A Context-Aware Click Model for Web Search

Jia Chen, Jiaxin Mao, Yiqun Liu[⋆], Min Zhang, Shaoping Ma
Department of Computer Science and Technology, Institute for Artificial Intelligence
Beijing National Research Center for Information Science and Technology
Tsinghua University, Beijing 100084, China
yiqunliu@tsinghua.edu.cn

ABSTRACT

To better exploit the search logs, various click models have been proposed to extract implicit relevance feedback from user clicks. Most traditional click models are based on probability graphical models (PGMs) with manually designed dependencies. Recently, some researchers also adopt neural-based methods to improve the accuracy of click prediction. However, most of the existing click models only model user behavior in query level. As the previous iterations within the session may have an impact on the current search round, we can leverage these behavior signals to better model user behaviors. In this paper, we propose a novel neural-based Context-Aware Click Model (CACM) for Web search. CACM consists of a context-aware relevance estimator and an examination predictor. The relevance estimator utilizes session context information, i.e., the query sequence and clickthrough data, as well as the pre-trained embeddings learned from a session-flow graph to estimate the context-aware relevance of each search result. The examination predictor estimates the examination probability of each result. We further investigate several combination functions to integrate the context-aware relevance and examination probability into click prediction. Experiment results on a public Web search dataset show that CACM outperforms existing click models in both relevance estimation and click prediction tasks.

CCS CONCEPTS

• **Information systems** → *Users and interactive retrieval*; Query log analysis;

KEYWORDS

Click Model, Web Search, Document Ranking, Click Prediction

ACM Reference Format:
Jia Chen, Jiaxin Mao, Yiqun Liu[⋆], Min Zhang, Shaoping Ma. 2020. A Context-Aware Click Model for Web Search. In *The Thirteenth ACM International Conference on Web Search and Data Mining (WSDM '20), February 3–7, 2020, Houston, TX, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3336191.3371819

1 INTRODUCTION

Understanding Web search user behavior is essential for improving the performance of retrieval systems. Summarizing the patterns of human behavior and further exerting them on user interaction simulation may help search engines better fulfill users' information needs. To this end, numerous *click models* have been proposed for Web search [9]. These click models act as the click simulators in a virtual environment if no real users are available. While click signals are vulnerable due to behavioral biases, e.g., the position bias [24], click models also estimate the unbiased relevance scores for query-document pairs to facilitate document ranking.

Most existing click models represent user behaviors as a sequence of observable or hidden states. They are normally constructed based on the probabilistic graphical model (PGM) framework. Researchers may first analyze the search log data and then manually design the dependencies in the PGM framework, e.g. Position-biased Model (PBM) assumes the examination probability of a document depends heavily on its position in the Search Engine Result Page (SERP) [12]. These model-driven methods can reason about user behavior through the dependencies between the events in PGMs. However, these dependencies have to be set manually and are likely to miss key aspects of user behavior [3, 4].

To better capture users' behavior patterns, Borisov et al. [3] propose a neural click model (NCM) for Web search. Instead of the traditional PGM-based framework, they adopt the distributed representation (DR) approach for user behavior representation. In NCM, user interactions are represented as a vector sequence. Experiment results show that it outperforms traditional click models constructed on the PGM framework. Although NCM has utilized the query-level interaction information, they ignore the interaction effects between different search iterations within a session. It also simply regards the relevance of a specific document as the predicted click probability when it is ranked at the first position on the SERP. Numerous studies have shown the great potential of considering the session context in various Information Retrieval (IR) tasks such as document ranking, query suggestion, and etc [1, 23, 36]. Therefore, the session context information may also be beneficial for improving the performance of click models.

Up till now, neither tradition click models nor the NCM have thoroughly modeled the relationship between examination, relevance and click. For NCM, the relationship between the predicted clicks and the estimated relevance scores is not explicitly modeled. For most PGM-based clicked models, they follow the *examination hypothesis* [9] that a user will click on a result if, and only if she has examined the result and is attracted by it. Generally, they regard the relevance as the attractiveness and directly multiply the examination probability with the relevance score (within the range of 0-1) to represent the click probability. This assumption is hard-coded and may restrain the system performance.

To shed light on the aforementioned issues, we propose a novel Context-Aware Click Model (CACM) in this paper. CACM consists of a relevance estimator and an examination predictor to output the relevance scores and the predicted examination probabilities, respectively. Based on an end-to-end neural network, the learning process of CACM is data-driven and more flexible. To better capture users' search intents, we also encode session context into

[⋆]Corresponding author

distributed vectors of the query and URL nodes by mining the "wisdom of crowds" buried in the session-flow graph. Last but not least, we investigate the examination hypothesis by comparing several combination functions of the relevance and examination factors to predict the click probabilities.

The main contributions of this paper are:

- We propose a novel neural-based click model named CACM. It models the session context with an end-to-end neural network and jointly learns the relevance and the click probability of a specific document.
- CACM achieves significantly better performance than existing click models in document ranking by utilizing the contextual information and the embeddings learned from the session-flow graph for relevance estimation. Besides, CACM can also achieve better click prediction performance.
- To explore the relationship of relevance, examination and clicks, we further design several combination functions to aggregate the estimated relevance scores and the predicted examination probabilities for click prediction. Experiment results show that exponential multiplication function outperforms the popular examination hypothesis which adopts simple multiplication.

2 RELATED WORK

2.1 Click Models

To simulate user behaviors and estimate the relevance of documents, numerous click models have been proposed for Web search. Most of the existing click models represent user behaviors as a sequence of observable and hidden events. They are normally based on the probabilistic graphical model (PGM) framework [26]. For example, the *cascade model* (CM) simply considers that a user sequentially scans an SERP from top to bottom until she finds a relevant document and clicks on it. Since CM only adapts to the queries with only one click, more sophisticated click models such as UBM [15], DBN [7], CCM [19] and DCM [20] have been proposed to address this problem. Many click models support a so-called examination hypothesis to cope with the position bias problem [12, 34]. To better model users' search intents, some researchers try to incorporate more information into the click models. Wang et al. [37] first analyze users' non-sequential examination behavior by eye-tracking and propose a novel Partially Sequential Click Model (PSCM). To better model clicks within a task, Zhang et al. [45] propose a task-centric click model (TCM) that considers two new user behavior biases in search tasks. As the proportion of vertical results increases these years, some new click models have been proposed to take the vertical bias into consideration [10, 30].

As the dependencies in traditional PGM-based click models should be designed manually, some neural network based approaches have been proposed for better user modeling [4, 28, 43, 46]. Borisov et al. [3] first attempt to use neural networks to model users' query-level interaction sequence. The Click Sequence Model (CSM) maintains an encoder-decoder framework to predict the order in which a user will interact with search engine results [4]. However, few of these studies consider session contexts or put much attention on the relationship between relevance, examination and clicks.

2.2 Search Session Modeling

Since user interaction behaviors can reflect their search intents, there is a broad spectrum of researches focus on modeling user behaviors by utilizing the session-level contextual information. Some models utilize the query sequence and the user interactions within a session for context-aware query suggestion [6, 13, 23, 36]. For instance, Jiang et al. [23] propose a reformulation inference network (RIN) that not only utilizes the pre-trained embeddings learned from a heterogeneous network but also embeds the user reformulation into distributed vectors. Inspired by RIN, we also build a session-flow graph and introduce the pre-trained embeddings into our model. Since query reformulation is the bottleneck issue in the usability of search engines, there are also some work aiming at the context-sensitive query auto-completion and query disambiguation task [2, 27, 35].

Besides to help users better issue their search queries, contextual information has also been considered for session-based retrievals [1, 5, 31, 40]. Xiang et al. [39] develop different ranking principles for different types of contexts and further adopt a learning-to-rank approach to integrate these principles into a state-of-the-art ranking model. Some models also utilize the Reinforcement Learning (RL) algorithms to model users' session-level query decision or reformulation states [18, 29, 33]. Previous studies have shown the great potential of considering the session context for IR tasks. In this work, we also encode the context information into distributed vectors for better user intent modeling.

3 PROBLEM STATEMENT

Before we zoom into the details of the model framework, we first formally define the problem in this paper. In Web search, a search session $\mathcal{S}$ can be formulated as a sequence of queries $\langle q_1, q_2, ..., q_L \rangle$ submitted by the user. For each query q_i in the session, the search engine will return top N corresponding results $\mathcal{D}_i = \langle d_{i,1}, d_{i,2}, ..., d_{i,N} \rangle$. Each of these results has three attributes: its URL identifier $\mathcal{U}_{i,j}$, its ranking position $\mathcal{P}_{i,j}$ in the document list, and its vertical type [1] $\mathcal{V}_{i,j}$. A user may click several results in the session, thus we define a click interaction as the pair of the result and its click variable $\mathcal{I}_{i,j} = \{d_{i,j}, C_{i,j}\}$, where $C_{i,j} = 1$ if $d_{i,j}$ has been clicked by the user and 0 if not. Then the problem of relevance estimation and click prediction can be defined as follows:

Definition 1. *For the n-th document in the l-th query ($d_{l,n}$), given the user's query history $Q = \langle q_1, q_2, \dots, q_l \rangle$ and the previous interactions $\mathcal{I} = \{\mathcal{I}_{i,j} | i \leq l, j < n\}$ within the session, we would like to estimate the context-aware relevance of $d_{l,n}$ as well as to predict whether it will be clicked by the user.*

In this paper, we not only adopt the click signals as the proxy of the relevance feedback but also use them for click prediction. To learn the relationship between relevance, examination, and clicks, we design both a relevance estimator and an examination predictor to output relevance scores and examination probabilities, respectively. We first use the relevance scores for the document ranking and then combine them with the estimated examination probabilities to predict the clicks.

4 MODEL FRAMEWORK

In this section, we will introduce the framework of the proposed Context-Aware Click Model (CACM), which is shown in Figure 1.

4.1 Relevance Estimator

The relevance estimator mainly consists of three components, including a query context encoder, a click context encoder, and a document encoder. In the next, we will introduce these three components respectively in detail.

[1]Also termed as "result type", which infers the presentation style of a search result. There are tens of result types in modern commercial search engines [44], such as the *organic result*, the *illustrated vertical*, the *encyclopedia vertical*, and etc.

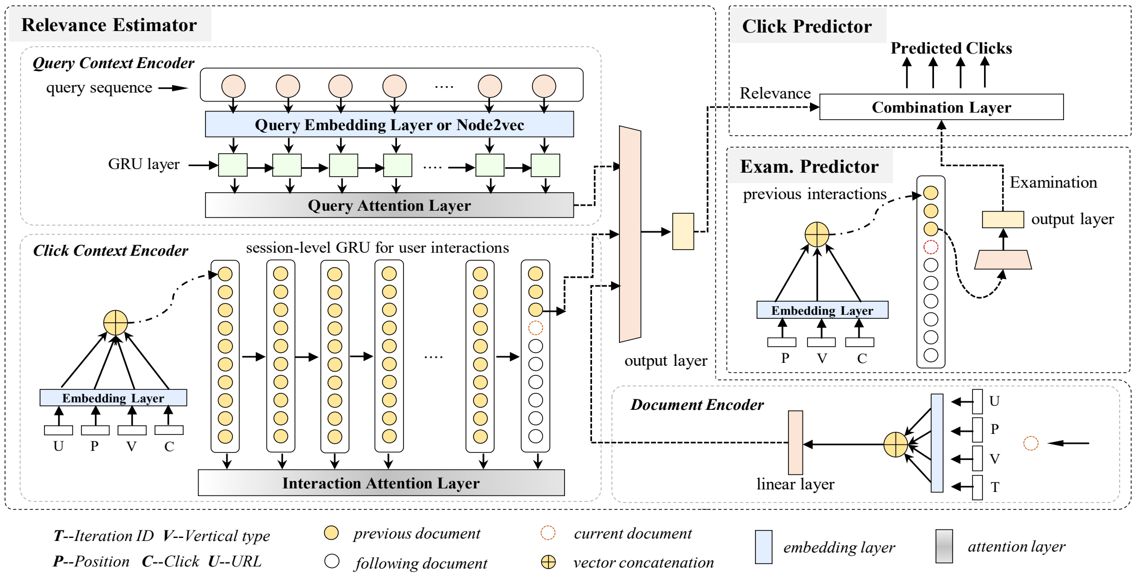


Figure 1: Framework of the proposed CACM. CACM consists of a relevance estimator and an examination predictor. The relevance estimator encodes both the inter-session contexts (i.e., the query context and the click context) and the current document to estimate the context-aware relevance $\mathcal{R}$. The examination predictor utilizes the intra-session context to predict the examination probability $\mathcal{E}$. Then through a combination layer, CACM integrates the $\mathcal{R}$ and $\mathcal{E}$ into the click prediction.

4.1.1 Query context encoder. CACM encodes the session-level query context because the queries submitted by a user imply to which extent she has acknowledged about the search task. We first encode each query (a one-hot query ID) in the sequence into fixed-length vectors. Normally, we can simply use an query-level embedding layer to encode a query q_i: $\mathbf{v}_{q_i} = \mathbf{Emb}_q(q_i)$, where $\mathbf{Emb}_q \in \mathbb{R}^{N_q \times l_q}$ is the query embedding layer, N_q is the total number of queries, and l_q is the size of the query embeddings.

Many existing studies have shown that incorporating the embeddings learned from the graph structure can improve the system performance [23]. Inspired by these works, we also construct a session-flow graph and attempt to mine the "wisdom of crowds" buried in this graph. Figure 2 shows an example of the constructed session-flow graph. There are three kinds of edges in the graph:

- *query-query*: the edges between each pair of two consecutive queries, which demonstrate the reformulation relationship between the two queries.
- *URL-URL*: the edges between a result and its follow-up, which represent the similarity of two contiguous documents and their positions in the SERP.
- *query-URL*: the edges between a query and each of its returned documents in the SERP, which imply the relevance of the document given the current query and the user's interaction.

To learn the representations better, we empirically design the weights of each edge in the graph. The query-query edges indicate the user's initiative reformulation behaviors, the query-URL edges can imply user's implicit feedback, while the URL-URL edges contain no human interaction information. Therefore, we consider the priority of the three edges as query-query > query-URL > URL-URL. For all query-query edges, we set their weight $\mathcal{W}_{q-q} = w$, where w is an adjustable parameter and $w > 1.0$. For a query-URL edge

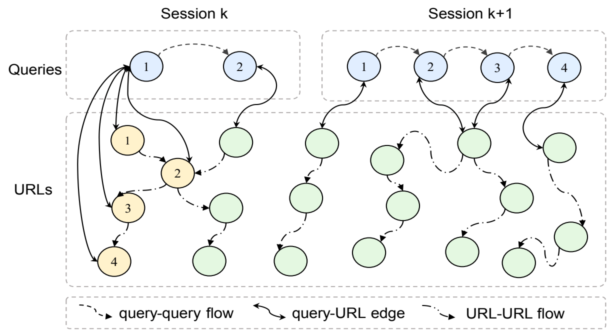


Figure 2: The session-flow graph which is used to model the query-level and session-level context information.

between the query q_i and the document $d_{i,j}$, we set its weight as:

$$\mathcal{W}_{q-u} = \begin{cases} 1, & C_{i,j} = 1, \\ -1, & C_{i,j} = 0, j < max(\mathcal{P}_{C_i}), \\ 0, & C_{i,j} = 0, j > max(\mathcal{P}_{C_i}); \end{cases} \tag{1}$$

where $max(\mathcal{P}_{C_i})$ denotes the position of the last clicked result in the i-th search round. Here we take the clicks as the positive feedbacks and the skips as the negative feedback. Other results are not considered as examined by users thus no weights are assigned. For an URL-URL edge between two documents $d_{i,j}$ and $d_{i,j+1}$, we define its weight according to:

$$\mathcal{W}_{u-u} = \frac{1}{\log_2(j+1)}; \tag{2}$$

here we introduce the rank of the latter document $d_{i,j+1}$ to describe the ranking position of the two documents.

Having constructed the session-flow graph, we then apply the node2vec [17] tool to learn the embeddings of queries ($\mathbf{v}_q$) and URLs ($\mathbf{v}_u$) on it. Other attributes of a result such as the vertical type are too sparse, so we do not learn their embeddings through the graph here. We then encode the sequence of query embeddings through a standard Gated Recurrent Unit (GRU) [11] with an attention mechanism. Specifically, a GRU encodes the query history by sequentially updating a hidden state. At the search iteration t, given the current query embedding $\mathbf{v}_{q_t}$ and the previous hidden state h_{t-1}, the current hidden state will be updated by $h_t = GRUCell(h_{t-1}, \mathbf{v}_{q_t})$. For a sequence of L queries, the GRU layer forms a sequence of L hidden representations: $H = [h_1, h_2, ..., h_T], \quad H \in \mathbb{R}^{l_h \times L}$, where l_h denotes the size of the GRU hidden unit. Previous queries may influence the user's action in the current search round differently. Therefore, to capture the different influence of each query in the query sequence, we apply a self-attention mechanism to form a context-attentive query sequence representation:

$$\mathcal{S}_{q,att} = \sum_{i=1}^{L} \alpha_q^i h_i, \quad \alpha_q^i = \frac{exp(h_L^T h_i)}{\sum_{k=1}^{L} exp(h_L^T h_k)}, \tag{3}$$

here we calculate the softmax value between the output of the last hidden layer h_L and each of the previous hidden representation, because a more similar previous query may have a larger impact on the current query. We then assign the importance factor α_q^i to each hidden layer and obtain the weighted sum $\mathcal{S}_{q,att}$ as the attentive session-level query sequence representation.

4.1.2 *Click context encoder.*

The interaction context within a session can provide valuable clues about a user's search intent. For instance, user clicks usually serve as a good proxy of relevance feedbacks [14, 24]. To consider this, we design a click context encoder to model users' session-level interactions. For the current document $d_{t,n}$, we first embed the user's interaction with each of its previous document. For a previous document $d_{i,j}$, CACM embeds its URL $\mathcal{U}_{i,j}$, position $\mathcal{P}_{i,j}$, vertical type $\mathcal{V}_{i,j}$ and click variable $C_{i,j}$ by four different embedding layers:

$$\mathbf{v}_u = \mathbf{Emb_u}(\mathcal{U}_{i,j}), \quad \mathbf{v}_p = \mathbf{Emb_p}(\mathcal{P}_{i,j}), \tag{4}$$

$$\mathbf{v}_v = \mathbf{Emb_v}(\mathcal{V}_{i,j}), \quad \mathbf{v}_c = \mathbf{Emb_c}(C_{i,j}), \tag{5}$$

where $\mathbf{Emb_u} \in \mathbb{R}^{N_u \times l_u}$, $\mathbf{Emb_p} \in \mathbb{R}^{N_p \times l_p}$, $\mathbf{Emb_v} \in \mathbb{R}^{N_v \times l_v}$, $\mathbf{Emb_c} \in \mathbb{R}^{N_c \times l_c}$, $N\_$ and $l\_$ denote the input size and the embedding size of each attribute, respectively. Specially, we set $N_p = 10$ (top 10 documents), $N_c = 2$ (whether be clicked or not), $N_v = 19$ (totally 19 vertical types), $l_p = 4$, $l_v = 8$, and $l_c = 4$. Specially, we can also replace the $\mathbf{v}_u$ with the pre-trained embeddings learned from the session-flow graph. The four embeddings are then concatenated together to represent an interaction round. Finally, we encode the previous interactions ($i \leq t, j < n$) with a session-level GRU layer:

$$\mathbf{v}_{\mathcal{I}_{i,j}} = [\mathbf{v}_u \oplus \mathbf{v}_p \oplus \mathbf{v}_v \oplus \mathbf{v}_c], \tag{6}$$

$$\mathcal{S}_c = GRU(\mathbf{v}_{\mathcal{I}_{1,1}}, ...\mathbf{v}_{\mathcal{I}_{t,n-1}}), \tag{7}$$

where $\oplus$ denotes the vector concatenation and $\mathbf{v}_{\mathcal{I}_{i,j}}$ is the representation of an interaction round. Previous interactions will also have different influences on the current user action, e.g. the action a user takes on a document similar to the current one may have a larger impact on her current decision. Similarly, we apply an interaction attention layer here to highlight the importance of these positions:

$$\mathcal{S}_{c,att} = \sum_{i=1,j=1}^{t,n-1} \alpha_c^{i,j} h_{i,j}, \ \alpha_c^{i,j} = \frac{exp(h_{t,n-1}^T h_{i,j})}{\sum_{p=1,q=1}^{t,n-1} exp(h_{t,n-1}^T h_{p,q})}, \tag{8}$$

where $h_{i,j}$ is the hidden representation when input $\mathbf{v}_{\mathcal{I}_{i,j}}$ to the GRU layer in Equation 7 and $\alpha_c^{i,j}$ denotes the importance of the j-th interaction in the i-th query.

4.1.3 *Document encoder.*

The current document $d_{t,n}$ will directly influence the user's interaction action if it is examined. So we encode the current document with an independent encoder. Because the position of the document appearing within the session affects its attractiveness to the users, we also embed the query iteration ID t besides its URL, rank, and vertical type: $\mathbf{v}_t = \mathbf{Emb_t}(t)$, where $\mathbf{Emb_t} \in \mathbb{R}^{N_t \times l_t}$, $N_t = 10$ and $l_t = 4$. Let $\mathcal{F}_d$ be a linear layer, we obtain the document vector by concatenating the four embeddings and feeding the long vector into an output layer:

$$\mathbf{v}'_{d_{t,n}} = [\mathbf{v}_u \oplus \mathbf{v}_p \oplus \mathbf{v}_v \oplus \mathbf{v}_t], \tag{9}$$

$$\mathbf{v}_{d_{t,n}} = Tanh(\mathcal{F}_d(\mathbf{v}'_{d_{t,n}})), \tag{10}$$

Finally, to combine the effects of the query context encoder, the click context encoder, and the document encoder, we concatenate the three output together and apply a two-layer MLP for the non-linearity of relevance estimation:

$$\mathbf{v}_{\mathcal{R}_{t,n}} = [\mathcal{S}_{q,att} \oplus \mathcal{S}_{c,att} \oplus \mathbf{v}_{d_{t,n}}], \tag{11}$$

$$\mathcal{R}_{t,n} = MLP(\mathbf{v}_{\mathcal{R}_{t,n}}), \tag{12}$$

where $\mathcal{R}_{t,n}$ is the estimated relevance representation for the current document $d_{t,n}$, the activation function for the first and the second layer in MLP is *tanh* and *sigmoid* function, respectively. We apply the *sigmoid* function to restrain the output of all relevance scores within the range of 0-1.

4.2 Examination Probability Predictor

Most existing click models assume that users' click actions are related to both the relevance of the document and the examination probability. We have introduced how to represent the relevance of a specific document by utilizing the session context in the previous subsection. To realize the click prediction, we should also estimate the examination probability of the document. According to the cascade assumption [12], a user scans documents on the SERP from top to bottom until she find a relevant one. Therefore, we assume that a user's examination action is only affected by her actions on previous results in the current query. For a document $d_{t,n}$, we first embed the previous interactions in q_t and then formulate the probability a user may examine it by an intra-query GRU:

$$\mathbf{v}_{\mathcal{I}_{t,j}} = [\mathbf{v}_p \oplus \mathbf{v}_v \oplus \mathbf{v}_c], \quad j < n, \tag{13}$$

$$\mathcal{E}'_{t,n} = GRU(\mathbf{v}_{\mathcal{I}_{t,1}}, ..., \mathbf{v}_{\mathcal{I}_{t,n-1}}), \tag{14}$$

where $\mathbf{v}_{\mathcal{I}_{t,j}}$ is the distributed representation of the j-th interaction in the current query. We assume that the examination probability is not affected by its content (users read the content only when the examination behavior happens) but by some other factors, hence we do not include the URL embedding here. After modeling the interaction sequence within q_t, we finally normalize the output range of the examination probability by:

$$\mathcal{E}_{t,n} = \sigma(\mathcal{F}_e(\mathcal{E}'_{t,n})), \tag{15}$$

here $\mathcal{E}_{t,n}$ is the predicted probability of a user may examine the document, $\mathcal{F}_e$ is a linear layer, and σ denotes the *sigmoid* function.

4.3 Click Prediction

Many existing click models follow the *examination hypothesis*: a user will click a document if and only if she examines the document and is attracted by the document [9]. The hypothesis can be represented by the following equation:

$$C_d = 1 \Leftrightarrow \mathcal{E}_d = 1 \text{ and } \mathcal{A}_d = 1, \quad (16)$$

where $C_d = 1$ denotes that the document d is clicked by the user, $\mathcal{E}_d$ and $\mathcal{A}_d$ are usually considered as two independent variables, which represent the examination probability and the attractiveness of the document, respectively. In this paper, we follow some of the previous work to refer the attractiveness as the relevance of the document. We then integrate the relevance scores and examination probabilities with a combination layer for click prediction.

4.3.1 Combination Layer. To further explore the adaptiveness of the examination hypothesis as well as the relationship between click, relevance, and examination, we have implemented five different combination functions, which are shown in Table 1.

Table 1: Five combination function for the relevance score and the examination probability (E.H is short for examination hypothesis, C–Click, $\mathcal{R}$–Relevance, $\mathcal{E}$–Examination, σ–sigmoid function, λ, μ, α, β are learnable hyperparameters).

function	Formula	Support E.H?
mul	$C = \mathcal{R} \cdot \mathcal{E}$	√
exp_mul	$C = \mathcal{R}^{\lambda} \cdot \mathcal{E}^{\mu}$	√
linear	$C = \alpha \cdot \mathcal{R} + \beta \cdot \mathcal{E}$	×
nonlinear	$C = MLP(\mathcal{R}, \mathcal{E})$	×
sigmoid_log	$C = 4\sigma(log(\mathcal{R})) \cdot \sigma(log(\mathcal{E}))$ $= 4\mathcal{R}\mathcal{E}/((\mathcal{R}+1)(\mathcal{E}+1))$	√

For *mul* function, we directly multiply the relevance score with the examination probability. To further enhance the fitting ability, we design *exp_mul* to add more parameters on the exponential of $\mathcal{R}$ and $\mathcal{E}$. For *nonlinear* function, we use a two-layer perceptron to take the relevance and examination as two features. We further devise the *sigmoid_log* function to integrate the sigmoid function and the logarithmic function, which can be finally deduced to a simple fraction. Among the five combination functions, only *mul*, *exp_mul*, and *sigmoid_log* support the examination hypothesis.

4.3.2 Model training. The whole framework of CACM is end-to-end, so we can train the model easily through the back-propagation algorithm. To better couple the relevance estimation and the click prediction task for learning the session context, we design a regularization based loss function as the optimizing object. To estimate the model parameters in CACM, we would like to minimize the training objective function as follows:

$$\mathcal{L}(\theta) = \mathcal{L}_{\mathcal{R}} + \mathcal{L}_C + \lambda||\theta||^2, \quad (17)$$

where $\mathcal{L}_{\mathcal{R}}$ and $\mathcal{L}_C$ is the loss for relevance estimation and click prediction respectively, θ denotes all parameters in CACM. To avoid overfitting, we add a regularization term for all parameters. Similar to multi-task learning technique [16], we impose two independent loss components. To learn user clicks better by modeling the session context, we set $\mathcal{L}_C$ as the cross-entropy with respect to the predicted clicks:

$$\mathcal{L}_C = -\frac{1}{N}\sum_n \sum_m (C_{n,m} \log \mathcal{P}_{n,m} + (1 - C_{n,m}) \log(1 - \mathcal{P}_{n,m})), \quad (18)$$

where N denotes the number of training batches, $C_{n,m}$ and $\mathcal{P}_{n,m}$ represent the click label and the predicted click probability of the m-th instance in the n-th training batch. On the other hand, we also use clicks as the proxy of weak relevance label for model training. However, there may be problems to directly use the same click signals for learning both elevance and clicks. To help the relevance estimator learn differently, we only use the click signals which have been examined by users as our training proxy in $\mathcal{L}_{\mathcal{R}}$, i.e., we regard the results ranked higher than the last clicked document within a query session as the examined ones as in many existing works [9]. The formula of $\mathcal{L}_{\mathcal{R}}$ is similar to Equation 18 except for that the predicted click $\mathcal{P}$ is replaced by the estimated relevance $\mathcal{R}$ and the subscript m should match the corresponding examined results.

Table 2: Basic statistics of the dataset we used in this paper.

	training	validating	testing
# sessions	117,431	13,154	16,570
# unique queries	35,903	9,373	11,391
avg. session length	2.4099	2.4012	2.4986

5 EXPERIMENTAL SETUPS

In this section, we will describe our experimental setups. We first outline the research questions in §5.1. We then briefly introduce the dataset in §5.2. The baseline systems and the evaluation metrics are then given in §5.3. In the end, we detailedly report the parameter setups we adopt to run our experiments in §5.4.

5.1 Research Questions

In this paper, we aim to answer the following research questions:

RQ1: How does CACM perform in terms of relevance estimation and click prediction compared to existing click models?

RQ2: Which combination function performs the best in terms of integrating the relevance and examination into click prediction?

RQ3: How will the query/URL embeddings learned from the session-flow graph and the attention mechanism affect the system ranking performance whether we incorporate them into the relevance estimator or not?

RQ4: Are $\mathcal{L}_{\mathcal{R}}$ and $\mathcal{L}_C$ both effective for model training? How will the system performance change if we remove one of them?

RQ5: How will the session-level contextual information affect the relevance estimation performance of CACM?

5.2 Dataset

We conduct our experiment on an open log-based session dataset named TianGong-ST [8] [2]. The data was refined from a 18-day user log recorded by a major Chinese commercial search engine *Sogou.com*. The dataset totally contains 147,155 refined search sessions and there are also 2,000 sessions with human-annotated relevance labels for documents in the last queries of them. We then randomly split the dataset into training, validating and testing set with a ratio of 8:1:1. To ensure proper evaluation, we filter a session in the validating and testing set if it contains queries which do not appear in the training set. We also include all the annotated sessions in the testing set to facilitate the evaluation of relevance estimation. We tune the parameters of CACM on training set and then stop the training according to its performance on the validating set. Some basic statistics of the dataset are shown in Table 2.

5.3 Baselines and Metrics

We compare CACM with both the traditional and neural click models. In this paper, we consider TECM [42], THCM [41], POM [38], DBN [7], UBM [15] and DCM [20] as the traditional click models, which are based on an open-source implementation [37]. In addition, we also re-implement the NCM (neural click model) according to the public paper [3] and take it as one of the baselines.

[2]This dataset is available at http://www.thuir.cn/tiangong-st/

CACM and most click models are content-independent, thus we do not consider content-based deep learning-to-rank methods such as DRMM [21], ARC-I/II [22] and DEUT [32]. For document ranking, we use the relevance scores estimated by the relevance estimator to rerank the document list and calculate the Normalized Discounted Cumulative Gain (NDCG) according to the human labels. As for click prediction, we report the click perplexity (PPL) [15] and the log-likelihood of each model. The definition of the click perplexity (PPL) at the rank r and the log-likelihood (LL) are as follows:

$$PPL@r = 2^{-\frac{1}{N}\sum_{i=1}^{N} C_{i,r} log\mathcal{P}_{i,r} + (1-C_{i,r}) log(1-\mathcal{P}_{i,r})}, \tag{19}$$

$$LL = \frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M} C_{i,j} log\mathcal{P}_{i,j} + (1-C_{i,j}) log(1-\mathcal{P}_{i,j}), \tag{20}$$

where the subscript r denotes a rank position in a result list, N is the total number of query sessions, and M is the number of results in a query. $C_{i,r}$ and $\mathcal{P}_{i,r}$ denote the true click signal and the predicted click probability of the r-th result in the i-th query session in the testing set. We then obtain the overall perplexity by averaging the values over all ranks. A lower value of PPL and higher value of LL indicates a better prediction performance.

5.4 Parameter Setups

We train CACM with a mini-batch size of 32 by using the Adam optimizer [25]. The hidden size is selected from {256, 512}. By applying the node2vec tool, we train the session-flow graph into node embeddings with size of {64, 128, 256, 512}. The initial learning rate and the dropout rate are selected from $\{10^{-2}, 10^{-3}, 10^{-4}\}$ and {0.1, 0.2, 0.3}, respectively. For each query, top 10 documents are considered for document ranking. To avoid system overfitting, we tune the weight decay parameter λ in $\{10^{-3}, 10^{-4}\}$. For the weight $\mathcal{W}_{q-q}$, we try some values and finally adopt the best value of 2.0. We also stop the training if the validation performance does not improve after five iterations. Finally, we adopt the iteration with the lowest validation click PPL for system evaluation in the testing set. A NVIDIA TITAN X GPU is used to train all deep models. We implement CACM on PyTorch [3] and share the source code in the link below [4].

6 RESULTS AND ANALYSIS

In this section, we will report the experiment results to answer the proposed research questions. We will first compare the testing performance of CACM with existing methods and then analyze the effect of each component.

6.1 Overall Performance

To answer **RQ1**, we train CACM based on both $\mathcal{L}_{\mathcal{R}}$ and $\mathcal{L}_{C}$. We first test the click prediction performance of each baseline system on our data. The results are shown in Table 3. We find that CACM significantly outperforms all baseline models. Besides, NCM performs the best among all the baseline systems, followed by DBN and UBM. NCM learns the distributed representations of queries and documents, thus can capture user behavior better. This is consistent with the results reported in the previous work [3].

To further investigate CACM's performance on relevance estimation, we use the output of the relevance estimator for the document ranking task. The performance of each model in document ranking is presented in Table 4. We observe that CACM outperforms all the baseline systems in terms of NDCG@1, 3, 5,10. Compared to traditional click models such as DBN and UBM, CACM maintains an end-to-end neural network which enables it to capture more subtle patterns in human behaviors. As for NCM, although it is neural-based, it ignores the context information within a session thus performs worse than CACM. Therefore, we can learn that the session context information is beneficial for user behavior modeling and should be considered for session-level relevance estimation.

[3]https://pytorch.org/
[4]https://github.com/xuanyuan14/CACM-master

Table 3: Click prediction performance of each model. Differences between all pairs of click models are statistically significant (p < 0.001).

Model	Perplexity	Log-likelihood
THCM	1.3407	-0.2421
TECM	1.7858	-0.3550
POM	1.4871	-0.3005
DCM	1.2800	-0.2289
DBN	1.2245	-0.1872
UBM	1.2129	-0.1775
NCM	1.2106	-0.1753
CACM	**1.2085**	**-0.1748**

Table 4: Comparison of the document ranking performance of each model, $*$ indicates a statistically significant improvement (p-value < 0.01) over the best baseline NCM.

Model	NDCG@1	NDCG@3	NDCG@5	NDCG@10
TECM	0.6345	0.6669	0.7018	0.8373
THCM	0.6617	0.6838	0.7115	0.8453
POM	0.6487	0.6704	0.7057	0.8399
UBM	0.6331	0.6692	0.7028	0.8375
DBN	0.6348	0.6681	0.7030	0.8376
NCM	0.7141	0.6993	0.7278	0.8562
CACM	**0.7222***	**0.7198***	**0.7417***	**0.8670***

6.2 Model Analysis

In this subsection, we will answer **RQ2-RQ5** by conducting several comparison tests and ablation studies to thoroughly investigate the effectiveness of CACM.

6.2.1 Combination Functions for Examination Hypothesis. To answer **RQ2**, we investigate the adaptiveness of each combination function by comparing their overall performances. From Table 5, we find that CACM with the *exp_mul* function has the best performance among all the combination functions. In addition, other functions such as *sigmoid_log* and *mul* achieve better click prediction performance than the *linear* and *nonlinear* functions. This indicates that the combination functions that support the examination hypothesis can better adapt the click signals in our dataset. Therefore, *exp_mul*, *mul* and *sigmoid_log* perform better than the rest two combination functions in click prediction.

To further explore the learning mechanism in each function, we check the values of their learned parameters. We observe that in *linear* and *nonlinear* functions, CACM assigns higher weights to the relevance than the examination when integrating them into clicks, i.e., in the *linear* function, we find $\alpha = 0.8838$ and $\beta = 0.3367$. So they focus more on the relevance estimation task and the click prediction performance drops a lot. We also find that $\lambda = 0.8954$ and $\mu = 0.9091$ in the *exp_mul* function. As for the *mul* function, the parameters can only be $\lambda = 1.0$ and $\mu = 1.0$ so its performance is a bit worse than the *exp_mul* function. Compared to other functions

Table 5: CACM's overall performance with different combination functions. The best results are given in bold. $*$ and $**$ indicate a statistically significant improvement in relevance estimation over the best baseline NCM at $p < 0.05$ and $p < 0.01$ level. Differences in click prediction between all pairs of click models are statistically significant ($p < 0.001$).

	Relevance Estimation				Click Prediction	
Model	NDCG@1	NDCG@3	NDCG@5	NDCG@10	PPL	Log-likelihood
DBN	0.6348	0.6681	0.7030	0.8376	1.2245	-0.1872
NCM	0.7141	0.6993	0.7278	0.8562	1.2106	-0.1753
CACM$_{linear}$	0.7084	0.7183**	0.7400**	0.8653**	1.2211	-0.1856
CACM$_{nonlinear}$	0.7147	0.7148**	0.7373**	0.8648**	1.2310	-0.1885
CACM$_{sigmoid\_log}$	0.6813	0.7059	0.7306	0.8583	1.2162	-0.1801
CACM$_{mul}$	0.6962	0.7089**	0.7332	0.8605*	1.2103	-0.1759
CACM$_{exp\_mul}$	**0.7222****	**0.7198****	**0.7417****	**0.8670****	**1.2085**	**-0.1748**

Table 6: Ablation study on the relevance estimator of CACM. We sequentially perform the following steps: I. remove the query iteration number in the document encoder; II. remove all the attention mechanisms; III. replace the pre-trained embeddings with a general embedding layer.

Model	NDCG@1	NDCG@3	NDCG@5	NDCG@10
0 CACM	**0.7542**	**0.7257**	**0.7484**	**0.8707**
I w/o IterID	0.7476	0.7240	0.7442	0.8681
II w/o Attn	0.7303	0.7201	0.7427	0.8675
III w/o Embed	0.7292	0.7135	0.7400	0.8648
NCM	0.7141	0.6993	0.7278	0.8562

that support the examination hypothesis, the *exp_mul* function owns more learnable parameters thus can fit the user behaviors more flexibly. Therefore, it has the best overall performance.

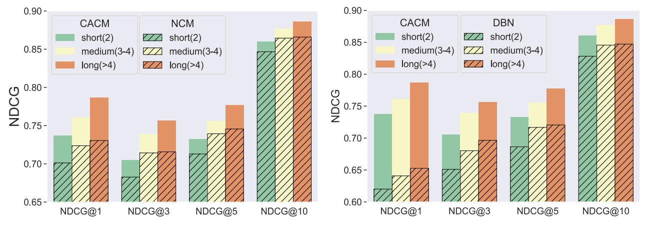
6.2.2 Ablation study. To begin with, we would like to answer **RQ3** by masking $\mathcal{L}_C$ in Equation 17 and training the relevance estimator with only $\mathcal{L}_R$. To investigate the independent relevance estimation performance, we conduct an ablation study by removing some components in the relevance estimator and then test the rest part of CACM. The comparison results are shown in Table 6.

We find that training the relevance estimator independently with $\mathcal{L}_\mathcal{R}$ can achieve better document ranking performance. As we can see that the metrics of CACM in Table 6 are higher than that in Table 4. This shows the effectiveness of our relevance estimator. The performance of CACM in document ranking drops when using both $\mathcal{L}_\mathcal{R}$ and $\mathcal{L}_C$ because there may be a trade-off between the two loss functions. The learning mechanism of CACM is different from the normal multiple-task learning models. They usually design two loss functions with different supervised signals for joint learning, while normally a click model only uses the click signals for both relevance estimation and click prediction.

From Table 6, we can also observe that the performance drops a lot when we remove the attention mechanism or the pre-trained embeddings, especially in NDCG@1 and NDCG@3. With the attention mechanism, the model can better represent the session context by emphasizing those more important components. On the other hand, the pre-trained embeddings contain the "wisdom of crowds" learned from the session-flow graph thus can also help CACM better capture users' session-level information needs and rank the high-quality documents higher. The query iteration ID can also boost the ranking performance by a small margin because it may provide information about the novelty of a document within a session. For deep neural frameworks, this feature can be learned through the input so the impact is not so obvious. All the variants of CACM achieve better ranking performance than NCM because they can capture users' search intents better by considering the session context.

Table 7: Ablation study on $\mathcal{L}_\mathcal{R}$ and $\mathcal{L}_C$.

Metric	$\mathcal{L}_\mathcal{R} + \mathcal{L}_C$	$\mathcal{L}_\mathcal{R}$	$\mathcal{L}_C$
NDCG@1	0.7222	0.7542	0.6635
NDCG@3	0.7198	0.7257	0.6451
NDCG@5	0.7417	0.7484	0.6824
NDCG@10	0.8670	0.8707	0.8357
Perplexity	0.2085	0.2375	0.2062
Log-likelihood	-0.1749	-0.1960	-0.1730



(a) Comparison between CACM and NCM. (b) Comparison between CACM and DBN.

Figure 3: The comparison of system ranking performance w.r.t. different session lengths (best viewed in color).

To answer **RQ4**, we further conduct another ablation study on $\mathcal{L}_\mathcal{R}$ and $\mathcal{L}_C$. Here we train the CACM with only $\mathcal{L}_\mathcal{R}$ or $\mathcal{L}_C$ and report the overall system performance on these two conditions. As shown in Table 7, we find that CACM trained with both loss functions can achieve the best overall performance. When we mask the $\mathcal{L}_\mathcal{R}$, the performance on document ranking drops a lot because there are no learning signals for the estimated relevance. On the other hand, if we only consider the $\mathcal{L}_\mathcal{R}$, the click prediction task will be underfitting. Here we learn that both $\mathcal{L}_\mathcal{R}$ and $\mathcal{L}_C$ are effective for model training. Considering the trade-off between the two tasks, the best choice is to combine the two loss functions together.

6.2.3 Performance across session lengths. To answer **RQ5**, we conduct an experiment by comparing the system performance across session lengths. Here we split all test sessions into three groups:

- Short sessions (with 2 queries) - 69.45% of the test data
- Medium sessions (with 3-4 queries) - 25.99% of the test data
- Long sessions (with at least 5 queries) - 4.56% of the test data

We then compare the document ranking performance of CACM with NCM and DBN (two best baselines) across different lengths of session contexts in Figure 3(a) and 3(b), respectively.

Table 8: Case study of a session with four queries. $I_{i,j}$ denotes the j-th interaction round in the i-th query, $\surd$ represents a user click. Here we illustrate the query and interaction attention for $d_{4,3}$ in the left subtable. Relevance estimation by each model for documents in Q4 are given on the right.

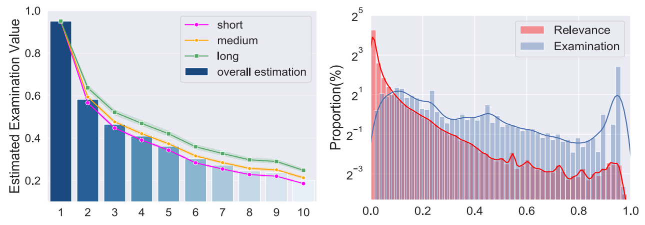
CACM Attention				Relevance Estimation		
Q1	Q2	Q3	Q4	Label	CACM	NCM
$\surd I_{1,1}$	$I_{2,1}$	$I_{3,1}$	$I_{4,1}$	4	0.8541	0.4031
$I_{1,2}$	$\surd I_{2,2}$	$\surd I_{3,2}$	$I_{4,2}$	3	0.4844	0.3368
$I_{1,3}$	$I_{2,3}$	$I_{3,3}$	$I_{4,3}{}^{*}$	0	0.0302	0.4519
$I_{1,4}$	$I_{2,4}$	$I_{3,4}$	$I_{4,4}$	0	0.0124	0.2048
$I_{1,5}$	$I_{2,5}$	$I_{3,5}$	$I_{4,5}$	2	0.0516	0.0421
$I_{1,6}$	$I_{2,6}$	$I_{3,6}$	$I_{4,6}$	2	0.1415	0.0483
$I_{1,7}$	$I_{2,7}$	$I_{3,7}$	$I_{4,7}$	1	0.0638	0.0166
$I_{1,8}$	$I_{2,8}$	$I_{3,8}$	$I_{4,8}$	0	0.0655	0.0173
$I_{1,9}$	$I_{2,9}$	$I_{3,9}$	$I_{4,9}$	2	0.0544	0.0130
$I_{1,10}$	$I_{2,10}$	$I_{3,10}$	$I_{4,10}$	2	0.0355	0.0160

* $\{d_{1,3}, d_{4,3}\}$, $\{d_{2,*}, d_{3,*}\}$, $\{d_{2,1}, d_{4,1}\}$ are the sets of same documents.
* On the left, the color depth of red and blue represent the weight of query and interaction attention, respectively. On the right, top 6 relevant documents estimated by each model for Q4 are highlighted in red.

In Figure 3(a), CACM and NCM perform better when a session is longer. By investigating the data, we find that users tend to repeat their queries at the end of a session and the frequency of queries in longer sessions is relatively higher. Therefore, both models can learn well in longer sessions. Nevertheless, we can observe that the improvement of CACM over NCM in medium or long sessions is obviously more significant than that in short sessions. Similar phenomenon can also be found in Figure 3(b). This indicates that CACM can capture users' information needs better by utilizing the contextual information. From the above results, it is evident that the session context can improve CACM in relevance estimation, which further boosts its ranking performance.

To further investigate how CACM works, we conduct a case study in Table 8. In this case, CACM outperforms NCM with a higher NDCG@3 value of 1.0 compared to 0.5837. We can observe that there is a recency effect in the interaction attention, which indicates that user behaviors in a nearby query are more important than those in the previous ones. This finding is consistent with the previous work [1]. Moreover, we find that the attention weight of Q4 is much higher than that of Q1-Q3, which implies the importance of the current query. In this case, we find that NCM has a poor NDCG because it regards $d_{4,3}$ as a high relevance document (a high relevance of 0.4519). However, we notice that in CACM, Q1>Q2 in query attention and $I_{1,3}$ has the largest interaction weight in Q1. This indicates CACM utilizes the contextual information that $d_{4,3}$ has showed up in Q1 but was not clicked, hence it regards it as an irrelevant document (a low relevance of 0.0302).

6.2.4 Estimated relevance and examination. Here we plot the estimated relevance and examination in Figure 4. From Figure 4(a), we can observe that the overall estimated examination probabilities are distributed with a top-heaviness across all ranks. There is a huge decay in examination from the top results to the bottom ones. This indicates that the examination predictor automatically learns the position bias through the data-driven training process. We also observe that CACM estimates higher examination probabilities at all ranks when a session is longer. In long sessions, user examination behaviors may be influenced by more complex factors thus they are less sensitive to the ranking positions. To further investigate



(a) Estimated examination at ranks.

(b) Distribution of estimated relevance and examination. (y-axis is in the log2 scale).

Figure 4: Estimated examination at ranks and the distribution of estimated relevance and examination in CACM.

how the CACM's outputs are like, we plot the distribution of its estimated relevance and examination in Figure 4(b). We see that estimated relevance scores mainly distribute in the [0.0, 0.2] interval, which implies that most results are marginally relevant. However, there are two peaks in the examination distribution, showing that both high and low examination probabilities account for a certain proportion. User examination behaviors may be more sensitive to the ranking positions, thus the estimated examination distributes more uniformly than relevance.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we propose a novel context-aware click model (CACM) for Web search. CACM consists of a relevance estimator as well as an examination predictor for both document ranking and click prediction. Specifically, the relevance estimator not only encodes the session-level context information into fixed-length vectors but also utilizes the pre-trained embeddings learned from a session-flow graph for session modeling. To explore the adaptiveness of the examination hypothesis, we further investigate the performances of several combination functions which integrate relevance and examination into click prediction.

Extensive experiments are conducted on an open web search session dataset. By answering the five research questions, we find that: 1) CACM can significantly outperform existing click models in both the relevance estimation and click prediction tasks. 2) The *exp_mul* function has the best overall performance among all the combination functions. In addition, those functions which support the examination hypothesis can better integrate relevance and examination into clicks. 3) The pre-trained embeddings and the two attention mechanisms are all beneficial for the context-aware relevance estimation. 4) Both the $\mathcal{L}_{\mathcal{R}}$ and $\mathcal{L}_{C}$ are crucial for modeling training. Since training the model with only one loss function will cause a huge performance drop on the other task, we should combine them for an optimal overall performance. 5) Incorporating the contextual information can improve the relevance estimation process, which results in better document ranking performance.

Through these experiments, we have a better understanding of the advantages and the limitations of CACM. The limitations can further inspire some future work. For example, due to the trade-off between $\mathcal{L}_{\mathcal{R}}$ and $\mathcal{L}_{C}$, we find that it is hard to jointly optimize the learning of both tasks through only click signals. More behavior signals should be exploited in the future for better session modeling.

8 ACKNOWLEDGEMENTS

This work is supported by the National Key Research and Development Program of China (2018YFC0831700) and Natural Science Foundation of China (Grant No. 61622208, 61732008, 61532011).

REFERENCES

[1] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2019. Context Attentive Document Ranking and Query Suggestion. *arXiv preprint arXiv:1906.02329* (2019).

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

[3] Alexey Borisov, Ilya Markov, Maarten de Rijke, and Pavel Serdyukov. 2016. A neural click model for web search. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 531–541.

[4] Alexey Borisov, Martijn Wardenaar, Ilya Markov, and Maarten de Rijke. 2018. A click sequence model for web search. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 45–54.

[5] Huanhuan Cao, Daxin Jiang, Jian Pei, Enhong Chen, and Hang Li. 2009. Towards context-aware search by learning a very large variable length hidden markov model from search logs. In *Proceedings of the 18th international conference on World wide web*. ACM, 191–200.

[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 *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 875–883.

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

[8] Jia Chen, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2019. TianGong-ST: A New Dataset with Large-scale Refined Real-world Web Search Sessions. In *Proceedings of the 28th ACM International on Conference on Information and Knowledge Management*. ACM.

[9] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. 2015. Click models for web search. *Synthesis Lectures on Information Concepts, Retrieval, and Services* 7, 3 (2015), 1–115.

[10] Aleksandr Chuklin, Pavel Serdyukov, and Maarten De Rijke. 2013. Using intent information to model user behavior in diversified search. In *European Conference on Information Retrieval*. Springer, 1–13.

[11] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[12] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. 2008. An experimental comparison of click position-bias models. In *Proceedings of the 2008 international conference on web search and data mining*. ACM, 87–94.

[13] Mostafa Dehghani, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. 2017. Learning to attend, copy, and generate for session-based query suggestion. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 1747–1756.

[14] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W Bruce Croft. 2017. Neural ranking models with weak supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 65–74.

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

[16] Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 109–117.

[17] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

[18] Dongyi Guan, Sicong Zhang, and Hui Yang. 2013. Utilizing query change for session search. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 453–462.

[19] Fan Guo, Chao Liu, Anitha Kannan, Tom Minka, Michael Taylor, Yi-Min Wang, and Christos Faloutsos. 2009. Click chain model in web search. In *Proceedings of the 18th international conference on World wide web*. ACM, 11–20.

[20] Fan Guo, Chao Liu, and Yi Min Wang. 2009. Efficient multiple-click models in web search. In *Proceedings of the second acm international conference on web search and data mining*. ACM, 124–131.

[21] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 55–64.

[22] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*. 2042–2050.

[23] Jyun-Yu Jiang and Wei Wang. 2018. RIN: Reformulation Inference Network for Context-Aware Query Suggestion. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 197–206.

[24] Thorsten Joachims, Laura A Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately interpreting clickthrough data as implicit feedback. In *Sigir*, Vol. 5. 154–161.

[25] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[26] Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

[27] Liangda Li, Hongbo Deng, Anlei Dong, Yi Chang, Ricardo Baeza-Yates, and Hongyuan Zha. 2017. Exploring query auto-completion and click logs for contextual-aware web search and query suggestion. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 539–548.

[28] Qiang Liu, Feng Yu, Shu Wu, and Liang Wang. 2015. A convolutional click prediction model. In *Proceedings of the 24th ACM international on conference on information and knowledge management*. ACM, 1743–1746.

[29] Jiyun Luo, Sicong Zhang, and Hui Yang. 2014. Win-win search: Dual-agent stochastic game in session search. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 587–596.

[30] Jiaxin Mao, Cheng Luo, Min Zhang, and Shaoping Ma. 2018. Constructing Click Models for Mobile Search. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 775–784.

[31] Bhaskar Mitra. 2015. Exploring session context using distributed representations of queries and reformulations. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. ACM, 3–12.

[32] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1291–1299.

[33] Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-oriented query reformulation with reinforcement learning. *arXiv preprint arXiv:1704.04572* (2017).

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

[35] Milad Shokouhi, Marc Sloan, Paul N Bennett, Kevyn Collins-Thompson, and Siranush Sarkizova. 2015. Query suggestion and data fusion in contextual disambiguation. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 971–980.

[36] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 553–562.

[37] Chao Wang, Yiqun Liu, Meng Wang, Ke Zhou, Jian-yun Nie, and Shaoping Ma. 2015. Incorporating non-sequential behavior into click models. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 283–292.

[38] Kuansan Wang, Nikolas Gloy, and Xiaolong Li. 2010. Inferring search behaviors using partially observable Markov (POM) model. In *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 211–220.

[39] Biao Xiang, Daxin Jiang, Jian Pei, Xiaohui Sun, Enhong Chen, and Hang Li. 2010. Context-aware ranking in web search. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 451–458.

[40] Xiaohui Xie, Jiaxin Mao, Yiqun Liu, Maarten de Rijke, Qingyao Ai, Yufei Huang, Min Zhang, and Shaoping Ma. 2019. Improving Web Image Search with Contextual Information. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM, 1683–1692.

[41] Danqing Xu, Yiqun Liu, Min Zhang, Shaoping Ma, and Liyun Ru. 2012. Incorporating revisiting behaviors into click models. In *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM, 303–312.

[42] Wanhong Xu, Eren Manavoglu, and Erick Cantu-Paz. 2010. Temporal click model for sponsored search. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 106–113.

[43] Hai-Tao Yu, Adam Jatowt, Roi Blanco, Joemon M Jose, and Ke Zhou. 2019. A Rank-biased Neural Network Model for Click Modeling. In *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval*. ACM, 183–191.

[44] Junqi Zhang, Yiqun Liu, Shaoping Ma, and Qi Tian. 2018. Relevance estimation with multiple information sources on search engine result pages. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 627–636.

[45] Yuchen Zhang, Weizhu Chen, Dong Wang, and Qiang Yang. 2011. User-click modeling for understanding and predicting search-behavior. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1388–1396.

[46] 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 *Twenty-Eighth AAAI Conference on Artificial Intelligence*.