Enhancing Potential Re-Finding in Personalized Search With Hierarchical Memory Networks

Yujia Zhou, Zhicheng Dou, *Member, IEEE*, and Ji-Rong Wen, *Senior Member, IEEE*

Abstract—The goal of personalized search is to tailor the document ranking list to meet user's individual needs. Previous studies showed users usually look for the information that has been searched before. This is called re-finding behavior which is widely explored in existing personalized search approaches. However, most existing methods for identifying re-finding behavior focus on simple lexical similarities between queries. In this paper, we propose a personalized framework based on hierarchical memory networks (MN) to enhance the identification of the potential re-finding behavior. Specifically, we explore the potential re-finding behaviors of users from two dimensions. (1) Granularity dimension. The framework carries out re-finding identification with external memories from word, sentence, and session levels. (2) Query intent dimension. Query-based re-finding and document-based re-finding are taken into account to cover user's different query intents. To enhance the interaction between different memory slots, we optimize the $READ$ operation of MN with two strategies that utilize the information in memory in a multi-hop way. Endowed with these memory networks, we can enhance user's potential re-finding behaviors and build a fine-grained user model dynamically. Experimental results on two datasets have a significant improvement over baselines, and the optimized $READ$ operation shows better performance.

Index Terms—Personalized search, re-finding identification, hierarchical memory networks

1 Introduction

Users usually get information from the internet by issuing a query to the search engine. Under the same query, most common search engines return the same result without distinction for all users. However, even for the same query, the real intentions of different users are often different, especially for ambiguous queries [1], [2]. Personalized search is a possible way to solve this problem. It tailors the original ranking of results to meet user's individual needs.

The key to personalized search is how to build user models accurately. Previous studies have shown that the user's query log contains plenty of personalized information that can help learn user profiles [3], [4], [5], [6], [7], [8], [9]. They extracted features from a large-scale click data to model the user. However, these manually designed features may not fully cover every aspect. With the emergence of deep learning, new personalization approaches were proposed to learn the semantic representation and extract features hidden in the search history automatically [5], [10], [11], [12]. They have successfully improved the quality of personalized search.

Although the strategies of personalization are different, most of them pointed out that users often seek information they have encountered before. This phenomenon is called re-finding behavior, which can be used to build user models in personalized search in a reliable way. Previous studies on modeling re-finding behavior attempted to examine the features from multiple angles to predict the clicks on viewed documents [13], [14]. However, these studies mainly identify the re-finding behaviour based on lexical similarity, which cannot cover semantically similar situations. In fact, some queries look different, but express the same intent, like "new Apple computer profile" and "new macbook introduction". The actual re-finding behavior in search engines is much more complicated than this. In this paper, our goal is to strengthen the identification of users' potential re-finding behaviors for personalized search, especially those that cannot be identified simply by lexical rules. Due to the powerful ability of deep learning to learn representation, we intend to apply it on capturing re-finding behavior in semantic and model the sequential information hidden in them.

Previous personalized search approaches with deep learning tried to build sequential user profiles over queries or sessions using the recurrent neural network (RNN) [10], [12]. These methods have been shown effective to model user interests over time by encoding historical interaction into a hidden state vector. However, this highly abstract encoding approach is not conducive to capture fine-grained user preference. Memory network has made progress on many sequential-based tasks (e.g., reading comprehension, sequential recommendation) due to the ability of extracting information from large-scale data and its great interpretability [15], [16]. Its advantages fit our needs for building a fine-grained user model based on re-finding. Motivated by the powerful representation ability of MN, we propose to enhance user's potential re-finding behavior based on it.

• *The authors are with the School of Information, Gaoling School of Artificial Intelligence, Beijing Key Laboratory of Big Data Management and Analysis Methods, DEKE, Renmin University of China, Beijing 100872, China. E-mail: {zhouyujia, dou}@ruc.edu.cn, jirong.wen@gmail.com.*

Manuscript received 21 Apr. 2020; revised 2 July 2021; accepted 27 Oct. 2021. Date of publication 9 Nov. 2021; date of current version 7 Mar. 2023.
This work was supported by National Natural Science Foundation of China under Grants 61872370 and 61832017, Beijing Outstanding Young Scientist Program under Grant BJJWZYJH012019100020098, China Unicom Innovation Ecological Cooperation Plan, and Intelligent Social Governance Platform, Major Innovation & Planning Interdisciplinary Platform for the Double-First Class Initiative, Renmin University of China.
(Corresponding author: Zhicheng Dou.)
Recommended for acceptance by Q. He.
Digital Object Identifier no. 10.1109/TKDE.2021.3126066

According to the user's information needs, we classify the re-finding behavior into two categories: tracking information about a certain topic or just for finding one document [17]. In the first case, users typically issue similar queries to get information. We can predict the user's next click behavior by analyzing his historical click data under these queries. In the second case, we are able to summarize the user's query habits for finding the document, and identify the re-finding by comparing the current query with his habits. To cover both cases, we design two separate memories, a query memory and a document memory, for storing user historical interactions from two different angels. In fact, users often issue a series of queries in a session for a single information need. They might show the same query intent over sessions. To identify this situation called session-based re-finding, we design an intent memory to store user past query intent of each session.

Specifically, we design a model for personalized search, which focuses on the re-finding behavior with external memories we stated above. Different historical behaviors have unequal contributions to the re-finding. Thus, we attempt to highlight relevant words and historical behaviors stored in the word memory, query memory and document memory based on their relevance to the current needs. And then we further model the session-based re-finding with the help of intent memory to build a more accurate user model. Finally, by matching the user model and the current needs, we compute the probability of the document being clicked under two types of re-finding and predict the personalized search results. Additionally, we extend our model to a multi-hop version, which can dig deeper into the effective information in the memory network.

Our main contributions are summarized as follows. First, we make use of external memories to enhance user re-finding behavior for personalized search in an interpretable way. Second, in order to cover more complex re-finding, we analyze user re-finding behavior from word, query, and document respectively, and further consider session-based re-finding. Third, based on the characteristic of re-finding that a document is more likely to be irrelevant if it has been ignored in history, we consider the negative impact of unclicked documents to model the user interests. This paper is an extended version of the WSDM 2020 paper [18]. The main extensions of this journal version are: (1) We extend our model to a hierarchical memory network, including three levels of re-finding identification from word, sentence, and session. Specifically, we add a word memory to emphasize the impact of important words in queries based on user past interactions, which is simply implemented by TF-IDF weights in the original paper. (2) We refine the $READ$ operation of memory networks with two optimization strategies to improve memory utilization, while in the original paper, we simply used a single vanilla attention. (3) We further test the effect of the model on the public dataset AOL search log. The performance on this dataset not only tests the personalization ability of the model, but also reflects the ability to match query and document in semantic.

In the rest of the paper, related works are summarized in Section 2. Our personalized model is introduced in Section 3. We demonstrate the experimental settings and results in Section 4, and draw the conclusion in Section 5.

2 Related Work

2.1 Personalized Search

Search results personalization has been shown to effectively improve the quality of search engines [4]. The main goal of personalized search is to re-rank the results to meet the individual needs of different users, depending on the user's interests. In traditional methods of personalized search, the main personalized features extracted from historical search data focus on click number and topic similarity [7], [19], [20], [21], [22]. The former is widely used due to its availability and reliability. Dou *et al.* [2] counted the number of clicks on the documents in history to re-rank the original document list. Teevan *et al.* [6] followed this approach to identify personal navigation with individual behavior for search results personalization. The topic-based features have gone through a transition from manual design to automated learning [19], [23]. Due to the incomplete category of manual design, such as Open Directory Project (ODP), some studies proposed to learn a latent topic of the document automatically with Latent Dirichlet Allocation (LDA) [7], [9], [20], [22]. With the emergence of learning to rank methods, recent studies [3], [23], [24] combined these two types of features to train a ranking model by the LambdaMART algorithm [25].

The above methods have made great progress, but the incomplete features are still a problem due to the limitations of manual design. Deep learning has become a possible solution to this problem. In the field of personalized search, Song *et al.* [5] proposed a general ranking model based on user individual adaptation. Li *et al.* [11] made use of semantic features powered by deep learning to improve the in-session contextual results. Ge *et al.* [10] used hierarchical recurrent neural networks to model user short- and long-term interests and highlighted the relevant interests by query-aware attention. Lu *et al.* [12] proposed a generative adversarial network framework to train the network with noisy click data. These methods make use of deep learning for semantic modeling and achieve better results. Different from previous studies, we attempt to combine deep learning with memory networks to enhance user re-finding behavior.

2.2 Re-Finding Identification

Re-finding behavior is a common phenomenon in information retrieval. Users often use the same or similar queries to retrieve previously viewed documents. Previous studies on re-finding behavior mainly focused on re-finding identification. Teevan *et al.* [14] analyzed the query log to predict whether the user will click on the same document when the user submitted a query that has ever issued. Tyler *et al.* [26] observed different types of re-finding behavior in inter-session and intra-session and measured the likelihood that re-finding behavior occurs at different positions in a session. Later, Tyler *et al.* [13] utilized re-finding for search results personalization. The results showed the reliability of re-finding prediction for personalized search. Elsweiler and Ruthven [17] performed a diary study that classified the re-finding tasks according to user's information needs. To more accurately identify the re-finding behavior, more kinds of features are used to model the query log. Kotov *et al.* [27] examined the features from three aspects: session-

based features, history-based features, and pairwise features. However, the above methods only consider lexical features, while ignoring the re-finding behavior based on semantic similarity. In this paper, we intend to combine the lexical and semantic features, and alleviate the problem of incomplete features with deep learning.

2.3 Memory Networks

Memory network was first proposed by Weston *et al.* [15] to solve the problem of insufficient representation ability of traditional deep learning models. They proposed a general memory network framework, including input, generation, output and response modules. Some subsequent studies have optimized the model structure based on it. Sukhbaatar *et al.* [28] put forward the end-to-end memory networks which improves the training process, and extended the model to multiple layers. Miller *et al.* [16] designed the key-value memory networks to fit the question-answering task. They used additional knowledge (knowledge base, Wikipedia) to find answers. Apart from these basic memory networks, some novel structures have been proposed to make up for the shortcomings of previous models. For instance, Henaff *et al.* [29] recorded the world state with recurrent entity networks. Chandar *et al.* [30] devised hierarchical memory networks to speed up training. And Liu *et al.* [31] constructed gated end-to-end memory networks which introduced the gate mechanism to achieve regularization of memory. In recent years, memory networks is popular in many research fields, such as dialogue systems, question answering systems, and recommendation systems. In this paper, we intend to apply it to personalized search for building fine-grained user profiles.

3 RPMN - A Hierarchical Memory Network Enhanced Re-Finding Model for Personalized Search

Tailoring the ranking of search results according to individual interest can improve the quality of the retrieval model. As we stated in Section 1, existing personalized search methods are weak in modeling potential re-finding behavior. Inspired by the ability of memory networks to capture fine-grained user preferences, we present a personalized search model with memory networks focusing on the re-finding behavior. With the help of external memories, we expect to screen out historical behaviors that are related to current needs and identify the re-finding behavior in semantic.

We define the notations used throughout the paper in Table 1. Suppose that for user u, his historical log U includes a series of issued queries and click information on the documents retrieved by search engine, i.e., $U = \{\{q_1, D_1\}, \ldots, \{q_i, D_i\}, \ldots, \{q_n, D_n\}\}$, where q_i is the i_{th} query in the query log and D_i is the document list retrieved for q_i. Given a new query q and its original search results $D = \{d_1, d_2, \ldots\}$, we predict the probability of each document being clicked according to personalized data U, and re-rank the document list D combining the relevance to the query q. The final probability of the document d being clicked is denoted as $p(d|q, U)$.

As we have introduced in Section 1, the re-finding behavior can be roughly summarized into two categories: using similar queries to find unspecified documents or just for finding a viewed document. For simplicity, we call these two categories query-based re-finding and document-based re-finding. The former focuses on the similarity between the candidate document and the user interested documents under similar queries, while the latter pays more attention to the historical queries containing the similar documents. We use $p(d|U^q)$ and $p(q|U^d)$ to represent the probability of the document being clicked under these two types of re-finding. The final probability consists of three parts

$$p(d|q, U) = \phi\big(p(d|U^q), p(q|U^d), p(d|q)\big), \tag{1}$$

where $p(d|q)$ represents the adhoc relevance between each candidate document and the query, and $\phi(\cdot)$ is a Multilayer Perceptron (MLP) with $\tanh(\cdot)$ as activation function, which is used to combine the three parts with different weights.

TABLE 1
Notations in the Paper

N.	Definition	N.	Definition
u	a user	U	u's historical log
Q	a set of queries	D	a set of documents
q	a query	d	a document
$q^{s(v)}$	the q's string (vector)	$d^{s(v)}$	the d's URL (vector)
q'	a refined query	d'	a refined document
M	an external memory	m	a slot of M
d^+	a satisfied document	d^-	a skipped document
d_q	average document of q	q_d	average query of d

The structure of our model is shown in Fig. 1. At first, we set the word memory to emphasize the words in the current query or document according to the words that the user has interacted in the history. Second, in order to handle the query-based re-finding and the document-based re-finding, we devise two external memories to highlight the historical behaviors from query and document respectively. And then, with the help of RNN, we construct the intent memory to model the re-finding over sessions. Finally, we get the probability by matching the user profiles with the current needs to re-rank the results. In the remaining parts of the section we will introduce the details.

3.1 Highlighting Relevant Historical Behaviors Dynamically

Although there is a large amount of personalized information in the query log, the same information contributes differently in different situations. So we expect to dynamically enhance the influence of relevant historical behaviors based on the current need, especially those with re-finding value. To utilize each query and document more comprehensively, we get their vector representation from two aspects. (1) Based on word embedding, which is good at capturing the relation at the semantic level. Their representation are computed by weighting the words together with re-finding weights from the word memory M^W. (2) Based on graph embedding, which measures the distance according to co-occurrence probability. This method constructs the historical interactions into a graph and learns the representation of each node. Finally, the representation of each item is generated by concatenating the vectors of two methods.

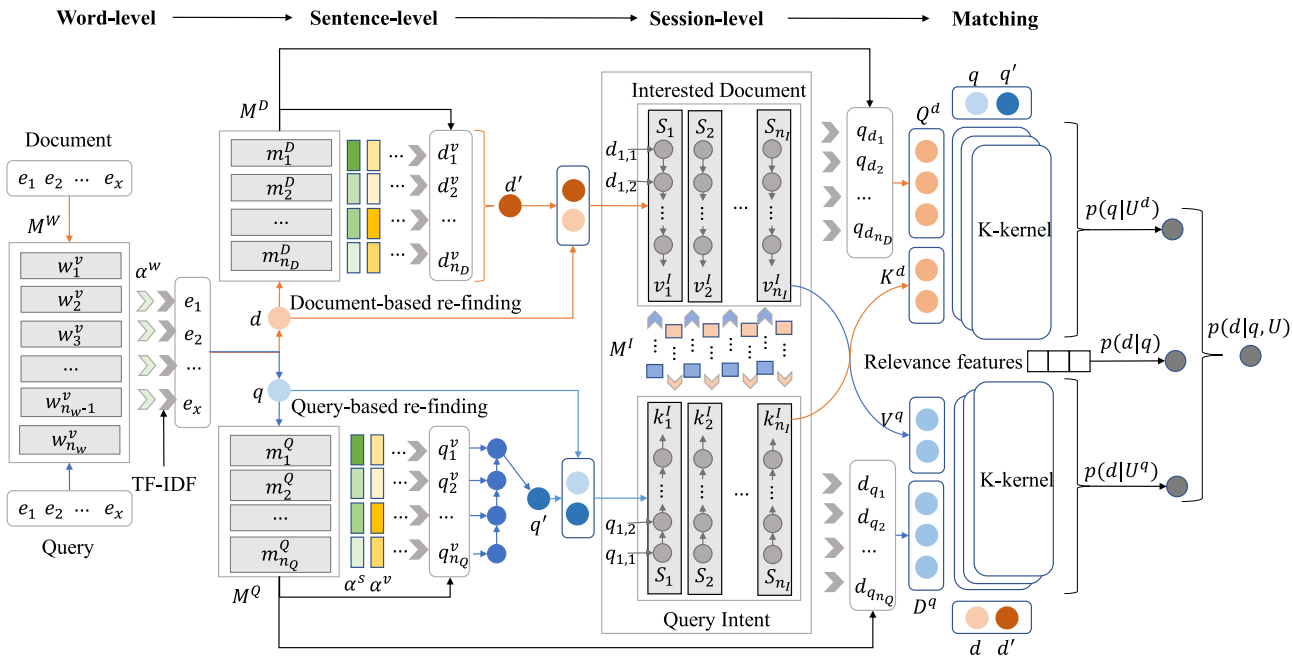


Fig. 1. The architecture of RPMN. Given a new query and a candidate document, relevant historical behaviors are highlighted by three external memories from word, query and document. After extracting session-based re-finding behavior using intent memory based on the current needs, personalized information for query-based re-finding (blue lines) and document-based re-finding (orange lines) are collected. Combining relevance features, we get the final probability for personalization.

As we discussed above, to deal with the re-finding behavior in personalized search, we use external memories which can store the query logs in detail to identify the re-finding behavior in an interpretable way. For covering two types of re-finding, we set up a query memory M^Q and a document memory M^D to record user historical behaviors. Note that our model builds memories for each user independently to store his personal behavior.

3.1.1 Word Memory

We build this memory for measuring the importance of each term in queries and documents. Higher weight should be given to the word that have been issued many times in the past. Traditional TF-IDF weight only considers the case of exact matches and global importance, but we should also focus on synonyms and personal information. We believe that the weight of the current word is determined by historically related words. The word memory records vector representations of words contained in the user's past queries and clicked documents, denoted as $M^W = \{w_1^v, \ldots, w_{n_W}^v\}$, where n_W is the number of words. Suppose that the current query q contains x terms, i.e., $q = \{e_1, \ldots e_x\}$, the re-finding weight α_i^w of the word e_i is computed by

$$\alpha_i^w = \sum_{j=1}^{n_W} \mathrm{S}(e_i, w_j^v), \tag{2}$$

where $\mathrm{S}(\cdot)$ is cosine similarity which keeps the values greater than a certain threshold, which is set to 0.7. These related words reflect the re-finding potential of the current word to which we need pay more attention. To consider the importance of words from both global and personal aspects, we combine the re-finding weight and TF-IDF weight together by normalizing the weights and averaging them. Finally, the vector representation of the current query q^v is a weighted summation of all words

$$q^v = \sum_{i=1}^{x} (\alpha_i^w + TF{-}IDF_{e_i}) e_i. \tag{3}$$

The query or document represented in this way contains word-level re-finding information, which can promote the identification of the re-finding behavior in the following.

3.1.2 Query Memory

Above we used word memory to strengthen the influence of key words, but sometimes the same words have different meanings in different queries. We next identify re-finding behavior from the query level for handling the query-based re-finding in sentence-level. The word memory can most directly and effectively reflect the user's interested words, while the query memory focuses more on the semantics of the entire sentence. They complement each other, and jointly carry out fine-grained modeling of user interests. Since that user behavior under similar queries are valuable to make a prediction, the main function of the query memory M^Q is to find out the historical queries that are related to the current query. Specifically, in addition to build user profiles using satisfied documents, we leverage the skipped documents to model user interests in reverse. The basic idea is if a user skipped a document before, it is more likely to skip it again when encountering the same document. A satisfied click usually refers to a click with more than 30s dwelling time or the last click in a session [3], [9], [10]. And a skipped document is defined as the unclicked document above a satisfied click.

Assume that there are n_Q memory slots in M^Q, i.e., $M^Q = \{m_1^Q, \ldots, m_{n_Q}^Q\}$. Each slot stores a query string, a query

vector and two average document vectors (satisfied and skipped), i.e., $m_i^Q = \{q_i^s, q_i^v, d_{q_i}^+, d_{q_i}^-\}$. Notice that the query stored in each slot is different. The $WRITE$ operation of query memory is defined as: there is a new interaction $\{q, D\}$ from u. We put the average vector d_q^+ of satisfied clicked documents and d_q^- of skipped documents into the memory. If query q has been issued before, we only modify the two average document vectors of corresponding slot

$$\begin{aligned} d_{q_i}^+(new) &\leftarrow \mathrm{GATE}(d_q^+, d_{q_i}^+(old)) \\ d_{q_i}^-(new) &\leftarrow \mathrm{GATE}(d_q^-, d_{q_i}^-(old)), \end{aligned} \tag{4}$$

where $\mathrm{GATE}(\cdot)$ is a gate to control the proportion of new information, $\mathrm{GATE}(a, b) = (1 - z_i) * a + z_i * b$, and the gate weight z_i is set to 0.5 in our model. Otherwise, we put the query string, the query vector and two average document vectors together, i.e., $\{q^s, q^v, d_q^+, d_q^-\}$, into a new slot (or replace the oldest one if there is no empty slot). Here we keep the memory in the chronological order to maintain the sequential information of historical interactions.

The $READ$ operation starts when the user issues a new query q, which is to learn the weight of each slot in M^Q based on the new query. Specifically, for covering more potential re-finding behavior, we compute the weight from the string level (lexical similarity) and the vector level (semantic similarity). Together, they determine the influence of each slot based on q. Formally, with respect to the query string q^s and the query vector q^v, the weight α_i^q of the i_{th} slot is defined as the combination of string level weight $\alpha_i^{q^s}$ and vector level weight $\alpha_i^{q^v}$

$$\alpha_i^q = \boldsymbol{\phi}(\alpha_i^{q^s}, \alpha_i^{q^v}). \tag{5}$$

For string level weight, we choose ten common ways of query change following previous work ("wordorder", "stemming", etc.) [13], [14]. We believe they contribute differently in re-finding. To learn the influence of each types, we devise a type memory M^T to store the matching types and their vector representation. Each representation is initialized by zero and will be updated when the new query comes. Formally, if the relation between the new query string q^s and a historic query string q_i^s belong to the j_{th} type, the new representation r_j of the type is

$$r_j(new) \leftarrow \mathrm{GATE}(f(q^v - q_i^v), r_j(old)), \tag{6}$$

where $f(\cdot)$ is to ensure that the value is the largest when the two queries are the same, and gradually decreases as the difference of them increases, defined as $f(x) = e^{-|x|}$. Given a new query q, we take out corresponding vectors according to the relationship between the historical queries in M^Q and the new query. If a query pair does not match any query change type, the relation vector is set to zero. We use R^{q^s} to represent the set of relation vectors based on q^s, and the string level weight $\alpha_i^{q^s}$ of slot m_i^Q is learned according to its relation vector $r_i^{q^s}$

$$\alpha_i^{q^s} = \frac{\exp(\boldsymbol{\phi}(r_i^{q^s}))}{\sum_{j=1}^{n} \exp(\boldsymbol{\phi}(r_j^{q^s}))}, \tag{7}$$

where the MLP $\boldsymbol{\phi}(\cdot)$ is to output a weight based on the relation vector. We use the function $\mathrm{softmax}(e_i)$ to represent $\frac{\exp(e_i)}{\sum_{j=1}^{n} \exp(e_j)}$ for short in the following.

For vector level weight, with respect to the current new query vector q^v, we highlight the relevant slots based on the topic similarity between query vectors. The weight $\alpha_i^{q^v}$ of slot m_i^Q is generated by the attention mechanism [32]

$$\alpha_i^{q^v} = \mathrm{softmax}(\boldsymbol{\phi}(q^v, q_i^v)). \tag{8}$$

Now we have learned the weight of each slot, which represents the contribution of each historical query to the current query in re-finding. Finally, we take the vectors from M^Q according to the learned weight and get three weighted sets: weighted historical query vector set $Q^q = \{\alpha_1^q q_1^v, \ldots \alpha_{n_Q}^q q_{n_Q}^v\}$, weighted satisfied document vector set $D^{+,q} = \{\alpha_1^q d_{q_1}^+, \ldots \alpha_{n_Q}^q d_{q_{n_Q}}^+\}$, weighted skipped document vector set $D^{-,q} = \{\alpha_1^q d_{q_1}^-, \ldots \alpha_{n_Q}^q d_{q_{n_Q}}^-\}$. And they act on calculating the final probability in Section 3.3.

3.1.3 Document Memory

The document memory is used to analyze the user's query habits based on each candidate document. For the document-based re-finding, we expect to focus on the queries that retrieve the documents which are related to the candidate document through the document memory $M^D = \{m_1^D, \ldots m_i^D, \ldots, m_{n_D}^D\}$. The method of constructing it is similar to the query-based memory. Each memory slot m_i^D consists of a document URL, a document vector and an average query vector, i.e., $m_i^D = \{d_i^s, d_i^v, q_{d_i}\}$. When a new interaction $\{q, D\}$ happens, the $WRITE$ operation forms document-query pairs with the satisfied documents in D and the query i.e., $\{\{d_1^+, q\}, \{d_2^+, q\}, \ldots\}$. And then we put each of them into the document memory M^D like query memory: modify the q_i by $\mathrm{GATE}(\cdot)$ if the document has satisfied before, or use a new (the oldest) slot to store it.

When evaluating a document d, we learn the weight α_i^d of each slot based on d by $READ$ operation. Due to the limited type of URL change, we only consider two types "the same" and "the same domain" of document change to learn the string level weight. And the vector level weight is also generated by attention mechanism, $\alpha_i^{d^v} = \mathrm{softmax}\big(\boldsymbol{\phi}\big(d^v, d_i^v\big)\big)$. By combining two parts of weight, we highlight user behaviors on relevant documents and get two weighted sets: weighted document vector set $D^d = \{\alpha_1^d d_1^v, \ldots, \alpha_{n_D}^d d_{n_D}^v\}$, weighted average query vector set $Q^d = \{\alpha_1^d q_{d_1}, \ldots, \alpha_{n_D}^d q_{d_{n_D}}\}$. They will contribute to the final probability along with the sets from the query memory.

3.2 Modeling Session-Based Re-Finding

In a large number of search behaviors, sometimes users do not get satisfied results by only one query. They often issue a query at the beginning of a session and reformulate it until getting a satisfied document [10]. We believe that user behaviors in a session reflect a query intent. Intuitively, the queries and click data in historical sessions are helpful when the user shows the same query intent next time. Therefore, we attempt to further analyze user re-finding behavior from the session-level. Specifically, we divide the query logs into different

sessions, $U = \{S_1, S_2, \ldots\}$, and construct an intent memory M^I which contains n_I slots to store the historical behaviors over sessions. Each memory slot m_i^I contains a query intent vector k_i^I of a session and an interested document vector v_i^I under the intent, denoted as $m_i^I = \{k_i^I, v_i^I\}$. The $WRITE$ and $READ$ operation of M^I will be introduced in the following.

3.2.1 Exploiting User Historical Intent With RNN

Assume that a user issues a series of queries $\{q_{i,1}, q_{i,2}, \ldots\}$ in the session S_i, and each query corresponds to an average satisfied document vector $\{d_{i,1}, d_{i,2}, \ldots\}$. In general, if the current query cannot meet the user's information needs, he will submit the next query until the information needs are met. So the latter query and the satisfied document in a session can better reflect the user's true intent. Inspired by the great success of RNN in modeling sequential data, we apply it to learn the representation of the session-based intent and interest. We adopt GRU [33] as the basic cell in our work. Two GRU layers are applied to model user query intent from $\{q_{i,1}, q_{i,2}, \ldots\}$ and user interested document from $\{d_{i,1}, d_{i,2}, \ldots\}$ in each session. The $WRITE$ operation of intent memory M^I is defined as: when a new interaction $\{q, D\}$ happens, if it belongs to an existed session in the slot m_i^I, we update the memory slot for this session regarding the query vector q^v and average satisfied document vector d^+ as the inputs of GRU

$$\begin{aligned} k_i^I(new) &\leftarrow \mathrm{GRU}(k_i^I(old), q^v), \\ v_i^I(new) &\leftarrow \mathrm{GRU}(v_i^I(old), d^+), \end{aligned} \tag{9}$$

where $\mathrm{GRU}(\cdot)$ is the GRU unit. The new state vector $k_i^I(new)$ can be calculated according to the inputs and previous state vector $k_i^I(old)$. If it belongs to a new session, we put it in a new (the oldest) slot and the previous state vector is initialized by zero vector.

3.2.2 Extracting Session-Based Information

Now we have recorded the query intent and corresponding interested document of each session in the intent memory, which allows us to explore the user's session-based re-finding behavior. Based on the two types of re-finding behavior, the $READ$ operation of intent memory includes two ways. For the query-based re-finding, we regard query intent as the key and user interested document as the value in M^I. Given a new query q, to more accurately express its intent, such as ambiguous queries, misspelled queries, etc., we generate a refined query vector according to the weighted historical query vector set $Q^q = \{\alpha_1^q q_1^v, \ldots \alpha_{n_Q}^q q_{n_Q}^v\}$ obtained in Section 3.1.2. For capturing the evolution of relevant queries over time in history, we also take a GRU layer to represent the current state vector h_n^q. And then we map it into the same dimension as the query vector by MLP to represent the refined query q'

$$q' = \phi(h_{n_Q}^q) = \phi(\mathrm{GRU}(h_{n_Q-1}^q, \alpha_{n_Q}^q q_{n_Q}^v)). \tag{10}$$

We learn the attentive weight of each slot based on the query vector q^v and the refined query vector q'. We have

$$\alpha_i^{I,q} = \mathrm{softmax}(\phi(k_i^I, [q^v, q'])). \tag{11}$$

Finally, we generate a set $V^q = \{\alpha_1^{I,q} v_1^I, \ldots, \alpha_{n_I}^{I,q} v_{n_I}^I\}$ by reading interested documents with query-aware weights to represent a probability distribution of different interests.

For the document-based re-finding, we exchange the roles of the two parts in M^I to evaluate what query intent the candidate document is likely to belong to, i.e., the key is interested document and the value is query intent. Since that URL is based on certain rules and changes less, we simply get the refined document vector by summing the elements of weighted document vector set D^d

$$d' = \sum_{i=1}^{n} \alpha_i^d d_i^v. \tag{12}$$

And the weights on query intents based on d^v and d' is

$$\alpha_i^{I,d} = \mathrm{softmax}(\phi(v_i^I, [d^v, d'])). \tag{13}$$

The probability distribution of historical intents based on d is denoted as the set $K^d = \{\alpha_1^{I,d} k_1^I, \ldots, \alpha_{n_I}^{I,d} k_{n_I}^I\}$. These two sets from intent memory are essential in the final probability.

3.3 Re-Ranking the Results

In this section, we compute the probability of each part in Eq. (1) using the personalized information we got above.

(1) For $p(d|U^q)$, we make use of the information which is collected for the query-based re-finding behavior. The notation U^q means the user interactions related to q, including (a) the weighted satisfied and skipped document vector sets $D^{+,q}$ and $D^{-,q}$ obtained in Section 3.1.2. (b) the estimated session-based interested documents V^q from the Section 3.2. In order to measure the positive and negative effects of historical behaviors, we calculate the probability of the two parts separately and use MLP to combine them, by

$$p(d|U^q) = \phi(p(d|U^{+,q}), p(d|U^{-,q})). \tag{14}$$

They can be measured by the matching the candidate documents and user personalized information. For a wider range of matches, we put d and the d' together as the target

$$\begin{aligned} p(d|U^{+,q}) &= F_k([d, d'], [D^{+,q}, V^q]), \\ p(d|U^{-,q}) &= F_k([d, d'], D^{-,q}), \end{aligned} \tag{15}$$

where F_k is the matching function which follows the idea of the previous model K-NRM [34]. It devises k kernels to cover different degrees of matching. And the number of kernel k is set to 11 in our model. Formally, after projecting all the vectors into the same semantic space, we form two translation matrices M_{ij}^+ and M_{ij}^- by cosine similarity. The matching function combines the scores of k kernels with MLP (using M_{ij}^+ as an example)

$$\begin{aligned} F_k(M_{ij}^+) &= \phi(f_1(M_{ij}^+), \ldots, f_o(M_{ij}^+), , \ldots, f_k(M_{ij}^+)), \\ f_o(M_{ij}^+) &= \sum_i log\left(\sum_j exp\left(-\frac{(M_{ij}^+ - \mu_o)^2}{2\sigma_o^2}\right)\right), \end{aligned} \tag{16}$$

where μ_o is evenly distributed between -1 and 1 according to k, and σ_o is set to 0.1 in our model. This approach gives us an opportunity to control the degree of matching by adjusting the kernel.

(2) For $p(q|U^d)$, which represents the probability of the document-based re-finding. And the notation U^d includes the information associated with d. (a) the weighted query vector sets Q^d in Section 3.1.3. (b) the estimated session-based query intents K^d in Section 3.2. Imitating the matching method of last part, we are able to get the probability by matching the personalized information with the new query q and the refined query q'

$$p(q|U^d) = F_k([q, q'], [Q^d, K^d]). \tag{17}$$

(3) For $p(q|d)$, following previous work [3], we extract lots of features for every document, including original position, click entropy, temporal weights and topical features. What's more, we add several additional features of the skipped document following our previous idea. The probability is computed by feeding these features $f_{q,d}$ into MLP with $tanh(\cdot)$ as the activation function

$$p(q|d) = \phi(f_{q,d}). \tag{18}$$

Finally, a personalized ranking list is generated by re-ranking the original search results according to the final probability $p(d|q, U)$. We train our model in a pairwise way based on the LambdaRank algorithm. The document pairs are formed by regarding the satisfied documents as positive samples and the skipped documents as negative samples. The distance dis_{ij} of the pair d_i and d_j is computed by $|p(d_i|q, U) - p(d_j|q, U)|$ with the normalization of logistic function. We choose weighted cross entropy between the true distance dis^t_{ij} and the predicted distance dis^p_{ij} as loss function, and we have

$$loss = -|\lambda_{ij}|\Big(dis^t_{ij} log(dis^p_{ij}) + (1 - dis^t_{ij}) log(1 - dis^p_{ij})\Big), \tag{19}$$

where the weight λ_{ij} is the change of ranking quality after swapping the pair d_i and d_j.

3.4 Optimizing Reading Operation of MN

The $READ$ operation of each memory we proposed above is the single-layer attention version, which has low utilization of the memory networks. Inspired by [16], [28], we extend our model to multi-layer reading to further exploit the information in each memory. This strategy can enhance the interaction between different memory slots, so as to more effectively use the information in memory. We design two optimization strategies shown in Fig. 2 to reuse the information in memory.

Multi-Hop Attention. The first optimized $READ$ operation is implemented by multi-hop attention. Each hop searches for similar information from the memory according to the input, and aggregates the information into a state vector with attentive weights, which will be used as the input of the next hop together with the original input. As the number of iterations increases, the information in the memory networks will be fully mined, and the final output will better reflect the user's personalized preferences.

We update the $READ$ operation with multi-hop attention for the word memory, query memory, document memory, and intent memory. Concretely, for a memory with n slots, given the input vector h, the attentive weights of the i_{th} hop $\{\alpha^i_1, \ldots, \alpha^i_n\}$ can be learned by Eq. 3.1.2 based on the input vector h_{i-1}. Next we construct user's preference representation p_i with weighted summation of values in each memory slot

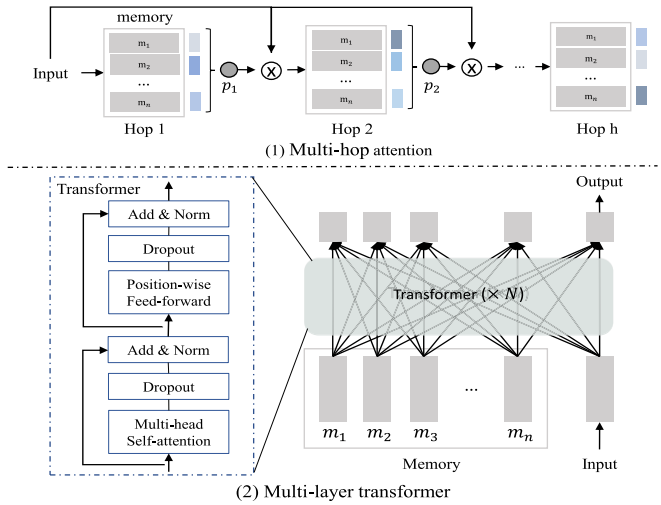


Fig. 2. The structure of optimized $READ$ operation.

$$p_i = \sum_{j=1}^{n} \alpha^i_j v_j, \tag{20}$$

where v_j is the value of j_{th} memory slot. In order to prevent the user preferences we collected from deviating from the original information need as the number of hops increases, we combine both of them as input for the next hop with a gate unit. We have

$$h_i = z * h_{i-1} + (1 - z) * p_i, \tag{21}$$

where z is the gate weight and it is set to 0.5 in our experiments. The input h_0 is the user information need fed into the memory. More hops can help extract more potentially relevant information, but this will ruin the identification of obvious re-finding behaviors. Therefore, we propose another reading strategy with transformer.

Multi-Layer Transformer. The vanilla attention mechanism sometimes cannot guarantee the stability of the effect during multi-hop reading. Transformer [35], a structure based on self-attention mechanism, shows stability in multi-layer structure. Therefore, we adopt the multi-layer transformer to implement the optimized reading operation.

Specifically, for a memory with n slots, the values of all memory slots are denoted as $V = \{v_1, \ldots, v_n\}$. Given an input vector h, we concatenate the values of the memory and the input vector, and apply a multi-layer transformer to mine the user preference. We have

$$o = \text{Transformer}^{last}([V, h]), \tag{22}$$

where o is the output of the model and represents the aggregation of information in memory based on the input vector. The function $\text{Transformer}^{last}(\cdot)$ is implemented by N-layer transformer encoder and takes the output of the last position. The transformer encoder consists of a Multi-head Self-attention (MS) layer and a Position-wise Feed-forward (PF)

TABLE 2
Basic Statistics of the Datasets

Type	AOL dataset	Commercial dataset
#days	91	58
#users	110,439	5,998
#queries	736,454	738,731
#sessions	279,930	276,047
Average query length	2.87	3.25
Average #click per query	1.11	0.46

layer. To keep valid training as the network goes deeper, residual connection is applied to each layer. Each transformer layer is defined as

$$\begin{aligned} \text{Transformer}(q) &= \text{LN}(M_q + \text{D}(\text{PF}(M_q))), \\ M_q &= \text{LN}(q + \text{D}(\text{MS}(q))), \end{aligned} \tag{23}$$

where $\text{LN}(\cdot)$ is layer normalization to stabilize the output. And $\text{D}(\cdot)$ is a dropout layer with 0.1 probability in our settings. The implementation details of the function $MS(\cdot)$ and $PF(\cdot)$ can refer to the previous work [35]. With the multi-layer transformer, the information between different memory slots can better interact to represent the user preference, and the advantages of memory network can be fully displayed to enhance the re-finding behavior.

3.5 Discussion

In summary, we propose a method to enhance the re-finding behavior in personalized search with memory networks. We make a brief discussion about usability of the model below.

Model Compression. Our model contains many external memories to store the user's behavior. Due to the limited memory space in the real scene, we compress the model to enhance its usability. We use a fixed size sliding window to control the size of the memory, and only keep the user's recent behavior. The size of the window can be adjusted according to the physical needs. The larger the window size, the stronger the storage capacity of the model.

Deployment on Edge Devices. Edge devices often have limited computing and storage capacity, which poses a challenge for the deployment of our model. In addition to the model compression, the construction of all external memories can be done offline and updated regularly. Moreover, the search engine can collect the user's behavior first, and after the accumulation reaches a certain number, upload them to the server to update the user's memories.

4 Experiments

4.1 Dataset

We experiment with query logs of a commercial search engine and AOL. The statistics are shown in Table 2.

Commercial dataset includes two month of non-personalized user click-through data in 2013. Each piece of data contains user anonymous ID, query string, query time, top URLs returned by the search engine, and click dwelling time. To ensure the validity of the data, we remove the users whose active time is less than 6 sessions (to make sure we have enough data to build user model) and the documents that cannot be accessed. To identify a session, we use the common approach of demarcating session boundaries by 30 minutes of user inactivity [36].

AOL dataset contains three month of data, which only collects the clicked documents. Following [37], the candidate documents are selected from the top documents ranked by BM25 algorithm [38]. Different from commercial dataset, following [39], the session boundaries are set with respect to the semantic similarity between two consecutive queries, and we sample 5 candidate documents per query for training and validation, 50 candidates to test the model. The document titles are regarded as the content for matching.

Since that personalized search is based on user historical interactions, we regard the first three quarters of data as historical information to build a basic user model, and the last quarter of data is divided into training set, validation set, and test set in a 4:1:1 ratio. Since the query time distribution of different users is uneven, the division is based on the number of sessions of each user during this period, so as to ensure that each part has at least one session data. For the two types of vector representation as we stated in Section 3.1, we train a word vector model with word2vec [40] for the method based on word embedding, and utilize node2vec [41] to learn the representations of graph embedding.

4.2 Baselines

We regard the original ranking as a basic baseline and consider traditional personalized methods based on re-finding and deep learning methods for performance comparison.

P-Click [2]. This method counts the click number on the same document under the same query in history, and generates personalized results by fusing the original ranking.

URP [13]. It extracts three types of feature (query change, personalized and shared) to identify the re-finding and uses it to predict user behaviors for personalized search.

SLTB [3]. It summarizes 102 features, including click-based features, topic-based features, short and long-term features, time decay etc., to train a ranking model by the LambdaMART algorithm.

HRNN [10]. This method models user short-term and long-term interests and highlight relevant interests dynamically using hierarchical RNN with query-aware attention. It is the first time to leverage sequential information with a deep learning framework.

PSGAN [12]. This is a personalized framework for dealing with the noisy click data based on generative adversarial network. We take the discriminator of the query generation based model as our baseline model, which is the state-of-the-art one among four variants of PSGAN.

4.3 Our Models

RPMN (Re-finding Plus by Memory Networks): It is our model proposed in Section 3 with vanilla attention reading operation. To validate the effectiveness of each component, we experiment with different combinations of the components. Specifically, we experiment with:

RPMN-WM. Word memory is removed and we only use the TF-IDF weight to aggregate the words.

RPMN-QM. Query memory is disabled and we assign the same weight to all historical queries.

TABLE 3
Overall Performances of Models on Commercial Dataset

Model	MAP	MRR	A.Clk.	#Better	#Worse	P-Imp.
Ori.	.7399	.7506	2.211	-	-	-
P-Click	.7509	.7634	2.189	3214	28	.0611
URP	.7742	.7802	2.070.	4631	50	.0884
SLTB	.7921	.7998	1.960	6224	81	.1170
HRNN	.8065	.8191	1.902	14608	2067	.2405
PSGAN	.8135	.8234	1.815	14675	1694	.2489
RPMN-WM	.8239†	.8343†	1.746†	14686	872	.2650†
RPMN-QM	.8195†	.8308†	1.763†	13621	630	.2493
RPMN-DM	.8207†	.8312†	1.756†	13580	**603**	.2490
RPMN-IM	.8226†	.8322†	1.749†	14372	800	.2584†
RPMN	.8238†	.8342†	1.745†	14735	890	.2656†
RPMN-A	.8249†	.8351†	1.735†	15082	1019	.2697†
RPMN-T	**.8260**†	**.8364**†	**1.730**†	**15372**	1059	**.2745**†

"†" indicates the model outperforms all baselines significantly with paired t-test at $p < 0.05$ level. The best results are shown in bold.

TABLE 4
Overall Performances of Models on AOL Dataset

Model	MAP	MRR	A.Clk.	#Better	#Worse	P-Imp.
Ori.	.2501	.2583	17.152	-	-	-
P-Click	.4224	.4298	16.526	148221	455	.1747
URP	.4652	.4744	15.132	228534	966	.2691
SLTB	.5072	.5194	13.926	310307	1480	.3652
HRNN	.5423	.5545	10.552	537758	8146	.6262
PSGAN	.5480	.5600	10.267	542403	7825	.6342
RPMN-WM	.5926†	.6049†	8.603†	651439	10892	.6502
RPMN-QM	.5834†	.5952†	8.782†	634664	9091	.6350
RPMN-DM	.5859†	.5973†	8.743†	634456	**8332**	.6383
RPMN-IM	.5902†	.6022†	8.628†	640416	10248	.6488†
RPMN	.5945†	.6072†	8.556†	654348	11313	.6522†
RPMN-A	.5968†	.6097†	8.537†	660265	11442	.6586†
RPMN-T	**.5995**†	**.6135**†	**8.498**†	**664554**	11890	**.6625**†

RPMN-DM. This method eliminates the document memory and treats the historical documents equally.

RPMN-IM. We remove the intent memory which is to model session-based re-finding behavior from the model.

RPMN-A. The read operation is replaced by multi-hop attention referring to Section 3.4.

RPMN-T. The read operation is replaced by multi-layer transformer referring to Section 3.4.

We experiment with multiple sets of parameters, including GRU hidden state size in $\{200, \mathbf{400}, 600\}$, the number of MLP hidden units in $\{64, 128, \mathbf{256}, 512\}$, transformer hidden state size in $\{128, 256, \mathbf{512}\}$, the number of heads in self-attention in $\{4, \mathbf{8}, 16\}$, word embedding size in $\mathbf{300}, 1000\}$, graph embedding size in $\{\mathbf{300}, 1000\}$, the number of kernel in $\{5, 7, 9, \mathbf{11}, 13\}$, learning rates in $\{10^{-2}, \mathbf{10^{-3}}, 10^{-4}\}$. Considering the performance of the model, training time, and memory usage, we choose the parameters in bold to train the model. In more detail about node2vec which we mentioned above, we regard queries, documents, words as the nodes of graph to learn the graph embedding. There are three types of edges: (1) two adjacent queries in the same session; (2) the document and the query to which it belongs; (3) the word and the query (or document) to which it belongs. The parameters p and q of node2vec are both set to 1.0 in our experiment. Due to the use of a large number of user history logs, there are plenty of parameters to learn, which takes about 12 hours per epoch with 1 GPU. After about two epochs of training, the model will reach convergence.

4.4 Evaluation Metrics

Based on the assumption that satisfied clicked documents are relevant and others are irrelevant, we choose three common evaluation metrics to measure the quality of the ranking list, i.e., Mean Average Precise (MAP), Mean Reciprocal Rank (MRR), and average click position (A.Clk.). What's more, due to the influence of the original ranking position bias, the reason why a document is not clicked may be that the position is too low. Based on the consideration that a satisfied clicked document is more relevant than the skipped documents and the next unclicked document, following [12], we construct the inverse document pairs by them and take three metrics #Better, #Worse, and P-Imp. to evaluate the results. The metric #Better shows the number of inverse document pairs on which the model ranks the satisfied clicked document higher than the skipped document. The metric #Worse counts the case that next unclicked document is ranked higher. The metric P-Imp. is defined as P-Imp=$\frac{\#\text{Better}-\#\text{Worse}}{\#\text{Pairs}}$, where #Pairs is the total number of inverse document pairs.

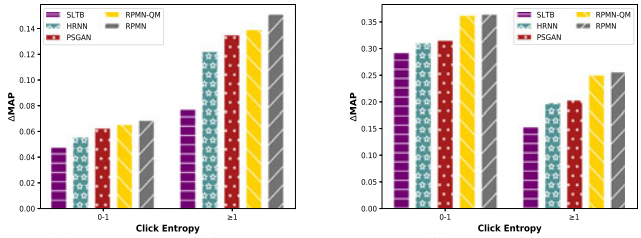
4.5 Overall Results and Analysis

We evaluate the results of the different methods on the test set. The overall results are shown in Tables 3 and 4. We find:

(1) Personalized baselines versus original ranking. All personalized strategies outperform the original ranking generated by search engine on both datasets. The result of P-Click shows that just using the exact matching based re-finding behavior is effective for personalization. URP analyzes a wider range of re-findings and gets a better performance. Their results prove the necessity of our work to model the re-finding behavior in a more holistic way. SLTB integrates all kinds of features and generates a ranking by the learning to rank method, which is more effective than traditional re-finding based features. HRNN and PSGAN prove the effectiveness of deep learning on building user profiles dynamically for personalization.

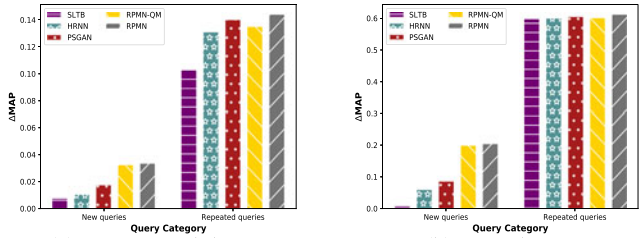
(2) Our methods versus baselines. Our proposed methods outperform baseline models in all evaluation metrics on both datasets. Compared with the best method PSGAN in baseline models, our models have significant improvements with paired t-test at $p < 0.05$ level on MAP. Specifically, the basic model RPMN has increased by 1.40% on MAP on commercial dataset, while this percentage increases to 8.91% on AOL dataset. The reason is that commercial dataset has a high-quality original ranking, which more tests the ability of personalizing the results. But the original ranking of AOL dataset is generated by BM25 algorithm, which still has great upside on navigational queries. As can be seen from the reduction of #Worse on commercial dataset, our personalized methods have a lower risk of mistakes when the quality of adhoc ranking is high.

(3) RPMN versus other methods we designed. The complete model outperforms other models that lack a memory on both datasets. Specifically, removing the query memory



(a) Commercial dataset (b) AOL dataset

Fig. 3. The results on queries with different click entropies.



(a) Commercial dataset (b) AOL dataset

Fig. 4. The results on repeat queries and new queries.

causes a significant decline. This indicates the query-based re-finding is common and it is feasible to capture user interested documents by analyzing the behaviors under similar queries. The model RPMN-DM reduces most on the metric #Better, showing that more pairs can be improved based on the user's query habits for finding a specific document by the document memory. It can be seen that eliminating the intent memory and the word memory have less influence on the model. The intent memory is used to discover re-finding behavior that is more implicit, and the word memory is focused on long queries which have unnecessary words. The optimized model RPMN-A and RPMN-T shows better performance and indicates the effectiveness of further utilizing information in memories.

In summary, the overall results prove that *memory networks are helpful for enhancing the re-finding behavior based on fine-grained personalized information, and improve the personalized results credibly*. To further analyze what kind of queries our model improves on, we test the performance of our model on the different query sets in the remaining parts of this section.

4.6 Results on Different Query Sets

To measure the main contribution of our model, we divide all test queries into different sets and test the effect of the model. We tried two ways of dividing in the following.

Informational Queries Versus Navigational Queries. Previous studies have shown that user queries can be divided into navigation queries and informational queries according to the intent [1], [2], [10]. The former refers to those queries whose purpose is clear and all users prefer the same document. The latter are generally those that are used to get various information or ambiguous queries. We divide the queries with the cutoff of click entropy at 1.0, which is an indicator to measure the potential for personalization. We choose three baseline models STLB, HRNN, PSGAN and two our models RPMN-QM, RPMN to compare. Finally, we compute their MAP improvements on two query sets.

As shown in Fig. 3, our models outperform the baselines on both query sets. For commercial dataset, all the personalized methods contribute more on informational queries (with larger click entropy) than navigational queries (with lower click entropy) on commercial dataset. Specifically, compared to the best baseline model PSGAN, our complete model RPMN has little improvement on navigational queries, but the performance on informational queries is much better. This shows RPMN is good at modeling fine-grained user personalized information to tailor the ranking. Comparing RPMN with RPMN-QM, we find the query memory contributes more on informational queries. It confirms the query-based re-finding usually happens for collecting information and it could be enhanced by our memory networks. For AOL dataset, we observe opposite results that the improvement on navigational queries is more obvious. A possible reason is that the original ranking of commercial dataset has performed well in navigational queries, while AOL dataset has a lower baseline.

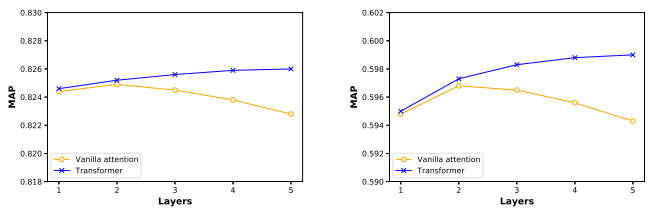
Repeated Queries Versus New Queries. In personalized search, user behaviors under relevant queries in history can provide essential information for building user models. For proving the effectiveness of our model on enhancing re-finding behavior, we categorize the queries into two sets: repeated queries (the queries the current user has issued before) and new queries (others), and test the performance on them with the same model settings as above.

From Fig. 4, we find that the results on two datasets are similar. All personalized models have improved search quality on both query sets, while the improvement on the repeat queries is much larger than that on the new queries. Compared to the best baseline model PSGAN, our model RPMN has a better performance on both parts and the improvement on new queries is more obvious. Intuitively, improving results on new queries is a more difficult task because of the lack of useful personalized information. Our model not only enhances the re-finding behavior from repeated queries, but also improves the potential re-finding in semantic from new queries. In addition, the results of RPMN-QM indicates removing query memory causes more decline on repeated queries, which proves the effectiveness of this memory to highlight the relevant queries.

4.7 Effect of Reading With Multiple Layers

Reading the memory with different hops makes different impact to build the user model. To explore the best reading strategy, we try different layers of transformer and observe the effect of the model on the two datasets separately. In addition, we choose vanilla attention to read the memory with multiple hops as a comparison. Based on the complete model RPMN, we set the reading hops from 1 to 5, and observe the changes of MAP.

The results are shown in Fig. 5. For vanilla attention, we find that on both datasets, two-layer reading is the best



(a) Commercial dataset (b) AOL dataset

Fig. 5. The performance of multi-layer reading operation.

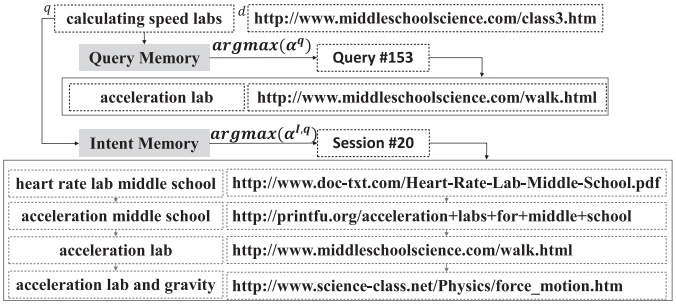


Fig. 6. The case study for interpretability of RPMN.

method, and more layers cannot further improve the MAP. When the number of layers is greater than three, the results are not even as good as the single-layer reading. A possible reason is that as the network structure deepens, the impact of obvious re-finding behaviors will be weakened. Multi-hop reading with transformer is a better strategy obviously. When only one layer transformer is used, the model effect is closest to vanilla attention. As the number of layers increases, the ranking quality is higher. This indicates the transformer is able to maintain the stability of the model when the network goes deeper.

4.8 Analysis on Interpretability of Our Model

Compared to previous personalization approaches based on deep learning, our model is more interpretable owing to the ability of memory networks to store valuable information. Recall that we highlight relevant queries by α^q and documents by α^d in Section 3.1, and measure the influence of session-based historical intent by $\alpha^{I,q}$ and $\alpha^{I,d}$. For simplicity, we present an example to analyze the interpretability of the model from the query. The analysis from the document can be analogized to this.

As shown in Fig. 6, given a new query "calculating speed labs", by looking at the content of the slot with the highest weight in query memory and intent memory, we can get the following explanation: the user interactions under the 153_{th} query "acceleration lab" is the most informative, and user intent in the 20_{th} session is highly similar to the current query intent. According to the satisfied documents under these queries, the candidate document has a high probability of being clicked. Similarly, from the angle of document, we can find out the most valuable historical satisfied document by document memory and infer the possible intent by intent memory. This example indicates our model handles the potential re-finding in semantic and external memories can explain the personalized results.

4.9 Analysis on Challenges From Data

In this part we will analyze how our model overcomes the challenges from the data: data sparsity and data noise.

Data Sparsity. For most users on the Internet, they have very few historical query logs and it is difficult to apply personalized strategies to them. Therefore, it is critical to make full use of the limited historical logs to model user interests. Our model takes advantage of the memory network on storing information, and combines deep learning to fully excavate the fine-grained interests reflected by users' historical behavior. The results on new queries that lack relevant history show that our model can overcome the data sparsity problem to a certain extent.

Data Noise. It is common for users to submit queries that contain noise, such as spelling errors, incomplete input, etc. In order to more realistically simulate user search behavior, we retain these noisy data. We believe these noise data also contains the personalized information. For example, a user often misspells some certain words. When the misspelled words are encountered for the second time, the model can identify it and understand the user's query intent.

5 Conclusion

In this paper, we made use of external memories to enhance the re-finding behavior that is difficult to identify based on the fine-grained user model. First, we construct a word memory to assign the re-finding weight to each word and generate vector representations. And then, we designed the memories for queries and documents to cover two types of re-finding behavior. In addition, endowed with the benefit from RNN on modeling sequential data, we further constructed an intent memory to extend the recognition of re-finding to session level. Finally, By matching the user information needs with the estimated user interests, we calculated the user's click probability on each candidate document, thereby personalizing the results. The $READ$ operation of memory networks can be optimized with multiple hops to strengthen the use of memory networks. Experimental results confirmed the effectiveness and interpretability of our proposed model, and showed the necessity of each memory and optimization strategy.

Acknowledgments

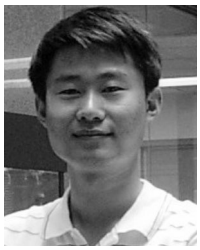
Zhicheng Dou is the corresponding author. I also wish to acknowledge the support provided and contribution made by Public Policy and Decision-making Research Lab of RUC.

References

[1] J. Teevan, S. T. Dumais, and D. J. Liebling, "To personalize or not to personalize: modeling queries with variation in user intent," in *Proc. SIGIR ACM*, 2008, pp. 163–170.
[2] Z. Dou, R. Song, and J.-R. Wen, "A large-scale evaluation and analysis of personalized search strategies," in *Proc. WWW ACM*, 2007, pp. 581–590.
[3] P. N. Bennett *et al.*, "Modeling the impact of short-and long-term behavior on search personalization," in *Proc. SIGIR ACM*, 2012, pp. 185–194.
[4] F. Cai, S. Liang, and M. De Rijke, "Personalized document re-ranking based on Bayesian probabilistic matrix factorization," in *Proc. SIGIR* ACM, 2014, pp. 835–838.
[5] Y. Song, H. Wang, and X. He, "Adapting deep ranknet for personalized search," in *Proc. WSDM ACM*, 2014, pp. 83–92.
[6] J. Teevan, D. J. Liebling, and G. R. Geetha, "Understanding and predicting personal navigation," in *Proc. WSDM* ACM, 2011, pp. 85–94.
[7] M. Harvey, F. Crestani, and M. J. Carman, "Building user profiles from topic models for personalised search," in *Proc. CIKM* ACM, 2013, pp. 2309–2314.
[8] T. Vu, D. Song, A. Willis, S. N. Tran, and J. Li, "Improving search personalisation with dynamic group formation," in *Proc. SIGIR*, 2014, pp. 951–954.
[9] T. Vu, A. Willis, S. N. Tran, and D. Song, "Temporal latent topic user profiles for search personalisation," in *Proc. ECIR*, 2015, pp. 605–616.

[10] S. Ge, Z. Dou, Z. Jiang, J.-Y. Nie, and J.-R. Wen, "Personalizing search results using hierarchical RNN with query-aware attention," in *Proc. CIKM* ACM, 2018, pp. 347–356.
[11] X. Li, C. Guo, W. Chu, Y.-Y. Wang, and J. Shavlik, "Deep learning powered in-session contextual ranking using clickthrough data," in *Proc. NIPS*, 2014, pp. 1–9.
[12] S. Lu, Z. Dou, X. Jun, J.-Y. Nie, and J.-R. Wen, "PSGAN: A minimax game for personalized search with limited and noisy click data," in *Proc. SIGIR*, 2019, pp. 555–564.
[13] S. K. Tyler, J. Wang, and Y. Zhang, "Utilizing re-finding for personalized information retrieval," in *Proc. CIKM* ACM, 2010, pp. 1469–1472.
[14] J. Teevan, E. Adar, R. Jones, and M. A. Potts, "Information re-retrieval: Repeat queries in Yahoo's logs," in *Proc. SIGIR* ACM, 2007, pp. 151–158.
[15] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *2014*, *arXiv:1410.3916*.
[16] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston, "Key-value memory networks for directly reading documents," 2016, *arXiv:1606.03126*.
[17] D. Elsweiler and I. Ruthven, "Towards task-based personal information management evaluations," in *Proc. SIGIR ACM*, 2007, pp. 23–30.
[18] Y. Zhou, Z. Dou, and J.-R. Wen, "Enhancing re-finding behavior with external memories for personalized search," in *Proc. WSDM*, 2020, pp. 789–797.
[19] P. N. Bennett, K. Svore, and S. T. Dumais, "Classification-enhanced ranking," in *Proc. WWW ACM*, 2010, pp. 111–120.
[20] M. J. Carman, F. Crestani, M. Harvey, and M. Baillie, "Towards query log based personalization using topic models," in *Proc. CIKM*, 2010, pp. 1849–1852.
[21] N. Matthijs and F. Radlinski, "Personalizing web search using long term browsing history," in *Proc. WSDM* ACM, 2011, pp. 25–34.
[22] T. Vu, D. Q. Nguyen, M. Johnson, D. Song, and A. Willis, "Search personalization with embeddings," in *Proc. ECIR*, 2017, pp. 598–604.
[23] R. W. White, W. Chu, A. Hassan, X. He, Y. Song, and H. Wang, "Enhancing personalized search by mining and modeling task behavior," in *Proc. WWW ACM*, 2013, pp. 1411–1420.
[24] M. Volkovs, "Context models for web search personalization," 2015, *arXiv:1502.00527*.
[25] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao, "Ranking, boosting, and model adaptation. Tecnical report," Tech. Rep. MSR-TR-2008–109, 2008.
[26] S. K. Tyler and J. Teevan, "Large scale query log analysis of re-finding," in *Proc. WSDM* ACM, 2010, pp. 191–200.
[27] A. Kotov, P. N. Bennett, R. W. White, S. T. Dumais, and J. Teevan, "Modeling and analysis of cross-session search tasks," in *Proc. SIGIR* ACM, 2011, pp. 5–14.
[28] S. Sukhbaatar *et al*., "End-to-end memory networks," in *Proc. Adv. Neural Inf. Process. Syst*., 2015, pp. 2440–2448.
[29] M. Henaff, J. Weston, A. Szlam, A. Bordes, and Y. LeCun, "Tracking the world state with recurrent entity networks," 2016, *arXiv:1612.03969*.
[30] S. Chandar, S. Ahn, H. Larochelle, P. Vincent, G. Tesauro, and Y. Bengio, "Hierarchical memory networks," 2016, *arXiv:1605.07427*.
[31] F. Liu and J. Perez, "Gated end-to-end memory networks," in *Proc. EACL*, 2017, pp. 1–10.
[32] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*.
[33] K. Cho *et al*., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. EMNLP*, 2014, pp. 1724–1734.
[34] C. Xiong, Z. Dai, J. Callan, Z. Liu, and R. Power, "End-to-end neural ad-hoc ranking with kernel pooling," in *Proc. SIGIR* ACM, 2017, pp. 55–64.
[35] A. Vaswani *et al*., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst*., 2017, pp. 5998–6008.
[36] R. W. White and S. M. Drucker, "Investigating behavioral variability in web search," in *Proc. WWW ACM*, 2007, pp. 21–30.
[37] W. U. Ahmad, K. Chang, and H. Wang, "Multi-task learning for document ranking and query suggestion," in *Proc. ICLR (Poster)*, 2018, pp. 1–14.
[38] S. Robertson *et al*., "The probabilistic relevance framework: Bm25 and beyond," *Found. Trends® Inf. Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.
[39] W. U. Ahmad, K.-W. Chang, and H. Wang, "Context attentive document ranking and query suggestion," 2019, *arXiv: 1906.02329*.
[40] T. Mikolov, Q. V. Le, and I. Sutskever, "Exploiting similarities among languages for machine translation," 2013, *arXiv:1309.4168*.
[41] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. SIGKDD* ACM, 2016, pp. 855–864.

Yujia Zhou received the BE degree in computer science and technology in 2019 from the Renmin University of China, where he is currently working toward the PhD degree with the School of Information. His research interests include information retrieval and data mining.



Zhicheng Dou (Member, IEEE) received the BS and PhD degrees in computer science and technology from Nankai University, in 2003 and 2008, respectively. He is currently an associate professor with the School of Information, Renmin University of China. From July 2008 to September 2014, he was with Microsoft Research as a researcher. His research interests include information retrieval, data mining, and big data analytics.

Ji-Rong Wen (Senior Member, IEEE) received the BS and MS degrees from the Renmin University of China, and the PhD degree from the Chinese Academy of Science, in 1999. He is a professor with the Renmin University of China. From 2000 to 2014, he was a senior researcher and research manager with Microsoft Research. His main research interests include web data management, information retrieval (especially web IR), and data mining.