A Hybrid Framework for Session Context Modeling

JIA CHEN, Tsinghua University, China
JIAXIN MAO[★], Renmin University of China, China
YIQUN LIU, Tsinghua University, China
ZIYI YE, Tsinghua University, China
WEIZHI MA, Tsinghua University, China
CHAO WANG, 6ESTATES PTE LTD, Singapore
MIN ZHANG, Tsinghua University, China
SHAOPING MA, Tsinghua University, China

Understanding user intent is essential for various retrieval tasks. By leveraging contextual information within sessions, e.g., query history and user click behaviors, search systems can capture user intent more accurately thus perform better. However, most existing systems only consider intra-session contexts and may suffer from the problem of lacking contextual information because short search sessions account for a large proportion in practical scenarios. We believe that in these scenarios, considering more contexts, e.g., cross-session dependencies, may help alleviate the problem and contribute to better performance. Therefore, we propose a novel Hybrid framework for Session Context Modeling (HSCM), which realizes session-level multi-task learning based on the self-attention mechanism. To alleviate the problem of lacking contextual information within current sessions, HSCM exploits the cross-session contexts by sampling user interactions under similar search intents in the historical sessions and further aggregating them into the local contexts. Besides, application of the self-attention mechanism rather than RNN-based frameworks in modeling session-level sequences also helps 1) better capture interactions within sessions, 2) represent the session contexts in parallelization. Experimental results on two practical search datasets show that HSCM not only outperforms strong baseline solutions such as HiNT, CARS, and BERTserini in document ranking but also performs significantly better than most existing query suggestion methods. According to the results in an additional experiment, we have also found that HSCM is superior to most ranking models in click prediction.

CCS Concepts: • **Information systems → Users and interactive retrieval**; **Retrieval models and ranking**.

Additional Key Words and Phrases: Document Ranking, Query Suggestion, Self-attention Mechanism

[★]Corresponding author

Authors' addresses: Jia Chen, Tsinghua University, 30 Shuangqing Rd, Haidian District, Beijing, China, chenjia0831@gmail.com; Jiaxin Mao[★], Renmin University of China, 59 Zhongguancun Street, Haidian District, Beijing, China, maojiaxin@gmail.com; Yiqun Liu, Tsinghua University, 30 Shuangqing Rd, Haidian District, Beijing, China, yiqunliu@tsinghua.edu.cn; Ziyi Ye, Tsinghua University, 30 Shuangqing Rd, Haidian District, Beijing, China, yeziyi1998@gmail.com; Weizhi Ma, Tsinghua University, 30 Shuangqing Rd, Haidian District, Beijing, China, mawz@tsinghua.edu.cn; Chao Wang, 6ESTATES PTE LTD, 79 AYER RAJAH CRESCENT #01-05, Singapore, chaowang0707@gmail.com; Min Zhang, Tsinghua University, 30 Shuangqing Rd, Haidian District, Beijing, China, z-m@tsinghua.edu.cn; Shaoping Ma, Tsinghua University, 30 Shuangqing Rd, Haidian District, Beijing, China, msp@tsinghua.edu.cn.

ACM Reference Format:
Jia Chen, Jiaxin Mao⋆, Yiqun Liu, Ziyi Ye, Weizhi Ma, Chao Wang, Min Zhang, and Shaoping Ma. 2021. A Hybrid Framework for Session Context Modeling. 1, 1 (May 2021), 34 pages. https://doi.org/10.1145/3448127

1 INTRODUCTION

Session-level user modeling has been an intriguing research topic in Information Retrieval (IR) for years. Despite the achievements that modern search engines have made, we still have a long way to go before fully understanding users' search intents. In complex search scenarios, people tend to submit a sequence of queries and interact with corresponding Search Engine Result Pages (SERPs) until their information needs are satisfied. Recently, numerous compelling session-level models [2, 12, 49] have shown that search contexts such as the query history and user click behaviors are effective for improving the system performance in various IR tasks. For example, Wu et al. [55] harness user browsing and click actions to formulate feedback memories, which enables their model to suggest diverse queries for the same search history under different user propensities. By representing previous query reformulations as distributed embeddings, Jiang et al. [29] propose an end-to-end network to infer users' next reformulation behaviors and thereby achieve great improvements on the query suggestion performance.

However, most existing models only exploit intra-session contexts for user modeling, which may cause some problems. Previous investigations have found that about 70%-80% of the real-world search sessions contain only two queries, not only for textual based retrieval [11] but also for image search [59]. Limited contextual information in these short sessions, e.g., deficient query history or sparse click signals, may raise problems for the system to learn user intents. On one hand, researchers usually tried to solve this condition by leveraging social relationships or behaviors from similar users in recommendation systems [52, 66]. In web search, however, it is hard to directly introduce user profile features due to user privacy. On the other hand, although many deep architectures learn high-dimension semantic features from training data, they model the cross-session dependencies (i.e., previous sessions of any user) implicitly hence can only partially solve the data-sparsity problem. To this end, we introduce a cross-session interaction aggregation module to explicitly augment user behaviors in other sessions guided by similar search intents for the current query. As it is non-trivial of protecting user privacy in Web search, here we anonymize all sessions and refer to the cross-session information as historical sessions by all users. We also design an algorithm to efficiently sample user interactions in these past sessions under similar information needs and further aggregate these behaviors into the local contexts.

Moreover, for lack of sequence transduction tools, existing work usually employ recurrent neural networks (RNNs) for multi-grained context modeling. Previous work has demonstrated that RNN-based frameworks are hard to parallelize [54], i.e., the representation of each unit in a sequence cannot be obtained simultaneously. This may cause two main problems: 1) It is hard for the machine to efficiently utilize GPU resources, 2) The framework may not capture some local interactions between two remote units within a sequence. To this end, some researchers propose pure attention-based models such as *Transformer* [54] and *BERT* [15] which model local interactions in natural language via the multi-head attention mechanism and have achieved outstanding performances in various NLP tasks. However, it is non-trivial to apply them in *interactive* IR tasks. Although it is convenient to obtain query/document embeddings via official pre-trained BERT interfaces and then use these embeddings for downstream training, session contexts can not be fully utilized due to the hard-coded pre-training process that only considers document-level parallelization (i.e., trained on Wikipedia text). Even if one integrates user interactions into the downstream RNN-based frameworks, the system can only realize partial parallelization by facilitating past session contexts [64]. Therefore, we build a pure

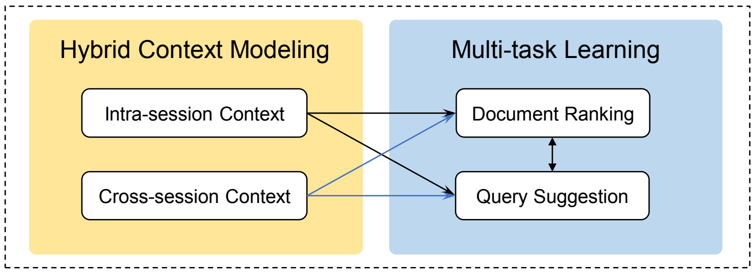


Fig. 1. An overview of HSCM. It exploits both the intra-session and cross-session contextual information to jointly optimize document ranking and query suggestion.

attention-based learning framework, which can better leverage session contexts by realizing more complete session-level modeling.

Regarding the behavior consistency within a series of the user's search actions, researchers also find that introducing multiple types of user behaviors may be helpful for analyzing one single retrieval task. As an example, a user may first examine or click on several results in the current query and then decide which query to be issued next. The query history, which indicates the evolution of user intent, may also help improve the document ranking. Ahmad et al. [2] present a context-aware neural ranking model that exploits users' on-task search activities to enhance retrieval performance on both document ranking and query suggestion tasks. Inspired by this mixing methodology, we also adopt the multi-task learning technique in our hybrid framework to better model search context information.

To shed light on the aforementioned issues, we propose a novel session-based model, namely HSCM (**H**ybrid framework for **S**ession **C**ontext **M**odeling). An overview of HSCM is presented in Figure 1. The main motivation is to utilize hybrid contextual information, i.e., intra-session and cross-session contexts, to further facilitate both document ranking and query suggestion in a multi-task learning manner. By sampling user behaviors under similar search intents in other sessions from historical user data, HSCM largely alleviates the dilemma that many search sessions lack sufficient contexts, e.g., query history and user click behaviors. To accelerate the sampling process on the whole training set, we apply a Maximum Inner Product Search (MIPS) algorithm for nearest-vectors-finding. Then through 1) the overall evaluation on two practical search datasets, 2) fine-grained investigations on session lengths and query frequencies, 3) the ablation study on several components, and 4) two case studies, we verify the effectiveness of HSCM on both document ranking and query suggestion tasks, the rationality of each component in it, its robustness on various session conditions, and its distinguished ability to deal with sessions with few contexts, respectively. Moreover, we have also shown the effectiveness of HSCM in click prediction via an additional experiment on TianGong-ST sessions.

To summarize, contributions of this work are listed as follows:

- We propose a novel Hybrid framework for Session Context Modeling (HSCM), which employs the self-attention mechanism to build a session-level multi-task learning architecture. Due to its pure attention-based nature, it can better exploit session contexts by realizing session-level parallelization.
- To our best knowledge, we are the first to explicitly incorporate a cross-session interaction aggregation module into session-based models. We build a click bipartite graph on the dataset before training and further add co-semantic dependencies between similar queries in an online manner. Cross-session user interactions are then

sampled based on this graph, which will be finally fuzed with the local session contexts. According to the experiment results, we find great potential of the cross-session module in dealing with sessions that lack contextual information as well as those long-tailed queries.

- A new content encoder is designed to transform all queries/documents from pre-trained word embeddings into fixed-size vectors in HSCM. Using content encoders rather than RNNs to embed contents manifests two advantages: 1) queries and documents are encoded at both text-level and session-level; 2) with a position encoding mechanism, it is easier to incorporate other dense information such as query co-occurrence frequencies into the sparse candidate embeddings.
- Experiments conducted on two public session datasets demonstrate that HSCM achieves state-of-the-art performances on both the document ranking and query suggestion tasks. Results in an additional experiment showed its great effectiveness in click prediction. Results of ablation studies have also indicated the effectiveness of the cross-session interaction aggregation module and other key components. Also, code and datasets are publicly available to facilitate the reproducibility of this work.

2 RELATED WORK

2.1 Session-based Retrieval

A broad spectrum of researches have shown the effectiveness of exploiting contextual information for user intent modeling [5, 7, 23, 47]. Numerous studies aim at utilizing query history and user interactive behaviors to optimize various IR tasks, e.g., session search, query suggestion, click prediction, etc. Xiang et al. [57] characterize search contexts into several types according to user reformulation behaviors and then develop different ranking principles for each type by using a learning-to-rank approach. To better infer session-level user intents, Cao et al. [6] model the evolution of user intents with a Hidden Markov Model (HMM). Some researchers focus on the query change over sessions to capture user intent shift [33, 53, 65]. Besides, some work also employs Reinforcement Learning (RL) algorithms to simulate user decisions or reformulation states [9, 21, 34, 36]. For instance, Guan et al. [21] aim at improving document ranking by utilizing query changes within sessions. Model-driven approaches achieve some success by reasoning user behaviors with pre-defined rules and dependencies. However, these manually designed methods can not generalize well once confronted with complex scenarios. Recently, with the emergence of many compelling deep learning-to-rank models, neural networks have also become a popular option for modeling session contexts [13, 14, 39]. For example, Chen et al. [12] propose a context-aware click model that leverages contextual information, i.e., query history and user clicks, to jointly optimize the document ranking and click prediction tasks. Besides the textual retrieval, contextual information has also been used for improving the performance of image search [59]. However, these models only consider intra-session contexts and perform worse in sessions with few contextual information, e.g., short sessions or sessions with only sparse user clicks. Therefore, we aim at leveraging cross-session contexts to alleviate the problem in this work.

2.2 Query Suggestion and Auto-completion

Whether users issue appropriate queries during web search has a great impact on the usability of search engines. Therefore, a large body of research aimed to improve users' search experience by providing services like query suggestions and auto-completion [6, 19, 49]. Early studies rely on the query association or similarities within search sessions for query suggestion, e.g., mining the association rules [18] or the co-occurrence relationships [25]. In addition, Cao et al. [6] attempted to learn associations between consecutive queries by Markov Models. These methods are simple and effective, but can also be largely restricted by the dataset, i.e., they may confront the data sparsity problem when a great proportion of the testing samples are unseen in the training set. To this end, researchers either leverage co-click information on the bipartite graph for query clustering [56] or use statistical features to learn user reformulating behaviors via machine learning [40, 46, 48]. By analyzing user reformulating behaviors, Jiang et al. [28] model syntactic reformulations based on session-level features including co-occurrence frequency, position in the session, and etc. As deep learning approaches flourished for years, Sordoni et al. [49] first adopt the hierarchical RNN-based framework to encode query history within a session for next query prediction. Dehghani et al. [14] incorporated a copy mechanism into the query decoder of their model under the observation that most query terms within a session are retained from the previously submitted queries. While these studies have been proved effective for query suggestion or auto-completion tasks, few of them have explicitly utilized the query dependencies between search sessions. Compared to some representation-based methods such as Reformulation Inference Network (RIN) [29] that implicitly encodes the cross-session dependencies into the vector space, we explicitly introduce a cross-session interaction aggregation module into our framework. Cross-session knowledge is integrated with local session contexts in a more straightforward manner to improve both session-based retrieval and query recommendation.

2.3 Attention Mechanism

The attention mechanism is first applied in neural machine translation (NMT), in which it assigns different weights for the words that may be concerned [3]. Due to its promising performance and considerable interpretability [20], the attention mechanism has been adapted to a wide range of tasks since then, e.g., text classification [62], recommender system [58, 63], query suggestion [14, 29], and so on. For lack of new transduction tools, these work mostly employ attention mechanisms with RNN-based architectures. However, there are limitations for RNN units: low parallelization and hard to perceive distant inputs. It is not until Vaswani et al. [54] propose *Transformer* that a new form of attention mechanism, i.e., multi-head self-attention with position encoding, can be adopted to replace traditional RNN units. Later, Devlin et al. [15] propose *BERT* by stacking a number of bi-directional transformer layers. It achieves state-of-the-art performance on 11 NLP tasks, showing the great learning power of transformers. Apart from NLP tasks, the self-attention mechanism has also shown its advantage in other domains, such as video dialogue systems [30]. It is convenient to apply *BERT* to in ad-hoc retrieval tasks. For instance, BERTserini [61] concats the content of query and document into an integrate sentence for classification. However, it is not suitable to straightly apply *BERT* in interactive search scenarios where multi-grained interactions (i.e., query-level/document-level/session-level, etc) should be kindly modeled. At this stage, the parameter size of *BERT* is huge for only modeling the textual interactions. Once we consider to model session-level interactions by using *BERT*, more parameters will be introduced for the complex, hierarchical framework. To this end, we simplify our model by borrowing just the self-attention mechanism and the position encoding layer from *Transformer*.

2.4 Multi-task Learning

Most of the existing methods only optimize one task at a time, which may ignore the dependencies of user behaviors across tasks. To this end, researchers begin to explore multi-task learning approaches for various IR tasks [26, 35, 38, 45]. The purpose of multi-task learning is to use one task to assist the other one by using extended cross-task data for regularization. For example, Huang et al. [26] jointly learn context-aware ranking and entity recommendation to improve entity suggestion for web search. Jiang et al. [29] combine discriminative and generative loss functions to learn user reformulation behaviors. Ahmad et al. [2] explicitly model the dependency between users' in-session query and click sequence by learning context attentive representations, which mutually enhances document ranking and query suggestion. However, most of these frameworks are RNN-based and sequentially model past session contexts in the training phase. Moreover, they usually adopt the log-likelihood function for query suggestion, which causes large parameter sizes and slow convergence speed. To better utilize the contextual information, we build a framework with more complete session-level parallelization by using pure attention mechanisms.

Differences between our work and previous ones lie in 1) cross-session dependencies are leveraged to better model user intents in sessions with fewer contextual information, which further facilitates both learning tasks; 2) we employ pure attention mechanism to build a session-level multi-task learning framework, which has been proved to be more effective and efficient than RNN-based ones; 3) we unify document ranking and query suggestion into discriminative tasks and handle the query suggestion task with newly designed content encoders which incorporate both dense and sparse features into query embeddings; 4) besides clicked documents, we also consider user non-click behaviors as negative feedbacks for interaction modeling, which is proved crucial for user intent modeling in our experimental results.

3 PROBLEM STATEMENT

Before we delve into the detailed model design, we first formulate the problems in this paper. In complex search scenarios, users may interact with a search system by reformulating search queries or clicking on some results before ending a search session. There may be some dependencies between a user's session-level behaviors since all of them head for the same purpose: to fulfill the user's search intents. Therefore, we aim at leveraging these dependencies to model the session-level contextual information and improve the performances of both the *document ranking* and *query suggestion* tasks in a multi-task learning manner.

Specifically, we represent a user's search history within a session as a sequence of queries $Q = \langle q_1, q_2, ..., q_L \rangle$, where each query q_i is associated with a list of returned documents $\mathcal{D}_i = \langle d_{i,1}, ..., d_{i,N} \rangle$. A user may click on several of these results, hence we formulate an interactive action on the j-th document in the i-th query as $\mathcal{I}_{i,j} = \{d_{i,j}, C_{i,j}\}$, where $C_{i,j}$ denotes whether $d_{i,j}$ has been clicked by the user. Then the two problems in this paper can be defined as:

- **Query Suggestion**: Given a user's previous interactions $\mathcal{I} = \{\mathcal{I}_{i,j} | i < l\}$ and the query history $Q_{qs} = \langle q_1, q_2, ..., q_{l-1} \rangle$, we would like to predict the next query q_l the user will submit. For simplicity, we only consider the discriminative query suggestion task here, i.e., the goal is to rank the target query q_l in the candidate query set as high as possible.
- **Document Ranking**: Given a user's previous interactions $\mathcal{I} = \{\mathcal{I}_{i,j} | i < l\}$ and the query sequence $Q_{dr} = \langle q_1, q_2, ..., q_{l-1}, q_l \rangle$, we would like to rank the documents in the current query q_l to achieve higher evaluation metrics, e.g., nDCG or MAP.

For simplicity, we use "*qs*" and "*dr*" as the subscripts to represent the query suggestion and document ranking tasks respectively in the rest of this paper.

4 METHODOLOGY

In this section, we present a **H**ybrid framework for **S**ession **C**ontext **M**odeling (HSCM). As revealed in Figure 2, HSCM models session-level contextual information with self-attention mechanisms from scratch. For both document ranking and query suggestion, HSCM collects and fuses four sources of contextual information to facilitate downstream learning: 1) *the Guiding Query*, 2) *the Query History*, 3) *the Intra-session Interaction Aggregation*, and 4) *the Cross-session*

Interaction Aggregation. Next, we will first describe the details of the content encoder, which is the basic unit in HSCM. We further sequentially elaborate on the representation of intra-session contexts (1,2,3) in Section §4.2, cross-session contexts (4) in Section §4.3, and finally end with task prediction and model training in Section §4.4.

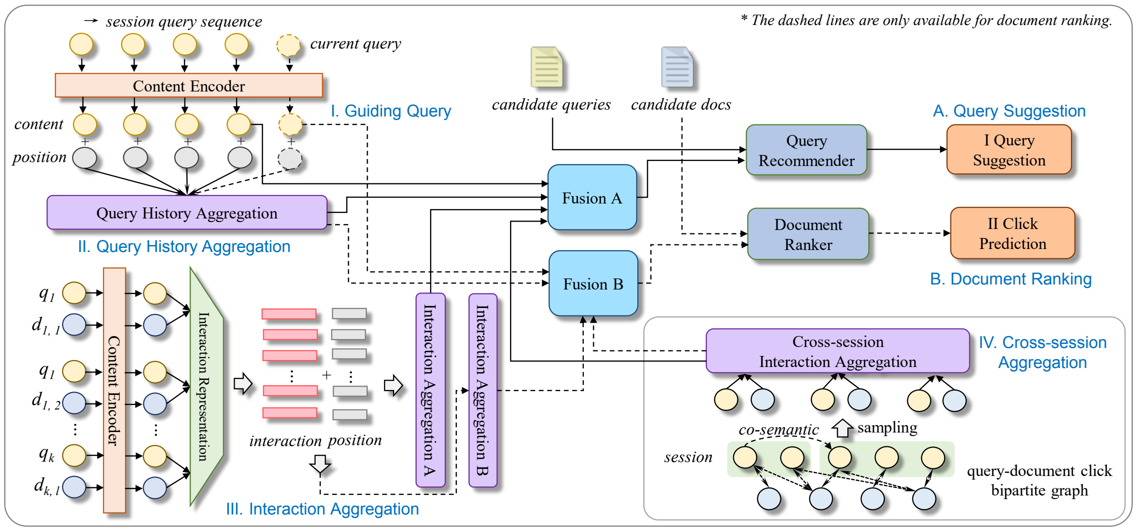


Fig. 2. The framework of HSCM. For both tasks, we utilize four sources of contextual information to enhance the model learning: 1) *the guiding query*, 2) *the query history aggregation*, 3) *the intra-session interaction aggregation*, and 4) *the cross-session interaction aggregation*. All sources are then fed into a fusion layer to contribute to either of the two tasks.

4.1 Content Encoder

To facilitate downstream calculations, we design a new content encoder to embed both queries and documents into fixed-length vectors with session-level contextual information (See Figure 3). Based on the self-attention mechanism, it can better represent both queries and documents by fully constructing interactions within a session. The input of a content encoder can be a query or a document segment [1] and the corresponding outputs are the session-interacted query embeddings and local document embeddings.

For a specific input sequence $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]$ (might be a query or a document segment), where $\mathbf{x}_i \in \mathbb{R}^{d_e}$ and d_e is the embedding size, we first apply a multi-head attention layer to calculate the local-interacted sequence:

$$\begin{aligned} \mathbf{X}' &= \text{MultiHead}(\mathbf{X}, \mathbf{X}, \mathbf{X}) \\ &= \text{Concat}(head_1, head_2, ..., head_h)\mathbf{W}^{\mathbf{O}} \end{aligned} \tag{1}$$

where $head_i = \text{Attention}(\mathbf{X}\mathbf{W}^{\mathbf{Q}}, \mathbf{X}\mathbf{W}^{\mathbf{K}}, \mathbf{X}\mathbf{W}^{\mathbf{V}})$, h is the number of heads, and Attention$(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ is the scaled dot-product attention mechanism [54]. Here we set $\mathbf{W}^{\mathbf{Q}}, \mathbf{W}^{\mathbf{K}}, \mathbf{W}^{\mathbf{V}} \in \mathbb{R}^{d_e \times d_e/h}$ to avoid parameter explosion with the increase of heads. Concat$(\cdot)$ is the vector concatenation operation along the second dimension over all heads. The

[1]Inspired by previous work which demonstrates the effectiveness of using topic structures in neural Information Retrieval [17, 51], we truncate all documents with a fixed length and then split them into segments. This can also alleviate the huge cost of computing self-attention over long sequences.

concatenated vector is then fed into an output weight matrix $\mathbf{W^O} \in \mathbb{R}^{d_e \times d_e}$ so that $\mathbf{X'}$ and $\mathbf{X}$ has the same shape. Let d_k be the dimensionality of $\mathbf{K}$, then the self-attention can be represented as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V} \tag{2}$$

The local-interacted input $\mathbf{X'} = [\mathbf{x}'_1, \mathbf{x}'_2, ..., \mathbf{x}'_n]$ is still a sequence of vectors. Hence, we embed the whole sequence into one distributed representation through a content embedding layer:

$$\mathbf{x}_i^p = \mathbf{x}'_i + pos(i);$$
$$\mathbf{Y} = \sum_{i=1}^{n} \alpha_i \odot \mathbf{x}_i^p, \alpha_i = \text{softmax}(\mathbf{W}_2^\alpha \tanh(\mathbf{W}_1^\alpha \mathbf{x}_i^p + \mathbf{b}^\alpha)). \tag{3}$$

where $\mathbf{Y}$ is the representation of the input sequence, i.e., the representation of a query or a document segment, $\mathbf{Y} \in \mathbb{R}^{d_h}$ and d_h is the hidden size. Here $pos(\cdot)$ is the sine-cosine position encoding mechanism as described in [54]. We employ this mechanism to add temporal information or dense features such as query frequencies over the local-interacted word embeddings. For example, when encoding session queries, we embed the temporal information into the query vectors. When we encode candidate queries in query suggestion, . $\mathbf{W}_1^\alpha/\mathbf{W}_2^\alpha$ and $\mathbf{b}^\alpha$ are the weights and bias for computing the importance factor α_i of each word, and $\odot$ is element-wise multiplication.

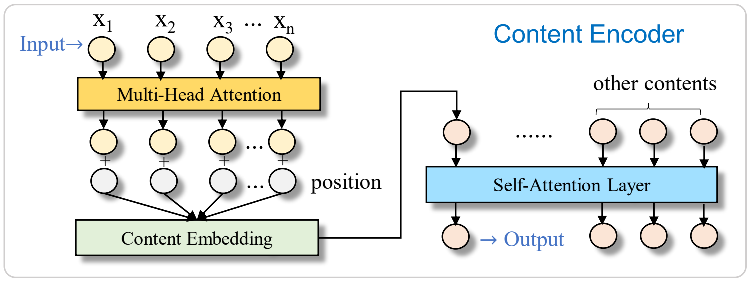


Fig. 3. The content encoder (CE) in HSCM. When encoding candidate queries in query suggestion, the self-attention layer is turned off and the features that position encoding layer embeds will be changed to the rank of co-occurrence frequency (in the descending order) of each candidate query.

Next, we adopt different strategies to calculate the output of the content encoder. For the current query q_l to be processed, let $C = \{q_j | \max(l - L, 1) \leq j \leq l\}$ be its query context, we first unify the length of C by padding [2] and then feed the representations of these queries (denoted as $\mathbf{Y}_c$) into a self-attention layer to obtain the session-level interacted query embeddings:

$$\mathbf{Q} = \text{Attention}(\mathbf{Y}_c, \mathbf{Y}_c, \mathbf{Y}_c). \tag{4}$$

where $\mathbf{Q} \in \mathbb{R}^{L \times d_h}$, L is the length of the padded query history, and d_h is the hidden size. Each element in $\mathbf{Q}$ is the representation of the corresponding query in the padded query context.

For a document, we can only obtain the segment embeddings through the aforementioned operations. Therefore, an additional aggregation layer is needed. Let $\mathbf{D} \in \mathbb{R}^{k \times d_h}$ be the segment embeddings for a document with k segments, we

[2]We truncate all query contexts at the cutoff L, and pad those contexts whose lengths are less than L with padding query IDs at the head of the sequence.

impose another content embedding layer similar to Equation 3 to aggregate the segment embeddings into a document embedding.

4.2 Intra-session Context Modeling

In this section, we consider leveraging three sources of intra-session information for user modeling: the *guiding query* **G**, the *query history aggregation* **H**, and the *intra-session interaction aggregation* **I**. Next, we will introduce these sources, respectively.

4.2.1 Guiding Query. We define guiding queries for two tasks differently. For document ranking, the guiding query is the current query. While in the query suggestion task, the current query, i.e., the intended query, is unknown and should be predicted. Therefore, we use the last query as the guiding query. The embedding of the guiding query (denoted as **G**) can be obtained via the content encoder in Section §4.1.

4.2.2 Query History. HSCM encodes the session-level query history because these queries imply to which extent a user has understood the search task. Besides, the reformulation chain also provides clues about the user's evolving intents. Given the query history in a session, we first apply the content encoder to obtain their embeddings $[\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_l]$, where $\mathbf{q}_i \in \mathbb{R}^{d_h}$. These query embeddings will then be aggregated into a whole representation via a positional aggregation layer:

$$\mathbf{q}_i^p = \mathbf{q}_i + pos(i);$$
$$\mathbf{H} = \sum_{i=1}^{n} \beta_i \odot \mathbf{q}_i^p, \beta_i = \text{softmax}(\mathbf{W}_2^\beta \tanh(\mathbf{W}_1^\beta \mathbf{q}_i^p + \mathbf{b}^\beta)). \tag{5}$$

where **H** is the presentation of query history and β_i denotes the weight of the i-th query. $\mathbf{W}_1^\beta$, $\mathbf{W}_2^\beta$ and $\mathbf{b}^\beta$ are the model parameters. Here we employ the position encoding to represent the temporal dependencies over the query sequence.

4.2.3 Intra-session Interaction Aggregation. User click behavior in sessions can reflect their propensity to a certain extent. Therefore, we regard user interactive behaviors on different documents as independent contextual information. Given the query history $Q = \langle q_1, q_2, ..., q_{l-1}\rangle$ and previous interactions $\mathcal{I} = \{\mathcal{I}_{i,j} \mid i < l\}$ for a certain query, we first apply the content encoder in Section §4.1 to embed each query and document into vectors. Previous work [2] only utilizes click signals for session modeling, which not only ignores the effects of non-clicks but also requires padding for batch processing. To this end, we embed both the clicked and non-clicked interactions into positive and negative feedback. For an L-length query history, we first sequentially collect the corresponding query embedding for each document within it and organize them as a set (denoted as **Q**). If we consider top k documents for each query, then $\mathbf{Q} \in \mathbb{R}^{kL\times d_h}$ (here we set $k = 10$ due to the setting of the dataset we used). Then let **D** be the embeddings of documents under these queries, and $\mathbf{C} \in \{0,1\}^{kL\times 1}$ be the binary click variables, session-level interactions can be represented as follows:

$$\mathbf{I}' = \mathbf{C}' * \text{ReLU}(\tanh(\mathbf{QMD})) \quad \mathbf{Q}, \mathbf{D} \in \mathbb{R}^{kL\times d_h} \tag{6}$$

$$where \quad C'_{i,j} = \begin{cases} 1, & C_{i,j} = 1; \\ -1, & else. \end{cases} \tag{7}$$

where $\mathbf{I}' \in \mathbb{R}^{kL\times d_h}$ represents all of the previous interactions, $\mathbf{C}'$ denotes the signed click signals (±1), and $\mathbf{M} \in \mathbb{R}^{d_h\times d_h\times d_h}$ represents the learnable bilinear layer which couples a pair of query and document. Employment of tanh and ReLU as activation function is to add nonlinear transformations and to filter some noisy interactions (e.g., accidental clicks), respectively.

Given the representation of all previous interactions $\mathbf{I}'$, we adopt different strategies for two tasks to aggregate them into a holistic interaction embedding. Let $\mathbf{I}_{i,j}$ be the representation of the j-th interaction in the i-th query, a position embedding is first added to each interaction representation to integrate the impact of result positions. Then for query suggestion, we straightly aggregate all the interactions without the involvement of the current query (Equation 9), which should be predicted. While for document ranking, we employ the query-aware aggregation, i.e., calculate the importance of each interaction under the guidance of the current query $\mathbf{q}_l^p$ (Equation 10).

$$\mathbf{I}_{i,j}^p = \mathbf{I}'_{i,j} + pos(j); \tag{8}$$

$$\mathbf{I}_{qs} = \sum_{i,j} \lambda_{i,j} \odot \mathbf{I}_{i,j}^p,\ \lambda_i = \text{softmax}(\tanh(\mathbf{W}^\lambda \mathbf{I}_{i,j}^p + \mathbf{b}^\lambda)); \tag{9}$$

$$\mathbf{I}_{dr} = \sum_{i,j} \mu_{i,j} \odot \mathbf{I}_{i,j}^p,\ \mu_i = \text{softmax}(\tanh(\mathbf{W}_1^\mu \mathbf{I}_{i,j}^p + \mathbf{W}_2^\mu \mathbf{q}_l^p + \mathbf{b}^\mu)). \tag{10}$$

where $\mathbf{I}_{qs}$ and $\mathbf{I}_{dr}$ denote the interaction aggregation embeddings for query suggestion and document ranking respectively, λ_i and μ_i are the importance factors for each previous interaction in two tasks, and $\mathbf{W}^\lambda/\mathbf{W}_1^\mu/\mathbf{W}_2^\mu/\mathbf{b}^\lambda/\mathbf{b}^\mu$ are the model parameters. Here the representation of each previous interaction $\mathbf{I}_{i,j}^p$ contributes to both $\mathbf{I}_{qs}$ and $\mathbf{I}_{dr}$, which will be used for the learning of both document ranking and query suggestion later.

4.3 Cross-session Context Modeling

To explicitly leverage the contexts from other sessions, we introduce a cross-session aggregation module. The main point of considering cross-session contexts is to improve the retrieval performance of cold-start queries (i.e., queries with less previous contextual information) and long-tailed queries that suffer severely from the data-sparsity problem via data augmentation. By bridging two queries across different sessions through co-clicked and co-semantic dependencies, we collect expanded queries for the guiding query and further aggregate the sampled user behaviors on them. Briefly, this process can be summarized in the following main steps:

(1) Construct a cross-session graph $\mathcal{G}$;
(2) For guiding query, sample user interactions guided by similar search intents from the constructed graph;
(3) Aggregate sampled behaviors into a distributed representation $\mathbf{I}^c$, which will be regarded as a new context;

In (1), we first build a click bipartite graph on session data and then add edges between queries with certain semantic relationships. The co-semantic connections can be very effective for intent modeling in some datasets, e.g., ones that are sparse in user click behaviors. For a specific query, we search for the top-3 most similar queries and add directed edges from it to these queries. Here query embeddings are represented as the average-pooled GloVe [42] vectors and the semantic similarity function we used is cosine similarity. A concern is that it is very time-consuming to search for the nearest neighbors for all queries in a large-scale dataset. Therefore, we normalize all query embeddings and employ a Maximum Inner Product Search (MIPS) algorithm to reduce the computation complexity [43]. Given a query point $q \in S$, where S is the set for all points, we would like to find the best matches concerning the cosine-similarity. According to Equation 12, it is obvious that the best match with cosine similarity is the best match with inner-products if all the points in the set S are normalized to the same length.

$$p = \arg\max_{r \in S} \frac{\langle q, r \rangle}{\|q\|\|r\|} = \arg\max_{r \in S} \frac{\langle q, r \rangle}{\|r\|} \tag{11}$$

$$\neq \arg\max_{r \in S} \langle q, r \rangle (\text{unless } \|\text{r}\| = \text{k}, \forall \text{r} \in \text{S}) \tag{12}$$

We implement a simple version of MIP Tree, which is a clever variant of the Ball Tree designed to solve the nearest-vector-finding problem. In practice, it can be several orders of magnitude faster than a linear scan over the entire collection. Ball trees are binary space-partitioning trees that have been widely used for the task of indexing the datasets. Every node in the tree represents a set of vectors and each node is subsequently indexed by a ball enclosing all the points in the node with its center, as depicted in Figure 4. The set of the point at a node is divided into two disjoint sets. This partitions the space into (possibly overlapping) hyper-spheres and forms the child nodes. The tree is built hierarchically and a node will be declared to be a leaf node if it contains a set with less than N_0 points. In this paper, we set $N_0 = 20$. Having constructed the ball tree, we can then apply a depth-first branch-and-bound algorithm to search

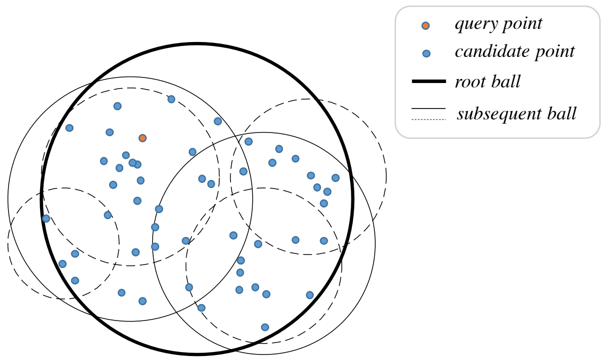


Fig. 4. An example of the ball tree. All points will be finally assigned to a leaf node.

for the best-matched nodes. To avoid data leakage, we only consider co-clicked dependencies for training sessions and add all semantic edges in an online manner, i.e., without preprocessing. In this way, HSCM cannot observe the session contexts in the validating/testing set and should infer the query recommendations and document relevances according to the co-clicked dependencies from the past observations and the newly added co-semantic relationships. Having built the graph based on the dataset, we can store the whole graph and load it next time if we need to add some new query nodes or to update the edges.

For (2), we start by searching for all connected query nodes for the guiding query on the graph via the breadth-first-search (BFS) algorithm. Here the search depth is restricted ≤ 3 to reduce the computation complexity as well as the number of extended queries. A broad extension can include many irrelevant queries that may largely influence system performance. Therefore, we only remain a small proportion of queries whose cosine similarity with the guiding query in query embedding is higher than a threshold of θ, where $\theta \in [0,1)$.

In the end, we randomly sample n queries based on the cosine similarities between these queries and the guiding one. In this regard, queries that are more similar to the guiding one have more chances to be selected. The detailed sampling process is formulated in Algorithm 1. Given a guiding query g and the cross-session graph $\mathcal{G}$, this algorithm will eventually return a set of sampled queries (denoted as $\mathcal{S}$). For each query in $\mathcal{S}$ (which may occur at a different time in

Algorithm 1 Sample extended queries for the guiding query

```
Require: The guiding query, $g$; The cross-session graph, $\mathcal{G}$;
Ensure: The set of sampled queries, $\mathcal{S}$;
1:  $\mathcal{S} = \emptyset, C = \emptyset$;
2:  Collect candidate queries $C$ for $g$ on $\mathcal{G}$ via BFS, depth $\leq 3$;
3:  $C = C - \{q|q \in C, cos\langle\vec{q},\vec{g}\rangle < 0\}$;
4:  for each $i \in [1, n]$ do
5:      Calculate sampling probabilities $\pi(q_j) = \frac{cos\langle\vec{q_j},\vec{g}\rangle}{\sum_{q\in C} cos\langle\vec{q},\vec{g}\rangle}$;
6:      Sample a query node $q$ from $C$ according to $\pi$;
7:      if $cos\langle\vec{q},\vec{g}\rangle \geq \theta$ then
8:          $\mathcal{S} = \mathcal{S} \cup q, C = C \setminus q$;
9:      end if
10: end for
11: return $\mathcal{S}$;
```

the training set), we randomly sample user behaviors on its top 10 documents. Cases with less than n expanded queries or sampled queries with no user interactions (i.e., queries that only occurs in the validating/testing set) will be padded. We then encode the expanded cross-session interactions into distributed representations via a similar aggregation layer in §4.2.3. Finally, the cross-session context is denoted as $\mathbf{I}^c_{qs}/\mathbf{I}^c_{dr}$ for two tasks.

4.4 Prediction and Model Training

4.4.1 Prediction. Having represented the four contexts $\mathbf{G}$, $\mathbf{H}$, $\mathbf{I}$, and $\mathbf{I}^c$ for a specific task, we further utilize them for the downstream prediction. We aggregate them through a concatenation layer (Fusion A/B in Figure 2) to obtain the session representation:

$$\mathbf{S}_{[\cdot]} = \tanh((\mathbf{W}_{[\cdot]} + \mathbf{W}_{share})([\mathbf{G}_{[\cdot]} \oplus \mathbf{H}_{[\cdot]} \oplus \mathbf{I}_{[\cdot]} \oplus \mathbf{I}^c_{[\cdot]}])) \tag{13}$$

where $\mathbf{S}_{[\cdot]} \in \mathbb{R}^{d_h}$ is the fused contextual embedding for a specific task (here "$[\cdot]$" can be qs or dr), and $\oplus$ denotes the vector concatenation. $\mathbf{W}_{[\cdot]}/\mathbf{W}_{share} \in \mathbb{R}^{d_h \times 4d_h}$ are the output linear layers. The difference is that $\mathbf{W}_{share}$ is shared between two tasks while $\mathbf{W}_{[\cdot]}$ is only private to the corresponding task.

For query suggestion, we first encode candidate queries into embeddings $\mathbf{Q}_{cand} = [\mathbf{q}^c_1, \mathbf{q}^c_2, ..., \mathbf{q}^c_{T_1}] \in \mathbb{R}^{T_1 \times d_h}$ through a content encoder, where T_1 is the number of candidate queries. Here we turn on the position encoding to embed the dense feature, i.e., the rank of a query within the candidate set in the descending order of co-occurrence frequency, into the vector space. We then predict the probability of each candidate query to be the intended one by calculating the similarities between the contextual representation and the candidate embeddings:

$$\mathcal{P}_{\mathbf{q}^c_i} = \text{sigmoid}(\mathbf{S}^T_{qs} \cdot \mathbf{q}^c_i) \tag{14}$$

where $\mathcal{P}_{\mathbf{q}^c_i}$ is the predicted score for the i-th query in the candidate set. Here we apply sigmoid as the activation function to enable the point-wise learning, which accelerates convergence.

For document ranking, we also embed the candidate documents with a content encoder. Suppose $\mathbf{D}_{cand} = [\mathbf{d}^c_1, \mathbf{d}^c_2, ..., \mathbf{d}^c_{T_2}] \in \mathbb{R}^{T_2 \times d_h}$ denotes the representation of candidate documents and T_2 is the number of candidate documents, we estimate the click probabilities of each document through another prediction layer:

$$\mathcal{P}_{\mathbf{d}^c_i} = \text{sigmoid}(\mathbf{S}^T_{dr} \cdot \mathbf{M}(\mathbf{d}^c_i)) \tag{15}$$

here $\mathcal{P}_{\mathbf{d}_i^c}$ denotes the predicted click probability of the i-th candidate document, and $\mathbf{M} \in \mathbb{R}^{d_h \times d_h}$ is the matching linear layer for coupling a document embedding and the contextual embedding $\mathbf{S}_{dr}$. We further rank all candidates according to predicted scores.

To facilitate the learning of document ranking along with query suggestion, we reuse the contextual information to represent the whole session for both tasks. However, there are differences between the two fused contextual embeddings. On one hand, the representation for query suggestion, which only leverages contexts before the current query, should be as similar to the current query as possible. On the other hand, the representation for document ranking should estimate the user's information needs. Here we prefer that this contextual embedding should be as similar to the clicked results by the user as possible. The two representations share some public contextual information thus can promote the multi-task learning process.

4.4.2 Model Training. We train HSCM end-to-end through the back-propagation algorithm based on multi-task learning. To estimate the parameters in HSCM, we would like to minimize the regularized loss function as follows:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{n} ((1 - \mu) \cdot \mathcal{L}_{qs}(q_n) + \mu \cdot \mathcal{L}_{dr}(q_n)) + \lambda ||\theta||^2 \tag{16}$$

where θ denotes all parameters in HSCM, $\mathcal{L}_{qs}$ and $\mathcal{L}_{dr}$ are the loss functions for query suggestion and document ranking, and $1 - \mu$ and $\mu \in (0, 1)$ are the weights for the two losses, respectively. N is the number of query sessions or subtasks in the training set and q_n is the n-th query. The regularization term is added to avoid model overfitting. For both tasks, we adopt the cross-entropy function as their learning objectives:

$$\mathcal{L}_{qs}(q_n) = -\sum_{a} (\mathcal{I}_{q_{n,a}} \log \mathcal{P}_{q_{n,a}} + (1 - \mathcal{I}_{q_{n,a}}) \log(1 - \mathcal{P}_{q_{n,a}})) \tag{17}$$

$$\mathcal{L}_{dr}(q_n) = -\sum_{b} (\mathcal{I}_{d_{n,b}} \log \mathcal{P}_{d_{n,b}} + (1 - \mathcal{I}_{d_{n,b}}) \log(1 - \mathcal{P}_{d_{n,b}})) \tag{18}$$

here $\mathcal{I}_{q_{n,a}}$ and $\mathcal{P}_{q_{n,a}}$ represent the observed and predicted probability of the a-th candidate query in the n-th subtask to be the intended next query. If $q_{n,a}$ is the next query then $\mathcal{I}_{q_{n,a}} = 1$, otherwise 0. Besides, $\mathcal{I}_{d_{n,b}}$ and $\mathcal{P}_{d_{n,b}}$ denote the observed and predicted click probability of the b-th candidate document in the n-th subtask, respectively. To stabilize the training process, we abandon a proportion of training instances. The filtering rules are:

- For query suggestion, we abandon an instance if 1) the intended query is not included in the candidate set; 2) there is only one candidate; 3) the candidate set does not exist, e.g., the first query within a session or the last query is out-of-vocabulary.
- For document ranking, we filter an instance if there is no clicked document in the current query.

5 EXPERIMENTAL SETUPS

To investigate the effectiveness of HSCM, we further conduct a series of experiments to address the following research questions:
RQ1: How does HSCM perform in document ranking?
RQ2: How does HSCM perform in query suggestion?
RQ3: How will different contexts, e.g., query history, intra-session interactions, and cross-session interactions affect overall system performance?
RQ4: Can cross-session interaction aggregation better handle search sessions with fewer intra-session contextual information?

5.1 Dataset

- **The TianGong-ST Dataset** We conduct our experiments on a public Chinese-centric session dataset, namely TianGong-ST [11]. The dataset was refined from a search log recorded by *Sogou.com*, which is among the top 3 popular search engines in China. We observe that the original dataset contains a small proportion of sessions with only repeated queries due to user paging-down behaviors. To tackle this problem, we filter these sessions. For all unique URLs, we crawl the web page content if it is missing in the TianGong-ST dataset and then discard those sessions with missing documents. Finally, there are 167,846 unique documents in total. We further split the remaining 120,256 sessions into training, validating, and testing set with a ratio of 8:1:1. The specifics of the dataset we used to run our experiments are provided in Table 1.

  For the query suggestion task, we use a huge background session dataset [10] to generate the candidate queries for each unique query [3]. We use this background dataset because it is also from *Sogou.com* and is publicly available. The top-20 queries with the highest frequencies co-occurring [4] with the last query in a search context in background sessions are regarded as the candidate queries. We further rank these candidates according to the predicted scores. For the document ranking task, we re-rank the top-10 documents in each query based on the click signals. To preprocess the corpus content, we adopt jieba_fast [5] for Chinese word segmentation. K-anonymity approach (K=5) is then adopted to preserve user privacy, i.e., only frequent words (about 359k) are retained in the word dictionary while the rest are represented as <UNK>. We further use pre-trained GloVe [42] word vectors as the initial embeddings to feed into HSCM.

Table 1. Basic statistics of our constructed dataset based on **TianGong-ST** and the background dataset from Sogou.

	All	**Training**	**Validating**	**Testing**	**Background** [10]
# sessions	120,256	96,248	11,994	12,014	1,383,134
# unique queries	18,125	16,547	5,710	5,769	194,792
avg. session length	2.2079	2.2076	2.2050	2.2134	2.5379
avg. query length	2.3805	2.3826	2.3800	2.3644	2.5985
avg. document length	1218.4	1213.7	1277.6	1197.1	–
avg. # click per query	0.7086	0.7075	0.7198	0.7062	–

- **The AOL Search Data** As TianGong-ST is Chinese-centric, we also conduct our experiments on the widely-used AOL search dataset [41]. This dataset was collected from March 1st, 2006 to May 31st, 2006, with a span of about 11 weeks. We split queries submitted by the same user into sessions with a 30-minute threshold. All sessions with less than two queries or without any clicks are discarded. To keep consistency with the TianGong-ST sessions, we also only consider the results on the first page for a specific query, i.e., top-10 documents. Since this log is outdated, it is really hard to crawl the textual contents for all documents. Here we obtain a title corpus provided by Ahmad et al. [2] and use document titles as their contents. We further remove all non-alphanumeric characters for all queries and documents. After that, queries that only contain punctuations will be eliminated. We then abandon a session if the title for any clicked result within it is missing.

  The AOL dataset only contains the clicked URLs for each query and does not record any other candidate documents. Therefore, we need to find the candidate documents and to restore the original log. To this end,

[3]This dataset does not contain click-through information thus we only use it to generate candidate queries for query suggestion.
[4]"Co-occurrence" refers to the following queries of the current one. For example, if we find a query sequence "AB" in a background session, then the frequency of query B being the candidate of A should be added by one.
[5]https://pypi.org/project/jieba-fast/0.42/

we recall the top 20 documents for each query ranked by the BM25 algorithm and fill them into the missing positions on the log one by one. Through the investigation of the title corpus, we observe that many clicked documents do not contain any overlapped words concerning the corresponding query. This may be because many websites have largely changed since then, as time went by. Keep using these URLs may hurt the system performance to a certain extent. Therefore, we remove all queries if any of their clicked document is not among the top 20 candidate documents collected before. Finally, we split the remaining sessions into the background, training, validating, and testing sets, respectively. Following previous work [2, 41], we use the first six weeks as the background set, the next five weeks as the training set, and the rest two weeks are divided into halves to build the validating and testing sets. Specifics for the AOL dataset can be found in Table 2. We can observe that the sessions in AOL dataset are longer than those in TianGong-ST, and the average query length is slightly longer. Moreover, it contains more unique queries, which may bring challenges for both tasks. Here we also aggregate the top-20 co-occurrences as the candidates for each query based on the background set. The word dictionary contains about 50k English words, each of which occurs at least 10 times in the corpus. Other settings are the same as those concerning TianGong-ST.

Table 2. Basic statistics of our constructed dataset based on the **AOL data**.

	All	**Training**	**Validating**	**Testing**	**Background**
# sessions	137,530	101,750	21,195	14,585	161,836
# unique queries	220,924	168,781	39,679	28,014	252,199
avg. session length	3.2778	3.2603	3.3021	3.3648	3.2452
avg. query length	2.8022	2.8020	2.8046	2.8004	2.7863
avg. document length	6.1024	6.1243	6.0291	6.0589	6.1377
avg. # click per query	0.3865	0.3873	0.3833	0.3854	0.3962

5.2 Baselines and Metrics

To evaluate the performance of HSCM, we compare it with three types of baseline methods: traditional models, deep ad-hoc models, and deep interactive models. For document ranking, we consider the following methods:

- **BM25** [44]: A widely-used probabilistic model based ranking function.
- **Rocchio** [31]: An interactive retrieval method. Here we use Rocchio-CLK, where clicked documents are regarded as positive feedbacks. For queries without previous clicks, we rank the documents randomly.
- **Rocchio+BM25**: A variant of Rocchio. As Rocchio can not rank the documents under the first query of a session, we use BM25 to rank them.
- **QCM** [65]: A query change based session search model. Here we use the same parameter setting as the original paper where $\mu = 5,000$, $\alpha = 2.2$, $\beta = 1.8$, $\epsilon = 0.07$, and $\delta = 0.4$.
- **Win-win** [36]: A reinforcement learning-based session search model. It develops a Partially Observed Markov Decision Process (POMDP) to model the exploration and exploitation of user state when reformulating queries within a session. We implemented a simplified version of it which adopts BM25 as the core search strategy according to the corresponding published paper since no public code is available so far.
- **DRMM** [22]: A basic ad-hoc neural ranking model that considers both the exact matches and the semantic similarities between the queries and documents. It uses matching histogram mapping as the interaction function. The interaction within the interval $[1, 1]$ is placed into 41 ordered bins in our implementation.

- **DSSM** [27]: A simple representation-based model with two-layer DNN, the hidden size of for each layer is 300 and 128 respectively.
- **ARC-I** [24]: A representation-based model that stacked convolutional layers and max-pooling layers to produce high-level representations of document and query. We use a two-layer CNN where the filter windows sizes are 3 and there are 256 and 128 feature maps for each filter respectively.
- **ARC-II** [24]: A neural ranking model that uses a CNN to map the word embeddings for query/document to an aggregated embedding. Here we use a two-layer CNN, where the size of kernels and pooling in both layers are set to (3×3).
- **HiNT** [17]: A hierarchical model that builds two matching layers to model the local relevance signals of a query-document pair and to further obtain the final relevance scores by accumulating local signals into different granularities. We set the max passage length to 24 and segment the document into passages for the TianGong-ST dataset (For the AOL dataset, documents are too short so we regard the whole document as a passage). The hidden size is set to 2 according to the performance in our validating set.
- **KNRM** [60]: A neural ranking model that adopts a kernel pooling strategy to model the matching signals at multiple semantic levels. In our experiments, we use the default setting in the original paper (totally 11 kernels, 10 for soft-matching, and 1 for exact-matching).
- **BERT** [15, 61]: A transformer-based pre-training model which has achieved state-of-the-art performance on a series of NLP tasks. Here we concatenate query and document into a long sentence ($< [CLS], Q, [SEP], D, [SEP] >$) as done in [61]. It will be then fed into BERT for binary classification. We fine-tuned the parameters according to the classification performance on the validating set.

For query suggestion, we select the following methods:

- **MPS**: Most popular suggestion, which is a maximum likelihood method that relies on "wisdom of the crowd" and ranks queries by the co-occurrence to the last query in the background set. The candidate queries in our experiments are generated by MPS.
- **Hybrid** [4]: A model that ranks the candidate queries based on a linear combination between their popularity (e.g., generated by MPS) and the similarities to recent queries.
- **QVMM** [23]: Query-based Variable Markov Model, which learns the probability of query transitions over sessions with the variable memory Markov model implemented by a suffix tree.
- **HRED-qs** [49]: An RNN-based deep neural model for query suggestion. We use a one-layer bidirectional LSTM as our encoder and the hidden size is 512.
- **Seq2seq+Attn.** [3]: An improved version of Seq2seq, which is a deep learning based model for query suggestion. The attention mechanism is employed to improve system performance.

We further use two models which also adopt the multi-task learning technique for training:

- **M-NSRF** [1]: A deep interactive architecture to learn both document ranking and query suggestion by modeling session contexts without click signals. The hidden neurons of query, document, session are set to 512, 512, 1024, respectively.
- **CARS** [2]: An improved version of M-NSRF that considers click signals in the session context. It uses a two-level hierarchical recurrent neural network to learn search context. The number of hidden neurons in each of its encoders and decoders were selected from $\{150, 256, 512\}$. The hyper-parameters λ_1, λ_2 , and λ_3 are set to 10^{-2}, 10^{-4}, and 10^{-1}.

Table 3. Comparison of several models in the utilization of various contexts. Here "dense features" can be the co-occurrence frequency of the candidates in query suggestion.

Model	*Guiding Query*	*Query History*	*Intra-session Clicks*	*Non-clicks*	*Dense Features*	*Crosss-session Contexts*	*External Corpus*
HiNT [17]	✓	×	×	×	–	×	×
Rocchio [31]	✓	×	✓	×	–	×	×
BERT [61]	✓	×	×	×	–	×	✓
Seq2seq [3]	✓	×	×	×	×	×	×
QVMM [23]	✓	✓	×	×	✓	×	×
Hybrid [4]	✓	×	×	×	✓	×	×
M-NSRF [1]	✓	✓	×	×	×	×	×
CARS [2]	✓	✓	✓	×	×	×	×
HSCM	✓	✓	✓	✓	✓	✓	×

As we use clicks as the ground truth labels, we further compare HSCM with some existing click models to properly evaluate their click prediction performances. Some aforementioned ranking baselines are considered for comparison, such as CARS, M-NSRF, HiNT, and KNRM. Output scores by these models have been normalized to represent click probabilities. We also compare them with some existing click models as follows.

- **UBM** [16]: A popular traditional click model that models user browsing behavior in Web search.
- **DBN** [8]: A popular traditional click model that is based on the dynamic Bayesian network.
- **CACM** [12]: A state-of-the-art click model that leverages contextual information within sessions to model user intents. As with most click models, CACM is content-free. In this paper, we use the click probabilities estimated by CACM for document ranking. Since the vertical type information of each result is missing, we set the same value (e.g., default=1, organic result) for all results.

We use open-source codes for all deep ranking models [6] while re-producing the rest traditional ones according to the published papers. For click models, we use the public implementation of UBM, DBN, and CACM [7]. The comparison of several typical models in the utilization of several sources of contextual information is given in Table 3.

To measure the performance of each model on the ranking task, we regard user clicks as the relevance labels and use them to calculate the Normalized Discounted Cumulative Gain (nDCG) and the Mean Average Precision (MAP) of a ranked list.

- **nDCG@k**: Let r_i be the relevance for the i-th document in the rank list, then DCG@k can be computed as Equation 19. Then the score of nDCG@k can be obtained by normalizing the DCG using ideal DCG@k.

$$DCG@k = \sum_{i=1}^{k} \frac{r_i}{\log_2(\mathrm{i}+1)} \tag{19}$$

$$nDCG@k = \frac{DCG@k}{IDCG@k} \tag{20}$$

[6] https://github.com/NTMC-Community/MatchZoo; https://github.com/wasiahmad/context_attentive_ir
[7] https://github.com/markovi/PyClick, https://github.com/xuanyuan14/CACM-master

- **MAP**: Let S be the number of testing cases, N_i be the number of clicked documents in query q_i, and $rank_{i,j}$ be the rank of the j-th clicked document in the query q_i, then MAP can be represented as follows:

$$MAP = \frac{1}{|S|}(\frac{1}{N_i}\sum_{j=0}^{N_i-1}\frac{j+1}{rank_{i,j}}) \tag{21}$$

As for query suggestion, we use Mean Reciprocal Rank (MRR) and the hit rate of the intended query at a specific rank k (HIT@k) in the candidate set as the evaluation metrics. Let $rank_i$ be the rank of the intended query in the candidates for the query q_i, then **MRR** and **HIT@k** can be computed as Equation 22 and 23, respectively.

$$MRR = \frac{1}{|S|}\sum_{q\in S}\frac{1}{rank_i} \tag{22}$$

$$HIT@k = \frac{1}{|S|}\sum_{q\in S}\mathbb{1}(rank_i <= k) \tag{23}$$

Here S is the set of all targeted queries and $\mathbb{1}$ is the indicator function. Due to the top-heaviness, we set the cutoff k=1, 3, 5 in this paper.

For click prediction, we report the click perplexity (PPL) [16] and the log-likelihood of each model. The definition of the click perplexity (PPL) at rank r and the log-likelihood (LL) are as follows:

$$PPL@r = 2^{-\frac{1}{N}\sum_{i=1}^{N}C_{i,r}logP_{i,r}+(1-C_{i,r})log(1-P_{i,r})}, \tag{24}$$

$$LL = \frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M}C_{i,j}logP_{i,j} + (1-C_{i,j})log(1-P_{i,j}), \tag{25}$$

Here the subscript r represents a rank position, N is the total number of query sessions in the testing set, and M is the number of results within a query. $C_{i,r}$ and $P_{i,r}$ denote true click labels and predicted click probability of the r-th result in the i-th query session, respectively. Then the overall perplexity is obtained by averaging the values over all ranks. Generally, a lower value of PPL and higher value of LL indicates a better prediction performance.

5.3 Parameter Setups

We train HSCM end-to-end with a mini-batch size of 4 by using the Adam optimizer [32]. The hidden size and embedding size of GloVe vectors are both 256. To stabilize the learning process, we clip the gradient within a maximum normalization of 2.0. The initial learning rate and the weight decay parameter λ are selected from $\{10^{-3}, 10^{-4}\}$ and $\{10^{-4}, 10^{-6}\}$, respectively. For all samples, we consider a unified context length of $L = 6$. To facilitate the batch processing in self-attention computation, we only utilize the first 24/480 words for each query/document in TianGong-ST. As for the AOL dataset, we use titles as the document content hence only the first 12 words are considered. All documents are then divided into 20/2 segments for TianGong-ST and AOL dataset respectively, each of which contains 24/6 words and will be fed into the content encoder. We also investigate the model performance across different number of head h in the multi-head attention mechanism (See Equation 1 for reference) and find the best number is four. According to experiment results, we empirically set the number of sampled cross-session queries n to three, the extension similarity threshold θ to 0.8, and the multi-task learning weight for document ranking μ to 0.9. The early-stop strategy is adopted when the validation performance does not improve after five iterations. All deep models are trained on an NVIDIA TITAN XP 12G GPU. The implementation code of HSCM is based on PyTorch [8] and is released in the link below [9].

6 RESULTS AND ANALYSIS

In this section, we report the experimental results to answer the aforementioned research questions. To begin with, we elaborate on the overall performance of HSCM on both document ranking and query suggestion. More analysis will then be introduced in the following subsections.

6.1 Document Ranking Evaluation

To answer **RQ1**, we report the ranking performances of all models in Table 4 and Table 5. According to the experiment results, we have several findings. First, Rocchio-CLK outperforms the traditional BM25 algorithm on both datasets, indicating the advantage of properly exploiting user click behaviors as positive feedbacks. Both the QCM and Win-win model perform poorly on two datasets. The default parameters of QCM from the original paper may not adapt to other datasets. As for Win-win, its performance is largely restricted by the core search strategy. Therefore, it can not achieve good performance by merely using BM25 for retrieval. Due to the sparse exact matches between queries and documents, BM25 does not perform well on the click-based evaluation. To our surprise, DRMM performs badly on two datasets. We guess this may be because it ignores the contextual and positional information of each word. As our task is based on document re-ranking, i.e., not recall-oriented, and most candidate documents under a specific query are highly similar in bag-of-words, DRMM may not accurately model the relevances between these query-document pairs. Among all deep learning-to-rank baselines, HiNT outperforms others on TianGong-ST thanks to its application of the topic structure. However, it performs slightly worse in the AOL dataset where the documents are too short, restricting its effectiveness of considering passage-level contents. There are still large margins between these neural ranking models and the deep session-level ones, e.g., M-NSRF and CARS. Although BERTserini is superior to most ad-hoc methods by adopting external corpus for model pre-training, it can not beat the session-level ones. These interactive models not only consider relationships between a query-document pair but also properly leverage user interactive behaviors within sessions to better capture their search intents. Therefore, they can easily achieve better ranking performance than those ad-hoc ones. By integrating the cross-session contexts and non-click behaviors, HSCM performs the best among all models. Its attention-based nature, which realizes a more complete parallelization across the whole session, also enables it to make better use of the intra-session contextual information.

On the AOL dataset, although HSCM performs significantly better than most models, we find that it does not show great superiority compared to the strongest baselines. We guess there are two main reasons: 1) Low document quality: the documents are too short in the AOL dataset, many documents are very similar in the content; 2) Information loss: original ranking lists are lost, which is hard for more sophisticated models to learn user examinations and to further boost the pairwise learning. Here we can only empirically restore the raw log data and generate the pseudo document lists using BM25 scores according to some previous work, which might not be so correct.

To investigate the robustness of HSCM, we analyze its performance across sessions with different lengths and queries of different frequencies. We first split all testing sessions into three groups according to their lengths: 1) short sessions (2 queries); 2) medium sessions (3-4 queries); 3) long sessions ($\geq$ 5 queries). As for query frequency, we split all testing queries into four intervals according to their occurrence frequency in the training set: $[0, 10)$, $[10, 100)$,

[8]https://pytorch.org/
[9]https://github.com/xuanyuan14/HSCM-master

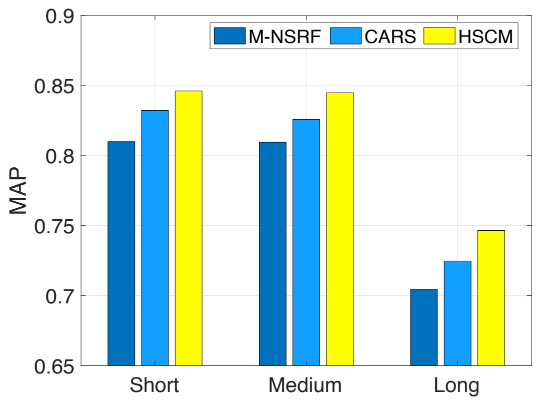
Table 4. Document ranking performance of each model on **TianGong-ST**. Note that $*$ denotes that compared to HSCM, the performance difference is statistically significant at a level of $p < 0.01$ using paired t-test. The best method in each group is underlined.

Model	nDCG@1	nDCG@3	nDCG@5	nDCG@10	MAP
Traditional Models					
BM25	0.1734*	0.3303*	0.4487*	0.5417*	0.3933*
Rocchio	0.3750*	0.4736*	0.5035*	0.6362*	0.5211*
Rocchio+BM25	0.4133*	0.5396*	0.6201*	0.6817*	0.5764*
QCM	0.1117*	0.2041*	0.2714*	0.4552*	0.2891*
Win-win	0.1743*	0.3303*	0.4489*	0.5419*	0.3935*
Deep Ad-hoc Models					
DRMM	0.1851*	0.3287*	0.4258*	0.5375*	0.3901*
DSSM	0.6489*	0.7331*	0.7679*	0.8130*	0.7480*
ARC-I	0.6529*	0.7603*	0.7980*	0.8249*	0.7618*
ARC-II	0.6503*	0.7583*	0.7952*	0.8234*	0.7598*
KNRM	0.6580*	0.7592*	0.7981*	0.8252*	0.7626*
HiNT	0.7140*	0.7921*	0.8212*	0.8506*	0.7968*
BERT	0.7161*	0.8111*	0.8381*	0.8597*	0.8079*
Deep Session Context-Aware Models					
M-NSRF	0.7230*	0.8072*	0.8363*	0.8599*	0.8086*
CARS	0.7447*	0.8320*	0.8564*	0.8751*	0.8286*
HSCM	**0.7755**	**0.8459**	**0.8681**	**0.8873**	**0.8450**

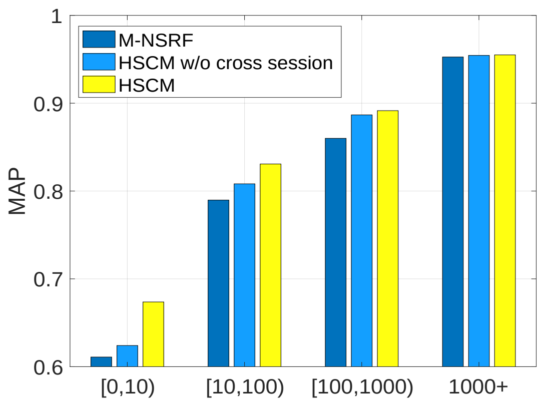
Table 5. Document ranking performance of each model on **AOL dataset**. Note that $*/\dagger$ denotes that compared to HSCM, the performance difference is statistically significant at a level of $p < 0.01/0.05$ using paired t-test. The best method in each group is underlined.

Model	nDCG@1	nDCG@3	nDCG@5	nDCG@10	MAP
Traditional Models					
BM25	0.1436*	0.3052*	0.3629*	0.5013*	0.3515*
Rocchio	0.2779*	0.3752*	0.4409*	0.5664*	0.4367*
Rocchio+BM25	0.3300*	0.3801*	0.4245*	0.5726*	0.4489*
QCM	0.1021*	0.2311*	0.3062*	0.4635*	0.3031*
Win-win	0.1654*	0.2955*	0.3689*	0.5068*	0.3585*
Deep Ad-hoc Models					
DRMM	0.0995*	0.2239*	0.3166*	0.4648*	0.3039*
DSSM	0.2700*	0.4851*	0.5538*	0.6167*	0.4955*
ARC-I	0.3222*	0.5631*	0.6223*	0.6635*	0.5541*
ARC-II	0.3070*	0.5415*	0.6044*	0.6515*	0.5388*
KNRM	0.2481*	0.4678*	0.5482*	0.6068*	0.4809*
HiNT	0.3451*	0.5908*	0.6449*	0.6806*	0.5762*
BERT	0.3593*	0.6119*	0.6635*	0.6931*	0.5935*
Deep Session Context-Aware Models					
M-NSRF	0.3657*	0.6171	0.6679	0.6967	0.5951†
CARS	0.3628*	0.6202	0.6698	0.6966	0.5965
HSCM	**0.3748**	**0.6213**	**0.6715**	**0.7004**	**0.6014**

$[100, 1000)$, and $[1000, +\infty)$. Table 6 presents the number and proportion of valid instances across all conditions in document ranking. Generally, AOL queries occur fewer times in the training set than those in TianGong-ST, i.e., queries



(a) Comparison on session lengths.



(b) Comparison on query frequencies.

Fig. 5. Comparison based on document ranking w.r.t. session lengths and frequencies of the current query on TianGong-ST.

are more sparse. Therefore, overall system ranking performances on AOL are worse than that on TianGong-ST. As the AOL dataset is outdated and low-quality, later we only perform more detailed analyses of HSCM on TianGong-ST.

Table 6. Statistics for valid queries within sessions of different lengths and with different frequencies in **document ranking**.

	TianGong-ST			AOL dataset		
Session Length	#Queries*	# Unique Queries	Proportion	#Queries	# Unique Queries	Proportion
Short	12,734	3,383	77.92%	8,490	5,545	45.93%
Medium	3,347	1,398	20.48%	6,620	4,147	35.81%
Long	261	164	1.60%	3,376	2,194	18.26%
All	16,342	4,442	100%	18,486	10,712	100%
Query Frequency	#Queries	# Unique Queries	Proportion	#Queries	# Unique Queries	Proportion
$[0, 10)$	3,044	2,472	18.67%	11,347	9,725	61.38%
$[10, 100)$	4,688	1,653	28.69%	2,995	933	16.20%
$[100, 1000)$	5,830	296	35.67%	744	50	4.02%
$[1000, \infty)$	2,780	21	17.01%	3,400	4	18.39%
All	16,342	4,442	100%	18,486	10,712	100%

* For document ranking, a query owns at least one clicked document is regarded valid.

For session length, we compare HSCM with the two most competitive baselines, i.e., M-NSRF and CARS. While for query frequency, we compare HSCM with its variant without cross-session contexts to verify the effectiveness of the corresponding module. As revealed in Figure 5(a), HSCM outperforms M-NSRF and CARS across all session lengths, which indicates its robustness on session lengths. Surprisingly, HSCM performs worse in long sessions. This may be caused by the task difficulty as other baselines also achieve relatively lower MAP values in long sessions. In Figure 5(b), we find that although HSCM outperforms M-NSRF and its variant in all intervals, the improvement in the $[0, 10)$ interval is the most evident (a 7.95% improvement over its variant). The observation verifies that considering cross-session contexts can make a further step in alleviating the data sparsity problem. From these results, we can answer **RQ1** that HSCM performs better than existing ranking models in various conditions.

Table 7. Comparison of query suggestion performance of each model on our testing set, best results are given in bold. Note that ∗ denotes that compared to HSCM, the performance difference is statistically significant at a level of p < 0.01 using paired t-test. To conduct the significance test on HIT@k, we treat each item as a binary value.

Model	**TianGong-ST**				**AOL dataset**			
	MRR	HIT@1	HIT@3	HIT@5	MRR	HIT@1	HIT@3	HIT@5
Traditional Models								
MPS	0.4124*	0.1846*	0.5574*	0.7004*	0.8171	0.7187	0.8988	0.9463
Hybrid (α=0.5)	0.4283*	0.1876*	0.5900*	0.7364*	0.8200	0.7233	0.9027	0.9419
QVMM	0.4847*	0.3423*	0.5374*	0.6350*	0.7608	0.6433	0.8559	0.9122
Deep Models								
Seq2seq+Attn.	0.5149*	0.3348*	0.6064*	0.7483*	0.8181*	0.7238*	0.8896*	0.9430*
M-NSRF	0.5174*	0.3250*	0.6232*	0.7798*	0.7176*	0.5552*	0.8591*	0.9410*
HRED-qs	0.5460*	0.3773*	0.6365*	0.7738*	0.7067*	0.6331*	0.7230*	0.7514*
CARS	0.5350*	0.3526*	0.6326*	0.7788*	0.8395*	0.7448*	0.9246*	0.9638*
HSCM	**0.7103**	**0.5593**	**0.8290**	**0.9183**	**0.8978**	**0.8299**	**0.9591**	**0.9786**

6.2 Query Suggestion Evaluation

To answer **RQ2**, we evaluate the discriminative ability of all models, i.e., whether they can rank the intended queries higher given a candidate list. As revealed in Table 7, HSCM achieves significantly better performance than all baseline models on both datasets, where the improvements are surprisingly huge. Compared to MPS, the Hybrid method (perform the best at $\alpha = 0.5$, which denotes the importance of hybrid information) introduces co-occurrence dependencies into query representation thus realizes a certain improvement. Among traditional methods, QVMM performs the best on TianGong-ST. It learns the probability of query transitions with a suffix tree based variable memory Markov model thus can better model the relationships between consecutive queries. However, it seems not effective on the AOL sessions where the queries are relatively rarer. In this scenario, QVMM may be confronted with many unseen query sequences thus perform worse. Among all baselines, CARS outperforms others, this shows the effectiveness of multi-task learning and leveraging contextual information (especially click signals) for query suggestion. Finally, we can observe huge improvements in HSCM over all baselines on both datasets. Compared to CARS, it not only employs a content encoder that embeds each candidate with both sparse features (textual information) and dense features (co-occurrence frequency) but also exploits additional contexts (e.g., cross-session contexts and non-click signals) to better model user intent hence performs significantly better.

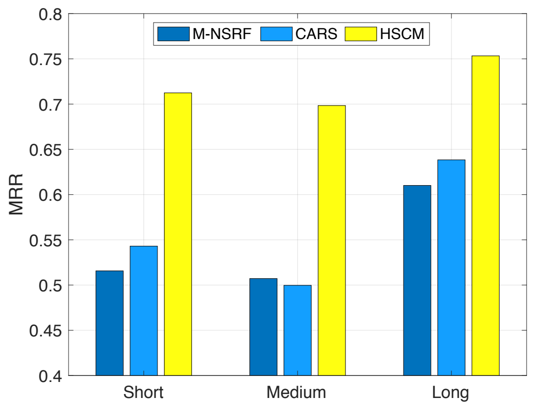
We further evaluate the finer-grained query suggestion performances for CARS, M-NSRF, and HSCM concerning various session lengths and query frequencies. Table 8 presents the number and proportion of valid instances across all conditions in query suggestion for the two datasets.

As shown in Figure 6(a), HSCM outperforms two other baseline models in query suggestion across sessions with different lengths. This again reveals the outstanding power of HSCM in predicting the next query. All of the three models perform better when a session is longer, which is consistent with the results reported in [2]. This can be explained by: more contextual information can assist these session-level models to infer user intents with less ambiguity, thus achieve better query suggestion performances. Furthermore, from Figure 6(b), we find that HSCM achieves the best MRR values in all frequency intervals. It can be easily observed that frequent queries amplify HSCM's performance on query suggestion, which is reasonable because adequate training data is beneficial for learning user intents. Similar to document ranking, HSCM also achieves a remarkable performance in sparse queries (i.e., the [0, 10) interval), which

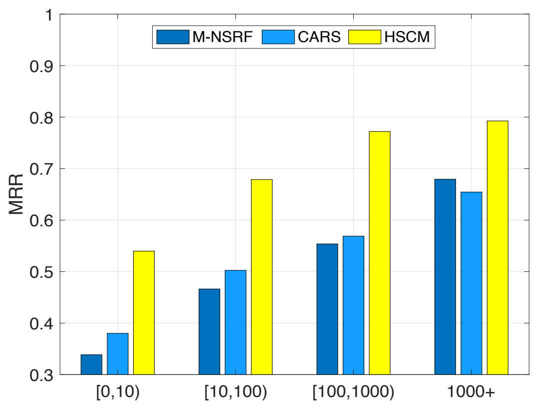
Table 8. Statistics for valid queries within sessions of different lengths and with different frequencies in **query suggestion**.

	TianGong-ST			AOL dataset		
Session Length	#Queries*	#Unique Queries	Proportion	#Queries	#Unique Queries	Proportion
Short	7,130	2,491	68.22%	948	370	21.64%
Medium	2,882	1,246	27.58%	1,532	543	34.98%
Long	439	196	4.20%	1,900	445	43.38%
All	10,451	3,103	100%	4,380	1,102	100%
Query Frequency	#Queries	#Unique Queries	Proportion	#Queries	#Unique Queries	Proportion
$[0, 10)$	1,732	1,460	16.57%	926	508	21.14%
$[10, 100)$	3,055	1,316	29.23%	1,210	529	27.63%
$[100, 1000)$	3,606	304	34.50%	560	61	12.79%
$[1000, \infty)$	2,058	23	19.69%	1,684	4	38.45%
All	10,451	3,103	100%	4,380	1,102	100%

* For query suggestion, a query that is within the candidate set of its previous query is regarded valid.



(a) Comparison on session lengths.



(b) Comparison on query frequencies.

Fig. 6. Comparison based on query suggestion w.r.t. session lengths and frequencies of the intended query on TianGong-ST.

confirms that the cross-session contexts can also help long-tailed queries in query suggestion. From the previous investigations, we can answer **RQ2** that HSCM significantly outperforms existing models in query suggestion in terms of both overall and finer-grained performances.

6.3 Click Prediction

In the previous sections, we have conducted a click-based evaluation for document ranking. To further evaluate the learning power of each model, we also compare the click prediction performances of them with some click models on the TianGong-ST dataset. Here we do not consider the AOL dataset because the original behavioral log is missing. The results are given in Table 9. We can obviously find that CACM performs best among all listed methods. Although performing worse than click models, HSCM is superior to other ranking models in both metrics, indicating its strong learning power. As all click models consider the ranking position of a document while ranking models do not, it is natural they can better predict user clicks by modeling the position bias.

Table 9. Comparison of click prediction performance of each model on TianGong-ST. Differences in click prediction between all pairs of click models are statistically significant (p < 0.001).

Model\ Metric	Perplexity (PPL)	Log-likelihood (LL)
Ranking Models		
KNRM	1.5420	-0.4330
HiNT	1.2425	-0.1978
M-NRSF	1.2333	-0.1924
CARS	1.2159	-0.1811
HSCM	1.2002	-0.1725
Click Models		
DBN	1.1953	-0.1765
UBM	1.1967	-0.1626
CACM	1.1781	-0.1500

Table 10. Ablation study on contextual information. Note that **C**, **H** and **I** are the abbreviations for cross-session contexts, query history and intra-session contexts, respectively.

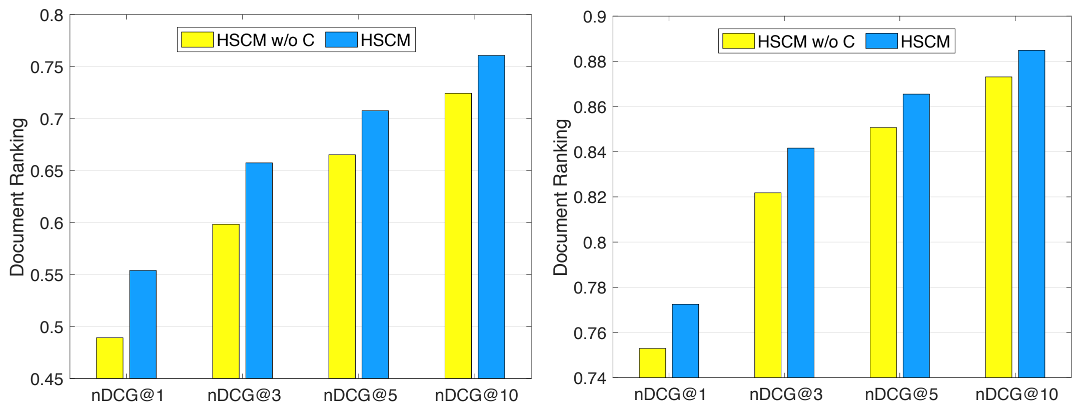
Document Ranking	nDCG@1	nDCG@3	nDCG@5	nDCG@10
HSCM	0.7755	0.8459	0.8681	0.8873
HSCM w/o **C**	0.7502(-3.26%)	0.8338(-1.43%)	0.8577(-1.20%)	0.8771(-1.15%)
HSCM w/o **C+H**	0.7491(-3.40%)	0.8319(-1.66%)	0.8561(-1.38%)	0.8760(-1.27%)
HSCM w/o **C+I**	0.7396(-4.63%)	0.8274(-2.19%)	0.8519(-1.87%)	0.8718(-1.75%)
HSCM w/o **C+N**	0.7375(-4.90%)	0.8236(-2.64%)	0.8488(-2.22%)	0.8699(-1.96%)
Query Suggestion	MRR	HIT@1	HIT@3	HIT@5
HSCM	0.7103	0.5593	0.8290	0.9183
HSCM w/o **C**	0.5872(-17.3%)	0.4132(-26.1%)	0.6963(-16.0%)	0.8214(-10.6%)
HSCM w/o **C+H**	0.5872(-17.3%)	0.4166(-25.5%)	0.6905(-16.7%)	0.8141(-11.3%)
HSCM w/o **C+I**	0.5852(-17.6%)	0.4136(-26.1%)	0.6900(-16.8%)	0.8085(-12.0%)
HSCM w/o **C+N**	0.5805(-18.3%)	0.4079(-27.1%)	0.6837(-17.5%)	0.8138(-11.4%)

6.4 Ablation Study and Analysis

To investigate the effectiveness of HSCM, we further conduct a series of ablation studies and comparison tests. In this subsection, we will sequentially answer **RQ3-RQ4** later.

6.4.1 Ablation study. To answer **RQ3**, we first alternatively wipe off three contexts: the cross-session interactions, the query history, and the intra-session interactions, from HSCM and then evaluate their performances. We do not wipe off the guiding query here due to its vital role in both tasks. For each context, we mask the corresponding component in Equation 13, respectively. As shown in Table 10, there are different degrees of performance degradation in both tasks when masking one of the three components, especially in document ranking. This implies that all of them facilitate user intent modeling. Later we will first delve into the effectiveness of the cross-session contextual information and then discuss the impact of the rest contexts.

First, we answer **RQ4** by verifying the effectiveness of the cross-session interaction aggregation module. Here we consider two kinds of queries for study: I) long-tailed queries; II) cold-start queries. For condition I, we select testing queries that occur less than 10 times in training sessions as the sparse queries and further compare the performances of HSCM with its variant that ignores the cross-session contexts in these queries. For condition II, we use all queries in testing sessions without any click signals in previous contexts (including the initial queries within a session). The results are shown in Figure 7 and Figure 8, respectively. On one hand, we find there are huge increases in nDCG values at all ranks by aggregating cross-session behaviors in both the long-tailed and cold-start conditions for document ranking, e.g., all the differences are significant at a level of $p < 0.01$ using paired t-test. Moreover, the improvements in long-tailed queries are more evident while the overall performances in the cold-start condition are much better. This may be because many cold-start queries are frequent enough in the training set for deep models to learn well, although there are not enough user clicks in the current session. These observations verify our assumption that sampling user behaviors in other sessions can largely boost ranking performance for these two kinds of queries in document ranking, especially for the long-tailed ones. On the other hand, there are larger promotions on cold queries for query suggestion (See Figure 8(b)). Similar to the ranking tasks, HSCM performs relatively worse on sparse queries due to the lack of training data. However, the difference is that, by incorporating the cross-session contexts, HSCM predicts the next query more accurately on cold queries. When users do not perform clicks on previous queries, it is hard for the system to capture user intents and further provide appropriate suggestions for them. In this scenario, utilizing the "wisdom of crowds" buried in the cross-session dependencies and integrating them into the local context may be a good option for user modeling.

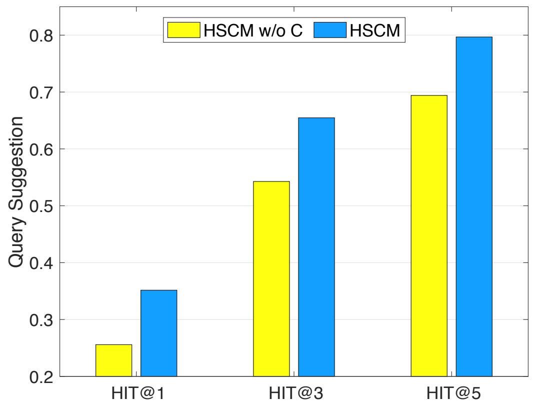


(a) Comparison in sparse queries (3,044 queries).

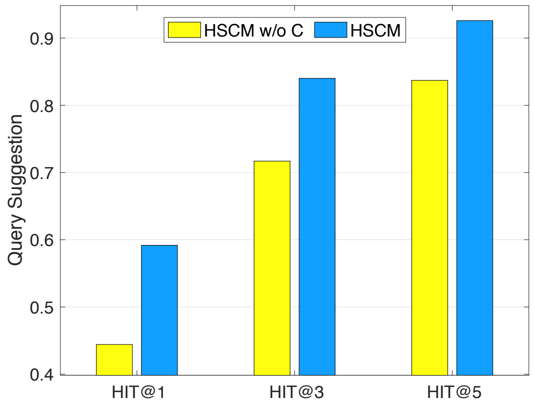
(b) Comparison in queries without previous clicks (10,266 queries).

Fig. 7. Ablation study for the cross-session module in document ranking on the TianGong-ST dataset.

For other contexts, we find that compared to query history, removing the intra-session interaction causes a greater decline in both tasks. Firstly, it indicates the importance of considering user interactive behaviors. Moreover, the guiding query in query suggestion, i.e., the last query, has already provided important information about the adjacent query. We find that most sessions in our data contain only two queries. Therefore, query history provides the same information as the guiding query in these sessions hence is not so essential for user modeling, especially in query suggestion (there are even small improvements at some metrics in query suggestion when ignoring query history). We also find that all the HSCM variants outperform CARS in query suggestion, which implies its effectiveness of inferring user intents by exploiting each session context. It is also worth mentioning that non-click information has the greatest impact on the overall performance compared to other intra-session contexts. By dispelling user non-click behaviors, we find that HSCM performs even worse than its variant whose interaction information is completely removed. This



(a) Comparison in sparse queries (1,732 queries).



(b) Comparison in queries without previous clicks (4,647 queries).

Fig. 8. Ablation study for the cross-session module in query suggestion on the TianGong-ST dataset.

shows that only considering user click behaviors will bring bias during session modeling and further decrease model effectiveness. However, most of the existing approaches ignore the impact of non-clicks and representing session-level embeddings with only clicked documents [2, 29, 31]. The introduction of learning bias may hurt the system performance to a certain extent.

6.4.2 Case study. To investigate the effectiveness of the cross-session module in HSCM, we list some examples of the sampled queries whose interactions are utilized for cross-session context modeling, in Table 11. We can easily observe that adding the similarity constraints can help extend queries with higher quality. From the column of $\theta = 0$, we find that there are many noisy queries (marked red in the background). These queries are searched through the co-click dependencies between sessions without considering the semantic similarities and may hurt the system performance by integrating the noisy interactions into the cross-session context representation. In addition, we find the problem is more serious for guiding queries with a large number of expanded nodes across all values of θ. This indicates the necessity of imposing some rules for sampling, especially for guiding queries with explosive extended nodes. As a result, if we add a similarity constraint, i.e., $\theta > 0$, then there will be more high-quality expansions (fewer red blocks appear in the third and fourth column of Table 11). Especially, the condition where $\theta = 0.8$ contains almost no noisy expansions. The model may even expand no new nodes for some queries (e.g., Amazon) to avoid hurting the system performance in this condition. To balance the trade-off between the expansion quality and number, we empirically select $\theta = 0.8$.

We further take document ranking as an example and conduct a case study in Table 12. We compare the ranking performance of HSCM with its variant without cross-session contexts and then select several cases where HSCM significantly performs better for analysis. In case one, we find that the long-tailed query "foreign exchange rate" is the first query within a session and it only occurs 14 times in the training sessions. Due to both the data sparsity and cold-start issues, HSCM without cross-session contexts only achieves a MAP value of 0.1667 in this case. However, when we search for similar extended queries and sample user behaviors on these queries as the augmented contexts, HSCM captures user clicks on URL1 in both the extended queries and outputs a higher relevance score for it (0.6796 compared to 0.1097). Therefore, it performs better by ranking the clicked document higher than before and achieves a

Table 11. Examples of sampled queries for cross-session modeling on TianGong-ST. The number of expanded query nodes for each guiding query is presented in the blankets in the first column. Noisy expansions are marked red in the background for each similarity threshold θ. Completely irrelevant queries are marked as " sample ", partially irrelevant queries are marked as " sample ".

Query (#Nodes)	Setting		
	θ=0	θ=0.5	θ=0.8
iphone (662)	second board calendar Xiangyang*	iphone assistant iphone old for new service iphone official	iphone assistant iphone old for new service
Attack on Titan(4)	*Attack on Titan comics* *Attack on Titan comics* chap.68 *Attack on Titan* complete works	*Attack on Titan* comics *Attack on Titan* comics chap.68 *Attack on Titan* live-action	*Attack on Titan* live-action *Attack on Titan* comics *Attack on Titan* comics chap.68
tgp Tencent Games Platform (1001)	*Voice*** online game weight rank The Roman Empire history	tga Tencent Games Platform Yaodou Games Platform tgp download official	tpg game platform lol† Tencent Games platform Tencent Games platform
Sanguosha§ online (21)	Sanguosha the web version 4399 Sanguosha	4399 Sanguosha the web version Sanguosha online desktop Sanguosha	Sanguosha Sanguosha Zhang Kun Sanguosha online desktop
Hearthstone (3888)	The game *AniPop* fortune-telling based on name Adobe+Photoshop+CS6	*Hearthstone*¶ Duowan dnf box *Hearthstone* box	*Hearthstone* new card *Hearthstone* decks *Hearthstone*¶
Amazon (26)	JD online shopping mall JD.com Amazon official	*Make Them Die Slowly*‡ Amazon official JD online shopping mall	None

*. A city in central China.
**. A TV programme.
†. League of Legends.
§. A card game based on the celebrities in the Three Kingdoms period of Chinese history.
¶. This query (炉石) is an abbreviation of the guiding query (炉石传说) in Chinese.
‡. A movie about the Amazonian cannibals.

MAP of 1.0. In case two, the query "*Tencent* Stock" is also rare in the training set, where *Tencent* is one of the world's largest Internet firms. Although it is not the initial query, there are no user clicks in its previous query "*Tencent*". In this condition, HSCM also aggregates the cross-session contexts where some users click on URL2 in the first extended query "Tencent+Stock". By ranking URL2 on the top with the largest predicted relevance score, HSCM also achieves an improved MAP value of 1.0. These users have similar information needs with the implicit user in the current query thus can provide useful behavioral signals. Through this case, we find that even without user profile features or social relationships as in recommendation, we can also leverage the cross-session contexts by sampling user behaviors on extended queries to improve the system performance. Results of the ablation study and case study have also answered **RQ3-RQ4** by showing the effectiveness of the cross-session aggregation module and other components in HSCM.

As HSCM achieves state-of-the-art performances in query suggestion on both datasets, we also investigate how the cross-session module benefits the system on TianGong-ST. We select the sessions that are largely improved after introducing the cross-session contexts and then screen out the ones where some key terms in the intended query appear in the extended queries but not in the query history. Several of these cases are listed in Table 13. Among all the improved cases, over 95% of them are short sessions, i.e., lack of session contexts, which again indicates the pivotal role of the cross-session module in predicting the next queries in short search sessions. Another finding is that both the co-click and co-semantic edges contribute to the query expansion and will further spur on the query suggestion performance. For example, to join the game "Sanguosha online", users need to first install the Adobe flash player plugin in their web browser to support the game platform to be normally presented. However, the two entities are not near

Table 12. A case study to investigate the effectiveness of the cross-session contextual information in document ranking on TianGong-ST. Note that valid documents denote the clicked documents ranked higher when considering cross-session contexts.

Case 1	**Query**	**Extended Queries**
	foreign exchange rate(外汇牌价)	1. Bank of China exchange rate (中国银行外汇牌价); 2. exchange rate Bank of China (外汇牌价中国银行).
	Valid Document(s)	**MAP Improvement**
	URL1="http://www.waihuipaijia.cn/"; Title1="Exchange rate list of Bank of China today (今日中国银行外汇牌价表)".	0.1667 ⇒ 1.0
Case 2	**Query**	**Extended Queries**
	Tencent Stock(腾讯股票)	1. *Tencent*+Stock (腾讯 + 股票); 2. *Tencent* Finance(腾讯财经); 3. *Tencent* Security(腾讯证券).
	Valid Document(s)	**MAP Improvement**

	URL2="http://quote.eastmoney.com/hk/00700.html";
	Title2="Share Price, Market, and Charts for Tencent Holdings (0700.HK) – eastmoney.com.
	(腾讯控股 (00700) 股票价格 _ 行情 _ 走势图—东方财富网)"
		0.3333 ⇒ 1.0

in the semantic space. The co-clicked edges between them can help for query expansion and further facilitate the system's query suggestion performance. We can observe that there are more extended queries highlighted in green in the third column of Table 13, suggesting the necessity of considering semantic relationships between queries, especially in sparse datasets (i.e., with few user clicks).

6.5 Parameter and efficiency

In this subsection, we will analyze the parameter size, convergence, and parameter sensitivity of HSCM.

6.5.1 Parameter size. To compare the robustness of each model in maintaining parameter sizes when using different sizes of vocabulary, we calculate the total number of trainable parameters for several competitive models. As presented in Figure 9, there are huge increases in parameter sizes for CARS and M-NSRF when building larger word dictionaries, e.g., CARS maintains 102.1M parameters when using the same vocabulary size as HSCM, a ninefold value compared to using 10k words. Moreover, when using more vocabulary, their training speed will also slow down due to heavy softmax computations over the huge vocabulary size in the log-likelihood function. The Seq2seq model with attention mechanism even suffers more from this problem because it contains several layers with similar computations, although it performs well in query suggestion. In contrast, HSCM does not depend on the log-likelihood function to infer the next predicted word neither needs to fine-tune the word embeddings during training, hence it is less likely to be influenced in terms of its parameter size and training speed when the vocabulary size grows. In many scenarios, a large word list is utilized to cover more query/document contents as possible. This alleviates the Out-of-vocabulary (OOV) problem and thus can boost the system performance. Therefore, HSCM will show its robustness in these scenarios. In this paper, we used BERT-base for document ranking on each dataset (Chinese and English, respectively). We can obviously find that the 110M parameters are about three folds of that in HSCM.

Table 13. A case study to investigate the effectiveness of the cross-session module in query suggestion on TianGong-ST. Note that valid terms denote the terms or phrases that appear in the extended queries but not in the query history, R' and R represent the predicted rank of the intended query within the candidate query list using HSCM without and with the cross-session module, respectively. Moreover, queries that are extended via the co-semantic edges are highlighted in green while the ones extended via the co-click edges are highlighted in blue.

Intended Query	Query History	Extended Queries	Valid Terms	R'	R
Guilin provident fund inquiry	provident fund	Liuzhou provident fund inquiry	inquiry	20	1
vegetable salad	what happens when you eat too much pomelo	yogurt fruit salad	fruit, salad	15	2
iPhone official website	iTunes official download	iPhone official website Chinese iPhone official website HongKong	official website, iPhone	13	3
Alibaba official website	Wangwang.taobao	Alibaba official login	official, Alibaba	15	3
iPhone 6 offer latest offer	Apple official, iPhone 6	iPhone 6 latest offer	latest, offer	12	3
Agents of S.H.I.E.L.D	Captain America II	Agents of S.H.I.E.L.D bar Agents of S.H.I.E.L.D II	Agents of S.H.I.E.L.D	9	1
adobe+flash+player	Sanguosha online	flash+player, adobe+flash	adobe, player, flash	10	2
Counter Strike	Cross Fire stand-alone	Counter Strike Online2, Counter Strike ol, Counter Strike 1.6	Counter Strike	12	5
Huitong	EMS	Best (Huitong) Express, Huitong Express	Huitong	8	2
My Love from the Star	List of South Korean dramas	Theme song of *My Love from the Star*	*My Love from the Star*	6	1

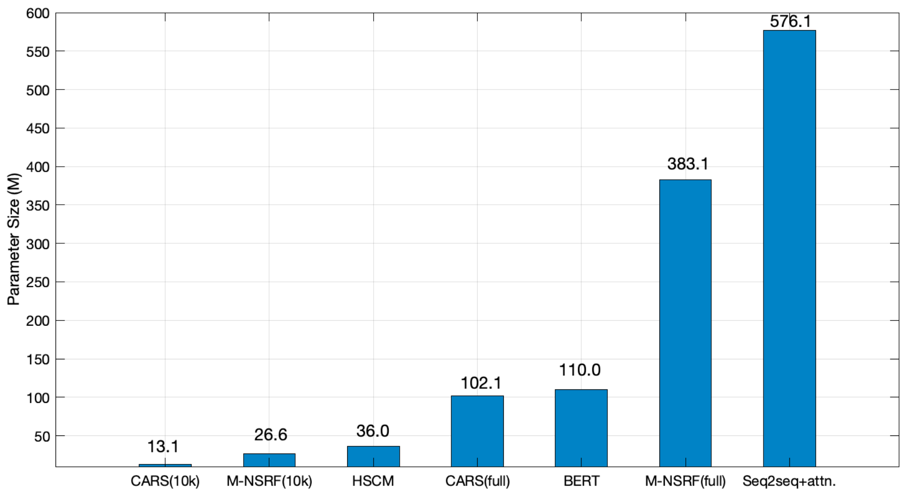
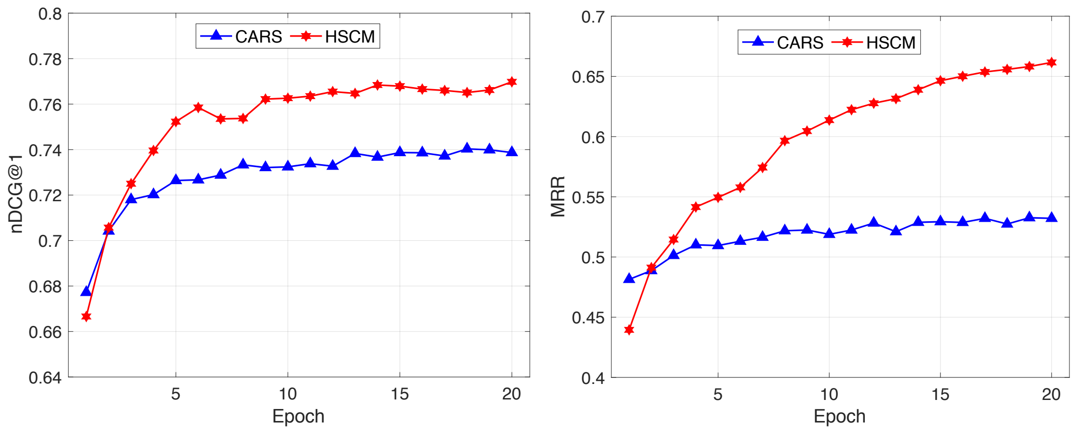


Fig. 9. Parameter size of each model. Although HSCM maintains a number of modules, it reuses several modules whose parameters are shared (e.g., intra-session and cross-session interaction aggregation modules). Therefore, the architecture is well-designed and the total parameter size is relatively small.

6.5.2 Convergence. To further investigate the efficiency of HSCM, we compare its learning performance on the validating set with that of CARS on the TianGong-ST dataset. To ensure fairness, we set both models with the same settings: initial learning rate = 10^{-3}, dropout rate = 0.2, weight decay = 10^{-6}, and both use the Adam optimizer. From the curves illustrated in Figure 10, we can infer that ceteris paribus, HSCM converges faster than CARS in both the document ranking and query suggestion tasks. In Figure 10(a), we find that HSCM achieves a considerable nDCG@1 value (0.7396) at the fourth epoch, which approximates the best result of CARS. This indicates that HSCM converges to the optimal system state much faster than CARS in terms of the document ranking. In query suggestion (see Figure 10(b)), CARS learns even more slowly because of its generative learning manner. It takes CARS about fourfold training epochs (epoch=17) to achieve comparable query suggestion performance as HSCM (epoch=4). A possible reason is that HSCM employs point-wise learning and trains both tasks in a discriminative manner, hence it can converge faster. The above investigations have indicated the efficiency of HSCM in convergence and its robustness in maintaining fixed-size parameters when considering different vocabulary sizes.

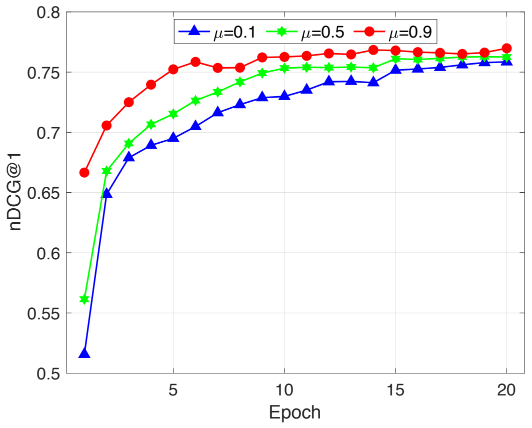


(a) Ranking performance at each epoch.

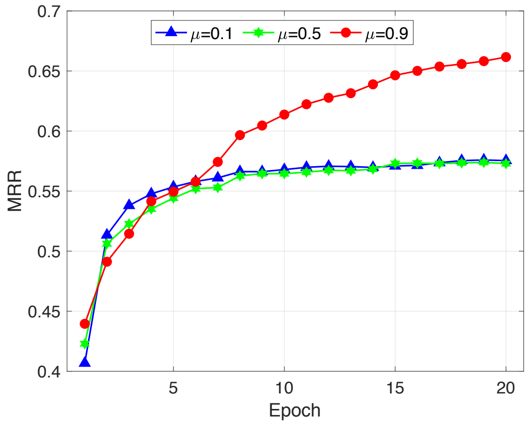
(b) Query suggestion performance at each epoch.

Fig. 10. Comparison of convergence speeds of HSCM and CARS on the validating sessions on TianGong-ST.

6.5.3 Parameter sensitivity. Here we aim to investigate the system performance on different parameter settings. Take the TianGong-ST dataset as an example, we select the weight of document ranking μ in $\{0.9, 0.5, 0.1\}$ and compare the corresponding system performances on the validating sessions within the first 20 epochs in both tasks, respectively. The results are shown in Figure 11. We can observe that for both tasks, HSCM not only achieves higher system performance but also converges faster than the other two conditions when $\mu = 0.9$. This finding is consistent with previous work that also assigns higher weights for document ranking, e.g., Ahmad et al. [2] also use a combination of $0.9/0.1$ as the learning weight for document ranking and query suggestion, respectively. The phenomenon implies that when training these two tasks via multi-task learning techniques, it is better to assign a much higher weight for the ranking task. A possible reason is that the supervision signals from the document ranker to the query recommender are more effective. On the contrary, assigning a larger weight for query suggestion (i.e., $\mu = 0.1$) does not necessarily improve the task performance. It is interesting to do more investigations about the dependencies between the two tasks when employing multi-task learning in the future.



(a) Document ranking performance on validating sessions.



(b) Query suggestion performance on validating sessions.

Fig. 11. Comparison of system convergence speeds across different values of the multi-task learning weight for document ranking μ on TianGong-ST.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a novel Hybrid framework for Session Context Modeling (HSCM). It employs the self-attention mechanism to build a session-level multi-task learning framework from scratch. Both document ranking and query suggestion are unified as discriminative tasks and are learned with intra-session and cross-session contextual information. An RNN-free content encoder is designed for encoding all textual contents into distributed representations at the session-level.

Extensive experiments are conducted on a large-scale practical session dataset. By answering several research questions, we conclude as follows: 1) HSCM significantly outperforms existing methods in document ranking and query suggestion in terms of both overall and finer-grained performances, e.g., sessions with different lengths and queries with various frequencies. Experimental results have shown its high effectiveness and robustness. 2) Intra-session contexts are beneficial for session-level user intent modeling. According to the ablation study, we find that user interactions and non-click data are relatively more important for model training. 3) By considering the cross-session contexts, HSCM performs significantly better in long-tailed queries or those with few previous contextual information with the session. We also instantiate how the cross-session contexts work in document ranking and query suggestion via two case studies.

Through the aforementioned investigations, we have a better understanding of HSCM, including its strengths and limitations. As HSCM largely alleviates the cold-start problem, it can be applied in various search scenarios that lack session contextual information besides the general Web search, e.g., image search [59] (also with few queries and sparse user clicks within sessions). However, for task-oriented search such as job search and product search, using the HSCM variant without the cross-session module may be effective enough as the contextual information such as queries or user interactions is sufficient for user intent modeling. For example, users may issue more queries with higher engagement in job search tasks and will click more results for comparison in order to find an ideal opportunity [37]. Moreover, job search is highly related to personal skills. While in product search, although users also submit few queries within a task, they would click more results to find a product with the best cost performance [50]. Therefore, intra-session contexts as well as the personalized information should be incorporated more for a better recommendation in job/product search. On the other hand, the limitations of HSCM may inspire our future work. For example, at this stage, HSCM indistinguishably aggregates the cross-session contexts for all queries. In the future, more intelligent techniques can be applied in the sampling process. Besides, how to balance the trade-off between utilizing intra-session contexts (for personalization) and cross-session contexts (for mitigating data-sparsity) also requires more in-depth investigations.

8 ACKNOWLEDGEMENTS

This work is supported by the National Key Research and Development Program of China (2018YFC0831700), Natural Science Foundation of China (Grant No. 61732008, 61532011, 61902209), Beijing Academy of Artificial Intelligence (BAAI), Tsinghua University Guoqiang Research Institute and Beijing Outstanding Young Scientist Program (NO. BJJWZYJH012019100020098). We also thank Jian-Yun Nie, Qingyao Ai, Xiang Wang and the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

[1] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2018. Multi-task learning for document ranking and query suggestion. In *International Conference on Learning Representations*.

[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*. 385–394.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[4] 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.

[5] 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*. ACM, 185–194.

[6] 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.

[7] 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.

[8] 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.

[9] Jia Chen, Yiqun Liu, Cheng Luo, Jiaxin Mao, Min Zhang, and Shaoping Ma. 2018. Improving session search performance with a Multi-MDP Model. In *Asia Information Retrieval Symposium*. Springer, 45–59.

[10] Jia Chen, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2019. Investigating Query Reformulation Behavior of Search Users. In *Information Retrieval*, Qi Zhang, Xiangwen Liao, and Zhaochun Ren (Eds.). Springer International Publishing, Cham, 39–51.

[11] 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.

[12] Jia Chen, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. A Context-Aware Click Model for Web Search. In *Proceedings of the Thirteenth ACM International Conference on Web Search and Data Mining*. ACM.

[13] Wanyu Chen, Fei Cai, Honghui Chen, and Maarten de Rijke. 2018. Attention-based hierarchical neural query suggestion. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 1093–1096.

[14] 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.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[16] 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.

[17] Yixing Fan, Jiafeng Guo, Yanyan Lan, Jun Xu, Chengxiang Zhai, and Xueqi Cheng. 2018. Modeling diverse relevance patterns in ad-hoc retrieval. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 375–384.

[18] Bruno M Fonseca, Paulo Braz Golgher, Edleno Silva de Moura, and Nivio Ziviani. 2003. Using association rules to discover search engines related queries. In *Proceedings of the IEEE/LEOS 3rd International Conference on Numerical Simulation of Semiconductor Optoelectronic Devices (IEEE Cat.*

No. 03EX726). IEEE, 66–71.

[19] Wei Gao, Cheng Niu, Jian-Yun Nie, Ming Zhou, Kam-Fai Wong, and Hsiao-Wuen Hon. 2010. Exploiting query logs for cross-lingual query suggestions. *ACM Transactions on Information Systems (TOIS)* 28, 2 (2010), 1–33.

[20] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. 2018. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA).* IEEE, 80–89.

[21] 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.

[22] 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.

[23] Qi He, Daxin Jiang, Zhen Liao, Steven CH Hoi, Kuiyu Chang, Ee-Peng Lim, and Hang Li. 2009. Web query recommendation via sequential query prediction. In *2009 IEEE 25th International Conference on Data Engineering.* IEEE, 1443–1454.

[24] 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.

[25] Chien-Kang Huang, Lee-Feng Chien, and Yen-Jen Oyang. 2003. Relevant term suggestion in interactive web search based on contextual information in query session logs. *Journal of the American Society for Information Science and Technology* 54, 7 (2003), 638–649.

[26] Jizhou Huang, Wei Zhang, Yaming Sun, Haifeng Wang, and Ting Liu. 2018. Improving Entity Recommendation with Search Log and Multi-Task Learning.. In *IJCAI.* 4107–4114.

[27] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management.* ACM, 2333–2338.

[28] Jyun-Yu Jiang, Yen-Yu Ke, Pao-Yu Chien, and Pu-Jen Cheng. 2014. Learning user reformulation behavior for query auto-completion. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval.* 445–454.

[29] 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.

[30] Weike Jin, Zhou Zhao, Mao Gu, Jun Yu, Jun Xiao, and Yueting Zhuang. 2019. Video Dialog via Multi-Grained Convolutional Self-Attention Context Networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 465–474.

[31] Thorsten Joachims. 1996. *A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization.* Technical Report. Carnegie-mellon univ pittsburgh pa dept of computer science.

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

[33] Nir Levine, Haggai Roitman, and Doron Cohen. 2017. An extended relevance model for session search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval.* 865–868.

[34] 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.

[35] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. (2015).

[36] 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.

[37] Jiaxin Mao, Damiano Spina, Sargol Sadeghi, Falk Scholer, and Mark Sanderson. 2019. Investigating the Learning Process in Job Search: A Longitudinal Study. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management.* 2461–2464.

[38] Kyosuke Nishida, Itsumi Saito, Atsushi Otsuka, Hisako Asano, and Junji Tomita. 2018. Retrieve-and-read: Multi-task learning of information retrieval and reading comprehension. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management.* ACM, 647–656.

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

[40] Umut Ozertem, Olivier Chapelle, Pinar Donmez, and Emre Velipasaoglu. 2012. Learning to suggest: a machine learning framework for ranking query suggestions. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval.* 25–34.

[41] Greg Pass, Abdur Chowdhury, and Cayley Torgeson. 2006. A picture of search. In *Proceedings of the 1st international conference on Scalable information systems.* 1–es.

[42] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation. (2014), 1532–1543.

[43] Parikshit Ram and Alexander G Gray. 2012. Maximum inner-product search using tree data-structures. *arXiv preprint arXiv:1202.6101* (2012).

[44] Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94.* Springer, 232–241.

[45] Bahar Salehi, Fei Liu, Timothy Baldwin, and Wilson Wong. 2018. Multitask Learning for Query Segmentation in Job Search. In *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval.* ACM, 179–182.

[46] Rodrygo LT Santos, Craig Macdonald, and Iadh Ounis. 2013. Learning to rank query suggestions for adhoc and diversity search. *Information Retrieval* 16, 4 (2013), 429–451.

[47] 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*. ACM, 43–50.

[48] Milad Shokouhi. 2013. Learning to personalize query auto-completion. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. 103–112.

[49] 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.

[50] Ning Su, Jiyin He, Yiqun Liu, Min Zhang, and Shaoping Ma. 2018. User intent, behaviour, and perceived satisfaction in product search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 547–555.

[51] Zhiwen Tang and Grace Hui Yang. 2019. Corpus-Level End-to-End Exploration for Interactive Systems. *arXiv preprint arXiv:1912.00753* (2019).

[52] Yuan Tian, Ke Zhou, Mounia Lalmas, Yiqun Liu, and Dan Pelleg. 2020. Cohort Modeling Based App Usage Prediction. (2020).

[53] 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*. 69–72.

[54] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[55] Bin Wu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2018. Query suggestion with feedback memory network. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 1563–1571.

[56] Wei Wu, Hang Li, and Jun Xu. 2013. Learning query and document similarities from click-through bipartite graph with metadata. In *Proceedings of the sixth ACM international conference on Web search and data mining*. 687–696.

[57] 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.

[58] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617* (2017).

[59] 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*. 1683–1692.

[60] 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*. ACM, 55–64.

[61] Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with bertserini. *arXiv preprint arXiv:1902.01718* (2019).

[62] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*. 1480–1489.

[63] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. 2018. Sequential recommender system based on hierarchical attention networks. In *the 27th International Joint Conference on Artificial Intelligence*.

[64] Fajie Yuan, Xiangnan He, Haochuan Jiang, Guibing Guo, Jian Xiong, Zhezhao Xu, and Yilin Xiong. 2019. Future Data Helps Training: Modeling Future Contexts for Session-based Recommendation. *arXiv* (2019), arXiv–1906.

[65] Sicong Zhang, Dongyi Guan, and Hui Yang. 2013. Query change as relevance feedback in session search. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. 821–824.

[66] Yuan Zhang, Dong Wang, and Yan Zhang. 2019. Neural IR Meets Graph Embedding: A Ranking Model for Product Search. In *The World Wide Web Conference*. ACM, 2390–2400.