Information Processing and Management xxx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

Hierarchical neural query suggestion with an attention mechanism[☆]

Wanyu Chen[*,1,a], Fei Cai[*,1,a], Honghui Chen[a], Maarten de Rijke[b]

[a] Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, Hunan, China
[b] Informatics Institute, University of Amsterdam, Amsterdam, the Netherlands

ARTICLE INFO

Keywords:
Neural methods for information retrieval
Query suggestion

ABSTRACT

Query suggestions help users of a search engine to refine their queries. Previous work on query suggestion has mainly focused on incorporating directly observable features such as query co-occurrence and semantic similarity. The structure of such features is often set manually, as a result of which hidden dependencies between queries and users may be ignored. We propose an Attention-based Hierarchical Neural Query Suggestion (AHNQS) model that uses an attention mechanism to automatically capture user preferences. AHNQS combines a session-level neural network and a user-level neural network into a hierarchical structure to model the short- and long-term search history of a user. We quantify the improvements of AHNQS over state-of-the-art recurrent neural network-based query suggestion baselines on the AOL query log dataset, with improvements of up to 9.66% and 12.51% in terms of Recall@10 and MRR@10, respectively; improvements are especially obvious for short sessions and inactive users with few search sessions.

1. Introduction

Modern search engines offer query suggestions to help users express their information needs effectively. Previous work on query suggestion, such as probabilistic models and learning to rank techniques, mainly relys on features indicating dependencies between queries and users, such as clicks and dwell time (Chen, Cai, Chen, & de Rijke, 2017). However, the structure of those dependencies is usually modeled manually. As a result, hidden relationships between queries and a user's behavior may be ignored. Recurrent neural network (RNN)-based approaches have been proposed to tackle these challenges. A query log can be treated as sequential data that can be modeled to predict the next input query. However, existing neural based methods only consider so-called current sessions (in which a query suggestion is being generated) as the search context for query suggestion (Onal et al., 2018).

Our *research goal* is to develop a neural query suggestion method that is able to capture the user's search intent by capturing both

[☆] A preliminary version of this paper appeared in the proceedings of SIGIR 2018 (Chen et al., 2018). In this extension, we (1) extend the neural query suggestion approach to model users' preference better by combining local (attention-based) and global session states; (2) investigate the performance of AHNQS with different session states, i.e., global vs. local vs. combined; (3) investigate the performance of our model with different numbers of users' search sessions, as we find that the majority of users in the AOL dataset only have a small number of sessions; and (4) include more related work and provide a more detailed analysis of the approach and experimental results.

[*] Corresponding authors.

E-mail addresses: wanyuchen@nudt.edu.cn (W. Chen), caifei@nudt.edu.cn (F. Cai), chenhonghui@nudt.edu.cn (H. Chen), derijke@uva.nl (M.d. Rijke).

[1] These authors contributed equally and are both corresponding authors.

https://doi.org/10.1016/j.ipm.2019.05.001
Received 17 October 2018; Received in revised form 10 March 2019; Accepted 2 May 2019
0306-4573/ © 2019 Elsevier Ltd. All rights reserved.

their short-term interests, as manifested during an ongoing search session, and their long-term interests, as manifested during earlier sessions. To this end we propose an AHNQS model that applies a user attention mechanism inside a hierarchical neural structure for query suggestion. The hierarchical structure contains two parts: a session-level RNN and a user-level RNN. The first captures queries in the current session and is used to model the user's short-term search context to predict their next query. The second captures the past search sessions for a given user and is applied to model their long-term search behavior to output a user state vector representing their preferences. We use the hidden state of the session-level RNN as the input to the user-level RNN; the user state of the latter is then used to initialize the first hidden state of the next session-level RNN.

In addition, we apply an attention mechanism inside the hierarchical structure of AHNQS that is meant to capture a user's preference towards different queries in a session. This addition is based on the assumption that different queries in the same session may express different aspects of the user's search intent (Bahdanau, Cho, & Bengio, 2015), e.g., queries with subsequent click behavior are more likely to represent the user's information need than those without. An attention mechanism can automatically assign different weights to hidden states of queries in the session-level RNN. The attentive hidden states together compose the session state, which we regard as a local session state. The local session state has the advantage of adaptively focusing on more important queries to capture users' main purpose in the current session. Besides, we also consider the final hidden state of the session-level RNN as a global session state, which acts as a vertical summary of the full sequence behavior. Then we use a combination of the global and local session state as the input for the user-level RNN.

We compare the performance of AHNQS against a state-of-the-art query suggestion baseline and variants of RNN-based query suggestion methods using the AOL query log. In terms of query suggestion ranking accuracy we establish improvements of AHNQS over the best baseline model of up to 9.66% and 12.51% in terms of Recall@10 and MRR@10, respectively. In addition, we investigate the impact on query suggestion performance of different session states, i.e., global vs. local vs. combined. The results show the effectiveness of the AHNQS model with the combined session state. Furthermore, we test the scalability of the AHNQS model across users with different numbers of sessions in their interaction history. The experimental results show that the performance of AHNQS is better than the best baseline model for users with varying degrees of activity.

Our contributions in this paper are:

1. We tackle the challenge of query suggestion in a novel way by proposing an Attention-based Hierarchical Neural Query Suggestion model, i.e., AHNQS, which adopts a hierarchical structure containing a user attention mechanism to better capture the user's search intent.
2. We analyse the impact of session length on query suggestion performance and find that AHNQS consistently yields the best performance, especially with short search contexts.
3. We examine the performance of AHNQS with different numbers of users' sessions. We find that AHNQS always yields better performance over the best baseline model, especially for users with few search sessions.

We describe related work in Section 2. Details of the attention-based hierarchical query suggestion model are described in Section 3. Section 4 presents our experimental setup. In Section 5, we report and discuss our results. Finally, we conclude in Section 6, where we also suggest future research directions.

2. Related work

Query suggestion can support users of search engines during their search tasks. A significant amount of work has gone into methods for formulating a better understandable query submitted by users (Cai et al., 2016; Cai & de Rijke, 2016a; 2016c; Smith, Gwizdka, & Feild, 2017; Vidinli & Ozcan, 2016a; 2016b). In recent years, deep learning techniques have been applied to a range of information retrieval tasks, often leading to a better understanding of user's search behavior (Borisov, Wardenaar, Markov, & de Rijke, 2018; Onal et al., 2018). In this section, we summarize traditional query suggestion methods in Section 2.1 and neural methods for query suggestion in Section 2.2.

2.1. *Traditional query suggestions methods*

Query suggestion methods that only rely on query co-occurrence are not able to satisfy all users in all contexts as they always provide the same list of suggestions to different users (Cai and de Rijke, 2016b; Chen et al., 2017). Thus, search query logs are an important resource to mine different users' search behavior. A query log can be partitioned into query sessions, i.e., sequences of queries issued by a unique user within a short time interval. Previously submitted queries may provide useful search context to reduce ambiguity of the current query and to produce more focused suggestions (Jiang, Ke, Chien, & Cheng, 2014). He et al. (2009) propose a context-aware method that uses a Variable Memory Markov model (QVMM) and builds a suffix tree to model the user query sequence. The method proposed by Cao et al. (2008) is similar but they build a suffix tree on clusters of queries and model transitions between clusters. However, for both methods, the number of parameters increases with the depth of the tree, which naturally leads to sparsity. Instead, our model can consider a flexible length of contexts with a fixed number of parameters. In addition, Santos, Macdonald, and Ounis (2013) and Ozertem, Chapelle, Donmez, and Velipasaoglu (2012) focus on learning to rank approaches for query suggestion. However, those approaches are trained with pairwise features, which cannot consider previous queries effectively.

Other work mainly focuses on personalized query suggestion methods based on a query-URL bipartite graph, which is constructed from click data with one type of vertex corresponding to queries and another type corresponding to URLs. Such query suggestion methods typically use a click graph representing the information flow in query logs with a Markov random walk model (Cui et al., 2011; Torres, Hiemstra, Weber, & Serdyukov, 2014). For instance, Ma, Yang, King, and Lyu (2008) develop a two-level query recommendation method based on two bipartite graphs (user-query and query-URL bipartite graphs) extracted from click data. Li, Yang, Liu, and Kitsuregawa (2008) use the connectivity of a query-URL bipartite graph through a novel two-phrase algorithm to recommend relevant queries that can improve the effectiveness of personalized query recommendation. Mei, Zhou, and Church (2008) propose a personalized query suggestion method by employing hitting time and creating pseudo query nodes in a click graph. Click graph-based approaches cannot extract features of users' long-term and short-term search histories, and thus fail to combine different types of user behavior effectively.

Unlike the work listed above, our method uses a session-level RNN and a user-level RNN to model a user's short-term and long-term search history, respectively, thereby extracting sequential behavior features in an effective way. In addition, we integrate this two level RNN structure into a hierarchical architecture to combine short-term and long-term search behavior simultaneously. An attention mechanism is applied inside the hierarchical structure to capture the user's search intent.

2.2. *Neural networks in query suggestion*

Neural networks have been applied in a variety of tasks in Information Retrieval, ranging from document ranking, and query understanding to user modeling (Borisov, Markov, de Rijke, & Serdyukov, 2016; He et al., 2018; Huang et al., 2013; Li, Ren, et al., 2017; Onal et al., 2018; Shen, He, Gao, Deng, & Mesnil, 2014). To some extent, we share similar targets with Borisov et al. (2016), who use distributed representations to model user browsing behavior in web search with a neural click model. We represent a user's short-term and long-term search behavior as a sequence of vector states that can describe a user's search intent. This kind of representation is richer than traditional methods and thereby enables us to capture more complex patterns of user search behavior.

Similar work has been done on neural query auto completion (QAC), i.e., a special type of query suggestion that suggests queries according to the input query prefix. Deep learning-based QAC models can be categorized into two groups: semantic models with convolutional neural networks and language models with recurrent neural networks (Mitra & Craswell, 2015; Park & Chiba, 2017). For RNN-based models, the probability for predicting the completed query is related to the length of the query, consequently favoring short query predictions (Fiorini & Lu, 2018). Using the RNN and CNN-based features developed in the work mentioned, a learning to rank-based method has been shown to achieve state-of-the-art performance on the neural query auto completion task, but with high computational costs (Fiorini & Lu, 2018).

In addition, Sordoni, Bengio, Vahabi et al. (2015) propose a hierarchical recurrent encoder-decoder model for generative query suggestion. The authors formulate a novel hierarchical neural network architecture and use it to produce query suggestions. It differs from our work in two important ways. On the one hand, it incorporates the neural query suggestion model as a feature into a learning to rank approach and thus belongs to the feature engineering approaches. On the other hand, their approach only considers users' short-term search history and, hence, it ignores long-term search behavior.

Another line of work similar to ours is due to Quadrana, Karatzoglou, Hidasi, and Cremonesi (2017), who propose a neural hierarchical session-based approach for recommender systems. In particular, the authors extend previous RNN-based session modeling with an additional RNN level that models the user activity across sessions and the evolution of their interests over time. We apply an attention mechanism inside the hierarchical structure that can capture a user's preference towards certain queries over others in a session. In addition, we propose a combined session state to capture both the user's sequential behavior and their main purpose in the current session, which we show to be useful in improving the query suggestion performance.

Compared to our previous work (Chen, Cai, Chen, & de Rijke, 2018), where we proposed an attention-based hierarchical neural query suggestion model (AHNQS) and quantified the improvement of AHNQS against the state-of-the-art baselines, we make new contributions in four directions. First, we combine the local (attention-based) states and the global session states to model users' preferences. Second, we investigate the performance of AHNQS with different session states to verify the utility of this combination (local and global session states). Third, we examine the performance of AHNQS with different numbers of users' search sessions, as we find that the majority of users in the AOL dataset only have a small number of sessions. Finally, we include more related work and provide a more detailed analysis of our experimental results.

3. Approach

Before introducing the AHNQS model, we introduce a neural query suggestion (NQS) model with session-level RNNs, and a hierarchical neural query suggestion (HNQS) model with hierarchical user-session RNNs.

3.1. *Session-level RNNs for query suggestion*

As in Sordoni et al. (2015), session-level RNNs are our starting point. In the neural-based query suggestion model (NQS), queries in the current session are taken as sequential input and used to output the probability of being the next query for the query suggestion candidates.

Formally, we assume that a query session $session_t$ contains N_t queries, denoted as $session_t = (q_{1,t}, q_{2,t}, q_{3,t}, ..., q_{N_t,t})$. As shown in Fig. 1, for generating the input vector of the network, we use a 1-of-N encoding of q_i, i.e., the vector length equals the number of unique queries V and only the coordinate corresponding to the ith query is one, the others are zero. We choose to use the Gated

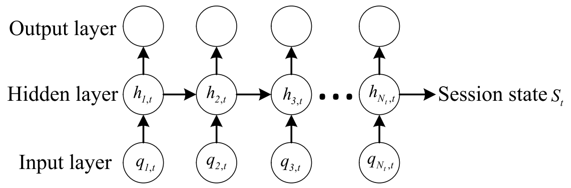


Fig. 1. Structure of the NQS model.

Recurrent Unit (GRU) (Cho, van Merrienboer, Gülçehre et al., 2014) as our non-linear transformation. The hidden state h_n can be calculated by using the previous hidden state h_{n-1} and the candidate update state $\hat{h}_n$:

$$h_n = (1 - u_n)h_{n-1} + u_n\hat{h}_n, \tag{1}$$

where the update gate u_n can be generated by:

$$u_n = \sigma(I_u q_{n,t} + H_u h_{n-1}). \tag{2}$$

The candidate update state $\hat{h}_n$ is calculated by:

$$\hat{h}_n = \tanh(Iq_{n,t} + H(r_n \cdot h_{n-1})), \tag{3}$$

where the reset gate r_n is:

$$r_n = \sigma(I_r q_{n,t} + H_r h_{n-1}), \tag{4}$$

where $I, I_u, I_r \in \mathbb{R}^{d_h \times V}$, $H, H_u, H_r \in \mathbb{R}^{d_h \times d_h}$, and d_h is the number of dimension of the hidden state; $\sigma(\cdot)$ denotes the sigmoid function. The H matrices are used to keep or forget the information in h_{n-1}. Finally, $q_{n,t}$ is the representation of the nth query in $session_t$.

We use $RNN_{session}$ and RNN_{user} to denote the GRU function. The final hidden state of a session-level RNN is used to indicate the session state, $S_t = h_{N_t,t}$. The output of the session-level RNN are the scores of query suggestion candidates being predicted as the next query:

$$s_{n,t} = g(\mathbf{W}_i h_{n,t}), \tag{5}$$

where $g(\cdot)$ is the activation function of the output layer, which can be either a softmax or a tanh , depending on the loss function of the neural network. In addition, $\mathbf{W}_i \in \mathbb{R}^{V \times d_h}$ is a parameter that can keep the dimension of the output equal to the number of queries. Thus we can generate a list of query suggestions in the test set according to the scores of query candidates.

We choose a pairwise loss function that forces positive query suggestion samples to be ranked higher than negative ones. There are several pairwise ranking loss functions, including cross-entropy and TOP1 (Hidasi, Karatzoglou, Baltrunas, & Tikk, 2016). In the field of recommender systems, TOP1 has proved to outperform others, so we set:

$$Loss = \frac{1}{N_S} \cdot \sum_{j=1}^{N_S} \sigma(s_{j,t} - s_{i,t}) + \sigma(s_{j,t}^2), \tag{6}$$

where $s_{j,t}$ and $s_{i,t}$ denote the score of a negative query candidate and a ground truth query, respectively, and N_S is the number of negative sampled query candidates, which are the queries that a user does not submit or input; as before $\sigma(\cdot)$ denotes the sigmoid function.

3.2. Hierarchical user-session RNN for query suggestion

Clearly, the NQS model only models the short-term search context. Next, we model the long-term search behavior of a given user with a user-level RNN, thereby producing the hierarchical NQS model (HNQS).

We assume that a user u has N_u query sessions, which is denoted as: $u(session) = (session_{1,u}, session_{2,u}, \ldots, session_{N_u,u})$. In a user-level RNN, the input is the session state S_t and the hidden state in user-level RNN can be calculated as:

$$h_{n,u} = RNN_{user}(h_{n-1,u}, S_{n,u}), \tag{7}$$

where $S_{n,u}$ is the session state of the nth query session of user u, which is equal to the last hidden state of the session-level RNN.

As shown in Fig. 2, we use the final hidden state of the user-level RNN to denote the user state, $U_t = h_{t,u}$, that contains the information about the search behavior from a user's past sessions and thus can be applied in the session-level RNN. In HNQS, the session-level RNN is initialized with a user state as follows:

$$h_{0,t} = \tanh(\mathbf{W} \cdot U_{t-1} + \mathbf{b_0}). \tag{8}$$

Updating:

$$h_{n,t} = RNN_{session}(h_{n-1,t}, q_{n,t}). \tag{9}$$

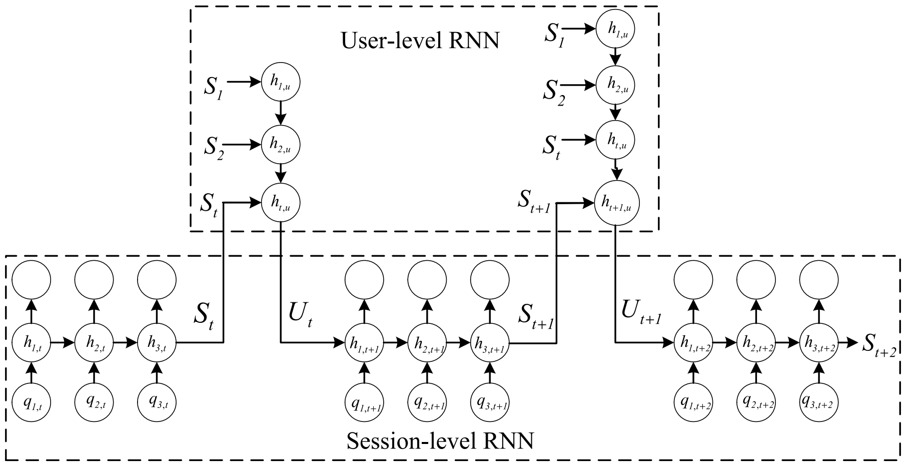


Fig. 2. Structure of the HNQS model.

Output:

$$s_{n,t} = g(\mathbf{W}_i h_{n,t}). \tag{10}$$

With this initialization strategy, the user's short-term search intent can be incorporated with his general preference. We choose to use only the initialization strategy for session-level RNN with user state U_{t-1}, instead of transporting the user information from U_{t-1} throughout the whole session-level RNN including initialization, updating and output. The GRU unit has both long and short term memory (Hochreiter & Schmidhuber, 1997) and can automatically transport the user state information within the network, which leads to a better performance when combined with our initialization strategy. If we choose to transport the user state throughout the whole session-level RNN, the user state information will be overloaded, which limits the performance of the model.

3.3. Attention-based hierarchical RNN for query suggestion

We assume that submitted queries that trigger subsequent click behavior have a better expression of the user's search intent. We hypothesize that queries in a session should have different weights to reflect the user's information need and employ an attention mechanism on top of the HNQS model to capture the user's preference for different queries in a session and then aggregate the representations of informative queries.

Fig. 3 shows how we update the user-level RNN in AHNQS with an attention mechanism as follows:

$$h_{t,u} = RNN_{user}(h_{t-1,u}, C_t), \tag{11}$$

where C_t is the attentive representation of the session state, a weighted sum of the hidden states $h_{j,t}$ from the session-level RNN, which is generated as:

$$C_t = \sum_{j=1}^{N_t} \alpha_{jt} h_{j,t}, \tag{12}$$

where α_{jt} is the normalized attention score for the jth query in session $session_t$, which is interpreted as the contribution of the query to the preference of the user:

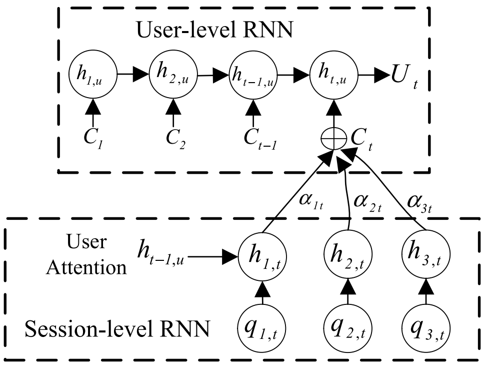


Fig. 3. The attention mechanism in the AHNQS model.

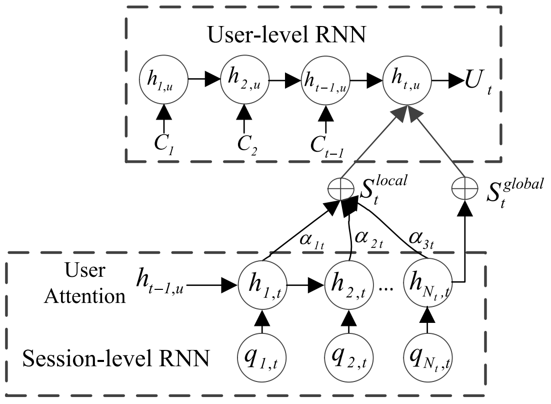


Fig. 4. Structure of the AHNQS model with combined session state.

$$\alpha_{jt} = \frac{\exp(e_{jt})}{\sum_{k=1}^{N_t} \exp(e_{kt})}, \tag{13}$$

where $e_{jt} = h_{t-1,u}^{\mathrm{T}} \mathbf{W}_a h_{jt}$ is the initial attention score computed with user state $h_{t-1,u}$ and hidden state h_{jt} in a session-level RNN. The parameters $\mathbf{W}_a$ can be jointly trained with the other components of AHNQS as the attention mechanism allows the gradient of the loss function to be backpropagated.

For the task of session-based query suggestion, the final hidden state of the session-level RNN can express a summary of the whole sequence behavior, which is called the *global session state*. As the attentive representation of the session state can adaptively select the important items in the current session to capture the user's main purpose, we regard C_t in (11) as the local session state. While the global session state indicates users' current search intents explicitly, the local session state denotes users' general preferences towards the queries in the current session. Thus, we combine the global and local session state by concatenating them to form an extended representation of a session as shown in Fig. 4.

We denote the global session state as $S_t^{global} = h_{N_t,t}$ and the local session state as $S_t^{local} = C_t$. Then, the combined session state can be generated by

$$S_t^{comb} = [S_t^{global}; S_t^{local}]\mathbf{W}_b^{\mathrm{T}} = \left[h_{N_t,t};\ \sum_{j=1}^{N_t} \alpha_{jt} h_{j,t} \right] \mathbf{W}_b^{\mathrm{T}}, \tag{14}$$

where $\mathbf{W}_b$ is used to keep the dimension of S_t^{comb} the same as S_t^{global} and S_t^{local}. Then, the hidden states in the user-level RNN in Eq. (12) can be updated with the combined session state as:

$$h_{t,u} = RNN_{user}(h_{t-1,u}, S_t^{comb}), \tag{15}$$

So far, we have developed the AHNQS model, which combines a hierarchical user-session RNN and an attention mechanism for query suggestion; the hierarchical structure models the user's short and long-term search behavior, while the attention mechanism captures the user's search preference.

The training process of AHNQS with combined session states is outlined in Algorithm 1.

We first initialize the parameters in the session-level RNN and user-level RNN in step 1. Then, for a session-level RNN, we initialize the first hidden state in $session_t$ with the former user state U_{t-1} in step and produce the hidden states in step 7. The prediction score for the next query is generated in step 8 and the loss of the whole network is calculated in step 9, which can be used during backpropagation to optimize the parameters. As for the user-level RNN, we calculate the attention weights with step 12 and step 13. Then we generate the combined session state S_t^{comb} for $session_t$ from step 14 and step 16. Finally, the tth hidden state $h_{t,u}$ and user state U_t in the user-level RNN are calculated in step 17 and step 18.

4. Experiments

We conduct our experiments on the AOL dataset to examine the effectiveness of AHNQS. We first list the research questions and the models used for comparison. After that, the datasets and experimental setup are described.

4.1. Research questions

We conduct experiments with the aim of answering the following research questions:

(RQ1) Do the hierarchical structure and attention mechanism incorporated in AHNQS help to improve the performance of the neural query suggestion model and outperform the state-of-the-art?

```
Input:  Epoches: training iterations;
        $U$: user set;
        $u(session)$: the session set of user $u$;
        $session_t$: the t-th session in $u(session)$;
        $query_{i,t}$: the i-th query in $session_t$;
        $N_t$: the number of queries in $session_t$;
        $N_S$: the number of negative sampled query candidates;
Output: the optimized parameters in session-level RNN and user-level RNN.
 1: randomly initialize the parameters and the hidden states in session-level RNN and user-level RNN.
 2: for epoch in range(Epoches) do
 3:   for $u \in U$ do
 4:     for $session_t \in u(session)$ do
 5:       for $query_{i,t} \in session_t$ do
 6:         $h_{0,t} = \tanh(\mathbf{W} \cdot U_{t-1} + \mathbf{b_0})$;   %% initialization strategy for session-level RNN
 7:         $h_{i,t} = RNN_{session}(h_{i-1,t}, q_{i,t})$;
 8:         $s_{i,t} = g(\mathbf{W}_i h_{i,t})$;   %% prediction score of next query
 9:         $Loss = \frac{1}{N_S} \cdot \sum_{j=1}^{N_S} \sigma(s_{j,t} - s_{i,t}) + \sigma(s_{j,t}^2)$;   %% loss calculation
10:         use back propagation to optimize the parameters.
11:       end for
12:       $e_{jt} = h_{t-1,u}^{\mathrm{T}} \mathbf{W}_a h_{jt}$;
13:       $\alpha_{jt} = \frac{\exp(e_{jt})}{\sum_{k=1}^{N_t} \exp(e_{kt})}$;   %% attention mechanism
14:       $S_t^{local} = \sum_{j=1}^{N_t} \alpha_{jt} h_{j,t}$;
15:       $S_t^{global} = h_{N_t,t}$;
16:       $S_t^{comb} = [S_t^{global}, S_t^{local}] \mathbf{W}_b^{\mathrm{T}}$;
17:       $h_{t,u} = RNN_{user}(h_{t-1,u}, S_t^{comb})$;
18:       $U_t = h_{t,u}$;
19:     end for
20:   end for
21: end for
22: return the parameters in session-level RNN and user-level RNN.
```

Algorithm 1. Attention-based Hierarchical Neural Query Suggestion.

(**RQ2**) What is the impact on query suggestion performance of session length, i.e., short vs. medium length vs. long sessions?
(**RQ3**) What is the impact on query suggestion performance of different session states, i.e., global vs. local vs. combined?
(**RQ4**) How does the performance of AHNQS vary across users with different numbers of search sessions?

4.2. Model summary

As AHNQS is based on a neural network to capture the dependencies between queries and users, we compare it with the state-of-the-art neural models for query suggestion. As an aside, Sordoni et al. (2015) incorporate a neural query suggestion model as a feature into a learning to rank approach and, hence, their approach belongs to the feature engineering approaches we do not compare with. We consider the following baselines for comparison:

ADJ original co-occurrence-based query suggestion method (Huang, Chien, & Oyang, 2003);
NQS a simple session-based RNN method for query suggestion (Hidasi et al., 2016), Section 3.1;
HNQS a hierarchical structure with user-session-level RNN for query suggestion, Section 3.2.

In addition, we consider two variants of our attention-based hierarchical neural query suggestion model:

AHNQS$_{local}$ an attention-based hierarchical RNN model for query suggestion, using the local session state S_t^{local}, Section 3.3;
AHNQS$_{combined}$ an attention-based hierarchical RNN model for query suggestion, using the combined session state $S_t^{combined}$, Section 3.3.

We list all the models to be compared in Table 1. It should be noticed that if we only use a global session state to update the user-

Table 1
An overview of models discussed in the paper.

Model	Description	Source
ADJ	An original co-occurrence-based query suggestion method.	Huang et al. (2003)
NQS	A simple session-based RNN method for query suggestion.	Hidasi et al. (2016), Section 3.1
HNQS	A hierarchical structure with user-session-level RNN for query suggestion.	Quadrana et al. (2017), Section 3.2
AHNQS$_{local}$	An attention-based hierarchical RNN model for query suggestion, using the local session state S_t^{local}.	This paper, Section 3.3
AHNQS$_{combined}$	An attention-based hierarchical RNN model for query suggestion, using the combined session state $S_t^{combined}$.	This paper, Section 3.3

level RNN, it is the same as the HNQS model. Thus, in the experiments on which we report below, we use HNQS to denote the model that only has a global session state.

4.3. Datasets and experimental setup

4.3.1. Dataset

We use the AOL query log and preprocess the dataset following Guo, Cheng, Xu, and Zhu (2011). Queries are separated into sessions by 30 minutes of inactivity. We remove queries with less than 20 occurrences and keep sessions whose length is larger than 5 as well as users with at least 5 sessions to provide sufficient user-session information. The training set consists all but the last 30 days in the search history; the test set consists of the last 30 days in the log after filtering out queries that do not exist in the training set. Table 2 details the statistics of the dataset used.

4.3.2. Parameter settings

We use GRUs as the RNN units and optimize the neural models using the TOP1 loss function and AdaGrad with momentum for 20 epochs. The number of hidden units is set to 100 in all cases and we use dropout regularization. We optimize the hyperparameters by running 100 experiments at randomly selected points of the parameter space. Optimization is done on a validation set, which is partitioned from the training set with the same procedure as the test set. We summarize the best performing parameters in Table 3.

In addition, in order to answer **RQ4**, we plot distributions of users with different numbers of sessions in the AOL dataset in Fig. 5.

The x-axis denotes the number of sessions while the y-axis indicates the number of users corresponding to the sessions. We see that although the maximum number of sessions that a user has is more than 150, the majority of users in the dataset only have a small number of sessions, which we regard as “inactive users.” In detail, 60.49% of the users have fewer than 8 sessions, 32.33% from 8 to 20 sessions, and only 7.18% of the users have more than 20 sessions. Thus, it is meaningful to investigate how the performance of the AHNQS models varies with different numbers of users’ sessions.

4.3.3. Training and evaluation

Research in natural language processing tasks often uses a fixed number of sequential words in a sentence to form a session and then put those sessions next to each other to form mini-batches. However, in query suggestion, the lengths of sessions are different and our goal is to capture how a session evolves over time, thus it is not suitable to set a fixed length for sessions. Besides, different users also have different numbers of sessions and we need to deal with sessions of the same user in order. Hence, we use parallel mini-batches with the identifications of users and sessions following Hidasi et al. (2016). We first group sessions by users and then sort sessions within each group by time. Then we order the users at random. Next, the first item of the first session of the first N users constitute the first mini-batch. The second item in each session server as the output for the first mini-batch and also constitute the next mini-batch. When a session ends, its next session of the user comes. When the sessions of a user have been processed completely, the next user is put in its place in the next mini-batch.

During training, we apply dropout in three ways: the calculation of the hidden states in the session-level RNN, the calculation of the hidden states in the user-level RNN, and the initialization for the first hidden state in the session-level RNN with a user state.

We evaluate the models by providing queries in a session one by one and measure the ranking performance of query suggestions with MRR and Recall on the test set.

Table 2
Dataset statistics.

Variable	Training	Test
# Queries	1,545,543	576,817
# Unique queries	61,641	33,519
# Sessions	166,414	67,716
# Users	23,308	19,255
Average # queries per session	9.28	8.52
Average # sessions per user	7.14	3.52

Table 3
Parameters used for each model.

Model	Batch	Dropout	Learning rate	Momentum
NQS	50	0.5	0.01	0.0
HNQS	50	0.1	0.1	0.0
AHNQS	50	0.1	0.1	0.0

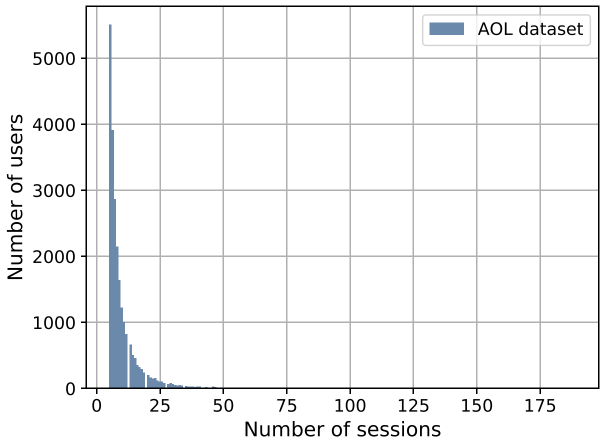


Fig. 5. Distribution of users with varying numbers of sessions in the AOL dataset.

5. Results and discussion

5.1. Performance of query suggestion models

To answer **RQ1**, we examine the query suggestion performance of the baselines as well as the AHNQS$_{local}$ and AHNQS$_{combined}$ models. Table 4 presents the results.

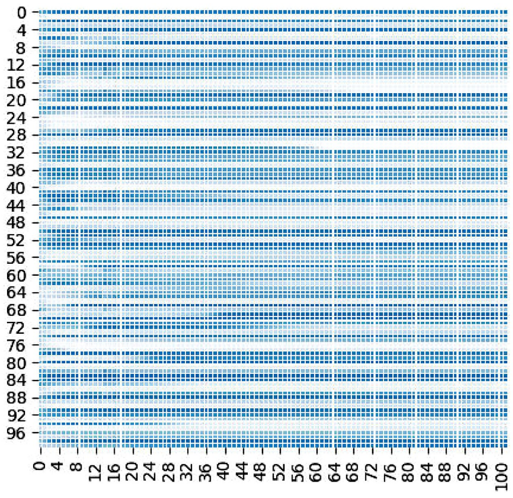
As shown in Table 4, amongst the baselines, ADJ outperforms NQS, with 9.74% and 12.58% improvements in terms of Recall@10 and MRR@10, respectively. This may be due to the fact that the NQS model (without knowing about individual users) fails to capture information from the past search history. HNQS shows improvements over ADJ of up to 15.07% and 13.75% in terms of Recall@10 and MRR@10, respectively. This demonstrates that the hierarchical structure can effectively incorporate a given user's previous search behavior and then improve the accuracy. Thus, in general, HNQS performs best among the baselines we consider and, hence, it is used as the baseline of choice in our later experiments.

Regarding our newly introduced models, the AHNQS models display a competitive performance compared to HNQS and ADJ. In particular, AHNQS$_{local}$ outperforms ADJ by 21.86% in terms of Recall@10 and 22.99% in terms of MRR@10, and beats HNQS by 5.9% in terms of Recall@10 and 8.13% in terms of MRR@10, respectively. The improvements are significant when tested with a *t*-test at the $\alpha = .01$ level. These findings indicate that attention can strengthen the model's ability to rank query suggestion candidates effectively. The best performance is obtained by AHNQS$_{combined}$; its improvements over AHNQS$_{local}$ are 3.55% in terms of Recall@10 and 4.05% in terms of MRR@10, respectively. This can be attributed to the utility of the combined session state.

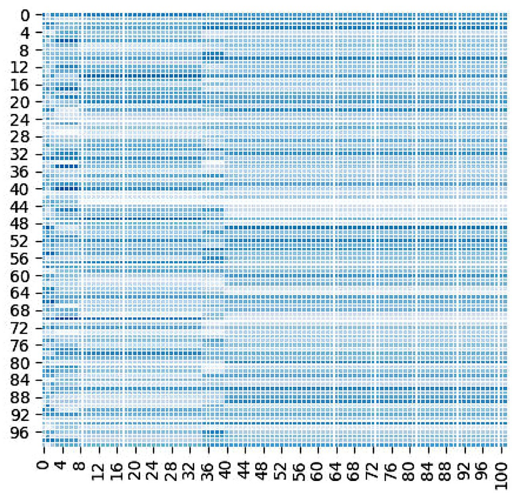
Interestingly, we see that the improvements of our AHNQS models against the baselines in terms of MRR@10 are more obvious than those in terms of Recall@10. For example, AHNQS$_{combined}$ shows an improvement over HNQS by 12.51% in terms of MRR@10 and 9.66% in terms of Recall@10. This means that the attention mechanism cannot only help to suggest the right query, but has a

Table 4
Performance of query suggestion models. The results by the best baseline and the best performer in each column are italic and in boldface, respectively. Statistical significance of pairwise differences of AHNQS$_{local}$ and AHNQS$_{combined}$ vs. the best baseline) is determined by a *t*-test (▲/▼ for $\alpha = 0.01$).

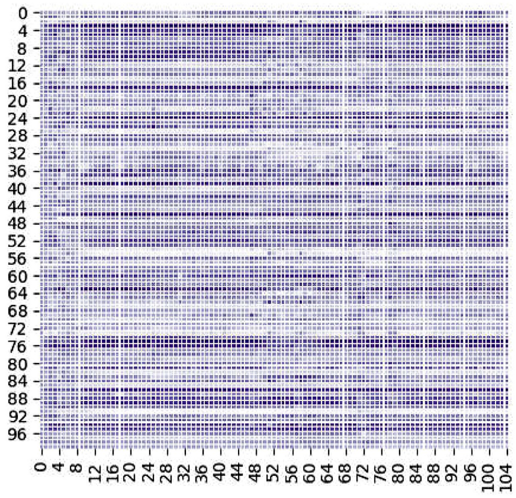
Model	Recall@10	MRR@10
ADJ	0.7072▲	0.6922▲
NQS	0.6444▲	0.6148▲
HNQS	*0.8138*▲	*0.7874*▲
AHNQS$_{local}$	0.8618▲	0.8514▲
AHNQS$_{combined}$	**0.8924**▲	**0.8859**▲



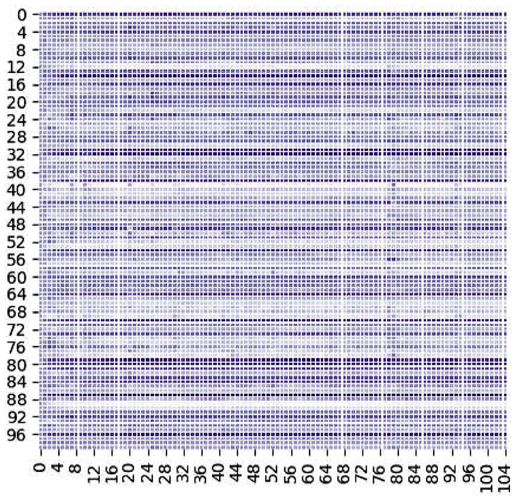
(a) Session-level RNN in NQS.



(b) Session-level RNN in HNQS.



(c) User-level RNN in HNQS.



(d) User-level RNN in AHNQS$_{local}$.

Fig. 6. Visualizing the hierarchical structure ((a) and (b)) and attention mechanism ((c) and (d)). The lighter the area in the plot, the more important the information is.

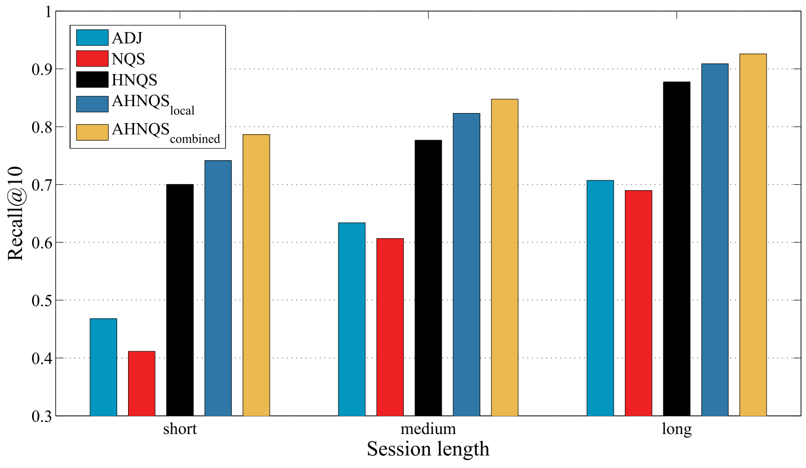
competitive performance in ranking it at the top position in the list of query suggestion.

5.2. Visualization of the hierarchical structure and attention mechanism

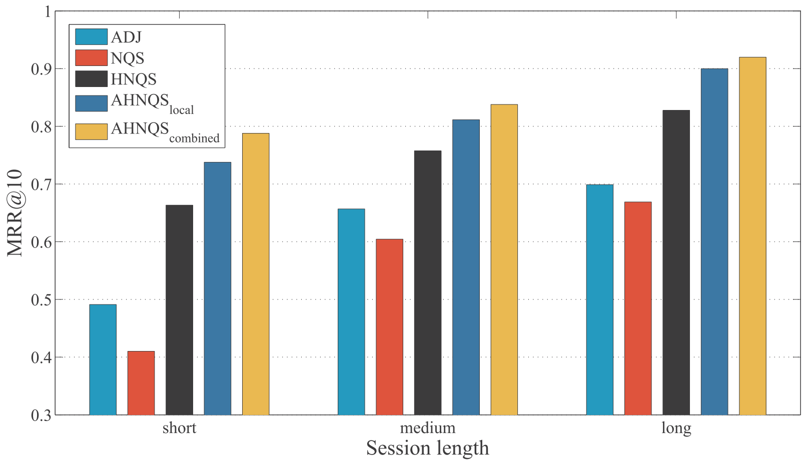
To determine the impact of the hierarchical structure and attention mechanism, we consider a sample session and user, and compare the hidden states of an RNN in NQS and HNQS, respectively in Fig. 6a and b, as well as the hidden states of an RNN in HNQS and AHNQS$_{local}$, respectively in Fig. 6c and d. The lighter the area in the plot, the more important the information is.

In Fig. 6a and b, the session contains 102 queries (x-axis); the number of hidden units is 100 (y-axis). Compared with Fig. 6a, we can see that the hierarchical structure modifies the user's search intent especially at the first positions in a session in Fig. 6b. This is because we initialize the session-level RNN with a user state U_t in HNQS. There is a fluctuation around the 36th to 40th queries in Fig. 6b, which may be due to the fact that we use a GRU unit inside the session-level RNN to transport the user state information within the network.

Turning to the attention mechanism (Fig. 6c and d), we select a user with 105 sessions (x-axis) and the number of hidden units in the user-level RNN is set to 100 (y-axis). Compared with Fig. 6c, the user's preference towards different information is more equally distributed inside the user-level RNN in Fig. 6d. Moreover, going from left to right there are fewer abrupt shifts form high interest (light) to low interest (dark) areas, or vice versa, in Fig. 6d than in Fig. 6c: the attention mechanism can help to describe a user's long-term search preferences towards different topics.



(a) Performance in terms of Recall@10.



(b) Performance in terms of MRR@10.

Fig. 7. Effect on performance of five models in terms of Recall@10 and MRR@10 with different session lengths, tested on the AOL log.

5.3. Impact of the current session length

For **RQ2**, we expect the current session length to have an impact on the performance of query suggestion models. We report separate results for short (2 queries), medium (3 or 4 queries), and long current sessions (at least 5 queries) on the test set in Fig. 7.

Clearly, as the session length increases, the performance in terms of Recall@10 of all query suggestion models improves and our AHNQS$_{combined}$ model always achieves the highest scores. As for the baselines, ADJ outperforms NQS; the margin between them goes down as the session length increases. HNQS performs better than both ADJ and NQS across all session lengths. The improvements of HNQS and AHNQS against ADJ (as well as NQS) are more obvious for short sessions than for long sessions. This is due to the fact that when predicting a user's search intent at the first position of a session, the hierarchical structure within RNN models can provide effective information from a user's past search history and thus can improve the accuracy for query suggestion.

For MRR@10, a similar trend is shown in Fig. 7a. Compared with the results in Fig. 7b, the AHNQS models show a larger improvement over HNQS. For AHNQS$_{local}$, the improvements are 11.23%, 7.13% and 8.74% in terms of MRR@10, for short, medium and long sessions, respectively, vs. improvements of 5.88%, 5.97% and 3.61% for Recall@10. As for AHNQS$_{combined}$, the improvements are 18.77%, 10.61% and 11.15% in terms of MRR@10, for short, medium and long sessions, respectively, vs. improvements of 12.31%, 9.12% and 5.53% for Recall@10. This confirms our intuition about attention mechanisms, i.e., that they help to improve the precision for query suggestion.

5.4. Impact of different session states

For **RQ3**, in order to investigate the impact of different session states used in our hierarchical structure, i.e., the global session state, the local session state, and the combined session state, we compare the performance of HNQS, AHNQS$_{local}$ and AHNQS$_{combined}$

Table 5
Performance comparison among query suggestion models with different session states when the cutoff *N* is 5, 10 and 15, respectively. *D* denotes the hidden state dimension in session-level RNN. The results by the best performer in each column are in boldface. Statistical significance of pairwise differences of AHNQS$_{combined}$ vs. HNQS is determined by a *t*-test (▲/▼ for $\alpha = 0.05$).

(a) Performance comparison at $N = 5$

	$D = 50$		$D = 100$	
Model	Recall@5	MRR@5	Recall@5	MRR@5
HNQS	0.7819▲	0.7537▲	0.7977▲	0.7816▲
AHNQS$_{local}$	0.8279▲	0.8208▲	0.8421▲	0.8499▲
AHNQS$_{combined}$	**0.8701**▲	**0.8629**▲	**0.8790**▲	**0.8837**▲

(b) Performance comparison at $N = 10$

	$D = 50$		$D = 100$	
Model	Recall@10	MRR@10	Recall@10	MRR@10
HNQS	0.7981▲	0.7646▲	0.8138▲	0.7874▲
AHNQS$_{local}$	0.8515▲	0.8301▲	0.8618▲	0.8514▲
AHNQS$_{combined}$	**0.8837**▲	**0.8657**▲	**0.8924**▲	**0.8859**▲

(c) Performance comparison at $N = 15$

	$D = 50$		$D = 100$	
Model	Recall@15	MRR@15	Recall@15	MRR@15
HNQS	0.8036▲	0.7673▲	0.8178▲	0.7882▲
AHNQS$_{local}$	0.8524▲	0.8336▲	0.8653▲	0.8537▲
AHNQS$_{combined}$	**0.8866**▲	**0.8671**▲	**0.8978**▲	**0.8864**▲

by varying the hidden state dimension in the session-level RNN from 50 to 100. Here, we should remind the reader that HNQS can be regarded as the model that only has a global session state, as we have explained in Section 4.2. We present the Recall and MRR scores when the cutoff *N* is set to 5, 10 and 15 in Table 5 as tested on the AOL log.

As shown in Table 5, we find that the HNQS and AHNQS$_{local}$ models, which only use a single way to represent the session state, do not perform well in terms of the two metrics. However, AHNQS$_{combined}$ yields the best performance under all experimental settings and its improvements over HNQS are significant when tested with a *t*-test at $\alpha = .05$. This indicates that merely considering the final hidden state or the sequential behavior in the current session may not allow the network to learn a good query suggestion model. In particular, AHNQS$_{local}$ performs better than HNQS on different hidden state dimensions over the three cutoff settings. This demonstrates that the local session state can give a better description of a user's search intent than the global session state.

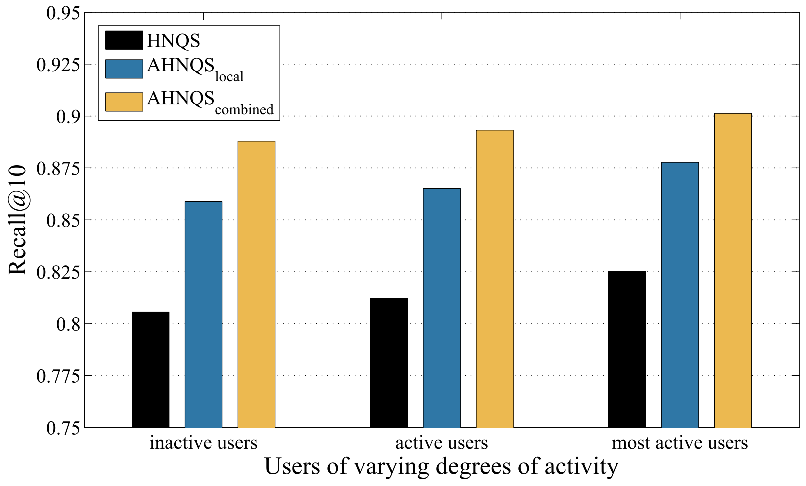
Regarding the impact of different hidden state dimensions, we can see that the query suggestion performance of the three models improves when the hidden state dimension increases from 50 to 100. However, the AHNQS$_{combined}$ model shows less improvements than the other two models. For example, the improvements of $D = 100$ over $D = 50$ are 1.97%, 1.21% and 0.98% of HNQS, AHNQS$_{local}$ and AHNQS$_{combined}$ in terms of Recall@10, and 2.98%, 2.56% and 2.33% in terms of MRR@10, respectively. This can be explained by the fact that AHNQS$_{combined}$ takes advantage both of the local and global session state, and thus can generate a better performance with a smaller hidden state dimension. In addition, comparing different cutoff settings, the performances of all models get improved with *N* increasing from 5 to 15. In addition, the improvements of $N = 15$ over $N = 5$ of our AHNQS models are smaller than of HNQS in terms of both metrics. This can be explained by the fact that the AHNQS models can rank the right query at the top position of the query suggestion list and thus the performance is affected little by an increase in the cutoff.

5.5. Scalability with different numbers of users' sessions

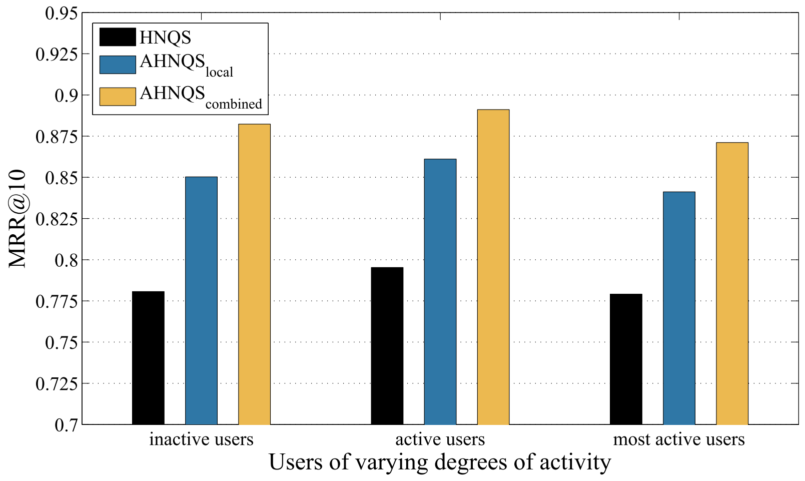
In Fig. 5, we have shown that most users have a very limited number of sessions (≤ 17) while there are some that have a particularly large number of sessions (≥ 150), which we regard as "inactive users" and "most active users," respectively. It is meaningful to investigate how the performance of AHNQS and HNQS varies across users with different numbers of sessions.

Following Prem, Hofman, and Blei (2015), we look at the performance for users of varying degrees of activity, measured by percentile. In Fig. 8, we first rank the users according to their number of sessions. The "inactive users" mark shows the mean performance across the bottom 10% of users, who are least active but constitute the majority of the dataset; the "active users" mark shows the mean performance for the remaining users except the top 10% most active users; the "most active users" mark shows the mean performance of the top 10% most active users, which form the smallest subset of the dataset.

As shown in Fig. 8, the AHNQS models outperform the best baseline model HNQS for users across all activity levels, i.e., inactive users, active users as well as most active users. In addition, the AHNQS$_{combined}$ model always achieves the best performance in terms of Recall@10 and MRR@10. In particular, AHNQS$_{combined}$ shows larger improvements over the HNQS model for the "inactive users"



(a) Performance in terms of Recall@10.



(b) Performance in terms of MRR@10.

Fig. 8. Performance of three models across users with different numbers of search sessions, tested on the AOL log.

than for the "active users" as well as the "most active users." For example, when the number of user sessions increases, i.e., testing on the group of "inactive users" to "active users" and "most active users" accordingly, the improvements decrease from 10.22% to 9.95% and 9.23% in terms of Recall@10, and from 13.01% to 12.04% and 11.81% in terms of MRR@10 on the AOL dataset, respectively. This demonstrates that AHNQS$_{combined}$ can effectively model a user's search intent even with a small number of user sessions.

We zoom in on the scores in terms of two metrics. In Fig. 8a, the performance of the three models in terms of Recall@10 monotonically improves when the number of user sessions goes up. However, as shown in Fig. 8b, the performance in terms of MRR@10 decreases when the number of user sessions increases, cf. "active users" vs. "most active users." That may be due to the fact that users with lots of search sessions may have very diverse search intents and thus it is difficult to improve the ranking accuracy of the models for such users. This naturally suggests that in future work we should consider extending the AHNQS$_{combined}$ model by integrating other strong signals for personalized query suggestion, such as dwell time or user profiles.

6. Conclusions and future work

We have proposed an attention-based hierarchical neural query suggestion model (AHNQS) that combines a hierarchical user-session RNN with an attention mechanism. The hierarchical structure, which incorporates a session-level and a user-level RNN, can model both the user's short-term and long-term search behavior effectively, while the attention mechanism captures a user's preference towards certain queries over others. For the session-level RNN, a combined session state is applied to capture both the user's sequential behavior and their main purpose in the current session, which is then used as the input for the user-level RNN. For the user-level RNN, we use the final hidden state to initialize the next session-level RNN, which can automatically transport the user information within the network.

Our experimental results show that: (1) the proposed AHNQS model helps to boost query suggestion performance in terms of MRR and Recall across sessions with various lengths; (2) using the combined session state in the AHNQS model achieves better performance than only using the local session state; (3) the AHNQS model yields better performance than the best baseline for inactive, active, as well has highly active users.

The theoretical implication of our research is that a hierarchical model which is combined with an attention mechanism for query suggestion can capture the dynamic search intents of a user. The practical implication of our research is that the improvements of AHNQS over the best baseline model are significant; they are especially prominent for short sessions and for inactive users with few search sessions, which is a realistic setup that online services are always confronted with Rashid, Karypis, and Riedl (2008). Compared to the state-of-the-art, AHNQS achieves improvements of 9.66% and 12.51% in terms of Recall@10 and MRR@10, respectively, on average over all users, and of 10.22% and 13.01% for inactive users.

As to future work, we plan to evaluate our model on other datasets so as to verify its robustness. We also want to investigate the performance of AHNQS when combining semantic similarity within the hierarchical structure (Meij, Bron, Hollink, Huurnink, & de Rijke, 2011), e.g., with different encoding methods for input queries (Karisani, Rahgozar, & Oroumchian, 2016; Li, Duan, et al., 2017; Li, Xing, Sun, & ma, 2016). As for attention strategies, we plan to apply different attention mechanisms to explore users' search intents, e.g., self-attention. In addition, signals for personalized query suggestion that we do not consider in this paper, such as user profiles and dwell time, could also be incorporated in our AHNQS model.

Acknowledgements

We would like to thank our anonymous reviewers for their helpful comments and valuable suggestions.

This research was supported by the National Natural Science Foundation of China under No. 61702526, the Defense Industrial Technology Development Program under No. JCKY2017204B064, the National Advanced Research Project under No. 6141B0801010b, Ahold Delhaize, the VSNU Vereniging van Universiteiten, the China Scholarship Council under No. 201803170244, All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

References

Bahdanau, D., Cho, K., & Bengio, Y. (2015). *Neural machine translation by jointly learning to align and translate. ICLR'15*1–15.

Borisov, A., Markov, I., de Rijke, M., & Serdyukov, P. (2016). *A neural click model for web search. WWW'16*. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee531–541.

Borisov, A., Wardenaar, M., Markov, I., & de Rijke, M. (2018). *A click sequence model for web search. SIGIR 2018: 41st International ACMSIGIR conference on research and development in information retrieval*. ACM45–54.

Cai, F., & de Rijke, M. (de Rijke, 2016a). Learning from homologous queries and semantically related terms for query auto completion. *Information Processing and Management, 52*(4), 628–643.

Cai, F., & de Rijke, M. (2016b). Selectively personalizing query auto-completion. *SIGIR 2016: 39th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 993–996). ACM.

Cai, F., & de Rijke, M. (2016c). A survey of query auto completion in information retrieval. *Foundations and Trends in Information Retrieval, 10*(4), 273–363.

Cai, F., Reinanda, R., & de Rijke, M. (2016). Diversifying query auto-completion. *ACM Transactions on Information Systems, 34*(4) 25:1–25:33.

Cao, H., Jiang, D., Pei, J., He, Q., Liao, Z., Chen, E., & Li, H. (2008). *Context-aware query suggestion by mining click-through and session data. KDD'08*. New York, NY, USA: ACM875–883.

Chen, W., Cai, F., Chen, H., & de Rijke, M. (2017). *Personalized query suggestion diversification. SIGIR'17*. New York, NY, USA: ACM817–820.

Chen, W., Cai, F., Chen, H., & de Rijke, M. (2018). *Attention-based hierarchical neural query suggestion. SIGIR'18*. New York, NY, USA: ACM1093–1096.

Cho, K., van Merrienboer, B., Gülçehre, Ç., et al. (2014). *Learning phrase representations using RNN encoder-decoder for statistical machine translation. EMNLP'14*. ACL1724–1734.

Cui, J., Liu, H., Yan, J., Ji, L., Jin, R., He, J., ... Du, X. (2011). *Multi-view random walk framework for search task discovery from click-through log. CIKM'11*. New York, NY, USA: ACM135–140.

Fiorini, N., & Lu, Z. (2018). *Personalized neural language models for real-world query auto completion. Proceedings of the 2018 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 3 (industry papers)*. Association for Computational Linguistics208–215.

Guo, J., Cheng, X., Xu, G., & Zhu, X. (2011). *Intent-aware query similarity. CIKM'11*. ACM259–268.

He, Q., Jiang, D., Liao, Z., Hoi, S. C. H., Chang, K., Lim, E., & Li, H. (2009). *Web query recommendation via sequential query prediction. 2009 IEEE 25th international conference on data engineering*1443–1454.

He, X., He, Z., Song, J., Liu, Z., Jiang, Y., & Chua, T. (2018). NAIS: Neural attentive item similarity model for recommendation. *IEEE Transactions on Knowledge and Data Engineering, 30*(12), 2354–2366.

Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2016). *Session-based recommendations with recurrent neural networks. ICLR'16*1–10.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

Huang, C.-K., Chien, L.-F., & Oyang, Y.-J. (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), 638–649.

Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., & Heck, L. (2013). *Learning deep structured semantic models for web search using clickthrough data. CIKM'13*. New York, NY, USA: ACM2333–2338.

Jiang, J.-Y., Ke, Y.-Y., Chien, P.-Y., & Cheng, P.-J. (2014). *Learning user reformulation behavior for query auto-completion. SIGIR'14*. New York, NY, USA: ACM445–454.

Karisani, P., Rahgozar, M., & Oroumchian, F. (2016). A query term re-weighting approach using document similarity. *Information Processing and Management, 52*(3), 478–489.

Li, C., Duan, Y., Wang, H., Zhang, Z., Sun, A., & Ma, Z. (2017). Enhancing topic modeling for short texts with auxiliary word embeddings. *ACM Transactions on Information Systems, 36*(2), 11:1–11:30.

Li, C., Xing, J., Sun, A., & ma, Z. (2016). *Effective document labeling with very few seed words: A topic model approach. CIKM'16*. New York, NY, USA: ACM85–94.

Li, J., Ren, P., Chen, Z., Ren, Z., Lian, T., & Ma, J. (2017). *Neural attentive session-based recommendation. CIKM'17*. New York, NY, USA: ACM1419–1428.

Li, L., Yang, Z., Liu, L., & Kitsuregawa, M. (2008). *Query-url bipartite based approach to personalized query recommendation. AAAI'08*. AAAI Press1189–1194.

Ma, H., Yang, H., King, I., & Lyu, M. R. (2008). *Learning latent semantic relations from clickthrough data for query suggestion. CIKM'08*. New York, NY, USA: ACM709–718.

Mei, Q., Zhou, D., & Church, K. (2008). *Query suggestion using hitting time. CIKM'08*. New York, NY, USA: ACM469–478.

Meij, E., Bron, M., Hollink, L., Huurnink, B., & de Rijke, M. (2011). Mapping queries to the linking open data cloud: A case study using dbpedia. *Journal of Web Semantics, 9*(4), 418–433.

Mitra, B., & Craswell, N. (2015). *Query auto-completion for rare prefixes. CIKM'15*. New York, NY, USA: ACM1755–1758.

Onal, K. D., Zhang, Y., Altingovde, I. S., Rahman, M. M., Karagoz, P., Braylan, A., ... Lease, M. (2018). Neural information retrieval: At the end of the early years. *Information Retrieval Journal, 21*(2–3), 111–182.

Ozertem, U., Chapelle, O., Donmez, P., & Velipasaoglu, E. (2012). *Learning to suggest: A machine learning framework for ranking query suggestions. SIGIR'12*. New York, NY, USA: ACM25–34.

Park, D. H., & Chiba, R. (2017). *A neural language model for query auto-completion. SIGIR'17*. New York, NY, USA: ACM1189–1192.

Prem, G., Hofman, J. M., & Blei, D. M. (2015). *Scalable recommendation with hierarchical poisson factorization. UAI'15*. AUAI Press326–335.

Quadrana, M., Karatzoglou, A., Hidasi, B., & Cremonesi, P. (2017). *Personalizing session-based recommendations with hierarchical recurrent neural networks. RecSys'17*. New York, NY, USA: ACM130–137.

Rashid, A. M., Karypis, G., & Riedl, J. (2008). Learning preferences of new users in recommender systems. *ACM SIGKDD Explorations Newsletter, 10*(2), 90.

Santos, R. L. T., Macdonald, C., & Ounis, I. (2013). Learning to rank query suggestions for adhoc and diversity search. *Information Retrieval, 16*(4), 429–451.

Shen, Y., He, X., Gao, J., Deng, L., & Mesnil, G. (2014). *A latent semantic model with convolutional-pooling structure for information retrieval. CIKM'14*. New York, NY, USA: ACM101–110.

Smith, C. L., Gwizdka, J., & Feild, H. (2017). The use of query auto-completion over the course of search sessions with multifaceted information needs. *Information Processing and Management, 53*(5), 1139–1155.

Sordoni, A., Bengio, Y., Vahabi, H., et al. (2015). *A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. CIKM'15*. ACM553–562.

Torres, S. D., Hiemstra, D., Weber, I., & Serdyukov, P. (2014). Query recommendation in the information domain of children. *Journal of the Association for Information Science and Technology, 65*(7), 1368–1384.

Vidinli, I. B., & Ozcan, R. (Ozcan, 2016a). New query suggestion framework and algorithms: A case study for an educational search engine. *Information Processing and Management, 52*(5), 733–752.

Vidinli, I. B., & Ozcan, R. (Ozcan, 2016b). New query suggestion framework and algorithms: A case study for an educational search engine. *Information Processing and Management, 52*(5), 733–752.