UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

A neural click model for web search

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

DOI
10.1145/2872427.2883033
Publication date
2016
Document Version
Author accepted manuscript
Published in
WWW'16

Link to publication

Citation for published version (APA):
Borisov, A., Markov, I., de Rijke, M., & Serdyukov, P. (2016). A neural click model for web search. In *WWW'16: proceedings of the 25th International Conference on World Wide Web : May 11-15, 2016, Montreal, Canada* (pp. 531-541). Association for Computing Machinery. https://doi.org/10.1145/2872427.2883033

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:07 Oct 2026

A Neural Click 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

Understanding user browsing behavior in web search is key to improving web search effectiveness. Many click models have been proposed to explain or predict user clicks on search engine results. They are based on the *probabilistic graphical model* (PGM) framework, in which user behavior is represented as a sequence of observable and hidden events. The PGM framework provides a mathematically solid way to reason about a set of events given some information about other events. But the structure of the dependencies between the events has to be set manually. Different click models use different hand-crafted sets of dependencies.

We propose an alternative based on the idea of distributed representations: to represent the user's information need and the information available to the user with a *vector state*. The components of the vector state are learned to represent concepts that are useful for modeling user behavior. And user behavior is modeled as a sequence of vector states associated with a query session: the vector state is initialized with a query, and then iteratively updated based on information about interactions with the search engine results. This approach allows us to directly understand user browsing behavior from click-through data, i.e., without the need for a predefined set of rules as is customary for PGM-based click models.

We illustrate our approach using a set of *neural click models*. Our experimental results show that the *neural click model* that uses the same training data as traditional PGM-based click models, has better performance on the click prediction task (i.e., predicting user click on search engine results) and the relevance prediction task (i.e., ranking documents by their relevance to a query). An analysis of the best performing neural click model shows that it learns similar concepts to those used in traditional click models, and that it also learns other concepts that cannot be designed manually.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*retrieval models*

Keywords

Click modeling; Deep learning; Web search; User behavior; Distributed representations; Recurrent neural networks

Copyright is held by the International World Wide Web Conference Committee (IW3C2). IW3C2 reserves the right to provide a hyperlink to the author's site if the Material is used in electronic media.
WWW 2016, April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4143-1/16/04.
DOI: http://dx.doi.org/10.1145/2872427.2883033.

1. INTRODUCTION

Understanding users' interaction behavior with a complex Information Retrieval (IR) system is key to improving its quality. In web search, the ability to accurately predict the behavior of a particular user with a certain information need, formulated as a query, in response to a search engine result page allows search engines to construct result pages that minimize the time that it takes users to satisfy their information needs, or increase the probability that users click on sponsors' advertisements.

Recently, many models have been proposed to explain or predict user behavior in web search; see [10] for an overview. These models, also called *click models* as the main observed user interaction with a search system concerns clicks, are used for click prediction and they may help in cases where we do not have real users to experiment with, or prefer not to experiment with real users for fear of hurting the user experience. Click models are also used to improve document ranking (i.e., infer document relevance from clicks predicted by a click model) [4, 12], improve evaluation metrics (e.g., model-based metrics) [5, 8, 35] and to better understand a user by inspecting the parameters of click models [13].

Existing click models are based on the *probabilistic graphical model* (PGM) framework [23], in which user behavior is represented as a sequence of observable and hidden events such as clicks, skips and document examinations. The PGM framework provides a mathematically solid way to reason about a set of events given information about other events. The structure of the dependencies between the events has to be set manually. Different click models use different hand-crafted sets of dependencies (represented as probabilistic graphical models), while all of them are, by necessity, simplifications and likely to miss key aspects of user behavior.

We propose an alternative to the PGM-based approach—the *distributed representation* (DR) approach—in which user behavior is represented as a sequence of *vector states* that capture the user's information need and the information consumed by the user during search. These vector states can describe user behavior from more angles than the binary events used in PGM-based models (such as whether a user examined a document, or whether a user is attracted by a document), which makes them attractive for learning more complex patterns of user behavior than those hard-coded in existing click models.

We illustrate the distributed representation-based approach using a set of *neural click models*, and compare them against traditional PGM-based click models on a click prediction task (i.e., predicting user clicks on search engine results) and a relevance prediction task (i.e., ranking documents by their relevance to a query). Our experimental results show (1) that the neural click model that uses the same training data as traditional PGM-based click models has better performance on both the click prediction task and the relevance

prediction task than the PGM-based click models; and (2) that the performance of this neural click model can be further improved by incorporating behavioral information over all query sessions that is (a) generated by a particular query, and (b) contains a particular document. We also conduct an analysis of the model's internal workings, which shows that our neural click models learn concepts such as "the current document rank" and "the distance to the previous click," which are used in the *user browsing model* [13], a state of the art PGM-based click model. We also show that the neural click models learn other concepts that cannot be designed manually.

The main contribution of our work is the introduction of a distributed representation-based approach for modeling user behavior and several neural click models, which learn patterns of user behavior directly from interaction data, unlike conventional PGM-based click models that require these patterns to be set manually.

2. RELATED WORK

We discuss two main types of related work: modeling click behavior of web search users and learning distributed representations of concepts.

2.1 Click models

Existing click models are based on the *probabilistic graphical model* (PGM) framework [23], in which user behavior is represented as a sequence of observable and hidden events such as clicks, skips and document examinations. Most probabilistic models of user behavior distinguish between two events (usually assumed independent): a document is *examined* by a user (E_d) and a document is *attractive* to a user (A_d).[1] Furthermore, most models make the *examination hypothesis* that a user clicks on a document ($C_d = 1$) if, and only if, she examined the document ($E_d = 1$) and was attracted by it ($A_d = 1$).[2] The examination probability is modeled differently by different click models, while the attractiveness probability is usually modeled with a parameter $\alpha_{q,d}$ that depends on a query and a document.

The *cascade model* (CM) [11] assumes that a user scans a *search engine result page* (SERP) from top to bottom until she finds a relevant document on which she clicks. In its canonical form, CM postulates that "a user who clicks never comes back, and a user who skips always continues," which limits its applicability to query sessions with exactly one click. This problem has been addressed in the *user browsing model* (UBM) [13], the *dynamic Bayesian network* (DBN) model [4], *dependent click model* (DCM) [19] and *click chain model* (CCM) [18] that use CM as their backbone. UBM introduces a set of examination parameters $\gamma_{r,r-r'}$ $(0 \leq r' < r \leq 10)$, and defines the examination probability of a document at rank r given that the previous click was at rank r' as $\gamma_{r,r-r'}$ ($r' = 0$ if none of the documents above r were clicked). DBN introduces a continuation parameter γ and per query-document pair *satisfactoriness* parameters $\beta_{q,d}$ that describe the *actual relevance* of the document d to the query q as opposed to the *attractiveness* parameters $\alpha_{q,d}$ that describe the *perceived relevance* of the document d to the query q. The examination probability of a document d is defined as the probability that a user was not satisfied by the documents ranked higher than d multiplied by the continuation parameter γ.

[1]The two-stage model also assumes that there is a skimming event prior to examination [25].

[2]Some recent click models [7, 34] relax this assumption to account for noise in user clicks: a user might click on an unattractive document or skip an attractive document because of carelessness.

Recently, a wide range of click models have been proposed that exploit additional information, e.g., information about the user, her current task, the result presentation, the content of results, and other search characteristics. These include the *personalized click model* [30], the *task-centric click model* [37], *intent-aware* modifications of UBM and DBN [9], the *federated click models* [6], the *vertical-aware click model* [33], the *content-aware click model* [34], and *noise-aware* modifications of UBM and DBN [7].

Our work differs from the work discussed above in two important respects. First, we model user browsing behavior on a SERP as a sequence of vectors. Such a representation allows us to describe user behavior from more angles than a sequence of predefined binary events, which is common in existing click models. Second, our approach does not require a manually designed set of rules that describes user browsing behavior on a SERP—such rules constitute the key ingredient of existing click models based on probabilistic graphical models. Instead, we learn such rules directly from past user sessions, which allows us to capture more complex patterns of user behavior than the ones that are set manually.

2.2 Distributed representations of concepts

The idea of modeling behavioral phenomena as an emergent process of interconnected network activities of simple units was originally introduced by cognitive scientists [14], and later developed into the *parallel distributed processing* approach [29], which forms the basis of the artificial neural networks used in image recognition [24], speech recognition [16], machine translation [31] and other fields [1]. The main idea is to represent the input data with one or many vectors, whose components (which may not be interpretable alone) work together to represent concepts that are useful in the task under consideration; these vectors are called *distributed (vector) representations* of the input data.

Recent work that demonstrates the importance of learning distributed representations is due to Mikolov et al. [27]; the authors represent words with vectors and train a neural network to predict the vector of a word given the vectors of surrounding words. The resulting word-specific vectors capture many linguistic regularities, which make them useful for a broad range of applications.

In neural image processing, distributed representations of images are learned directly from image pixels. These image-specific vectors capture many visual regularities, which make them useful for image classification [24]. In neural speech recognition, distributed representations of audio signals are learned from the raw audio data, and then converted into sequences of words [16]. In neural machine translation, a sequence of words in one language is first transformed into an internal distributed representation and then converted into a sequence of words in another language [31].

Different applications use different network architectures to construct effective representations of their data. *Convolutional neural networks* (CNN) are used in image processing to abstract from the exact pixel locations and generalize to unseen images with similar pixel configurations [24]. *Recurrent neural networks* (RNN) are used in language modeling [26], speech recognition [16] and machine translation [31] to process word sequences of arbitrary length.

In this work we introduce distributed representations of user information needs and search results for modeling user browsing behavior in web search. To the best of our knowledge this is the first time such representations have been developed for this purpose.

3. METHOD

Below, we propose a general *neural click model framework*, in which user behavior is modeled as a sequence of distributed repre-

sentations of the user's information need and the information consumed by the user during search. The principal advantage of the proposed framework over existing click models is that patterns of user behavior can be learned directly from interaction data, which allows us to capture more complex patterns of user behavior than the ones that are hard-coded in existing click models.

3.1 Neural click model framework

We model user browsing behavior in web search as a sequence of vector states $(\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2, \dots)$ that describes the information consumed by the user as it evolves within a query session, i.e., $\mathbf{s}_{r-1}$ denotes the information consumed by the user before examining document d_r at rank r. We initialize the vector state $\mathbf{s}_0$ with a user query q, and iteratively update a vector state $\mathbf{s}_r$ to the vector state $\mathbf{s}_{r+1}$ based on user interactions i_r and the next document d_{r+1}:

$$\mathbf{s}_0 = \mathcal{I}(q), \quad (1)$$

$$\mathbf{s}_{r+1} = \mathcal{U}(\mathbf{s}_r, i_r, d_{r+1}). \quad (2)$$

We learn the mappings $\mathcal{I}(\cdot)$ and $\mathcal{U}(\cdot)$ to produce vector states $\mathbf{s}_0$, $\mathbf{s}_1, \mathbf{s}_2, \dots$ that are useful for predicting user clicks on a SERP. We predict the probability of a click on document d_{r+1} ($C_{r+1} = 1$) given the query q, interactions $i_1, \dots, i_r$ and documents $d_1, \dots, d_{r+1}$ from the vector state $\mathbf{s}_{r+1}$ using a function $\mathcal{F}(\cdot) \mapsto [0, 1]$:

$$P(C_{r+1} = 1 \mid q, i_1, \dots, i_r, d_1, \dots, d_{r+1}) = \mathcal{F}(\mathbf{s}_{r+1}). \quad (3)$$

Overall, the process of modeling user browsing behavior on a SERP can be unfolded in the following steps.

1. (a) A user starts a search session by issuing a query q.
   (b) The vector state is initialized with q: $\mathbf{s}_0 = \mathcal{I}(q)$; the previous interactions are empty: $i_0 = \varnothing$.
2. (a) The user examines document d_1.
   (b) The vector state is updated with the examined document and the previous interactions: $\mathbf{s_1} = \mathcal{U}(\mathbf{s}_0, i_0, d_1)$.
3. (a) The user clicks on the examined document d_1 with probability $\mathcal{F}(\mathbf{s}_1)$ and skips it with probability $1 - \mathcal{F}(\mathbf{s}_1)$.
   (b) The user interactions i_1 are set using the information about the observed user interactions with d_1.
4. The user continues examining documents at ranks $r > 1$ repeating steps 2 and 3, where the indices 0 and 1 are replaced with $r - 1$ and r, respectively.

See Figure 1 for an illustration.

Notice that similar to many existing click models, we do not explicitly model search abandonment (the user stops examining the SERP), but we train our model to predict low click probabilities for documents that are unlikely to be examined by the user.

The above is a general framework for modeling user browsing behavior on a SERP. In fact, most existing click models can be described by the mappings $\mathcal{I}(\cdot), \mathcal{U}(\cdot)$ and the function $\mathcal{F}(\cdot)$; see Appendix for the descriptions of $\mathcal{I}(\cdot), \mathcal{U}(\cdot)$ and $\mathcal{F}(\cdot)$ used in DBN and UBM. However, our approach differs from UBM and DBN in an important aspect. The mappings $\mathcal{I}(\cdot)$ and $\mathcal{U}(\cdot)$ in UBM and DBN have no parameters that can be tuned during training, which means that all relevant information for predicting clicks on the next documents is included or discarded manually (according to the rules specified when designing the probabilistic graphical model). In our approach, these mappings are learned to collect information that is useful for predicting clicks on the next documents.

Now that we have specified our general approach to modeling user browsing behavior on a SERP, we need to detail how we represent the query q, the interactions i_r and the document d_{r+1} (§3.2), and how we learn the mappings $\mathcal{I}(\cdot)$, $\mathcal{U}(\cdot)$ and the function $\mathcal{F}(\cdot)$ (§3.3).

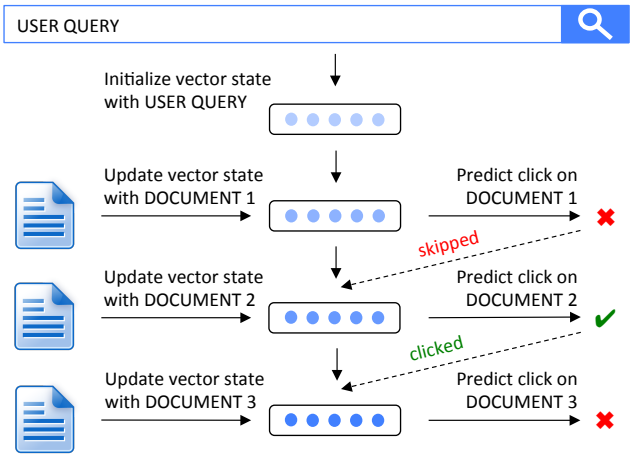


Figure 1: Modeling user browsing behavior on a SERP in the neural click model framework.

3.2 Representations of queries, documents and interactions

We describe three sets of representations for a query q, a document d and interactions i.[3] Set 1 (QD) operates on the basis of query-document pairs similarly to traditional click models (e.g., DBN, UBM). Set 2 (QD+Q) extends Set 1 by considering all query sessions generated by the query q (including ones whose SERPs do not contain the document d). Set 3 (QD+Q+D) extends Set 2 by considering all query sessions whose SERPs contain the document d (including ones generated by queries other than the query q).

3.2.1 *Set 1 (QD)*

The first set of representations operates at the level of query-document pairs. It uses a trivial representation $\mathbf{q}^{(1)}$ of the query q, a query-dependent representation $\mathbf{d}^{(1)}$ of the document d and a click-based representation $\mathbf{i}^{(1)}$ of the interactions i.

Represention $\mathbf{q}^{(1)}$. We represent the query q with a zero vector of size 1.

Represention $\mathbf{d}^{(1)}$. We represent the document d by historical user interactions, observed on SERPs generated by the query q and containing the document d. User interactions on a SERP can be described in a number of ways: by clicks, dwell times, mouse movements, etc. We describe them by *click patterns*—the sets of documents that received a click—because clicks are the most widely logged form of user interaction. For example, the clicks on the first and third positions define the click pattern $[1, 0, 1, 0, 0, \dots]$; the click on the second position defines the click pattern $[0, 1, 0, 0, \dots]$.

Since there are $2^{10} = 1024$ possible click patterns and a document can be presented on any of the 10 positions, we represent the document d with a vector of size of 10240. In each component of the vector, we store the number of times a particular click pattern was observed on SERPs generated by the query q when presenting the document d at a particular rank.

Represention $\mathbf{i}^{(1)}$. We represent interactions i with a binary vector of size 1: the value 0 denotes the fact that a user skipped the previous document, the value 1 denotes the fact that a user clicked on the previous document.

[3] We drop the indices in d_{r+1} and i_r to simplify the notation.

3.2.2 Set 2 (QD+Q)

The second set of representations extends Set 1 by considering all query sessions generated by the query q. This is useful in cases when query sessions generated by the query q have different SERPs, which happens for various reasons: a change in the global ranking algorithm, due to a document index update or due to personalization. In particular, the representations used in Set 1 ignore the potentially useful information about user interactions in query sessions generated by the query q, whose SERPs do not contain the document d. Set 2 aggregates information about all query sessions generated by the query q in a representation $\mathbf{q}^{(2)}$. The representations of the document d and the interactions i stay the same as in Set 1, i.e., $\mathbf{d}^{(1)}$ and $\mathbf{i}^{(1)}$ respectively.

Represention $\mathbf{q}^{(2)}$. We represent the query q by click patterns, observed on SERPs generated by the query q. Since there are $2^{10} = 1024$ possible click patterns, we represent the query q with a vector of size 1024. In each component of the vector $\mathbf{q}^{(2)}$, we store the number of times a particular click pattern was observed on SERPs generated by the query q.

3.2.3 Set 3 (QD+Q+D)

The third set of representations extends Set 2 by considering all query sessions whose SERPs contain the document d. This is particularly useful for rare queries, as it allows us to collect behavioral information over a larger number of observations than only considering query sessions generated by the query q, as done in $\mathbf{d}^{(1)}$. This behavioral information can be biased due to the fact that SERPs containing document d can be generated by queries with different intents.

However, most documents are presented in query sessions generated by queries with the same or similar intents. And even the behavioral information collected over query sessions generated by queries with different intents tells, e.g., about global attractiveness of document d, which might be useful for explaining so-called *sudden interest* clicks on a SERP. Thus, Set 3 extends the representation $\mathbf{d}^{(1)}$ by including information about all query sessions containing the document d (which we refer to as a representation $\mathbf{d}^{(3)}$ of the document d). To sum up, Set 3 uses the $\mathbf{q}^{(2)}$ representation of query q, the concatenation of the $\mathbf{d}^{(1)}$ and $\mathbf{d}^{(3)}$ representations of document d and the $\mathbf{i}^{(1)}$ representation of the interactions i.

Representation $\mathbf{d}^{(3)}$. We represent document d by click patterns observed on SERPs that contain the document d (but not necessarily generated by the query q). Since a document can be presented on any of the 10 positions and there are $2^{10} = 1024$ possible click patterns, we represent the document d with a vector of size 10240. In each component of the vector, we store the number of times a particular click pattern was observed on all SERPs containing the document d at a particular rank.

3.3 Implementations of $\mathcal{I}$, $\mathcal{U}$ and $\mathcal{F}$

Now that we have described the representations that we aim to use, we need to detail how we implement the mappings $\mathcal{I}(\cdot), \mathcal{U}(\cdot)$ and the function $\mathcal{F}(\cdot)$ that form a key ingredient of our neural click model framework (§3.1). Following the literature on recurrent neural networks, we propose two configurations: the *RNN configuration* in §3.3.1 and the *LSTM configuration* in §3.3.2. We describe how to learn the parameters of these configurations in §3.3.3.

Below, we use the following notation: we write $\mathbf{q}$ to denote the vector representation of the query q, $\mathbf{d}_r$ to denote the vector representation of the document d_r, $\mathbf{i}_r$ to denote the vector representation of the interactions i_r and $\mathbf{c}_r$ to denote the vector of size 1, whose single component equals $P(C_r = 1 \mid q, i_1, \ldots, i_{r-1}, d_1, \ldots, d_r)$.

3.3.1 RNN configuration

The RNN configuration is illustrated in Figure 2a. It uses a fully-connected layer[4] to initialize the vector state $\mathbf{s}_0$ and a simple recurrent connection to propagate the information from the vector state $\mathbf{s}_r$ to the vector state $\mathbf{s}_{r+1}$:

$$\begin{aligned} \mathbf{s}_0 &= g_1(\mathbf{W}_{qs}\mathbf{q} + \mathbf{b}_1), \\ \mathbf{s}_{r+1} &= g_2(\mathbf{W}_{ss}\mathbf{s}_r + \mathbf{W}_{is}\mathbf{i}_r + \mathbf{W}_{ds}\mathbf{d}_{r+1} + \mathbf{b}_2). \end{aligned}$$

The functions $g_1(\cdot)$ and $g_2(\cdot)$ denote element-wise non-linear transformations.[5] The matrices $\mathbf{W}_{qs}, \mathbf{W}_{ss}, \mathbf{W}_{is}, \mathbf{W}_{ds}$ and the bias vectors $\mathbf{b}_1, \mathbf{b}_2$ are the parameters of $\mathcal{I}(\cdot), \mathcal{U}(\cdot)$, which are to be learned during training.

The probability $\mathbf{c}_{r+1}$ is computed using a fully-connected layer with one output unit and the sigmoid activation function:

$$\mathbf{c}_{r+1} = \sigma(\mathbf{W}_{sc}\mathbf{s}_{r+1} + \mathbf{b}_3). \quad (4)$$

The matrix $\mathbf{W}_{sc}$ and the bias vectors $\mathbf{b}_3$ are the parameters of $\mathcal{F}(\cdot)$, which are to be learned during training. The sigmoid function $\sigma(x)$ is used to ensure that the output falls in the interval (0, 1).

3.3.2 LSTM configuration

A possible problem of the RNN configuration is the *vanishing and exploding gradient* problem described by Bengio et al. [2]: after applying a few non-linear transformations, the norm of the gradient gets either too small (the vanishing gradient problem, where no learning is happening) or too large (the exploding gradient problem, where the values of the network parameters become unstable). To account for this problem, Hochreiter and Schmidhuber [20] propose the *long short-term memory* (LSTM) block. The LSTM block contains a memory cell (i.e., a vector) and three gate units (i.e., vectors of the same size as the memory cell): the *input gate* that filters out irrelevant information in the input vector, the *forget gate* that filters out information in the vector state that is no longer needed and the *output gate* that controls information in the output vector. It has been shown that the gate mechanism helps to alleviate the vanishing and exploding gradient problems; this yields better results than simple recurrent neural networks that do not use it [15, 20, 22].

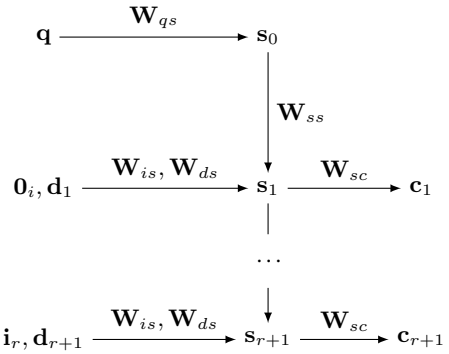
The LSTM configuration is illustrated in Figure 2b. Unlike the RNN configuration, which propagates the information from the vector state $\mathbf{s}_r$ to the vector state $\mathbf{s}_{r+1}$ directly, the LSTM configuration propagates it through the LSTM block, which, as said, helps to mitigate the vanishing and exploding gradient problem. The click probability $\mathbf{c}_r$ is computed as in the RNN configuration (Eq. 4). The parameters of the LSTM configuration, i.e., the parameters of the LSTM block and the parameters of the function $\mathcal{F}(\cdot)$, are learned during training.

3.3.3 Training RNN and LSTM configurations

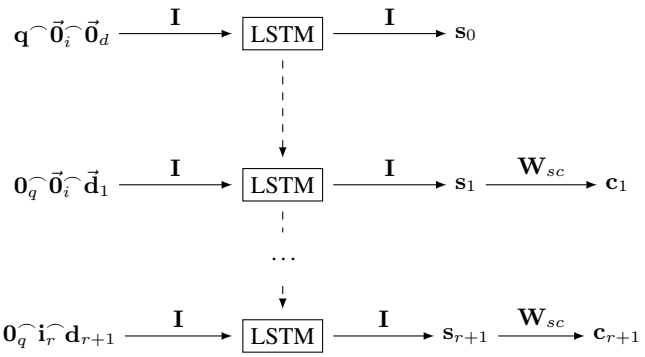
Similar to PGM-based click models, both RNN and LSTM configurations are trained by maximizing the likelihood of observed click events. In particular, we optimize the logarithm of the likelihood function using the *stochastic gradient descent* (SGD) algorithm with mini-batches. The learning rates for each parameter are adjusted according to the *ADADELTA* algorithm [36] (we use the default values of $\epsilon = 10^{-6}$ and $\rho = 0.95$). We also use the *gradient clipping* technique [28] to alleviate the exploding gradient problem [2] (we set the value of the threshold $= 1$).

[4] We refer the reader to [1] for explanations and background material on neural networks.

[5] The possible choices are the sigmoid function, the hyperbolic tangent and the rectified linear unit [1].



(a) The RNN configuration.



(b) The LSTM configuration.

Figure 2: Two possible implementations of the mappings $\mathcal{I}(\cdot)$, $\mathcal{U}(\cdot)$ and the function $\mathcal{F}(\cdot)$. We use $\mathbf{q}$ to denote the vector representation of the query q, $\mathbf{i}_r$ to denote the vector representation of the interactions i_r, $\mathbf{d}_r$ to denote the vector representation of the document d_r, $\mathbf{s}_r$ to denote the vector state after examining the document d_r and $\mathbf{c}_r$ to denote the vector of size 1, whose component equals $P(C_r = 1 \mid q, i_1, \ldots, i_{r-1}, d_1, \ldots, d_r)$. We use the symbol $\frown$ to denote the concatenation of two vectors and write $\vec{\mathbf{0}}_q$, $\vec{\mathbf{0}}_d$, $\vec{\mathbf{0}}_r$ to denote zero vectors of the same size as the query, document and interactions representations, respectively. The matrices $\mathbf{W}_{qs}, \mathbf{W}_{ss}, \mathbf{W}_{is}, \mathbf{W}_{ds}$ denote the projections applied to the vectors $\mathbf{q}$, $\mathbf{s}_r$, $\mathbf{i}_r$, $\mathbf{d}_{r+1}$; the matrix $\mathbf{I}$ denotes an identity matrix. The rectangles labeled LSTM denote the *long short-term memory* block [20] that is used to alleviate the *vanishing and exploding gradient* problem [2]. The matrix $\mathbf{W}_{sc}$ denotes the projection matrix from the vector state $\mathbf{s}_{r+1}$ to the vector $\mathbf{c}_{r+1}$. After each projection a non-linear transformation is applied. The LSTM block contains non-linear transformations inside.

We do not provide the expressions for computing the gradients of the logarithm of the likelihood function with respect to the configurations' parameters, because such expressions can be computed automatically using symbolic differentiation in math packages such as Theano [3].

The main message to take away from this section is that we use distributed representations (sequences of vector states as detailed in §3.1) to model user browsing behavior. We use neural click models to learn those representations. We write $\text{NCM}^{\text{Y}}_{\text{X}}$ to denote a neural click model with representation X (QD, QD+Q, QD+Q+D) and configuration Y (RNN, LSTM). The neural click models can be used to simulate user behavior on a SERP and to infer document relevance from historical user interactions. We estimate the relevance of a document d to a query q using the probability of click on d when d appears on the first position, i.e., $P(C_1 = 1 \mid q, d)$.

4. EXPERIMENTAL SETUP

In this section we describe our experimental setup. The research questions are outlined in §4.1. The dataset and baselines are described in §4.2 and §4.4. The evaluation methodology is given in §4.3. The experiments that we conduct to answer our research questions are outlined in §4.5.

4.1 Research questions

We seek to answer the following research questions.

RQ1 Does the distributed representation-based approach that models user behavior as a sequence of distributed vector representations have better predictive abilities than the PGM-based approach that models user behavior as a sequence of observed and hidden events?

RQ2 Does the LSTM configuration have better learning abilities than the RNN configuration?

RQ3 Does the representation $\mathbf{q}^{(2)}$ of a query q as defined in §3.2.2 provide the means to transfer behavioral information from historical query sessions generated by the query q to new query sessions generated by the query q?

RQ4 Does the representation $\mathbf{d}^{(3)}$ of a document d as defined in §3.2.3 provide the means to transfer behavioral information from historical query sessions whose SERPs contain the document d, to new query sessions whose SERP contain the document d?

RQ5 Do the neural click models produce better document rankings than the PGM-based models?

RQ6 Do the neural click models operate with concepts similar to the ones used in the PGM-based models? In particular,

(a) Do they learn to account for the document rank similarly to most PGM-based click models?

(b) Do they learn to account for the distance to the previous click similarly to the state-of-the-art UBM model?

4.2 Dataset

We use the Yandex Relevance Prediction dataset,[6] which contains 146,278,823 query sessions sampled from logs of Yandex, a major commercial search engine in Russia. There are a total of 30,717,251 unique queries and 117,093,258 unique documents in this dataset. The query sessions are ordered by time. We split the query sessions into two equal parts, and use the earlier query sessions to train click models and the later query sessions to evaluate their prediction performance.

The Yandex Relevance Prediction dataset also contains human-generated binary relevance labels for 41,275 query-document pairs (4991 queries; on average, each query is associated with 8 documents). We use them to evaluate the ranking performance of our click models.

4.3 Evaluation methodology

We consider the click prediction task (i.e., predicting user clicks on search engine results) and the relevance prediction task (i.e., ranking documents by their relevance to a query).

[6] http://imat-relpred.yandex.ru/en/datasets (last visited January 31, 2016).

Click prediction. The task is to predict user clicks on a SERP. The training and test sets consist of query sessions separated by time: the training set comprises query sessions from an earlier period; the test set comprises query sessions from a later period.

Following Dupret and Piwowarski [13], we evaluate the quality of click prediction using the *perpexity* metric, which measures how "surprised" the model is upon observing a particular set of clicks. In particular, we compute perplexity of a model M on a set of sessions S separately for each rank r as follows:

$$p_r(M) \quad = \quad 2^{-\frac{1}{|S|}\sum_{s \in S} \log_2 P_M\left(C_r = c_r^{(s)}\right)},$$

where $P_M(C_r = c_r^{(s)})$ denotes the probability of observing a click event $c_r^{(s)}$ (i.e., click or skip) at rank r in a query session s, as predicted by the click model M. The total perplexity of the click model M is calculated by averaging perplexities over all positions. Lower values of perplexity correspond to higher quality of a model.

We also report the logarithm of the likelihood function $\mathcal{L}(M)$ for each click model M, averaged over all query sessions S in the test set (all click models are learned to optimize the likelihood function):

$$\log \mathcal{L}(M) \quad = \quad \frac{1}{|S|} \sum_{s \in S} \log P_M \left(C_1 = c_1^{(s)}, \ldots, C_n = c_n^{(s)}\right),$$

where $P_M C_1 = c_1^{(s)}, \ldots, C_n = c_n^{(s)})$ denotes the probability of observing a particular sequence of clicks $c_1^{(s)}, \ldots, c_n^{(s)}$ in a query session s according to the click model M. We refer to this statistic as *log-likelihood* in the rest of the paper.

We perform significance testing using a paired t-test on per query session scores; the differences are considered statistically significant for p-values lower than 0.05.

Relevance prediction. Here, the task is to rank documents by their estimated relevance to a query. The training set consists of all query sessions, while the test set consists of query-document pairs that occur at least once in the training set and that have a human generated relevance label. We use the training set to train click models, and the test set to evaluate the quality of document rankings produced by click models.

Following Chapelle and Zhang [4], we evaluate the quality of document rankings using the mean *normalized discounted cumulative gain* (NDCG) [21]. We report NDCG scores at truncation levels 1, 3, 5 and 10.

We perform significance testing using a paired t-test on per query scores; the differences are considered statistically significant for p-values lower than 0.05.

4.4 Baselines

We use DBN, DCM, CCM and UBM as baseline click models.[7] Following [10], we set the priors of all click model parameters to a Beta distribution with $\alpha = 1$ and $\beta = 2$, and the number of EM iterations to 50.

The baseline performance on the click prediction task and the relevance prediction task is given in Tables 1 and 2. Table 1 shows that UBM is the best for click prediction (in terms perplexity and log-likelihood); Table 2 shows that CCM is the best for ranking (in terms of NDCG scores). These results agree with earlier work [17].

4.5 Experiments

We design our experiments to answer our research questions.

[7]We use the PyClick implementation available at https://github.com/markovi/PyClick (last visited January 31, 2016).

Table 1: Performance of the baseline click models on the click prediction task. Differences between all pairs of click models are statistically significant ($p < 0.001$). The best results are given in bold.

Click model	Perplexity	Log-likelihood
DBN	1.3510	−0.2824
DCM	1.3627	−0.3613
CCM	1.3692	−0.3560
UBM	**1.3431**	**−0.2646**

Table 2: Performance of the baseline click models on the relevance prediction task. Improvements of DCM and CCM over DBN and UBM are statistically significant ($p < 0.001$). Differences between (1) DBN and UBM, (2) DCM and CCM are not statistically significant ($p > 0.05$). The best results are given in bold.

	NDCG			
Click model	@1	@3	@5	@10
DBN	0.717	0.725	0.764	0.833
DCM	0.736	0.746	0.780	0.844
CCM	**0.741**	**0.752**	**0.785**	**0.846**
UBM	0.724	0.737	0.773	0.838

Experiment 1. To answer RQ1, we compare the performance of $\text{NCM}_{\text{QD}}^{\text{RNN}}$ against UBM on the click prediction task (UBM is the best performing baseline click model on this task).

Experiment 2. To answer RQ2, we compare the performance of $\text{NCM}_{\text{QD}}^{\text{RNN}}$ against $\text{NCM}_{\text{QD+Q+D}}^{\text{LSTM}}$ on the click prediction task; the best configuration (RNN or LSTM) is then denoted with B_2.

Experiment 3. To answer RQ3, we compare the performance of $\text{NCM}_{\text{QD}}^{\text{B}_2}$, and $\text{NCM}_{\text{QD+Q}}^{\text{B}_2}$ on the click prediction task.

Experiment 4. To answer RQ4, we compare the performance of $\text{NCM}_{\text{QD+Q}}^{\text{B}_2}$, and $\text{NCM}_{\text{QD+Q+D}}^{\text{B}_2}$ on the click prediction task.

Experiment 5. To answer RQ5, we compare the performance $\text{NCM}_{\text{QD}}^{\text{RNN}}$, $\text{NCM}_{\text{QD}}^{\text{LSTM}}$ $\text{NCM}_{\text{QD+Q}}^{\text{B}_2}$ and $\text{NCM}_{\text{QD+Q+D}}^{\text{B}_2}$ against CCM on the relevance prediction task (CCM is the best performing baseline click model on this task).

Experiment 6. To answer RQ6, we analyze vector states $\mathbf{s}_r$ with respect to document ranks and distances to the previous click. The analysis is performed by visualizing the vector states $\mathbf{s}_r$ of the best performing neural click model on the click prediction task in a two-dimensional space using the *t-SNE* dimensionality reduction technique [32].[8] We compute the vector states $\mathbf{s}_r$ on a uniformly sampled subset of the test query sessions.

In all experiments, we use vector states of size 256. The neural click models are trained using mini-batches of 64 query sessions and the parameters specified in §3.3.3.

5. RESULTS

We present the outcomes of the experiments described in §4.5 and provide answers to our research questions stated in §4.1.

5.1 Click prediction task

[8]t-SNE is a state of the art method for visualizing high dimensional data that preserves distances between the data points.

The results for the click prediction task are given in Table 3 and Figures 3, 4. Table 3 lists the overall performance of the click models considered in terms of perplexity and log-likelihood. Figure 3 plots perplexity vs. the number of times a query occurred in the training set. Figure 4 shows perplexity values at different ranks.

Table 3: Performance on the click prediction task. Differences between all pairs of click models are statistically significant ($p < 0.001$). The best results are given in bold.

Click model	Perplexity	Log-likelihood
UBM	1.3431	−0.2646
$\text{NCM}_{\text{QD}}^{\text{RNN}}$	1.3379	−0.2564
$\text{NCM}_{\text{QD}}^{\text{LSTM}}$	1.3362	−0.2547
$\text{NCM}_{\text{QD+Q}}^{\text{LSTM}}$	1.3355	−0.2545
$\text{NCM}_{\text{QD+Q+D}}^{\text{LSTM}}$	**1.3318**	**−0.2526**

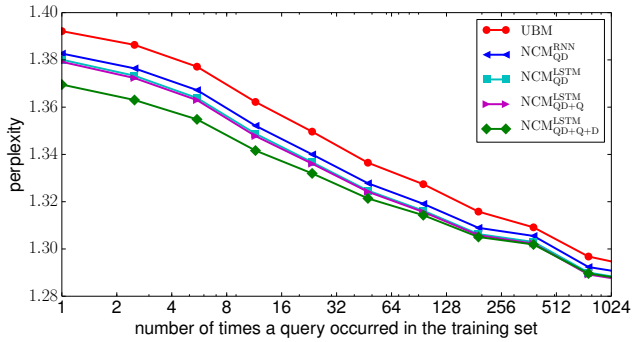


Figure 3: Perplexity for different query frequencies. (Best viewed in color.)

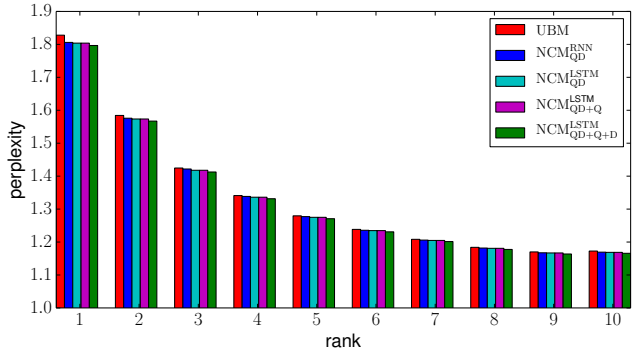


Figure 4: Perplexity for different ranks. (Best viewed in color.)

RQ1. Table 3 shows that $\text{NCM}_{\text{QD}}^{\text{RNN}}$ outperforms the best performing PGM-based click model, UBM, in terms of perplexity and log-likelihood by a large margin. Figure 3 shows that $\text{NCM}_{\text{QD}}^{\text{RNN}}$ has lower perplexity than UBM for all query frequencies. Figure 4 also shows that $\text{NCM}_{\text{QD}}^{\text{RNN}}$ performs better than or, at least, as good as UBM at all ranks.

From the above results we conclude that the DR-based approach, which models user behavior as a sequence of distributed vector representations and learns patterns of user behavior directly from data, has better predictive abilities than the PGM-based approach used in state-of-the-art click models DBN, DCM, CCM and UBM, which models user behavior as a sequence of observed and hidden events and which requires a carefully hand-crafted set of rules describing user behavior (i.e., a probabilistic graphical model).

RQ2. Table 3 shows that $\text{NCM}_{\text{QD}}^{\text{LSTM}}$ outperforms $\text{NCM}_{\text{QD}}^{\text{RNN}}$ in terms of perplexity and log-likelihood. $\text{NCM}_{\text{QD}}^{\text{LSTM}}$ also performs better than, or at least as good as, $\text{NCM}_{\text{QD}}^{\text{RNN}}$ for all query frequencies (Figure 3) and at all ranks (Figure 4).

From the above results, we conclude that the introduction of the LSTM block helps to improve the learning abilities of the neural click models. Therefore, we use the LSTM configuration in the subsequent experiments.

RQ3. Table 3 shows that $\text{NCM}_{\text{QD+Q}}^{\text{LSTM}}$ outperforms $\text{NCM}_{\text{QD}}^{\text{LSTM}}$ in terms of perplexity and log-likelihood by a small but statistically significant margin ($p < 0.001$). Figure 3 shows that $\text{NCM}_{\text{QD+Q}}^{\text{LSTM}}$ consistently outperforms $\text{NCM}_{\text{QD}}^{\text{LSTM}}$ in terms of perplexity for all queries, with larger improvements observed for less frequent queries. In addition, Figure 4 shows that $\text{NCM}_{\text{QD+Q}}^{\text{LSTM}}$ performs as good as $\text{NCM}_{\text{QD}}^{\text{LSTM}}$ in terms of perplexity at all ranks.

From the above results, we conclude that the representation $\mathbf{q}^{(2)}$ of a query q provides the means to transfer behavioral information between query sessions generated by the query q. And this, in turn, helps to better explain user clicks on a SERP.

RQ4. Table 3 shows that $\text{NCM}_{\text{QD+Q+D}}^{\text{LSTM}}$ outperforms $\text{NCM}_{\text{QD+Q}}^{\text{LSTM}}$ in terms of perplexity and log-likelihood. Furthermore, Figure 3 shows that $\text{NCM}_{\text{QD+Q+D}}^{\text{LSTM}}$ consistently outperforms $\text{NCM}_{\text{QD+Q}}^{\text{LSTM}}$ in terms of perplexity for rare and torso queries, with larger improvements observed for less frequent queries. Finally, Figure 4 shows that $\text{NCM}_{\text{QD+Q+D}}^{\text{LSTM}}$ outperforms $\text{NCM}_{\text{QD+Q}}^{\text{LSTM}}$ in terms of perplexity at all ranks.

From the above results, we conclude that the representation $\mathbf{d}^{(3)}$ of a document d provides the means to transfer behavioral information between query sessions, whose SERPs contain the document d. And this, in turn, helps to better explain user clicks on a SERP.

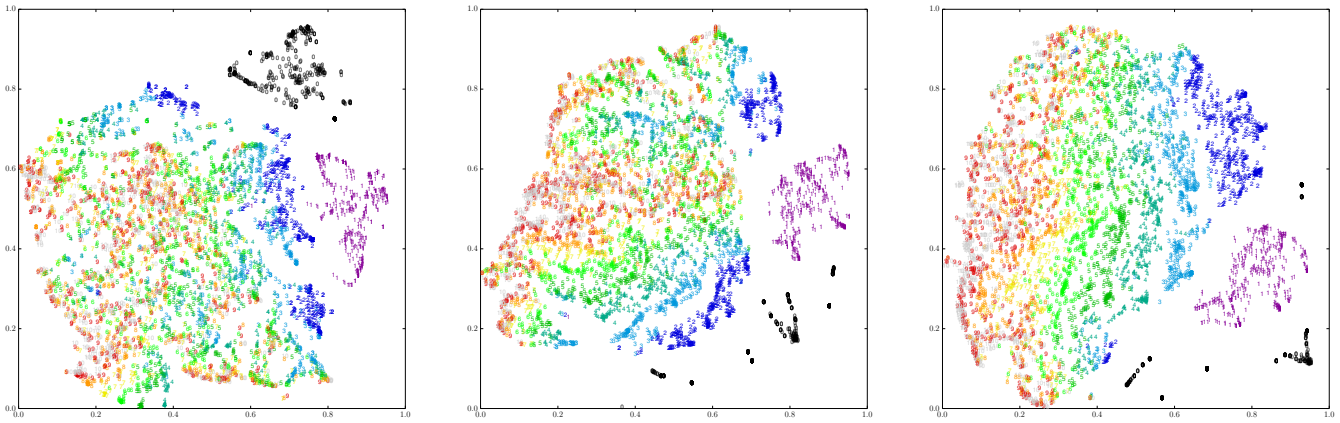
5.2 Relevance prediction task

The results for the relevance prediction task are given in Table 4, which lists the performance of the click models we consider in terms of NDCG scores at truncation levels 1, 3, 5 and 10.

Table 4: Performance on the relevance prediction task. Improvements of (1) $\text{NCM}_{\text{QD}}^{\text{RNN}}$, $\text{NCM}_{\text{QD}}^{\text{LSTM}}$ and $\text{NCM}_{\text{QD+Q}}^{\text{LSTM}}$ over CCM and (2) $\text{NCM}_{\text{QD+Q}}^{\text{LSTM}}$ over the other neural click models are statistically significant ($p < 0.05$). The best results are given in bold.

	NDCG			
Click model	@1	@3	@5	@10
CCM	0.741	0.752	0.785	0.846
$\text{NCM}_{\text{QD}}^{\text{RNN}}$	0.762	0.759	0.791	0.851
$\text{NCM}_{\text{QD}}^{\text{LSTM}}$	0.756	0.759	0.789	0.850
$\text{NCM}_{\text{QD+Q}}^{\text{LSTM}}$	**0.775**	**0.773**	**0.799**	**0.857**
$\text{NCM}_{\text{QD+Q+D}}^{\text{LSTM}}$	0.755	0.755	0.787	0.847

RQ5. Table 4 shows that all neural click models outperform the best performing PGM-based click model, CCM, in terms of NDCG. The differences between $\text{NCM}_{\text{QD}}^{\text{RNN}}$ and $\text{NCM}_{\text{QD}}^{\text{LSTM}}$ are not statistically significant. $\text{NCM}_{\text{QD+Q}}^{\text{LSTM}}$ outperforms $\text{NCM}_{\text{QD}}^{\text{LSTM}}$ and $\text{NCM}_{\text{QD}}^{\text{LSTM}}$ by a large margin. Interestingly, the performance of $\text{NCM}_{\text{QD+Q+D}}^{\text{LSTM}}$ falls behind that of $\text{NCM}_{\text{QD+Q}}^{\text{LSTM}}$.



(a) All query sessions.

(b) Query sessions generated by queries that occur one time in the training set.

(c) Query sessions with no clicks generated by queries that occur one time in the training set.

Figure 5: Two-dimensional t-SNE projections of vector states $\mathbf{s}_r$ for different ranks r. Colors correspond to ranks: black 0; purple 1; dark blue 2; light blue 3; light blue-green 4; green 5; light green 6; yellow 7; orange 8; red 9; grey 10. (Best viewed in color.)

This shows that the neural click models produce better document rankings than the PGM-based models. The differences between the neural click models can be explained as follows. When ranking a query-document pair (q, d), $\text{NCM}^{\text{LSTM}}_{\text{QD}}$ uses behavior information from historical query sessions generated by the query q and whose SERPs contain the document d. $\text{NCM}^{\text{LSTM}}_{\text{QD+Q}}$ also uses behavioral information from all historical query sessions generated by the query q, which helps, e.g., to distinguish highly personalized SERPs and to discount observed clicks in these sessions. $\text{NCM}^{\text{LSTM}}_{\text{QD+Q+D}}$ also uses behavioral information from all historical query sessions, whose SERP contain the document d. However, this global information does not tell us much about the relevance of the document d to the query q. It does, though, inform us about the attractiveness of the document d, which leads to improvements on the click prediction task (see Experiment 4).

5.3 Concepts learned by NCM

The results of the analysis of the $\text{NCM}^{\text{LSTM}}_{\text{QD+Q+D}}$ vector states are given in Figures 5 and 6. Figure 5 shows the t-SNE projections of the vector states $\mathbf{s}_r$ for different ranks r. It contains Figures 5a, 5b and 5c, in which the vector states $\mathbf{s}_r$ are generated for different sets of query sessions: Figure 5a uses a uniformly sampled subset of query sessions in the test set (S_a); Figure 5b uses the query sessions in S_a that are generated by queries that occur one time in the training set (S_b); Figure 5c uses the query sessions in S_b that contain no clicks (S_c). Figure 6 plots the t-SNE projections of the vector state $\mathbf{s}_7$ for different distances to the previous click.[9] Here, we compute the vector states for the query sessions in S_a, and then filter out some vector states to construct a balanced set that contains equal number of vector states for each distance $d = 0, 1, \ldots, 6$.

RQ6 (a). Figure 5a shows how the vector states $\mathbf{s}_r$ for different ranks r are positioned in the space learned by $\text{NCM}^{\text{LSTM}}_{\text{QD+Q+D}}$. We find that the subspaces of $\mathbf{s}_0$ and $\mathbf{s}_1$ are well separated from the subspaces of $\mathbf{s}_r$ computed at lower positions; the subspaces of $\mathbf{s}_2$ and $\mathbf{s}_3$ are also separated from the subspaces of $\mathbf{s}_r$ computed for other ranks, but have a significant overlap with each other. The subspaces of vector states $\mathbf{s}_r$ for ranks $r > 3$ have large overlaps with each other. We hypothesize that this is due to the fact that other

[9]We choose $\mathbf{s}_7$ because the rank 7 is low enough to see a sufficient number of distances but not too low to suffer from sparsity.

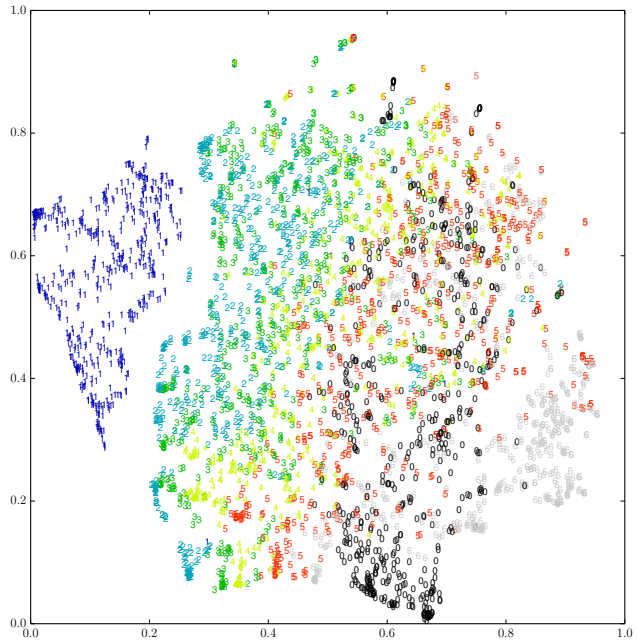


Figure 6: Two-dimensional t-SNE projections of the vector state $\mathbf{s}_7$ for different distances d to the previous click. Colors correspond to distances: black 0; blue 1; blue-green 2; green 3; green-yellow 4; red 5; grey 6. (Best viewed in color.)

factors (e.g., query frequency and clicks on previous documents) are becoming more important for modeling user behavior as rank r increases.

We test our hypothesis by fixing some of these factors. Figure 5b shows that for query sessions generated by queries of similar frequencies (in our case, by queries that occur one time in the training set), the subspaces of the vector states $\mathbf{s}_0, \ldots, \mathbf{s}_6$ are much better separated than the subspaces of the vector states $\mathbf{s}_0, \ldots, \mathbf{s}_6$ computed for query sessions generated by all queries (Figure 5a). Furthermore, Figure 5c shows that for query sessions generated by queries of similar frequencies and having the same click pattern (in

our case, no clicks) the subspaces of $\mathbf{s}_r$ are even better separated by ranks. This is intuitive, because the less information there is to explain user behavior (each query occurred only once and no clicks were observed), the more $\text{NCM}^{\text{LSTM}}_{\text{QD+Q+D}}$ learns to rely on ranks.

Interestingly, Figure 5b shows that the subspaces of the vector states $\mathbf{s}_r$ for $r > 1$ consist of more than one dense clusters (see, e.g., $\mathbf{s}_2$). We explain this by the fact that other factors, such as clicks on previous documents, are also memorized by $\text{NCM}^{\text{LSTM}}_{\text{QD+Q+D}}$. In particular, Figure 5c shows that for query sessions generated by queries of the same frequency and having the same click pattern, the subspaces of the vector states consist of single dense clusters.

From the above results, we conclude that $\text{NCM}^{\text{LSTM}}_{\text{QD+Q+D}}$ learns the concept "current document rank" although we do not explicitly provide this concept in the document representation. In particular, $\text{NCM}^{\text{LSTM}}_{\text{QD+Q+D}}$ strongly relies on the current document rank to explain user browsing behavior on top positions. To explain user browsing behavior at lower positions, $\text{NCM}^{\text{LSTM}}_{\text{QD+Q+D}}$ considers other factors to be more important. However, $\text{NCM}^{\text{LSTM}}_{\text{QD+Q+D}}$ still discriminates most other ranks (we find this by limiting the set of query sessions, which are used to compute the vector states $\mathbf{s}_r$, to query sessions generated by queries of similar frequencies and having a particular set of clicks).

RQ6 (b). Figure 6 shows how the vector states $\mathbf{s}_7$ for different distances to the previous click are positioned in the vector state space learned by $\text{NCM}^{\text{LSTM}}_{\text{QD+Q+D}}$. Here, the distance to the previous click varies from 0 (no clicks above rank 7) to 6 (a click on a document at rank 6). We find that the subspace of the vector states $\mathbf{s}_7$ in query sessions containing a click at rank 6 (which corresponds to distance $d = 1$) is well separated from the subspace of $\mathbf{s}_7$ in other query sessions. The subspace corresponding to query sessions containing a click on the first position ($d = 6$) is also well separated from the subspace corresponding to other query sessions. The subspaces of the vector states $\mathbf{s}_7$ for distances $d \in \{0, 2, \ldots, 5\}$ have a significant overlap with each other. Still, the subspace corresponding to $d = 2$ is well separated from the subspace corresponding to $d = 5$. Interestingly, the subspace corresponding to query sessions containing no clicks on the first six documents ($d = 0$) has a larger overlap with the subspace corresponding to query sessions containing a click on the second position ($d = 5$) than with the subspace corresponding to query sessions containing a click on the first position ($d = 6$).

These results show that $\text{NCM}^{\text{LSTM}}_{\text{QD+Q+D}}$ learns the concept of distance to the previous click, although this information is not explicitly provided in the document representation. In particular, the information about a click on the previous document is particularly important. $\text{NCM}^{\text{LSTM}}_{\text{QD+Q+D}}$ also memorizes whether a user clicked on the first document. $\text{NCM}^{\text{LSTM}}_{\text{QD+Q+D}}$ distinguishes well between the distances 2 and 5, but the subspaces of the vector states computed for the pairs of distances $(2, 3)$, $(3, 4)$, $(4, 5)$ have a large overlap.

Zooming out, what we have learned from the t-SNE analysis of the $\text{NCM}^{\text{LSTM}}_{\text{QD+Q+D}}$ vector states is that $\text{NCM}^{\text{LSTM}}_{\text{QD+Q+D}}$ learns to account (a) for the document rank, and (b) for the distance to the previous click—the two most important concepts used in UBM, the state-of-the-art PGM-based click model. However, these are not the only concepts learned by $\text{NCM}^{\text{LSTM}}_{\text{QD+Q+D}}$. All t-SNE projections contain a large number of clusters (of different density and size) that group vector states by their similarities in the vector state space learned by $\text{NCM}^{\text{LSTM}}_{\text{QD+Q+D}}$. The large clusters are easily interpretable (e.g., they group vector states by rank, distance to the previous click). Smaller clusters are less easily interpretable, but their existence indicates that $\text{NCM}^{\text{LSTM}}_{\text{QD+Q+D}}$ also operates with concepts that are not hard-coded in PGM-based click models.

6. CONCLUSION AND FUTURE WORK

In this paper, we have used *distributed representations* to model user browsing behavior in web search. We represented user browsing behavior as a sequence of vector states that describes the information need of a user and the information available to the user during search. In contrast to existing click models, which represent user browsing behavior as a sequence of predefined binary events, the proposed representation is much richer, and thereby enables us to capture more complex patterns of user browsing behavior than existing click models. The proposed approach allows us to learn patterns of user behavior directly from interaction data, circumventing the need for a predefined set of rules, which are a key ingredient of existing click models.

We evaluated our approach on the click prediction task (i.e., to predict user clicks on search engine results) and the relevance prediction task (i.e., to rank documents by relevance based on historical user interactions). Our experimental results show that the proposed approach (1) has better predictive performance in terms of perplexity and log-likelihood for all query frequencies and at all ranks than DBN and UBM; and (2) produces better document rankings in terms of NDCG than DBN and UBM. We also showed how to go beyond the level of query-document pairs and incorporate information (1) about all query sessions generated by a given query and (2) about all query sessions containing a given document. The first yields a small improvement on the click prediction task and a considerable improvement on the relevance prediction task. The second yields a large improvement on the click prediction task and a significant deterioration on the relevance prediction task. Finally, we provided an analysis of the best performing model on the click prediction task, which showed that (1) it learned concepts such as "the current document rank" and "the distance to the previous click," which are used in UBM; and that (2) it also learned other concepts that cannot be designed manually.

As to future work, first, we want to extend our models to non-linear user browsing behaviors, e.g., examining the first three documents and then clicking on the second document. Second, we want to consider other types of (1) user action, e.g., clicking on a sponsor advertisement, zooming on a result (in mobile search), reformulating a query; (2) query, e.g., audio queries (in voice search), image queries (in image search), foreign language queries (in cross-lingual search); (3) document, e.g., image results (in image search); and (4) interaction, e.g., mouse movements. Third, we want to extend the modeling scope from a search engine result page to a search session. Fourth, we want to incorporate information about the user, e.g., user profile, location, time, etc.

Another interesting future direction is to analyze the vector states that we used in this paper. A sequence of vector states provides a rich representation of the information need of a user and its evolution during the search. The information encoded in such a state can be used to detect positive abandonment or to distinguish user frustration from exploration of search engine results.

Acknowledgments. This research was supported by Amsterdam Data Science, the Dutch national program COMMIT, 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 Swiss National Science Foundation under project number P2T1P2_152269. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

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

[2] 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.
[3] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *SciPy*, 2010.
[4] O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *WWW*, pages 1–10. ACM, 2009.
[5] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *CIKM*, pages 621–630. ACM, 2009.
[6] 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.
[7] W. Chen, D. Wang, Y. Zhang, Z. Chen, A. Singla, and Q. Yang. A noise-aware click model for web search. In *WSDM*, pages 313–322. ACM, 2012.
[8] A. Chuklin, P. Serdyukov, and M. de Rijke. Click model-based information retrieval metrics. In *SIGIR*, pages 493–502. ACM, 2013.
[9] 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.
[10] A. Chuklin, I. Markov, and M. de Rijke. *Click Models for Web Search*. Morgan & Claypool, 2015.
[11] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *WSDM*, pages 87–94. ACM, 2008.
[12] G. Dupret and C. Liao. A model to estimate intrinsic document relevance from the clickthrough logs of a web search engine. In *WSDM*, pages 181–190. ACM, 2010.
[13] G. E. Dupret and B. Piwowarski. A user browsing model to predict search engine click data from past observations. In *SIGIR*, pages 331–338. ACM, 2008.
[14] J. L. Elman. *Rethinking innateness: A connectionist perspective on development*, volume 10. MIT press, 1998.
[15] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *IEEE Transactions on Neural Networks*, 18(5):602–610, 2005.
[16] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, pages 6645–6649. IEEE, 2013.
[17] A. Grotov, A. Chuklin, I. Markov, L. Stout, F. Xumara, and M. de Rijke. A comparative study of click models for web search. In *CLEF*, 2015.
[18] F. Guo, C. Liu, A. Kannan, T. Minka, M. Taylor, Y.-M. Wang, and C. Faloutsos. Click chain model in web search. In *WWW*, pages 11–20. ACM, 2009.
[19] F. Guo, C. Liu, and Y. M. Wang. Efficient multiple-click models in web search. In *WSDM*, pages 124–131. ACM, 2009.
[20] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
[21] K. Järvelin and J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *SIGIR*, pages 41–48. ACM, 2000.
[22] R. Jozefowicz, W. Zaremba, and I. Sutskever. An empirical exploration of recurrent network architectures. In *ICML*, pages 2342–2350, 2015.
[23] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.
[25] Y. Liu, C. Wang, K. Zhou, J. Nie, M. Zhang, and S. Ma. From skimming to reading: A two-stage examination model for web search. In *CIKM*, pages 849–858. ACM, 2014.
[26] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048, 2010.
[27] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.
[28] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*, 2012.
[29] D. E. Rumelhart, J. L. McClelland, P. R. Group, et al. *Parallel distributed processing*, volume 1. IEEE, 1988.
[30] S. Shen, B. Hu, W. Chen, and Q. Yang. Personalized click model through collaborative filtering. In *WSDM*, pages 323–332. ACM, 2012.
[31] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.
[32] L. Van der Maaten and G. Hinton. Visualizing data using t-SNE. *JMLR*, 9(2579-2605):85, 2008.
[33] 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.
[34] H. Wang, C. Zhai, A. Dong, and Y. Chang. Content-aware click modeling. In *WWW*, pages 1365–1376, 2013.
[35] E. Yilmaz, M. Shokouhi, N. Craswell, and S. Robertson. Expected browsing utility for web search evaluation. In *CIKM*, pages 1561–1564. ACM, 2010.
[36] M. D. Zeiler. ADADELTA: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.
[37] Y. Zhang, W. Chen, D. Wang, and Q. Yang. User-click modeling for understanding and predicting search-behavior. In *KDD*, pages 1388–1396. ACM, 2011.

APPENDIX

Similarly to the neural click model framework described in §3.1, traditional click models can be fully described by mappings $\mathcal{I}(\cdot)$, $\mathcal{U}(\cdot)$ and the function $\mathcal{F}(\cdot)$ (see Eqs. (1), (2) and (3)). Below, we specify these parameters for UBM and DBN.

UBM. The vector state $\mathbf{s}_r$ can be represented with a tuple of four integer values (q, d, r, r'), where the first component q denotes a query ID, the second component d denotes the ID of a currently examined document, the third component r denotes the rank of the currently examined document and the last component r' denotes the rank of the previously clicked document.

The mapping $\mathcal{I}(\cdot)$ initializes $\mathbf{s}_0$ by setting the first component to the ID of a user's query and the other components to zero. The mapping $\mathcal{U}(\cdot)$ updates the previous state $\mathbf{s}_r$ to the next state $\mathbf{s}_{r+1}$ as follows: the second component is set to the ID of the next document d_{r+1}, the third component is incremented by one, and the fourth component is set to the third component of $\mathbf{s}_r$, if the previous document d_r was clicked. The interaction variable i_r denotes a click event: $i_r = 1$ if d_r was clicked, and $i_r = 0$ otherwise.

The function $\mathcal{F}(\cdot)$ computes the probability that a user clicks on the currently examined document as a product of the examination

probability $\gamma_{r,r-r'}$ and the attractiveness probability $\alpha_{q,d}$. Both $\gamma_{r,r-r'}$ and $\alpha_{q,d}$ are the parameters of the function $\mathcal{F}(\cdot)$.

Overall, UBM can be formalized as follows.

$$\begin{aligned}
\mathcal{I}(q) &= (\text{QUERY_ID}(q), 0, 0, 0) \\
\mathcal{U}(\mathbf{s}_r, i_t, d_{r+1}) &= (\mathbf{s}_r[1], \text{DOC_ID}(d_{r+1}), \mathbf{s}_r[3] + 1, h(\mathbf{s}_r, i_r)) \\
h(\mathbf{s}_r, i_r) &= \begin{cases} \mathbf{s}_r[3] & \text{if } \; i_r = 1 \\ \mathbf{s}_r[4] & \text{otherwise} \end{cases} \\
\mathcal{F}(\mathbf{s}_{r+1}) &= \gamma_{\mathbf{s}_{r+1}[3], \mathbf{s}_{r+1}[3] - \mathbf{s}_{r+1}[4]} \cdot \alpha_{\mathbf{s}_{r+1}[1], \mathbf{s}_{r+1}[2]}
\end{aligned}$$

DBN. The vector state $\mathbf{s}_r$ can be represented with a tuple of two integer and one floating-point values (q, d, ϵ), where the first component q denotes a query ID, the second component d denotes the ID of a currently examined document, and the third component ϵ denotes the probability of examining the current document.

The mapping $\mathcal{I}(\cdot)$ initializes $\mathbf{s}_0$ by setting the first component to the ID of a user's query, the second component to zero and the third component to one. The mapping $\mathcal{U}(\cdot)$ updates the previous state $\mathbf{s}_r$ to the next state $\mathbf{s}_{r+1}$ as follows: the second component is set to the ID of the next document d_{r+1}, and the third component is updated according the function $g(\mathbf{s}_r, i_r)$:

$$g(\mathbf{s}_r, i_r) = \begin{cases} (1 - \beta_{\mathbf{s}_r[0], \mathbf{s}_r[1]})\gamma & \text{if } \; i_r = 1 \\ \dfrac{(1 - \alpha_{\mathbf{s}_r[0], \mathbf{s}_r[1]})\mathbf{s}_r[3]\gamma}{1 - \alpha_{\mathbf{s}_r[0], \mathbf{s}_r[1]}\mathbf{s}_r[3]} & \text{otherwise} \end{cases}$$

where γ, $\beta_{q,d}$, and $\alpha_{q,d}$ are the parameters of the mapping $\mathcal{U}(\cdot)$.

The above formula can be interpreted as follows (see [10, Chapter 3] for more details). If a user clicks on the current document ($i_r = 1$), then according to DBN she continues examining other documents if she is not satisfied with the current one ($1 - \beta_{q,d}$) and explicitly decides to continue (γ). The user skips the current document ($i_r = 0$) with probability $(1 - \alpha_{q,d})\epsilon / (1 - \alpha_{q,d}\epsilon)$, i.e., the user examines the current document (ϵ), but is not attracted by it ($1 - \alpha_{q,d}$), normalized by the total probability of no click ($1 - \alpha_{q,d}\epsilon$). In the case of a skip, the user continues examining other documents with probability γ.

The function $\mathcal{F}(\cdot)$ computes the probability that a user clicks on the currently examined document as a product of the examination probability ϵ and attractiveness probability $\alpha_{q,d}$. Here, $\alpha_{q,d}$ is the parameter of $\mathcal{F}(\cdot)$, which is shared with the mapping $\mathcal{U}(\cdot)$.

Overall, DBN can be formalized as follows.

$$\begin{aligned}
\mathcal{I}(q) &= (\text{QUERY_ID}(q), 0, 1) \\
\mathcal{U}(\mathbf{s}_r, i_t, d_{r+1}) &= (\mathbf{s}_r[1], \text{DOC_ID}(d_{r+1}), g(\mathbf{s}_r, i_r)) \\
\mathcal{F}(\mathbf{s}_{r+1}) &= \mathbf{s}_{r+1}[3] \cdot \alpha_{\mathbf{s}_{r+1}[1], \mathbf{s}_{r+1}[2]}
\end{aligned}$$