- and voice-based control, such as swipes, touch and voice conversations – and that have not been studied extensively [29]. Our work differs from these publications as we remain focused on click signals only and especially on sequences click signals.

Relevant examples of the studies that zoom in on specific aspects of multiple click behavior include work on repeat behavior such as repeated examinations or clicks [35, 50], which can be interpreted as strong evidence of the value ascribed to the result being examined or

clicked again. In a similar vein, Scaria et al. [40] consider back clicks and last clicks; in their view, back clicks suggest a lack of progress on the current navigational path and, depending on contextual factors, last clicks mark success or failure. Hassan et al. [19] and Odijk et al. [36] focus on aspects of click sequences, including the number of clicks, their dwell time, and features to capture whether the user was clicking on the same results or results from the same domain multiple times, indicative of difficulty locating a particular resource. Williams and Zitouni [47] examine whether sequences of user interactions over time can be used to differentiate between good and abandonment and train an LSTM to distinguish between the two types of behavior. Especially relevant for our paper is the work by Wang et al. [43], who consider non-sequential examination and click behavior, both through an eye-tracking study and a log-based study. They arrive at several behavioral principles, for instance(i) between adjacent clicks, users tend to examine search results in a single direction without changes, and the direction is usually consistent with that of clicks; and (ii) although the examination behavior between adjacent clicks can be regarded as locally unidirectional, users may skip a few results and examine a result at some distance from the current one following a certain direction.

6.2 Modeling user interactions

To understand, describe and predict various types of user interactions discussed above, a number of user interaction models have been proposed aimed at modeling clicks [9], mouse movements [11], dwell time [27, 32], etc.

So far, modeling user clicks in search has attracted the most attention [9]. Click models usually represent clicks as binary random variables and construct a *probabilistic graphical model* (PGM) that describes the dependencies between clicks and other (usually hidden) random variables, such as attractiveness (i.e., whether a snippet is attractive to a user given a query) and examination (i.e., whether a snippet is examined by a user) [8, 10, 12, 16, 17]. The advantage of PGM-based click models is that they intuitively describe user click behavior and can predict future clicks based on past observations [9]. Some click models take into account the order in which a user interacts with the results in order to better model and predict clicks [33, 43, 44, 49, 51]. However, such models either do not aim at predicting click sequences [33, 43, 44, 49] or consider only very short sequences of clicks [51].

Recently, neural click models have been proposed [5, 53]. The advantage of these models is that they do not require manually constructed PGMs to describe and predict user clicks, but rely on raw click data to learn hidden click patterns. Neural click models have better click prediction accuracy, but suffer from uninterpretability of the learned neural model as opposed to easily interpretable PGM-based click models. Also, as before, neural click models cannot predict sequences of clicks.

In addition to clicks, mouse movements between search results and various search-related timings have been studied and modeled. The probability of hovering over one element of a SERP after hovering over another element is predicted using the Farley-Ring model in [11]. Dwell time is modeled though Weibull and gamma distributions in [27, 32]. More timings, such as time between clicks, time to first/last click, etc., are considered in [4], where, in addition to the above distribution-based models, a context bias is modeled using neural networks.

What our work adds on top of the work listed above is our focus on *sequences* of clicks, and in particular on describing a set of *probably correct* click sequences.

7 CONCLUSION AND FUTURE WORK

In this paper, we studied the problem of predicting sequences of user interactions and, in particular, sequences of clicks. We formally defined the problem of click sequence prediction and introduced the notion of probably correct click sequences. Furthermore, we proposed CSM, a neural network based model for predicting a probability distribution over click sequences. We advocated for using the K most probable click sequences predicted by CSM as a set probably correct click sequences. And suggested to use these K click sequences to reason about the properties of the probability distribution over click sequences, such as the expected number of clicks and the expected order of clicks.

We evaluated the quality of CSM on a publicly available dataset. First, we showed that even for modest thresholds the K (larger than the threshold) most probable click sequences predicted by the CSM constitute a substantial part of the total probability mass assigned by the CSM to all possible click sequences, and thus can be regarded as probably correct click sequences predicted by CSM. We proposed to judge the success of a click sequence model $\mathcal{M}$ by the fact that the observed click sequence occurs in the list of the K most probable click sequences predicted by $\mathcal{M}$. We measured performance of CSM using recall@K, a metric that is also used to evaluate the performance of approximate nearest neighbor search methods [24, 34]. Our results showed that recall@K grows fast with K, starting from 47.24% at $K = 1$ and reaching 96.26% at $K = 1024$. We also found that recall@K increases slower with K in query sessions with larger number of clicks and in query sessions where users click on higher ranked results after clicking on a lower ranked result.

We also evaluated CSM on three prediction tasks: (i) predicting the number of clicks, (ii) predicting non-consecutive click sequences, and (iii) predicting clicks. The first two tasks were proposed in our work for the first time and the last one is a standard task used to evaluate click models. We found that CSM shows reasonable performance on the first two tasks, outperforming naive baselines that predict (i) that a user will click on $\leq L$ results, and (ii) that a user will click on the results in a non-consecutive order with a constant probability optimized on the training set. Finally, we observed that CSM reaches state-of-the-art performance on the task of predicting clicks, outperforming PGM-based click models DBN [8], DCM [17], CCM [17] and UBM [12] by a large margin, and matching the results of the recently proposed NCM [5], which is also implemented as a neural network.

In contrast to previous studies, which focus on modeling and predicting separate interaction events (e.g., a click on a result or mouse movement between two results) and properties of these separate events (e.g., time between clicks), our work focuses on understanding, modeling and predicting sequences of these events.

As to future work, we see two main directions:(i) to consider other representations of the user's query and the results returned by a search engine, and (ii) to extend CSM to non-linear SERP layouts. The user's query can be represented by its text, and the results by

their content (title, snippet and main content). We believe that using content-based representations will allow us to learn more interesting dependencies between the results, and improve the performance for rare queries. The encoder proposed in §3.1 makes use of the fact that search results are presented as a list. Recommender systems present their results using non-linear layouts. Generalizing the encoder will make CSM suitable for applications outside of web search.

Acknowledgements

This research was partially supported by Ahold Delhaize, Amsterdam Data Science, the Bloomberg Research Grant program, the China Scholarship Council, the Criteo Faculty Research Award program, Elsevier, the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement nr 312827 (VOX-Pol), the Google Faculty Research Awards program, the Microsoft Research Ph.D. program, the Netherlands Institute for Sound and Vision, the Netherlands Organisation for Scientific Research (NWO) under project nrs CI-14-25, 652.002.001, 612.001.551, 652.001.003, and Yandex. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Eugene Agichtein, Eric Brill, and Susan Dumais. 2006. Improving web search ranking by incorporating user behavior information. In *SIGIR*. ACM, 19–26.
[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).
[3] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* 5, 2 (1994), 157–166.
[4] Alexey Borisov, Ilya Markov, Maarten de Rijke, and Pavel Serdyukov. 2016. A context-aware time model for web search. In *SIGIR*. ACM, 205–214.
[5] Alexey Borisov, Ilya Markov, Maarten de Rijke, and Pavel Serdyukov. 2016. A neural click model for web search. In *WWW*. International World Wide Web Conferences Steering Committee, 531–541.
[6] Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. 2008. Context-aware query suggestion by mining click-through and session data. In *KDD*. ACM, 875–883.
[7] Olivier Chapelle, Thorsten Joachims, Filip Radlinski, and Yisong Yue. 2012. Large-scale validation and analysis of interleaved search evaluation. *TOIS* 30, 1 (2012), 6.
[8] Olivier Chapelle and Ya Zhang. 2009. A dynamic bayesian network click model for web search ranking. In *WWW*. ACM, 1–10.
[9] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. 2015. *Click models for web search*. Morgan & Claypool.
[10] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. 2008. An experimental comparison of click position-bias models. In *WSDM*. ACM, 87–94.
[11] Fernando Diaz, Ryen White, Georg Buscher, and Dan Liebling. 2013. Robust models of mouse movement on dynamic web search results pages. In *CIKM*. ACM, 1451–1460.
[12] Georges E. Dupret and Benjamin Piwowarski. 2008. A user browsing model to predict search engine click data from past observations.. In *SIGIR*. ACM, 331–338.
[13] Steve Fox, Kuldeep Karnawat, Mark Mydland, Susan Dumais, and Thomas White. 2005. Evaluating implicit measures to improve web search. *TOIS* 23, 2 (2005), 147–168.
[14] Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711* (2012).
[15] Artem Grotov, Aleksandr Chuklin, Ilya Markov, Luka Stout, Finde Xumara, and Maarten de Rijke. 2015. A comparative study of click models for web search. In *CLEF*. Springer, 78–90.
[16] Fan Guo, Chao Liu, Anitha Kannan, Tom Minka, Michael Taylor, Yi-Min Wang, and Christos Faloutsos. 2009. Click chain model in web search. In *WWW*. ACM, 11–20.
[17] Fan Guo, Chao Liu, and Yi Min Wang. 2009. Efficient multiple-click models in web search. In *WSDM*. ACM, 124–131.
[18] Ahmed Hassan. 2012. A semi-supervised approach to modeling web search satisfaction. In *SIGIR*. ACM, 275–284.
[19] Ahmed Hassan, Ryen W. White, Susan T. Dumais, and Yi-Min Wang. 2014. Struggling or exploring?: Disambiguating long search sessions. In *WSDM*. ACM, 53–62.
[20] Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. 2012. Estimating interleaved comparison outcomes from historical click data. In *CIKM*. ACM, 1779–1783.
[21] Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. 2013. Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. *Information Retrieval Journal* 16, 1 (2013), 63–90.
[22] Jeff Huang, Ryen W. White, and Susan Dumais. 2011. No clicks, no problem: Using cursor movements to understand and improve search. In *SIGCHI*. ACM, 1225–1234.
[23] Bouke Huurnink, Laura Hollink, Wietske van den Heuvel, and Maarten de Rijke. 2010. Search behavior of media professionals at an audiovisual archive: A transaction log analysis. *JASIST* 61, 6 (June 2010), 1180–1197.
[24] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2011. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence* 33, 1 (2011), 117–128.
[25] Jyun-Yu Jiang, Yen-Yu Ke, Pao-Yu Chien, and Pu-Jen Cheng. 2014. Learning user reformulation behavior for query auto-completion. In *SIGIR*. ACM, 445–454.
[26] Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *KDD*. ACM, 133–142.
[27] Youngho Kim, Ahmed Hassan, Ryen W White, and Imed Zitouni. 2014. Modeling dwell time to predict click-level satisfaction. In *WSDM*. ACM, 193–202.
[28] Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2015).
[29] Julia Kiseleva and Maarten de Rijke. 2017. Evaluating personal assistants on mobile devices. In *CAIR*. ACM.
[30] Elad Kravi, Ido Guy, Avihai Mejer, David Carmel, Yoelle Maarek, Dan Pelleg, and Gilad Tsur. 2016. One query, many clicks: Analysis of queries with multiple clicks by the same user. In *CIKM*. ACM, 1423–1432.
[31] Yanen Li, Anlei Dong, Hongning Wang, Hongbo Deng, Yi Chang, and ChengXiang Zhai. 2014. A two-dimensional click model for query auto-completion. In *SIGIR*. ACM, 455–464.
[32] Chao Liu, Ryen W White, and Susan Dumais. 2010. Understanding web browsing behaviors through weibull analysis of dwell time. In *SIGIR*. ACM, 379–386.
[33] Yiqun Liu, Xiaohui Xie, Chao Wang, Jian-Yun Nie, Min Zhang, and Shaoping Ma. 2016. Time-aware click model. *TOIS* 35, 3 (2016), 16:1–16:24.
[34] Marius Muja and David G Lowe. 2009. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP* 2, 331-340 (2009), 2.
[35] Doug Oard and Jinmook Kim. 1998. Implicit feedback for recommender systems. In *Proceedings of the AAAI Workshop on Recommender Systems*. AAAI.
[36] Daan Odijk, Ryen W. White, Ahmed Hassan Awadallah, and Susan T. Dumais. 2015. Struggling and success in web search. In *CIKM*. ACM, 1551–1560.
[37] Neil O'Hare, Paloma de Juan, Rossano Schifanella, Yunlong He, Dawei Yin, and Yi Chang. 2016. Leveraging user interaction signals for web image search. In *SIGIR*. ACM, 559–568.
[38] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026* (2013).
[39] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *ICML*. 1310–1318.
[40] Aju Thalappillil Scaria, Rose Marie Philip, Robert West, and Jure Leskovec. 2014. The last click: Why users give up information network navigation. In *WSDM*. ACM, 213–222.
[41] Anne Schuth, Katja Hofmann, and Filip Radlinski. 2015. Predicting search satisfaction metrics with interleaved comparisons. In *SIGIR*. ACM, 463–472.
[42] Anne Schuth, Harrie Oosterhuis, Shimon Whiteson, and Maarten de Rijke. 2016. Multileave gradient descent for fast online learning to rank. In *WSDM*. ACM, 457–466.
[43] Chao Wang, Yiqun Liu, Meng Wang, Ke Zhou, Jian-yun Nie, and Shaoping Ma. 2015. Incorporating non-sequential behavior into click models. In *SIGIR*. ACM, 283–292.
[44] Kuansan Wang, Nikolas Gloy, and Xiaolong Li. 2010. Inferring search behaviors using partially observable Markov (POM) model. In *WSDM*. ACM, 211–220.
[45] Yue Wang, Dawei Yin, Luo Jie, Pengyuan Wang, Makoto Yamada, Yi Chang, and Qiaozhu Mei. 2016. Beyond ranking: Optimizing whole-page presentation. In *WSDM*. ACM, 103–112.
[46] Ryen W. White. 2016. *Interactions with search systems*. Cambridge University Press.
[47] Kyle Williams and Imed Zitouni. 2017. Does that mean you're happy?: RNN-based modeling of user interaction sequences to detect good abandonment. In *CIKM*. ACM, 727–736.
[48] Wei Wu, Hang Li, and Jun Xu. 2013. Learning query and document similarities from click-through bipartite graph with metadata. In *WSDM*. ACM, 687–696.
[49] Xiaohui Xie, Jiaxin Mao, Maarten de Rijke, Ruizhe Zhang, Min Zhang, and Shaoping Ma. 2018. Constructing an interaction behavior model for web image search. In *SIGIR*. ACM.
[50] Danqing Xu, Yiqun Liu, Min Zhang, Shaoping Ma, and Liyun Ru. 2012. Incorporating revisiting behaviors into click models. In *WSDM*. ACM, 303–312.
[51] Wanhong Xu, Eren Manavoglu, and Erick Cantu-Paz. 2010. Temporal click model for sponsored search. In *SIGIR*. ACM, 106–113.
[52] Yandex. 2014. Personalized web search challenge. https://www.kaggle.com/c/yandex-personalized-web-search-challenge. (2014).
[53] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. 2014. Sequential click prediction for sponsored search with recurrent neural networks. In *AAAI*. AAAI Press, 1369–1375.