Integrating Representation and Interaction for Context-Aware Document Ranking

HAONAN CHEN, ZHICHENG DOU, QIANNAN ZHU, and XIAOCHEN ZUO,
Gaoling School of Artificial Intelligence, Renmin University of China, China
JI-RONG WEN, Beijing Key Laboratory of Big Data Management and Analysis Methods, China

Recent studies show that historical behaviors (such as queries and their clicks) contained in a search session can benefit the ranking performance of subsequent queries in the session. Existing neural context-aware ranking models usually rank documents based on either latent representations of user search behaviors or the word-level interactions between the candidate document and each historical behavior in the search session. However, these two kinds of models both have their own drawbacks. Representation-based models neglect fine-grained information on word-level interactions, whereas interaction-based models suffer from the length restriction of session sequence because of the large cost of word-level interactions. To complement the limitations of these two kinds of models, we propose a unified context-aware document ranking model that takes full advantage of both representation and interaction. Specifically, instead of matching a candidate document with every single historical query in a session, we encode the session history into a latent representation and use this representation to enhance the current query and the candidate document. We then just match the enhanced query and candidate document with several matching components to capture the fine-grained information of word-level interactions. Rich experiments on two public query logs prove the effectiveness and efficiency of our model for leveraging representation and interaction.

CCS Concepts: • **Information systems → Retrieval models and ranking;**

Additional Key Words and Phrases: Document ranking, session search, neural-IR

ACM Reference format:
Haonan Chen, Zhicheng Dou, Qiannan Zhu, Xiaochen Zuo, and Ji-Rong Wen. 2023. Integrating Representation and Interaction for Context-Aware Document Ranking. *ACM Trans. Inf. Syst.* 41, 1, Article 21 (January 2023), 23 pages.
https://doi.org/10.1145/3529955

This work was supported by National Natural Science Foundation of China (grant nos. 61872370 and 61832017), Beijing Outstanding Young Scientist Program (grant no. BJJWZYJH012019100020098), and Intelligent Social Governance Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China. I also wish to acknowledge the support provided and contribution made by Public Policy and Decision-making Research Lab of RUC.

Authors' addresses: H. Chen, Z. Dou (corresponding author), Q. Zhu, and X. Zuo, Gaoling School of Artificial Intelligence, Renmin University of China, No. 59 Zhongguancun Street, Haidian District, Beijing, 100872, China; emails: hnchen@ruc.edu.cn, dou@ruc.edu.cn, zhuqiannan@ruc.edu.cn, zuoxc@ruc.edu.cn; J.-R. Wen, Beijing Key Laboratory of Big Data Management and Analysis Methods, No. 59 Zhongguancun Street, Haidian District, Beijing, China; email: jrwen@ruc.edu.cn.

1046-8188/2023/01-ART21 $15.00
https://doi.org/10.1145/3529955

1 INTRODUCTION

The search engine has become an increasingly popular way for people to get information from the Web. It receives a query and returns a ranked document list for the user to browse and click. Usually, users' search intents are complex so that they need to try multiple queries and browse several websites to obtain the information that satisfies their search intent. This series of user search behaviors (i.e., issued queries and browsed documents) is referred to as a search session [18, 29]. Leveraging contextual information in a session has been proved useful for inferring a user's current search intent and improve ranking [1–3, 24, 31]. Figure 1 shows an example, which helps us understand how a user's search context benefits the user's current search. Supposing a user is issuing a query "apple" and has searched "microsoft." The current search intent is likely to be browsing information about the "apple" company. However, if the user has searched "banana" a minute before, the user may be looking for the fruit "apple" now. From this example, we can see that the query "apple" is ambiguous and has different meanings under different search contexts. By considering the historical queries ("banana" or "microsoft"), we can identify the real intent of the current query and improve the ranking performance. Hence, utilizing contextual information of the current search session is valuable for mining a user's actual search intent, which is the primary motivation of context-aware document ranking.

There are already some early works focused on session context modeling [3, 4, 13, 26, 31]. Most extract specific features from the search session to analyze the user's current search intent. For example, Shen et al. [26] use statistical language models to model session context. Recently, with the widespread use of deep learning technologies in the artificial intelligence field, some neural context-aware document ranking models have been proposed [1, 2, 24]. Following [1], we roughly divide existing neural context-aware document ranking models into two categories: **representation-based** models [1, 2] and **interaction-based** models [24]. Representation-based models usually encode queries, documents, and session context into hidden representations and compute the ranking scores of the candidate documents based on the encoded history representation and the representation of the current query. In contrast, interaction-based models often rank documents by the fine-grained term-level interactions between queries and documents. For example, Qu et al. [24] concatenate all search behaviors of the session into a long sequence and put it into BERT to get the word-level interaction-based contextual representations for ranking.

These neural context-aware ranking models have achieved great performance. Nevertheless, both kinds of models have their own advantages and drawbacks. Representation-based models overlook the fine-grained word-level interactions [1, 2]. Interaction-based models try to compute the interactions between every two words in the session sequence (i.e., historical search behaviors, the current query, and candidate documents), which leads to a larger calculation cost with the increasing length of the input sequence. There are a few works that attempt to integrate both interaction-based features and representation-based features for ad-hoc search [17, 23], which do not include the session context into interactions. However, to the best of our knowledge, there has not been any attempt to leverage both representation- and interaction-based approaches for context-aware document ranking. In this article, we attempt to take full advantage of representation and interaction to perform session search.

The challenge of leveraging representation and interaction for context-aware document ranking is how to incorporate session context into word-level interactions while reducing calculation cost. In contrast to HBA-Transformers [24] making every behavior (historical queries and documents) in the search session interact with each other on the word level, we attempt to encode the session history into a latent representation and use it to enhance the word-level interactions between the current query and the candidate documents. In other words, **instead of matching every query**

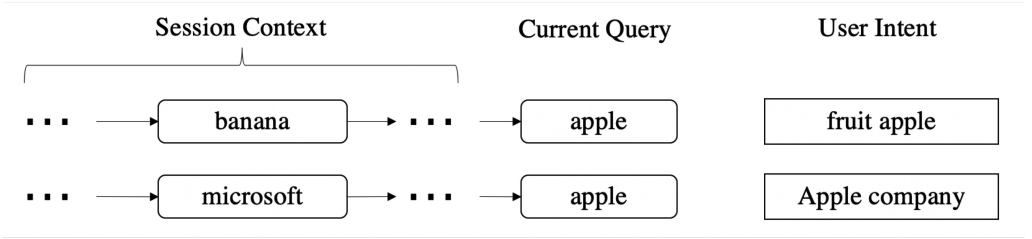

Fig. 1. An example of session contexts reflecting different search intents under the same query.

and document in a session, we alternatively use the history representation to enhance the last ones of the session (i.e., the current query and the candidate document) and then match the last ones on the word level. This will significantly reduce the cost of interaction and, at the same time, it keeps the signals of historical representations. This is a lightweight integration of both representation and interaction signals for context-aware ranking. In addition, we generate supplemental queries to interact with the candidate document. These supplemental queries are supposed to contain richer information about the current user intent than the original queries and will help improve ranking quality.

We propose a **R**epresentation and **I**nteraction-fused **C**ontext-aware document **R**anking model (RICR), which can utilize the advantages of both representation- and interaction-based approaches. As shown in Figure 2, RICR is composed of three modules:

(1) The **session history encoder** module attempts to represent the session history into a latent representation. It first uses the **behavior encoding** sub-module to obtain an attentive representation for each historical behavior with respect to the current query. This sub-module consists of the term-level query-aware attention mechanism and the inner-attention mechanism. Term-level query-aware attention can capture word-level interactions between the current query and the historical behaviors. In addition, the inner-attention mechanism [19] encodes different weights of words in the current query and uses them to integrate the word-level interaction between the current query and a specific behavior for the attentive representation of that behavior. Then, the **session history encoder** module applies a Gated Recurrent Unit (GRU) [6] on the attentive representations of historical behaviors to obtain the sequential information of the session context.
(2) Next, the **information enhancing** module employs another two GRUs with the encoded history representation as the initial hidden state to learn the enhanced word-level representations of the current query and the candidate documents. This module uses the final hidden state of the GRU used for enhancing the current query to select a supplemental query. This can help our model deal with a user's search intent being more complex than the issued query, which contains only a small set of keywords.
(3) Finally, with the enhanced and supplemental representations ready, the **matching and scoring** module computes the matching score for document ranking with the Conv-KNRM [8] component.

Rich experiments on an AOL search log and Tiangong-ST search log show that our model not only yields a state-of-the-art performance but also reduces considerable calculation cost versus interaction-based context-aware models (e.g., HBA-Transformers). RICR manages to cut over 80% of parameters versus HBA and reduces training and inference cost considerably. More analysis of the cost reduction is provided in Section 5.6. In addition, we conduct analysis of the supplemental

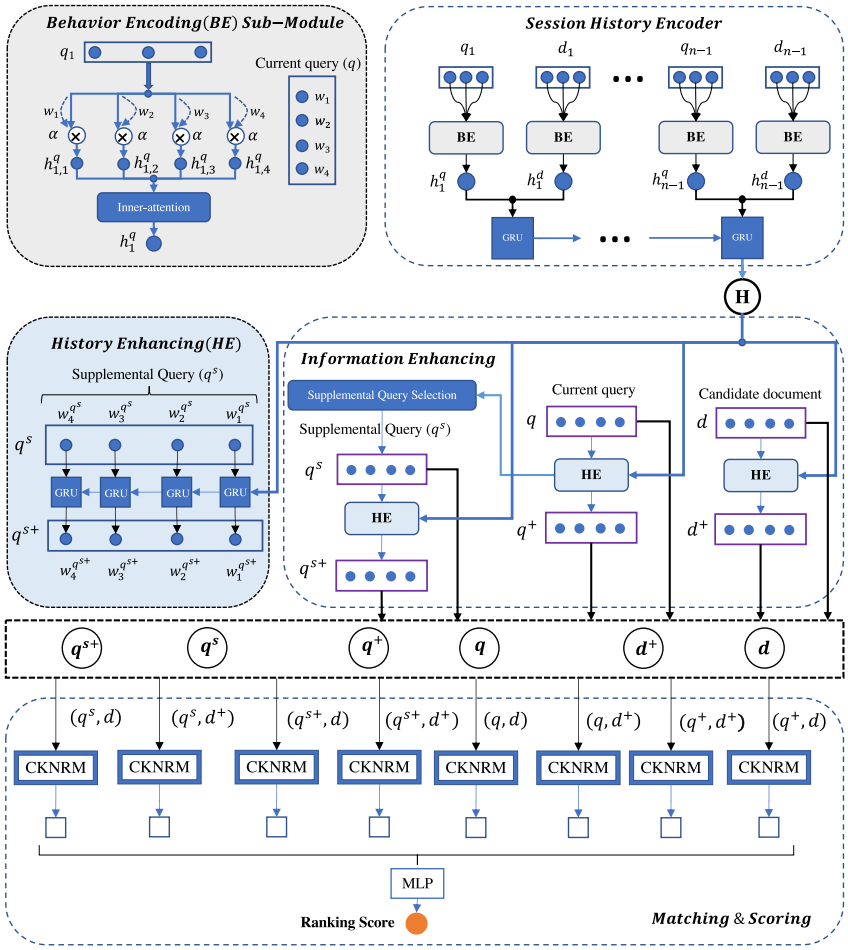

Fig. 2. The detailed structure of RICR. Our model has three modules. (1) Session history encoder: RICR uses the Behavior Encoding (BE) sub-module to encode historical behaviors. BE consists of the term-level query-aware attention mechanism and the inner attention mechanism. Then, we use a GRU to model the sequential history information and take the last hidden state of it as the overall history representation (**H**). (2) Information enhancing: RICR uses **H** as the initial hidden state of two GRUs to enhance q and d. We also use the last hidden state of the GRU used for enhancing q to select a supplemental query q^s for q and enhance q^s with a GRU as well. (3) Matching and scoring: With the enhanced and supplemental representations ready, RICR computes the matching score for document ranking with the Conv-KNRM component.

query selection module in Section 5.5, which proves that RICR manages to select a supplemental query that can enhance the information obtained by the original query.

Our main contributions can be summarized as follows:

(1) We propose a context-aware document-ranking model, which takes full advantages of both representation and interaction with low calculation cost.

(2) We develop a Behavior Encoding sub-module that utilizes the word-level information of the current query to learn historical behaviors' attentive representations.

(3) We use the overall representation of the session history to enhance the word-level representations of the current query and the candidate documents for later matching. We also select a

supplemental query from the query database to enhance our understanding of the current query, which can make our model more robust.

The rest of the article is organized as follows. Related works, including representation-based and interaction-based models, are briefly introduced in Section 2. We introduce details of our context-aware document-ranking model RICR in Section 3. In Section 4, we describe the datasets, experimental settings, and selected baselines. In Section 5, we compare and analyze the experimental results. We present our conclusions in Section 6.

2 RELATED WORK

2.1 Neural Ranking

There are already ad-hoc neural ranking models that only use the current query to re-rank the candidate document. These ad-hoc models can be split into two groups: representation-based models [15] and interaction-based models [8, 15, 23, 32]. Representation-based ad-hoc models usually encode the current query and the candidate document into hidden vectors, then compute the ranking score. ARC-I [15], for example, uses Convolutional Neural Networks (CNNs) to represent the current query and the candidate document, and utilizes a Multi-Layer Perceptron (MLP) to calculate the ranking score. Interaction-based ad-hoc models calculate the ranking score based on word-level interaction-based information between the current query and the candidate document. For example, on the interaction matrix of the current query and the candidate document, ARC-II [15] uses a 2D-CNN. On a word-by-word basis, Xiong et al. [32] extract the aspects of interaction between the current query and the candidate document. Kernel-pooling is used to provide soft match signals for ranking.

The main difference between representation-based and interaction-based ad-hoc models is that representation-based models encode query and documents into latent vectors whereas those that are interaction based mine the information from word-level interactions between query and documents.

2.2 Context-Aware Ranking

Previous studies have revealed that queries issued by users are usually short and ambiguous [7, 27] and that these queries' intents are hard to understand. Modeling a user's search context inside the current search session has been proved to help understand a user's real search intent [3, 11, 18, 33]. There are already early works focused on modeling session context [3, 4, 13, 26, 31]. For example, Bennett et al. [3] have proved contextual information inside the current search session–that is, short-term session behaviors–to be important for improving the performance of retrieval. This work unifies prior works on short- and long-term behaviors' contribution to personalized retrieval. Their work also shows that as the search session progresses, the role of the current session's previous behaviors in satisfying the user's search intent becomes increasingly important. Shen et al. [26] use statistical language models to model contextual session information. White et al. [31] mine other users' search sessions that are similar to the current one and use them to identify the documents that may have a high rank. They generate features to build rich models of a current user's search task for finding similar tasks in the history search log. These traditional retrieval models have made great progress, but most focus only on specific features, overlooking those that may be equally valuable.

With the recent emergence of deep learning, lots of neural information retrieval models have been studied [1, 2, 8, 24, 32]. These deep models gradually resolve the problem that the extracted features are limited. Following [1], we roughly divided existing neural context-aware document ranking models into two categories: representation-based and interaction-based models.

2.2.1 Representation-Based Models. Representation-based models usually encode user search behaviors and the session context into hidden representations and compute the ranking scores of the candidate documents based on both the session context and the current query.

Ahmad et al. [1] propose a multi-task neural session relevance framework (M-NSRF) to predict a user's click in the current session and the user's next query jointly. They use long short-term memory (LSTM) [14] to model queries, documents, and the sequence of historical queries into latent representations. Then, they concatenate the representations of current query and session history and take the result as the user's search intent, re-ranking candidates based on this operation.

Ahmad et al. [2] supplement their previous work [1] by adding an attention mechanism into encoding queries and documents. They also model the sequence of previously clicked documents in the current session as clickthrough information into the context representation. Though representation-based models can include rich history information of the current session inside latent vectors, they inevitably lose the fine-grained information of word-level interactions between queries and documents.

2.2.2 Interaction-Based Models. There are already some ad-hoc retrieval models that focus on word-level interaction [8, 17, 23, 32]. However, only a few works pay attention to an interaction-based neural ranking model for session search. Qu et al. [24] design a Hierarchical Behavior Aware Transformer (HBA-Transformer). They regard queries, clicked documents, and skipped documents all as user behaviors and concatenate all behaviors in a session into a long sequence. They put the sequence into a BERT [9] encoder to make every two behaviors in it interact with each other and get their contextual word-level interaction-based representations. Then, they use a transformer structure with behavior embedding and relative position embedding to further enhance the representations. Finally, the representation of the first token ("[CLS]") is used to calculate the ranking score. This is the state-of-the-art method.

Interaction-based models mainly focus on fine-grained information, which makes them able to represent user behaviors with word-level interactions. Nevertheless, interaction-based models often cannot handle a long sequence of behaviors because of high calculation cost. Qu et al. [24] use a history window to resolve this issue. They only concatenate user's historical behaviors in a fixed window size in the encoding stage. Still, they will inevitably neglect some useful information of those behaviors that are not in the window.

In our work, we design a neural context-aware document-ranking model that leverages representation and interaction. Instead of making every two search behaviors interact with each other, RICR enhances the current query and the candidate document with the encoded session context and then matches them. Experimental results show that our method is simple but effective.

3 OUR METHOD

The main goal of our model is to capture the fine-grained information of word-level interactions with the session context taken into consideration while reducing the calculation cost. Rather than matching every single query and document in a session, RICR uses the encoded representation of session history to enhance the word-level interaction between the current query and the candidate document, which manages to significantly reduce the cost.

3.1 Problem Definition

Before shedding light on RICR, we first need to state the task and some notations. In a search session, to express a complex intent, a user might need to try several different queries and browse some websites to obtain adequate information. The user's former search history, that is, search context, has influence on the user's current search activity. Therefore, it is crucial to use session

context to facilitate current search. The search context $\mathcal{S}$ is composed of the historical queries and their corresponding clicked documents:

$$\mathcal{S} = \{(q_1, D_1), (q_2, D_2), \ldots, (q_{n-1}, D_{n-1})\}, \tag{1}$$

where q_i is the i-th query of the session and $D_i = \{d_{i,1}, \ldots, d_{i,M}\}$ refers to the corresponding list of clicked documents. As stated in [2, 24], the skipped documents have little value; thus, we do not include them in $\mathcal{S}$. The goal of context-aware ranking is to rank the candidate document list $\mathcal{D}_n$ based on both the current query q_n and the session context $\mathcal{S}$. Specifically, we rank $\mathcal{D}_n$ by the ranking scores of every document in it. The ranking score of a candidate document d under the session history $\mathcal{S}$ and the current query q (short for q_n) is denoted as $P(d|\mathcal{S}, q)$.

Note that when no ambiguity is involved, we will refer to q and d as the current query and the corresponding candidate document to be ranked, respectively.

3.2 Overview

In this work, we propose a context-aware document-ranking model to encode $\mathcal{S}$ into a latent representation and use it to enhance the word-level representations of q and d. Based on these enhanced representations, RICR obtains fine-grained information of word-level interactions in the matching module. Specifically, as shown in Figure 2, RICR can be divided into three main modules: a *session history encoder module*, *information enhancing module*, and *matching and scoring module*. With the statement and notations stated in Section 3.1, we give a brief introduction of these three modules as follows.

(1) Session History Encoder. As shown in the upper right part of Figure 2, the goal of this module is to encode the session context $\mathcal{S}$ into a single latent representation for further usage in the enhancing module. RICR first uses a Behavior Encoding (BE) sub-module (illustrated in the upper left part of Figure 2) to encode each behavior in $\mathcal{S}$. The BE sub-module consists of the term-level query-aware attention mechanism and the inner-attention mechanism [19], which can utilize the word-level information of q to encode historical behaviors. Then, a GRU [6] is applied to the attentive representations of encoded historical behaviors to obtain the sequential information of $\mathcal{S}$. We use the final hidden state of this GRU as the overall history representation.

(2) Information Enhancing. As shown in the middle of Figure 2, the goal of this module is to use the overall history representation to enhance the information of queries and documents on the word level. RICR applies two GRUs with the encoded history as their initial states to encode the sequential information buried in the word sequences of q and d, respectively, which can also incorporate historical information into the enhancement. In addition, we use the final hidden state of the GRU used for enhancing q to select a supplemental query q^s to make our model more robust.

(3) Matching and Scoring. As shown in the lower part of Figure 2, the goal of this module is to gain fine-grained information of word-level interactions. With the original and enhanced representations of q, q^s, and d ready, RICR utilizes several matching components (Conv-KNRM [8]) to compute the matching scores of all paired combinations between these queries and documents.

3.3 Session History Encoder

With this module, RICR attempts to model session history $\mathcal{S}$ into a single vector for further usage in the enhancing module. To obtain this latent representation, we use the Behavior Encoding (**BE**) sub-module at the lower level to encode historical search behaviors and a GRU at the higher level to model the sequential structure of session histories.

(1) **Encoding Historical Search Behaviors.** In this part, for each behavior in $\mathcal{S}$, given a sequence of T words $\{o_1, \ldots, o_T\}$, we first embed them into d_w-dimensional vectors $\{\mathbf{w}_1, \ldots, \mathbf{w}_T\}$ using a pretrained word-embedding model word2vec [22]. Instead of those complex embedding

models, such as BERT [9], we choose word2vec as our initialized embedding model to reduce the cost. As stated in Section 3.1, there may be several clicked documents ($D_i = \{d_{i,1}, \ldots, d_{i,M}\}$) for each historical query q_i. We simply calculate the mean of these documents' word embedding vectors to get a fixed-length vector for further performing of the attention mechanism, that is, $d_i = \text{mean}(D_i)$. Note that we need the word-level information in the information enhancing module. Thus, we take the average of these word embeddings over all clicked documents instead of their tokens. Let's assume that the shape of the tensor of clicked documents is [the batch size, the length of session history, the number of clicked documents, the number of words in the document, the size of word embeddings]; the average is taken on the third dimension, that is, document-level. We will refer to d_i as the aggregated representation of the corresponding clicked documents of q_i. We leave a more advanced approach to aggregating clicked documents as our future work.

Over the word-embedding layer, we encode historical search behaviors using the BE sub-module as follows:

$$\mathbf{h}_i^q = \text{BE}(q_i), \tag{2}$$

$$\mathbf{h}_i^d = \text{BE}(d_i), \tag{3}$$

where $\mathbf{h}_i^q$ and $\mathbf{h}_i^d$ are the output representations of q_i and d_i, and the BE sub-module consists of the **term-level query-aware attention** mechanism and the **inner-attention** mechanism. Let us take the encoding of q_i as example to walk through the structure of BE.

q_i will first go through the term-level query-aware attention mechanism. The motivation of this mechanism is that different words of the current query q have different focuses on the words of a historical behavior. For example, the current issued query is "Tencent manager" and a historical query is "Alibaba president." The word "manager" will pay more attention to "president" than "Alibaba." However, for "Tencent," "Alibaba" draws more attention. We use the term-level query-aware attention mechanism to capture these various focuses. We compute the focused result of $\mathbf{q}_{n,j}$ (we use it to represent the embedding vector of the j-th word in q to avoid ambiguity) on q_i as follows:

$$\mathbf{h}_{i,j}^q = \text{softmax}\left(\frac{(\mathbf{q}_{n,j}\mathbf{W}^q + \mathbf{b}^q)(\mathbf{q}_i\mathbf{W}^k + \mathbf{b}^k)^T}{\sqrt{d_w}}\right)(\mathbf{q}_i\mathbf{W}^v + \mathbf{b}^v), \tag{4}$$

where $\mathbf{q}_i$ is the embedding of q_i, $\mathbf{h}_{i,j}^q$ is the attentive representation that aggregates the focuses of $\mathbf{q}_{n,j}$ on every word of q_i, $\mathbf{W}^q$, $\mathbf{b}^q$, $\mathbf{W}^v$, $\mathbf{b}^v$, $\mathbf{W}^k$, and $\mathbf{b}^k$ are the parameters to apply linear transformations on representations, and d_w is the dimension of word embeddings. Note that all $\mathbf{W}$ of the linear transformations in the term-level query-aware attention mechanism do not change the dimension of the vector to which it is applied.

Then, we apply the inner-attention mechanism [19] on the obtained attentive representations to identify the weights of words in q. The intuition here is that the importance of each word in q is different with respect to the encoding of q_i. For example, the current query is "history of China." The weights of "history" and "China" should be higher than that of "of." We re-weight the word-level attentive representations of a behavior, for example, $\mathbf{h}_i^q$, as follows:

$$\mathbf{h}_i^q = \sum_{j=1}^{|q|} \alpha_j \mathbf{h}_{i,j}^q, \tag{5}$$

$$\alpha_j = \text{softmax}\left(\tanh(\mathbf{h}_{i,j}^q\mathbf{W}_2 + \mathbf{b}_2)\mathbf{W}_1 + \mathbf{b}_1\right), \tag{6}$$

where $\mathbf{h}_i^q$ is the weighted output representation of q_i, $\tanh(\cdot)$ is a tangent activation function, $\mathbf{W}_2 \in \mathbb{R}^{d_w \times d_w}$, $\mathbf{W}_1 \in \mathbb{R}^{d_w \times 1}$, and $\mathbf{b}_2$ and $\mathbf{b}_1$ are the parameters of a two-layer perceptron to compute the

attention weight. Through this, the word of q that is more informative, its corresponding focused result on a search behavior (e.g., $\mathbf{h}_{i,j}^{q}$), would have a larger weight in the aggregated representation of that behavior ($\mathbf{h}_{i}^{q}$).

Note that we use the developed BE sub-module instead of other sophisticated structures such as Transformer [28] because we attempt to encode each historical behavior with only its own information and take q into consideration to reduce the cost.

(2) **Modeling Sequential Session Histories.** In a search session, a user often issues several queries, browses the returned documents, and clicks on some of them. The sequential information of the session histories is crucial for inferring the user's current search intent. For example, a user's current query is "population." Without any historical information, we may be confused about the user's intent. However, if we know that the user's last two queries are "China" and "capital," we will infer that the user's current intent is searching for "population of China's capital," which is a concatenation of the sequential history information. In this part, we attempt to model this kind of sequential information. Specifically, for the representations of each historical query (q_i) and the corresponding clicked documents (d_i), we first combine them to get a fixed-length vector of the behaviors made at timestamp i. Then, intending to utilize the recurrent structure of a Recurrent Neural Network (RNN) to model sequential information explicitly, we use a GRU to model the sequential session histories. We use a GRU rather than LSTM [14] to reduce the cost.

For the attentive representations (obtained in Equations (2) and (3)) of a historical query ($\mathbf{h}_{i}^{q}$) and its corresponding clicked documents ($\mathbf{h}_{i}^{d}$), we use a multi-layer perceptron (MLP) with the tanh(·) as the activation function to combine them into a fixed-length vector:

$$\mathbf{h}_i = \text{MLP}\left(\mathbf{h}_{i}^{q}, \mathbf{h}_{i}^{d}\right), \tag{7}$$

where $\mathbf{h}_i$ is the combined representation of the behaviors at timestamp i. Then, a GRU is used as the encoder of sequential historical information. It computes the hidden state of each step as follows:

$$\mathbf{s}_i = \text{GRU}(\mathbf{s}_{i-1}, \mathbf{h}_i), \tag{8}$$

where $\mathbf{s}_i \in \mathbb{R}^{d_h}$ is the hidden state at the i-th behavior and d_h is the dimension of the GRU's hidden unit. We take the last hidden state $\mathbf{s}_{n-1}$ as the overall encoded history representation $\mathbf{H}$.

3.4 Information Enhancing

In this module, RICR first utilizes two GRUs with the encoded history $\mathbf{H}$ as their initial hidden states to enhance the information of q and d. Then RICR uses the final hidden state of the GRU used for enhancing q to select a supplemental query and enhances it as well. These two ways of information enhancing are illustrated as follows.

(1) **Enhancing the representation of q and d with the Session History.** The main goal of our model is to obtain the information of word-level interactions along with contextual information, while simultaneously reducing the cost. Thus, instead of matching every behavior in the search session, we attempt to use the history ($\mathbf{H}$) to enhance the current query q and the candidate document d, and then match them. Most works represent d independently from the session context [1, 2, 11]. However, we believe it should also be represented contextually. For example, let us suppose that d is "mouse Amazon." If a historical clicked document is "computer mouses," we will know that d may lead the user to where one can purchase computer mouses, not animal mice. In the following, we describe how we utilize the encoded history $\mathbf{H}$ to enhance the information of q and d.

We start by using two GRUs to enhance the word-level representations of q and d, respectively. The recurrent structure of an RNN can give the words contextual representations by encoding the

sequential information of the sentence. In addition, we use the encoded history $\mathbf{H}$ as the initial hidden state of the GRU to add session history information into the enhancing process. We obtain the word-level enhanced representations of q and d as follows:

$$q^+ = \left\{\mathbf{w}_1^{q+}, \ldots, \mathbf{w}_T^{q+}\right\} = \text{Enhance}\left(\left\{\mathbf{w}_1^{q}, \ldots, \mathbf{w}_T^{q}\right\}\right), \tag{9}$$

$$d^+ = \left\{\mathbf{w}_1^{d+}, \ldots, \mathbf{w}_T^{d+}\right\} = \text{Enhance}\left(\left\{\mathbf{w}_1^{d}, \ldots, \mathbf{w}_T^{d}\right\}\right), \tag{10}$$

where $\{\mathbf{w}_1, \ldots, \mathbf{w}_T\}$ are the embedded word vectors from the word-embedding layer and $\{\mathbf{w}_1^+, \ldots, \mathbf{w}_T^+\}$ are the enhanced embeddings. The following gives the process of word-level enhanced representations:

$$\mathbf{s}_i = \text{GRU}(\mathbf{s}_{i-1}, \mathbf{w}_i), \mathbf{s}_0 = \mathbf{H}, \tag{11}$$

$$\mathbf{w}_i^+ = \text{MLP}(\mathbf{s}_i), \tag{12}$$

where $\mathbf{s}_0$ is the initial hidden state, $\mathbf{s}_i \in \mathbb{R}^{d_h}$ is the hidden state at the i-th word, d_h is the dimension of the GRU's hidden unit, $\mathbf{w}_i^+$ is the enhanced word representation, MLP$(\cdot)$ is a multi-layer perceptron with tanh$(\cdot)$ as the activation function.

(2) **Enhancing q with a Supplemental Query.** A user issuing a query might input only a set of keywords of interest. For example, a user issues a query "wedding songs" whereas the actual search intent is to find a song to dance with at the user's son's wedding. This kind of search intent is not fully expressed by the issued query, which makes it hard to satisfy, especially with a lack of historical behaviors. To deal with this, we attempt to mine a supplemental query for q from the query database to help us understand it, which can make our model more robust. Note that the query database we use here is the query set of the training dataset. We select 9 candidate queries from the database for each query. Following [21], we evaluate a candidate's supplemental rate based on the following function:

$$\text{sup}(q^c|q) = \text{spe}(q^c|q) + \text{sim}(q^c|q), \tag{13}$$

where:

(1) q^c is the candidate query, $\text{sup}(q^c|q)$ is the supplemental rate of q^c against q;
(2) $\text{spe}(q^c|q) = \frac{\text{len}(q^c)-\text{len}(q)}{\text{len}(q)}$ when every word of q^c appears in q, otherwise, $\text{spe}(q^c|q) = 0$. This component computes the specificity between q^c and q.
(3) $\text{sim}(q^c|q)$ is the similarity between q^c and q. We use the Python class SequenceMatcher[1] to compute the similarity here. We choose it because it is a human-friendly longest contiguous and junk-free sequence comparator. We leave more advanced approaches to calculating similarity, for example, semantic similarity, as our future work.

We use the function above to choose the top 9 queries in the database as candidates for every query. Following [21], to ensure that if q matches the user's current search intent or all candidates are worse than q, we also add q into the candidate set, which makes each candidate set contain 10 queries.

To choose one from the candidates, we first use the mean of the word-embedding vectors to represent them. Then, for each candidate, we concatenate its representation with the final hidden state of the GRU that is used to enhance q ($\mathbf{s}_T^q$, introduced in Equation (11)). Next, we apply an MLP with relu$(\cdot)$ as the activation function on it:

$$p_i = \text{MLP}\left(\left[\mathbf{s}_T^q, q_i^c\right]\right), \tag{14}$$

[1] https://docs.python.org/3/library/difflib.html.

where q_i^c is the i-th candidate. Then, p_i goes through a softmax function to get the probability of selecting q_i^c.

Finally, the q_i^c that has the largest p_i is selected as the supplemental query q^s. The intuition here is that we utilize the information of both $\mathcal{S}$ and q to infer which candidate we should select. In the case that our model tries to degenerate to q, it will select q from the candidates. Moreover, we obtain the enhanced version of the supplemental query with the same process illustrated in Section 3.4:

$$q^{s+} = \text{Enhance}(q^s). \tag{15}$$

3.5 Matching and Scoring

In this section, we describe how we perform word-level interactions between queries and documents and how we integrate different aspects of interactions to get an overall score.

In this module, to capture the fine-grained information of word-level interactions, we use several matching components to calculate the ranking score of d. There are some promising matching components for re-ranking, such as KNRM [32], Conv-KNRM [8], Duet [23], and so on. RICR uses Conv-KNRM as its matching component because of its ability to model n-gram soft matches, which can capture the word-level information more thoroughly. We construct RICR on top of Conv-KNRM instead of BERT because we want to model long-sequence session context with less GPU memory and lower computation cost in comparison with BERT-based models. (BERT-based models are often incapable of dealing with long session sequences. HBA simply uses a small history window to deal with only a short session sequence.) There will be a more thorough discussion of efficiency in Section 5.6. We obtain four scores to get a thorough understanding of the relevance between q and d: $P(d, q)$, $P(d, q^+)$, $P(d^+, q)$, and $P(d^+, q^+)$. We obtain another four scores to get a supplemental understanding of the relevance by matching q^s and d: $P(d, q^s)$, $P(d^+, q^s)$, $P(d, q^{s+})$, and $P(d^+, q^{s+})$. q and d are the original current query and candidate document. q^+ and d^+ are the information-enhanced version of them. q^s and q^{s+} are defined in Equation (15). Each score is computed as follows:

$$P(d, q) = \text{CKNRM}(d, q), \tag{16}$$

where P(d, q) is the matching score between a query and the corresponding candidate document. CKNRM is short for Conv-KNRM.

Let's shed light on $P(d, q)$ as an example. The Conv-KNRM component first applies convolution filters to compose n-grams from the text. We first construct a similarity matrix M. Each element $M_{i,j}$ of the matrix M is the embedding similarity between the i-th word q_i of q and the j-th word d_j of d by cosine similarity. Then, we use some RBF kernels on the similarity matrix M to convert word-level interactions to multi-level soft-match features $\phi(M)$ between the query and document:

$$\phi(M) = \sum_{i=1}^{T} \log \vec{K}(M_i), \tag{17}$$

$$\vec{K}(M_i) = \{K_1(M_i), \ldots, K_K(M_i)\}, \tag{18}$$

$$K_k(M_i) = \sum_j \exp\left(-\frac{(M_{i,j} - \mu_k)^2}{2\sigma_k^2}\right), \tag{19}$$

where T is the number of words in q; K is the number of RBF kernels; μ_k and σ_k are the mean and variance of the k-th RBF kernel, respectively; and $\phi(M)$ is the obtained ranking features.

Table 1. Statistics of all Datasets

AOL	Training	Validation	Test	
# sessions	219,748	34,090	29,369	
# queries	566,967	88,021	76,159	
average # queries per session	2.58	2.58	2.59	
# candidates per query	5	5	50	
average query length	2.86	2.85	2.9	
average document length	7.27	7.29	7.08	
average # clicks per query	1.08	1.08	1.11	
Tiangong-ST	**Training**	**Validation**	**Test (Click)**	**Test (Relevance)**
# sessions	143,155	2,000	2,000	2,000
# queries	344,806	5,026	4,420	2,000
average # queries per session	2.41	2.51	2.21	1.00
# candidates per query	10	10	10	10
average query length	2.89	1.83	3.26	3.92
average document length	8.25	6.99	8.76	10.11
average # clicks per query	0.94	0.53	0.78	6.48

In total, we use eight Conv-KNRM components with different parameters to calculate the matching scores of all combinations. All eight scores are combined using an MLP to get the overall ranking score:

$$P(d|\mathcal{S},q) = \Phi\Big(P\left(d,q\right), P\left(d^{+},q\right), P\left(d,q^{+}\right), P\left(d^{+},q^{+}\right), \\ P\left(d,q^{s}\right), P\left(d^{+},q^{s}\right), P\left(d,q^{s+}\right), P\left(d^{+},q^{s+}\right)\Big), \tag{20}$$

where $\Phi(\cdot)$ is an MLP with $\tanh(\cdot)$ as the activation function, $P(d|\mathcal{S},q)$ is the overall ranking score for d with respect to the current query q and the session context $\mathcal{S}$.

3.6 Model Learning

To train our model, we apply a standard pairwise learning-to-rank (LTR) algorithm. For the usage of pairwise loss, we craft pairwise training documents on the search log in the data preprocessing stage. The positive samples are the clicked documents and the negative samples are the skipped documents. The ranking loss for q is computed as follows:

$$\mathcal{L}_R(q) = \sum_{(d^p,d^n)\in\mathcal{D}_q^{p,n}} \max\left(0, 1 - P(d^p|\mathcal{S},q) + P(d^n|\mathcal{S},q)\right), \tag{21}$$

where $\mathcal{D}_q^{p,n}$ is the crafted paired document set for q, d^p is the clicked document, and d^n is the skipped document. By this loss function, we train our model to re-rank the positive samples higher than the corresponding negative samples.

4 EXPERIMENTAL SETTINGS

4.1 Datasets and Evaluation Metrics

We evaluate our model on two public search logs. The statistics of these two datasets are shown in Table 1.

4.1.1 AOL Search Log. We use the one provided by Ahmad et al. [2]. Each query in the training and validation datasets has 5 candidate documents. For each query in the testing dataset, there are

50 candidate documents retrieved by BM25 [25]. As suggested in [2, 10, 16], we use only the title as the content for each document.

4.1.2 Tiangong-ST Search Log. The Tiangong-ST Search Log [5] is collected from Sogou, a Chinese commercial search engine. It consists of 18-day user-issued queries, their top 10 results, and the click information made on the results. Among all sessions, there are 2,000 sessions whose last query has human relevance labels. We use these sessions as the test set. For the rest of the sessions, we use the last 2,000 sessions as the validation set and the remaining sessions as the training set. In the training and validation sets, each document has a label that shows whether it was clicked by the user. **For the testing dataset, because only the last query in each session has an annotated relevance score, we construct two testing sets based on the original testing data:**

(1) Tiangong-ST-Click: In this testing set, we do not use the last query of each session. Each document has a label that shows whether it is clicked by a human.

(2) Tiangong-ST-Relevance: In this testing set, only the last query in a session that has a manual annotation is used. Each of these queries is manually annotated with a five-graded relevance score. More details about the relevance score can be found in [5]. When counting the statistics of this dataset, we take the documents with the relevance score higher than 1 as the clicked documents. Therefore, the average number of clicked documents for each query of this testing set seems to be larger than those of the training and validation sets in Table 1. Note that when a context-aware model is being evaluated on this dataset, the previous queries of each session (i.e., queries in another testing set) are still used as session context.

Note that following [2, 10, 16], we use only the title as the content for each document.

4.1.3 Evaluation Metrics. To evaluate our model, we use Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), and Normalized Discounted Cumulative Gain (NDCG) [30] as metrics. NDCG includes NDCG@1, NDCG@3, NDCG@5, and NDCG@10, where NDCG@k indicates NDCG at position k. For Tiangong-ST-Relevance, the relevance label has five levels, 0 to 4, which is not suitable for using MAP or MRR. Hence, we considered documents with labels larger than one to be relevant for computing MAP and MRR. All evaluation results are calculated by TREC's evaluation tool (trec_eval) [12].

4.2 Baselines

We use two kinds of baseline models for comparison to prove our model's effectiveness.

(1) Ad-hoc Models: These models do not utilize any information from search history, that is, they only use q to re-rank d.

- **BM25** [25] is a classical retrieval algorithm that ranks d based on the terms of q appearing in d.
- **ARC-I** [15] represents q and d with CNNs. Then, it uses an MLP to calculate the ranking score.
- **ARC-II** [15] is interaction based. A 2D-CNN is utilized on the interaction matrix of the current query and the candidate document.
- **KNRM** [32] extracts the features of interaction between q and d on the word level. Kernel pooling is used to provide soft match signals for ranking.
- **Conv-KNRM** [8] is an extension of KNRM, which models n-gram soft matches with CNNs.
- **Duet** [23] integrates both representation-based features and interaction-based features to rank d for ad-hoc search.

(2) Context-aware Models: These models utilize both $\mathcal{S}$ and q to calculate the ranking score of d.

- **M-NSRF** [1] predicts the user's click in the current session and the user's next query jointly. It models queries, documents, and session history information into continuous vectors. Then, it computes the ranking score based on these representations.
- **CARS** [2] solves query suggestion tasks and document-ranking tasks simultaneously. In contrast to previous works [1], it adds attention mechanism into encoding queries and documents. It also encodes the sequence of clicked documents in the current session into a context representation.
- **HBA-Transformers** [24] concatenates all search behaviors of the session into a sequence and puts it into BERT to get word-level interaction-based contextual representations. Then, HBA uses a hierarchical behavior attention module that consists of behavior embedding and position embedding to further enhance the representations. Finally, the representation of the token [CLS] is used to calculate the ranking score. This is the state-of-the-art method.[2]

4.3 Experiment Setup

We re-implemented all baselines. All baselines used the same pairwise ranking loss as RICR. For ARC-I, ARC-II, Duet, M-NSRF, and CARS, we selected their hyperparameters following [2]. As for KNRM, Conv-KNRM, and HBA, we selected their hyper-parameters following their original papers [8, 24, 32]. As suggested in [24], we fine-tuned the BERT encoder of HBA during training. All models (including RICR) were trained for 10 epochs on the AOL dataset and 5 epochs on the Tiangong-ST dataset. Note that we did not use an early-stopping strategy and selected the best result among all epochs as the final result.

To finalize the parameters of our model, we tried multiple sets of experiments on the validation set. The decided parameters are as follows. We set the pretrained 100-dimensional word2vec model [22] as the initial word embedding, the dimension of GRU's hidden unit as 256, the maximum length of a session as 7, the maximum length of a query as 7, the maximum document length as 15, the dropout rate as 0.1, the learning rate as 0.001, and the training batch size as 512. As for the Conv-KNRM components, their kernel pooling layers all have 21 kernels with different parameters, and one is used for exact matching ($\mu = 1.0$ and $\sigma = 0.001$). Other hyperparameters of Conv-KNRMs are set as suggested in [8]. We use AdamW [20] as the optimizer, and train our model on a 12G TITAN V GPU. The code is released on GitHub at https://github.com/haon-chen/RICR.

5 RESULTS AND ANALYSIS

5.1 Overall Performance

The overall results on all datasets are presented in Table 2. We find that all neural models outperform the traditional model BM25 significantly, which indicates that the task we study is difficult and meaningful. It can be clearly observed that RICR outperforms all baseline models on all datasets in terms of all metrics. This indicates that our model successfully takes full advantage of representation and interaction to improve re-ranking performance. For example, our model has achieved about 13.13% improvement on NDCG@1 compared with the state-of-the-art baseline HBA-Transformers on the Tiangong-ST-Click set and about 3.29% improvement in the terms of NDCG@1 on the AOL set. Further, we determined the following.

[2]There are some slight differences between our re-implemented results and those of the original paper on HBA. This is because of different batch size settings. We use a considerably smaller batch size than HBA does (32 vs. 512) due to the limitation of computing resources.

Table 2. Overall Results on All Three Testing Datasets

Dataset	Model	MAP	MRR	NDCG@1	NDCG@3	NDCG@5	NDCG@10
AOL	BM25	0.2200	0.2271	0.1195	0.1862	0.2136	0.2481
	ARC-I	0.3559	0.3661	0.2032	0.3308	0.3773	0.4242
	ARC-II$_\diamond$	0.4114	0.4217	0.2507	0.3945	0.4396	0.4837
	KNRM$_\diamond$	0.3861	0.3954	0.2268	0.3640	0.4115	0.4578
	Conv-KNRM$_\diamond$	0.4282	0.4380	0.2634	0.4156	0.4598	0.5007
	Duet$_\diamond$	0.4268	0.4364	0.2616	0.4134	0.4580	0.5002
	M-NSRF*	0.4308	0.4479	0.2951	0.4297	0.4832	0.5214
	CARS*	0.4363	0.4457	0.3005	0.4313	0.4801	0.5309
	HBA$^\star_\diamond$	0.5273	0.5382	0.3770	0.5254	0.5597	0.5916
	RICR$^\star_\diamond$	**0.5338**†	**0.5450**†	**0.3894**†	**0.5267**†	**0.5648**†	**0.5971**†
	Improv.	1.23%	1.26%	3.29%	0.25%	0.91%	0.93%
Tiangong-ST-Click	BM25	0.2963	0.3073	0.1181	0.2085	0.2910	0.4649
	ARC-I	0.6657	0.6899	0.5368	0.6474	0.7015	0.7557
	ARC-II$_\diamond$	0.6684	0.6995	0.5451	0.6470	0.7103	0.7602
	KNRM$_\diamond$	0.6733	0.6952	0.5356	0.6629	0.7149	0.7615
	Conv-KNRM$_\diamond$	0.6925	0.7138	0.5575	0.6769	0.7248	0.7761
	Duet$_\diamond$	0.6952	0.7145	0.5594	0.6838	0.7335	0.7776
	M-NSRF*	0.6849	0.7111	0.5649	0.6741	0.7182	0.7746
	CARS*	0.6923	0.7128	0.5682	0.6829	0.7297	0.7774
	HBA$^\star_\diamond$	0.6961	0.7185	0.5658	0.6855	0.7366	0.7790
	RICR$^\star_\diamond$	**0.7472**†	**0.7697**†	**0.6401**†	**0.7450**†	**0.7822**†	**0.8174**†
	Improv.	7.34%	7.13%	13.13%	8.68%	6.19%	4.93%
Tiangong-ST-Relevance	BM25	0.7837	0.8225	0.6029	0.6646	0.7072	0.8541
	ARC-I	0.7901	0.8580	0.7271	0.7263	0.7451	0.8781
	ARC-II$_\diamond$	0.7977	0.8621	0.7390	0.7463	0.7588	0.8842
	KNRM$_\diamond$	0.8139	0.8915	0.7429	0.7489	0.7561	0.8896
	Conv-KNRM$_\diamond$	0.8132	0.8921	0.7498	0.7474	0.7593	0.8889
	Duet$_\diamond$	0.8025	0.8762	0.7389	0.7357	0.7572	0.8844
	M-NSRF*	0.8077	0.8811	0.7154	0.7329	0.7503	0.8805
	CARS*	0.8112	0.8850	0.7389	0.7428	0.7492	0.8846
	HBA$^\star_\diamond$	0.8142	0.8929	0.7598	0.7509	0.7617	0.8893
	RICR$^\star_\diamond$	**0.8147**†	**0.8937**†	**0.7670**†	**0.7636**†	**0.7740**†	**0.8934**†
	Improv.	0.06%	0.09%	0.95%	1.69%	1.61%	0.46%

"HBA" is short for HBA-Transformers. "Improv." reflects improvements of RICR over HBA-Transformers. "$\star$" indicates this model is context-aware. "$\diamond$" indicates this model utilizes interaction-based features. "†" indicates our model outperforms all baselines significantly ($p < 0.05$ in two-tailed paired t-test). The best performance is in bold and the second-best performance is underlined.

(1) RICR performs better than all ad-hoc models, which indicates the importance of modeling session history. The ad-hoc models do not take the session context into consideration, while RICR does. Therefore, the higher performance of RICR compared with ad-hoc models suggests that modeling session context is important. In addition, RICR obtains the best performance among all context-aware models, which proves its effectiveness for modeling session context. Intriguingly, we find that most existing context-aware models perform worse than the interaction-based ad-hoc model Conv-KNRM on the Tiangong-ST set. The reason may be that these models fail to capture fine-grained interactions on the word level; thus, they are unable to take full advantage of historical information.

(2) Interaction-based methods generally outperform representation-based ones. We find that even without the usage of session context, the interaction-based model Conv-KNRM still performs better than representation-based context-aware models on the Tiangong-ST-Click

Table 3. Performance of Ablated Models on All Datasets

Dataset	Metric	w/o HE		w/o DE		w/o BE		w/o SQS		RICR (Full)
AOL	MAP	0.4454	−16.56%	0.5087	−4.93%	0.5271	−1.26%	0.5287	−0.96%	**0.5338**
	MRR	0.4552	−16.48%	0.5192	−4.97%	0.5381	−1.27%	0.5391	−1.08%	**0.5450**
	NDCG@1	0.2796	−28.20%	0.3566	−9.51%	0.3803	−2.34%	0.3813	−2.08%	**0.3894**
	NDCG@3	0.4355	−17.32%	0.5014	−5.04%	0.5201	−1.25%	0.5219	−0.91%	**0.5267**
	NDCG@5	0.4779	−15.39%	0.5399	−4.61%	0.5577	−1.26%	0.5593	−0.97%	**0.5648**
	NDCG@10	0.5190	−13.08%	0.5750	−3.84%	0.5914	−0.95%	0.5929	−0.70%	**0.5971**
Tiangong-ST-Click	MAP	0.7216	−3.43%	0.7402	−0.94%	0.7329	−1.91%	0.7409	−0.84%	**0.7472**
	MRR	0.7425	−3.53%	0.7622	−0.98%	0.7513	−2.39%	0.7631	−0.86%	**0.7697**
	NDCG@1	0.5998	−11.00%	0.6281	−1.85%	0.6099	−4.72%	0.6284	−1.83%	**0.6401**
	NDCG@3	0.7180	−3.62%	0.7359	−1.22%	0.7297	−2.05%	0.7404	−0.62%	**0.7450**
	NDCG@5	0.7612	−3.96%	0.7752	−0.90%	0.7699	−1.57%	0.7771	−0.65%	**0.7822**
	NDCG@10	0.7978	−3.62%	0.8121	−0.65%	0.8060	−1.39%	0.8126	−0.59%	**0.8174**
Tiangong-ST-Relevance	MAP	0.8127	−0.25%	0.8133	−0.17%	0.8139	−0.10%	0.8144	−0.03%	**0.8147**
	MRR	0.8890	−0.53%	0.8881	−0.63%	0.8927	−0.11%	0.8912	−0.28%	**0.8937**
	NDCG@1	0.7605	−0.85%	0.7612	−0.76%	0.7643	−0.35%	0.7647	−0.30%	**0.7670**
	NDCG@3	0.7511	−1.64%	0.7537	−1.30%	0.7546	−1.18%	0.7524	−1.47%	**0.7636**
	NDCG@5	0.7705	−0.45%	0.7693	−0.61%	0.7673	−0.87%	0.7692	−0.62%	**0.7740**
	NDCG@10	0.8901	−0.37%	0.8902	−0.36%	0.8909	−0.28%	0.8912	−0.25%	**0.8934**

set. In addition, the interaction-based context-aware model HBA-Transformers outperforms all other baselines on all datasets. These can indicate the effectiveness of mining the information of interactions between queries and documents. Compared with all interaction-based models, RICR achieves better performance, which proves that it manages to take full advantage of both representation and interaction to improve re-ranking performance.

5.2 Ablation Analysis

To prove the effectiveness of our model, we design several variants of RICR to evaluate the importance of the components. We conduct ablation experiments on all three datasets as follows.

- **RICR w/o HE.** We remove the queries and documents obtained by history enhancing (HE, introduced in Section 3.4): d^+, q^+, and q^{s+}. In other words, we use only these two matching scores: $P(d, q^s)$, $P(d, q)$ to re-rank d.
- **RICR w/o DE.** We remove the history-enhanced candidate documents (DE, introduced in Section 3.4) and the corresponding four matching scores: $P(d^+, q^+)$, $P(d^+, q)$, $P(d^+, q^{s+})$, and $P(d^+, q^s)$.
- **RICR w/o BE.** We remove the behavior-encoding sub-module (BE, introduced in Section 3.3), which consists of the term-level query-aware attention mechanism and the inner-attention mechanism. Instead, we simply encode each historical search behavior by averaging the embedding vectors of its words.
- **RICR w/o SQS.** We abandon the supplemental query selection (SQS, introduced in Section 3.4) and the corresponding four matching scores: $P(d, q^s)$, $P(d^+, q^s)$, $P(d, q^{s+})$, and $P(d^+, q^{s+})$.

The results of ablation experiments are shown in Table 3, from which we can see that all three ablated models underperform the full model. We can further obtain the following conclusions.

(1) Utilizing contextual information of session history is necessary. In the information-enhancing module, RICR uses the encoded session history as the initial state of three GRUs to learn the enhanced word representations of q, q^s, and d, respectively. This adds the information of $\mathcal{S}$ into the latter matching module. Our model's performance decreases after discarding the matching of enhanced queries and documents. For example, it makes our model decrease 13.08%

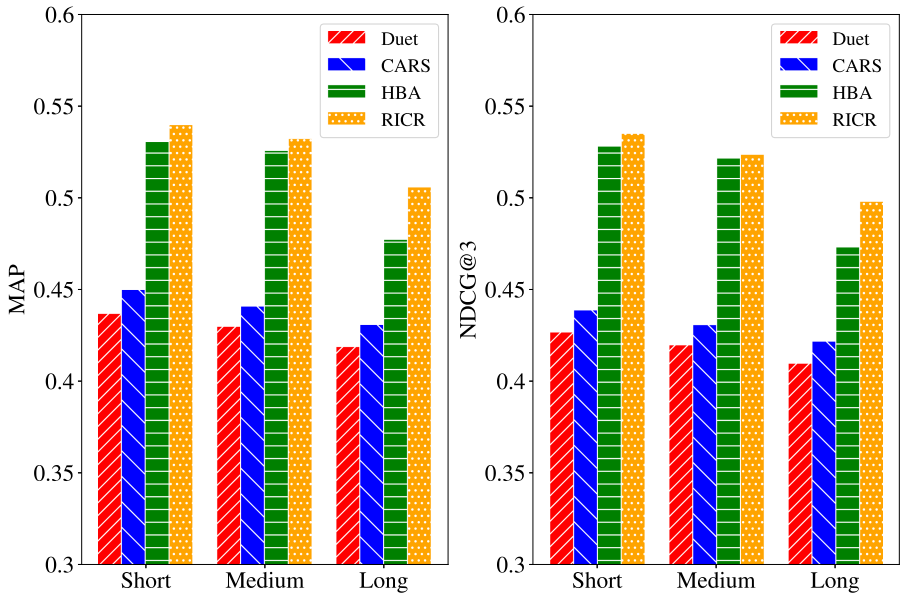

Fig. 3. Performance on sessions with different lengths.

in terms of NDCG@10 on the AOL dataset. The significant decrease shows that using contextual information (i.e., $\mathcal{S}$) is necessary for inferring a user's intent. It is consistent with our analysis in Section 5.1.

(2) Enhancing the candidate document with session history is effective. In contrast to most works that represent d independently from the session context [1, 2, 11], RICR enhances the candidate document with session history. The performance of RICR decreases after discarding the corresponding matching scores of the enhanced document. For example, it makes our model decrease 9.51% in terms of NDCG@1 on the AOL dataset. This proves the effectiveness of enhancing the candidate document.

(3) It is important to utilize the information of the current query to encode historical behaviors. RICR uses the Behavior Encoding (BE) sub-module to encode every behavior in $\mathcal{S}$. It consists of the term-level query-aware attention mechanism and the inner-attention mechanism. Term-level query-aware attention can encode each behavior in $\mathcal{S}$ with the word-level interactions between the behavior and q. The inner-attention mechanism can identify the different levels of importance of words in q, and use this information to encode the representations of behaviors in $\mathcal{S}$. From Table 3, we find that removing BE causes a decrease in our model's performance. For example, NDCG@1 decreases 2.34% on the AOL dataset. This confirms the effectiveness of our BE sub-module. We conduct more analysis of BE in Section 5.4.

(4) The supplemental query that we select can make our model more robust. In the information-enhancing module, we select a supplemental query (q^s) based on both session context and q. It can help our model to deal with the user issuing a small set of keywords as q while the search intent is more complex than this. After we abandon the supplemental query selection (SQS) and its corresponding four matching scores, the performance of our model decreases. For example, removing SQS causes a 1.83% decrease in terms of NDCG@1 on the Tiangong-ST-Click dataset. This proves the effectiveness of our supplemental selection module. More analysis is provided in Section 5.4.

Table 4. An Example Search Session with Three Queries

q	q_1	q_2
popular	**a song** for my son	groom and mother wedding **dance songs**
wedding	a **song** for my **son**	**groom** and mother **wedding** dance songs
songs	a **song** for **my** son	groom and mother wedding **dance songs**

q_1 and q_2 are the historical queries, q is the current query. Bold words of q_1 and q_2 are given higher weights with respect to a specific word of q by term-level query aware attention mechanism. Bold words of q are given higher weights by inner-attention mechanism.

5.3 Effect of Session Length

To study the impact of context information on sessions with different lengths, we split the test set of the AOL dataset into three bins:

(1) Short sessions (with 2 queries) — 66.5% of the test set;
(2) Medium sessions (with 3–4 queries) —27.24% of the test set;
(3) Long sessions (with 5+ queries) — 6.26% of the test set.

Note that following [2], we filter out sessions with only one query, that is, without context information.

We compare RICR with Duet, CARS, and HBA-Transformers on the AOL dataset and present the results on MAP and NDCG@3 in Figure 3. We can obtain the following conclusions from the experiment:

(1) RICR manages to take full advantage of representation and interaction to model session context. We clearly find that RICR outperforms all context-aware document ranking models on all three groups of sessions. This proves RICR's effectiveness in learning context information. RICR first utilizes the BE sub-module with a GRU to encode the session context into a latent vector. Then, it uses this latent representation to enhance the word-level interaction between the current query and the candidate document for scoring. Through this, RICR manages to model and utilize the session context.

(2) Modeling historical information is essential for improving ranking performance. It is evident that the ad-hoc ranking model Duet performs worse than all three context-aware document ranking models on all bins of sessions. This demonstrates the importance of modeling session context once again.

(3) RICR can handle long sessions better than HBA-Transformers. From Figure 3, we find that all models' performance on long sessions decreases drastically. However, RICR's performance is relatively stable compared with the BERT-based model HBA-Transformers. For example, HBA decreases 10.23% in terms of NDCG@3 on long sessions compared with that on medium sessions, whereas RICR only decreases 4.90%. This supports our claim that RICR can handle long sequences better than the SOTA model HBA-Transformers.

5.4 Case Study of Behavior Encoding and Supplemental Query Selection

To further verify the effectiveness and interpretability of the BE sub-module, we illustrate a qualitative example from the AOL search log in Table 4. The original session has four queries: "a song for my son" (q_1), "groom and mother wedding dance songs" (q_2), "popular groom and mother wedding dance songs" and "popular wedding songs" (q). We take the last one as the query being issued, that is, q, and abandon the third because it is almost identical to q_2. In Table 4, we highlight two words for each query that gain higher attention weights. For example, the words "wedding" and "songs" of q are in bold because they are given higher weights by the inner-attention mechanism,

Table 5. An Example Query "Popular Wedding Songs" with Its Candidate Set

Index	Candidate query
1	listen to most popular wedding songs
2	wedding songs
3	✓**family wedding songs**
4	wedding songs example
5	wedding processional songs
6	free wedding songs
7	popular love songs
8	money dance wedding songs
9	new wedding tradition songs
10	popular wedding songs

The selected supplemental query is in bold and marked with a "✓".

which indicates that they are more informative than other words in q. For q_1 in the second row, the words "song" and "son" are in bold because they are given higher weights with respect to the word "wedding" of q by the term-level query-aware attention mechanism. Like these two examples, most of the given attention weights are reasonable and interpretable. This proves the value of our BE sub-module.

We also take a look at how our model selects the supplemental query for q. The candidates and the selected result are presented in Table 5. It can be observed that the selected query "family wedding songs" is clearly consistent with the user's current search intent, which is to find a song to dance with at the son's wedding, whereas q is too simple to infer this intent, which indicates that the supplemental query can make our model more robust. We can also infer that our model selects the supplemental query based on both the historical information and q, which is consistent with our claim in Section 3.4.

5.5 Quality of Selected Supplemental Queries

To further study the quality of the supplemental query selected by RICR, we take a look at some statistics of the selected supplemental queries of the AOL dataset, presented in Table 6. The first row of this table is the rank of the supplemental rate of each candidate, which is computed by Equation (13). The second row is the proportion of the queries being selected. As stated in Section 3.4, to ensure that if the original query matches the user's current search intent or all candidates are worse than the original query, we also add it into the candidate set and place it at the head of the candidate set, that is, rank 0. We find that about 35% of supplemental queries end up being the same as the original query, that is, about 65% of the supplemental queries have the information that the original query does not have. We also use a sentence-transformer model[3] to compute the semantic similarity between the supplemental queries and the original. The average similarity is **0.6452**. This indicates that RICR manages to select a supplemental query that can enhance the information obtained by the original query.

5.6 Cost Comparison between RICR and HBA-Transformers

The main goal of our model is to capture the information of word-level interactions along with contextual information while reducing the cost. To investigate how our model performs with

[3]https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2.

Table 6. Statistics of the Selected Supplemental Queries of AOL Dataset

Supplemental Rate Rank	**0**	1	2	3	4	5	6	7	8	9
Proportion	**0.3544**	0.0588	0.0653	0.0584	0.0591	0.1020	0.0859	0.0762	0.0695	0.0704

The original query is in bold.

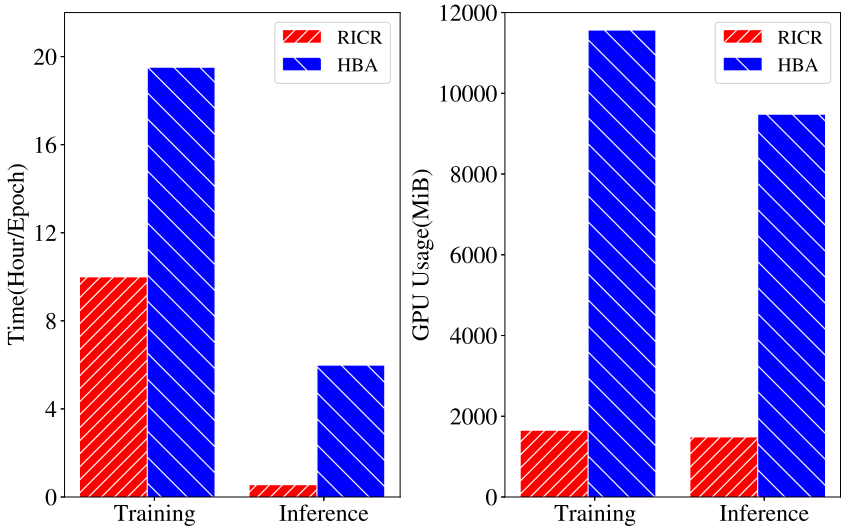

Fig. 4. The training and inference cost of RICR and HBA. The batch size for training is 16 and is 100 for inference. The experiment is conducted on a 12G TITAN V GPU.

respect to cost reduction, we compare the cost of our model and HBA-Transformers in two ways: an analysis of the number of parameters and an experiment on training and inference cost.

5.6.1 The Number of Parameters. HBA-Transformers has two main components: a BERT Encoder (BERT-Base-Uncased) and a Hierarchical Behavior Attention module. The BERT-Base model has a position-embedding layer and 12 encoders. Each encoder has two main parts: a Multi-Head Attention module and a Feed Forward layer. Each part in each encoder has a residual connection around it and is followed by a layer-normalization step. Most parameters of HBA-Transformers fall into the BERT Encoder. The Hierarchical Behavior Attention module has a two-level attention mechanism, which also requires training.

For RICR, most parameters fall into four parts: a Word-Embedding layer, a BE sub-module, four GRUs, and eight Conv-KNRM components. The BE sub-module, which is used to encode historical behaviors, consists of a term-level query-aware attention and an inner attention. Among the four GRUs, one is used to encode the sequential information of $\mathcal{S}$; the other three are used for enhancing word-level representations of q, d, and q^s. Conv-KNRM is used to capture the fine-grained word-level interactions between queries and documents.

The main difference that allows our model to reduce the cost is that in the history-encoding stage, instead of making every two search behaviors interact with each other, RICR obtains the interaction information between the current query and each historical behavior only. This can be observed in the difference between our BE sub-module and HBA's BERT Encoder.

We count the number of the parameters that require training for both models. The results are that HBA-Transformers has 117,044,738 parameters and RICR has 23,043,270 parameters. It is

clear that **our model cuts about 80% of the parameters compared with HBA-Transformers**, which achieves our goal to reduce the calculation cost.

5.6.2 Training and Inference Cost. To obtain a more straightforward view of the calculation cost of RICR and HBA, we record their training and inference costs on the AOL dataset on a 12G TITAN V GPU. We record the occupation of GPU memory and the runtime as the cost. We record the total runtime on all queries as the training and inference time. The IO, tokenization, and word embedding have all been considered. The results are presented in Figure 4. For the training stage, it is evident that our model saves over a half training time period compared with HBA and, at the same time, takes only about 14.4% of HBA's GPU memory usage. This proves our model's offline training efficiency. Furthermore, for the inference stage, our model uses only 15.8% of HBA's GPU memory and spends 9.7% of HBA's inference time. This confirms RICR's online inference efficiency.

We have compared the training cost, inference cost, and the number of trainable parameters of RICR and HBA. From these analyses and experiments, we conclude that our model manages to reduce the calculation cost considerably.

6 CONCLUSION AND FUTURE WORK

In this work, we propose a context-aware document-ranking model RICR that leverages representation and interaction. We first use word-level attention mechanisms and RNN to encode the session history. We then use this encoded history to enhance the representations of the current query and the corresponding candidate document. To make our model more robust, we also select a supplemental query for the current query. Finally, we use several matching components to obtain the fine-grained information of word-level interactions. Our model manages to incorporate session context into the word-level interactions while reducing calculation cost. Experimental results prove the effectiveness and the efficiency of our model.

For future work, we are interested in (1) developing more advanced approaches instead of just averaging the word embedding to aggregate clicked documents; (2) studying more effective approaches to calculate the similarity between the current query and the candidate queries to select the supplemental query; (3) exploring how our proposed model performs on long document contents instead of only titles; and (4) extending our work to different scenarios, such as personalized search.

REFERENCES

[1] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2018. Multi-task learning for document ranking and query suggestion. In *6th International Conference on Learning Representations (ICLR'18), Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

[2] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2019. Context attentive document ranking and query suggestion. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19), Paris, France, July 21–25, 2019*, Benjamin Piwowarski, Max Chevalier, Éric Gaussier, Yoelle Maarek, Jian-Yun Nie, and Falk Scholer (Eds.). ACM, 385–394. https://doi.org/10.1145/3331184.3331246

[3] Paul N. Bennett, Ryen W. White, Wei Chu, Susan T. Dumais, Peter Bailey, Fedor Borisyuk, and Xiaoyuan Cui. 2012. Modeling the impact of short- and long-term behavior on search personalization. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'12)*, ACM, 185–194. https://doi.org/10.1145/2348283.2348312

[4] Ben Carterette, Paul Clough, Mark Hall, Evangelos Kanoulas, and Mark Sanderson. 2016. Evaluating retrieval over sessions: The TREC session track 2011–2014. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'16)*, ACM, 685–688. https://doi.org/10.1145/2911451.2914675

[5] 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 Conference on Information and Knowledge Management (CIKM'19), Beijing, China, November 3–7, 2019*, Wenwu Zhu, Dacheng Tao, Xueqi Cheng, Peng

Cui, Elke A. Rundensteiner, David Carmel, Qi He, and Jeffrey Xu Yu (Eds.). ACM, 2485–2488. https://doi.org/10.1145/3357384.3358158

[6] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14), October 25–29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, Alessandro Moschitti, Bo Pang, and Walter Daelemans (Eds.). ACL, 1724–1734. https://doi.org/10.3115/v1/d14-1179

[7] Steve Cronen-Townsend, W. Bruce Croft, et al. 2002. Quantifying query ambiguity. In *Proceedings of HLT*, Vol. 2. Citeseer, 94–98.

[8] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional neural networks for soft-matching N-grams in ad-hoc search. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM'18), Marina Del Rey, CA, February 5–9, 2018*, Yi Chang, Chengxiang Zhai, Yan Liu, and Yoelle Maarek (Eds.). ACM, 126–134. https://doi.org/10.1145/3159652.3159659

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'19), Minneapolis, MN, June 2–7, 2019, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, 4171–4186. https://doi.org/10.18653/v1/n19-1423

[10] Jianfeng Gao, Xiaodong He, and Jian-Yun Nie. 2010. Clickthrough-based translation models for web search: From word models to phrase models. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM'10, Toronto, Ontario, Canada, October 26–30, 2010*, Jimmy Huang, Nick Koudas, Gareth J. F. Jones, Xindong Wu, Kevyn Collins-Thompson, and Aijun An (Eds.). ACM, 1139–1148. https://doi.org/10.1145/1871437.1871582

[11] Songwei Ge, Zhicheng Dou, Zhengbao Jiang, Jian-Yun Nie, and Ji-Rong Wen. 2018. Personalizing search results using hierarchical RNN with query-aware attention. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM'18), Torino, Italy, October 22–26, 2018*, Alfredo Cuzzocrea, James Allan, Norman W. Paton, Divesh Srivastava, Rakesh Agrawal, Andrei Z. Broder, Mohammed J. Zaki, K. Selçuk Candan, Alexandros Labrinidis, Assaf Schuster, and Haixun Wang (Eds.). ACM, 347–356. https://doi.org/10.1145/3269206.3271728

[12] Christophe Van Gysel and Maarten de Rijke. 2018. Pytrec_eval: An extremely fast Python interface to trec_eval. In *41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR'18), Ann Arbor, MI, July 08–12, 2018*, Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz (Eds.). ACM, 873–876. https://doi.org/10.1145/3209978.3210065

[13] Christophe Van Gysel, Evangelos Kanoulas, and Maarten de Rijke. 2016. Lexical query modeling in session search. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval (ICTIR'16), Newark, DE, September 12–16, 2016*, Ben Carterette, Hui Fang, Mounia Lalmas, and Jian-Yun Nie (Eds.). ACM, 69–72. https://doi.org/10.1145/2970398.2970422

[14] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[15] 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 27: Annual Conference on Neural Information Processing Systems 2014, December 8–13 2014, Montreal, Quebec, Canada*, Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger (Eds.). 2042–2050. https://proceedings.neurips.cc/paper/2014/hash/b9d487a30398d42ecff55c228ed5652b-Abstract.html.

[16] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *22nd ACM International Conference on Information and Knowledge Management (CIKM'13), San Francisco, CA, October 27 - November 1, 2013*, Qi He, Arun Iyengar, Wolfgang Nejdl, Jian Pei, and Rajeev Rastogi (Eds.). ACM, 2333–2338. https://doi.org/10.1145/2505515.2505665

[17] Aaron Jaech, Hetunandan Kamisetty, Eric K. Ringger, and Charlie Clarke. 2017. Match-Tensor: A deep relevance model for search. *CoRR* abs/1701.07795 (2017). arXiv:1701.07795 http://arxiv.org/abs/1701.07795.

[18] Rosie Jones and Kristina Lisa Klinkner. 2008. Beyond the session timeout: Automatic hierarchical segmentation of search topics in query logs. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM'08), Napa Valley, California, October 26–30, 2008*, James G. Shanahan, Sihem Amer-Yahia, Ioana Manolescu, Yi Zhang, David A. Evans, Aleksander Kolcz, Key-Sun Choi, and Abdur Chowdhury (Eds.). ACM, 699–708. https://doi.org/10.1145/1458082.1458176

[19] Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning natural language inference using bidirectional LSTM model and inner-attention. *CoRR* abs/1605.09090 (2016). arXiv:1605.09090 http://arxiv.org/abs/1605.09090.

[20] Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations (ICLR'19), New Orleans, LA, May 6–9, 2019*. OpenReview.net. https://openreview.net/forum?id=Bkg6RiCqY7.

[21] Shuqi Lu, Zhicheng Dou, Jun Xu, Jian Yun Nie, and Ji Rong Wen. 2019. PSGAN: A minimax game for personalized search with limited and noisy click data. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*, ACM, 555–564. https://doi.org/10.1145/3331184.3331218

[22] Tomas Mikolov, Kai Chen, G. S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR* 2013 (01 2013).

[23] 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 (WWW'17), Perth, Australia, April 3–7, 2017*, Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy Gabrilovich (Eds.). ACM, 1291–1299. https://doi.org/10.1145/3038912.3052579

[24] Chen Qu, Chenyan Xiong, Yizhe Zhang, Corby Rosset, W. Bruce Croft, and Paul Bennett. 2020. Contextual re-ranking with behavior aware transformers. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20), Virtual Event, China, July 25–30, 2020*, Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 1589–1592. https://doi.org/10.1145/3397271.3401276

[25] Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval* 3, 4 (2009), 333–389. https://doi.org/10.1561/1500000019

[26] Xuehua Shen, Bin Tan, and Chengxiang Zhai. 2005. Context-sensitive information retrieval using implicit feedback. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'05)*, ACM, 43–50. https://doi.org/10.1145/1076034.1076045

[27] Craig Silverstein, Monika Rauch Henzinger, Hannes Marais, and Michael Moricz. 1999. Analysis of a very large web search engine query log. *SIGIR Forum* 33, 1 (1999), 6–12. https://doi.org/10.1145/331403.331405

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017, Long Beach, CA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[29] Hongning Wang, Yang Song, Ming Wei Chang, Xiaodong He, Ryen W. White, and Wei Chu. 2013. Learning to extract cross-session search tasks. In *Proceedings of the 22nd International World Wide Web Conference (WWW'13)*, International World Wide Web Conferences Steering Committee / ACM, 1353–1363. https://doi.org/10.1145/2488388.2488507

[30] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. 2013. A theoretical analysis of NDCG type ranking measures. In *Conference on Learning Theory*. PMLR, 25–54.

[31] Ryen W. White, Wei Chu, Ahmed Hassan, Xiaodong He, Yang Song, and Hongning Wang. 2013. Enhancing personalized search by mining and modeling task behavior. In *Proceedings of the 22nd International World Wide Web Conference (WWW'13)*, International World Wide Web Conferences Steering Committee / ACM, 1411–1420. https://doi.org/10.1145/2488388.2488511

[32] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7–11, 2017*, Noriko Kando, Tetsuya Sakai, Hideo Joho, Hang Li, Arjen P. de Vries, and Ryen W. White (Eds.). ACM, 55–64. https://doi.org/10.1145/3077136.3080809

[33] Yujia Zhou, Zhicheng Dou, and Ji-Rong Wen. 2020. Encoding history with context-aware representation learning for personalized search. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20), Virtual Event, China, July 25–30, 2020*, Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 1111–1120. https://doi.org/10.1145/3397271.3401175

Received 10 September 2021; revised 22 February 2022; accepted 30 March 2022