Enhancing Re-finding Behavior with External Memories for Personalized Search

Yujia Zhou[2], Zhicheng Dou[1,2], and Ji-Rong Wen[3,4]
[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]School of Information, Renmin University of China
[3]Beijing Key Laboratory of Big Data Management and Analysis Methods
[4]Key Laboratory of Data Engineering and Knowledge Engineering, MOE
zhouyujia@ruc.edu.cn,dou@ruc.edu.cn,jirong.wen@gmail.com

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 to construct memory networks (MN) to support the identification of more complex re-finding behavior. Specifically, incorporating semantic information, we devise two external memories to make an expansion of re-finding based on the query and the document respectively. We further design an intent memory to recognize session-based re-finding behavior. Endowed with these memory networks, we can build a fine-grained user model dynamically based on the current query and documents, and use the model to re-rank the results. Experimental results show the significant improvement of our model compared with traditional methods.

CCS CONCEPTS

• **Information systems** → **Personalization**;

KEYWORDS

Personalized search, Re-finding, Memory networks

ACM Reference Format:
Yujia Zhou, Zhicheng Dou, and Ji-Rong Wen. 2020. Enhancing Re-finding Behavior with External Memories for Personalized Search. In The Thirteenth ACM International Conference on Web Search and Data Mining (WSDM '20), February 3–7, 2020, Houston, TX, USA. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3336191.3371794

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 [8, 25]. 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, 5, 12, 23, 26, 31, 32]. 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 [10, 17, 18, 23]. 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 [24, 28], such as query change features (e.g. "WordOrder", "StopWords", "TermAdded"), session features (e.g. "the position in a session"), rank features (e.g. "rank for URL associated with the first finding query instance") and so on. 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 enhance the potential re-finding behavior that is difficult to identify. Due to the powerful ability of deep learning to learn representation automatically, 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, 18]. These methods have been shown effective to model user interests over time by encoding historical interaction into a hidden state vector. However, limited by the storage capacity of a hidden vector, it is weak in capturing fine-grained user preference. And this highly abstract encoding approach is not conducive to identifying re-finding behavior. 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

WSDM '20, February 3–7, 2020, Houston, TX, USA

ACM ISBN 978-1-4503-6822-3/20/02…$15.00
https://doi.org/10.1145/3336191.3371794

great interpretability [14, 21, 33]. Its advantages perfectly fit our needs for building a fine-grained user model based on re-finding. Motivated by the powerful storage capacity of MN, we propose to enhance the quality of user models on re-finding based on it.

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 [9]. 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 and corresponding interested document of each session.

Specifically, we design a model RPMN 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 historical behaviors stored in the query memory and the 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.

Our main contributions are summarized as follows: (1) We make use of external memories to enhance user re-finding behavior for personalized search in an interpretable way. (2) In order to cover more complex re-finding, we analyze user re-finding behavior from query and document respectively, and further consider session-based re-finding. (3) 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.

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 [5]. 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 different personalization algorithms, the representation and modeling methods of user interests are also different. However, the main idea of them is to model user preferences based on user history search behavior.

In traditional methods of personalized search, the main personalized features extracted from historical search data focus on click number and topic similarity [2, 6, 12, 19, 22, 30]. The former is widely used due to its availability and reliability. Dou et al. [8] counted the number of clicks on the documents in history to re-rank the original document list. Teevan et al. [26] followed this approach to identify personal navigation with individual behavior for search results personalization. Studies have shown that not all queries are suitable for personalization [8, 25]. Sometimes blindly personalizing search results for navigational queries (like "google.com") may reduce the quality of the results. The click entropy is a valid indicator to measure the potential of personalization of the query. The topic-based features have gone through a transition from manual design to automated learning [2, 22, 34]. 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) [6, 12, 30, 32]. With the emergence of learning to rank methods, recent studies [3, 29, 34] combined these two types of features to train a ranking model by the LambdaMART algorithm [36], which is an extension of LambdaRank [4]. Although the above methods have made great progress, the incomplete user features are still a problem due to the limitations of manual design. Deep learning has become a possible solution to this problem.

The main advantage of deep learning is the word embedding can be learned automatically, which means more potential user preferences in the deep semantic space can be explored. In the field of personalized search, Song et al. [23] proposed a general ranking model based on user individual adaptation. Li et al. [17] 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. [18] 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. [24] 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. [27] 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. [28] utilized re-finding for search results personalization. The results showed the reliability of re-finding prediction for personalized search. Elsweiler and Ruthven [9] 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. [16] examined the features from three aspects: session-based features, history-based features, and pairwise features. However, the above methods still have the limitations of manually designed

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

features, and most of them only consider lexical features, while ignoring the re-finding behavior based on semantic similarity. For modeling complex user re-finding behavior in various scenarios, we intend to combine the lexical and semantic features, and alleviate the problem of incomplete features with deep learning.

3 RPMN - A 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. $U = \{\{q_1, D_1\}, ..., \{q_i, D_i\}, ..., \{q_n, D_n\}\}$, 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 $D = \{d_1, d_2, ...\}$, 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:

$$p(d|q, U) = \phi\left(p\left(d|U^q\right), p(q|U^d), p\left(d|q\right)\right) \quad (1)$$

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 in our model, which is used to combine the three parts with different weights.

The structure of our model is shown in Figure 1. At first, 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 generate the refined vectors and 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 TF-IDF weights. (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.

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 Query Memory. We construct this memory for handling the query-based re-finding. 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, 10, 32]. 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. $M^Q = \{m_1^Q, ..., m_{n_Q}^Q\}$. 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

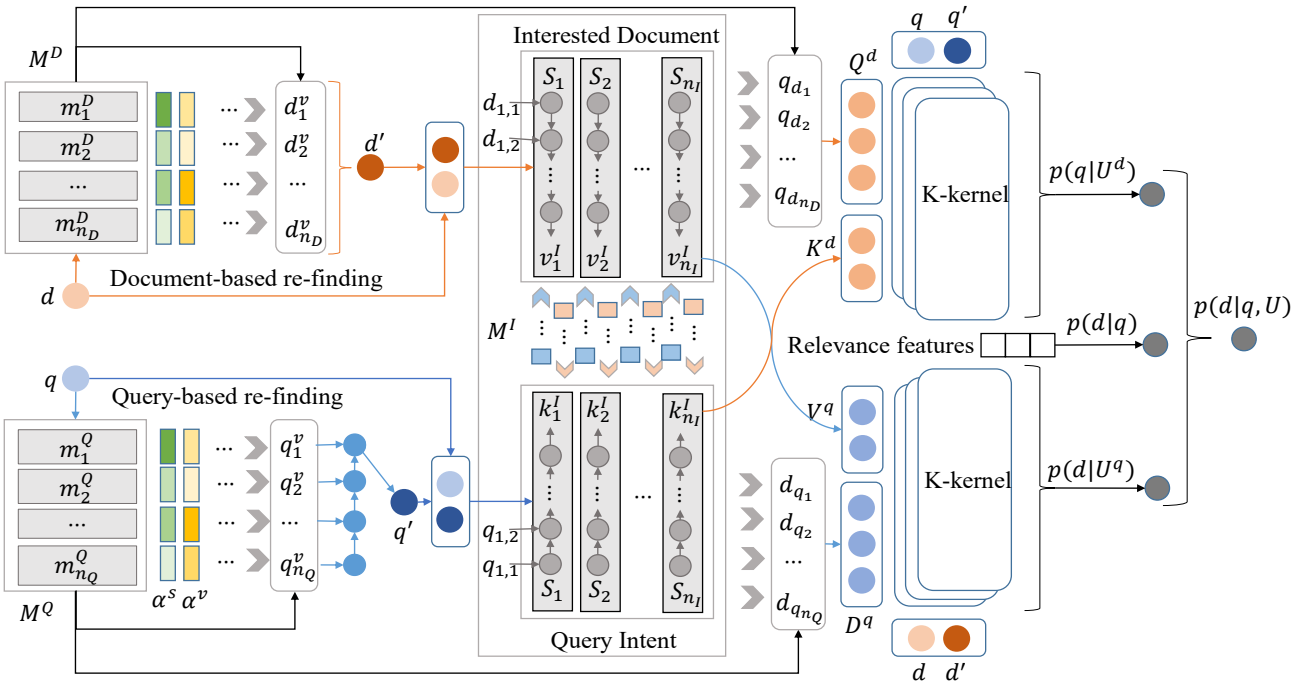

Figure 1: The architecture of RPMN. Given a new query and a candidate document, relevant historical behaviors are highlighted by two external memories from query and document. After extracting session-based re-finding behavior using intent memory based on the current needs, personalized information for 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.

the two average document vectors of corresponding slot:

$$d_{q_i}^+(new) \leftarrow \text{GATE}(d_q^+, d_{q_i}^+(old))$$
$$d_{q_i}^-(new) \leftarrow \text{GATE}(d_q^-, d_{q_i}^-(old)) \tag{2}$$

where GATE$(\cdot)$ is a gate to control the proportion of new information, 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 $\alpha_i^{q^v}$:

$$\alpha_i^q = \phi(\alpha_i^{q^s}, \alpha_i^{q^v}). \tag{3}$$

For string level weight, we choose ten common ways of query change following previous work ("wordorder", "stemming", etc.) [24, 28]. 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:

$$r_j(new) \leftarrow \text{GATE}(f(q^v - q_i^v), r_j(old)), \tag{4}$$

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 $r_i^{q^s}$:

$$\alpha_i^{q^s} = \frac{\exp(\phi(r_i^{q^s}))}{\sum_{j=1}^n \exp(\phi(r_j^{q^s}))}, \tag{5}$$

where the MLP $\phi(\cdot)$ is to output a weight based on the relation vector. We use the function 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 [1]:

$$\alpha_i^{q^v} = \text{softmax}(\phi(q^v, q_i^v)). \tag{6}$$

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 $Q^q = \{\alpha_1^q q_1^v, ...\alpha_{n_Q}^q q_{n_Q}^v\}$, weighted satisfied document vector set $D^{+,q} = \{\alpha_1^q d_{q_1}^+, ...\alpha_{n_Q}^q d_{q_{n_Q}}^+\}$, weighted skipped document vector set $D^{-,q} = \{\alpha_1^q d_{q_1}^-, ...\alpha_{n_Q}^q d_{q_{n_Q}}^-\}$. And they act on calculating the final probability in Section 3.3.

3.1.2 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 $M^D = \{m_1^D, ...m_i^D, ..., m_{n_D}^D\}$. 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. $\{\{d_1^+, q\}, \{d_2^+, q\}, ...\}$. And then we put each of them into the document memory M^D like query memory: modify the q_i by 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:

$$\alpha_i^{d^v} = \text{softmax}\left(\phi\left(d^v, d_i^v\right)\right). \tag{7}$$

By combining two parts of weight, we highlight user behaviors on relevant documents and get two weighted sets: weighted document vector set $D^d = \{\alpha_1^d d_1^v, ..., \alpha_{n_D}^d d_{n_D}^v\}$, weighted average query vector set $Q^d = \{\alpha_1^d q_{d_1}, ..., \alpha_{n_D}^d q_{d_{n_D}}\}$. They will contribute to the final probability along with the sets obtained 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, $U = \{S_1, S_2, ...\}$, 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 $\{q_{i,1}, q_{i,2}, ...\}$ in the session S_i, and each query corresponds to an average satisfied document vector $\{d_{i,1}, d_{i,2}, ...\}$. 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 and widespread application of RNN in modeling sequential data, we apply it to learn the representation of the session-based intent and interest. A major shortcoming of RNN is that when dealing with long sequences, there is a problem of vanishing gradient. Thus, more complicated structures based on RNN, such as Gated Recurrent Unit (GRU) [7] and Long Short-Term Memory (LSTM) [13], were proposed to solve the problem. We adopt GRU as the basic cell in our work since it is simpler than LSTM and is easier to train. Two GRU layers are applied to model user query intent from $\{q_{i,1}, q_{i,2}, ...\}$ and user interested document from $\{d_{i,1}, d_{i,2}, ...\}$ 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:

$$\begin{aligned} k_i^I(new) &\leftarrow \text{GRU}(k_i^I(old), q^v), \\ v_i^I(new) &\leftarrow \text{GRU}(v_i^I(old), d^+), \end{aligned} \tag{8}$$

where 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 from query and document. 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. According to the two types of re-finding behavior we introduced above, the *READ* operation of intent memory also 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 $Q^q = \{\alpha_1^q q_1^v, ...\alpha_{n_Q}^q q_{n_Q}^v\}$ obtained in Section 3.1.1. 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':

$$q' = \phi(h_{n_Q}^q) = \phi(\text{GRU}(h_{n_Q-1}^q, \alpha_{n_Q}^q q_{n_Q}^v)). \tag{9}$$

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

$$\alpha_i^{I,q} = \text{softmax}(\phi(k_i^I, [q^v, q'])). \tag{10}$$

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

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:

$$d' = \sum_{i=1}^{n} \alpha_i^d d_i^v. \tag{11}$$

And the weights on query intents with respect to d^v and d' is:

$$\alpha_i^{I,d} = \text{softmax}(\phi(v_i^I, [d^v, d'])). \tag{12}$$

The probability distribution of historical intents based on d is denoted as the set $K^d = \{\alpha_1^{I,d} k_1^I, ..., \alpha_{n_I}^{I,d} k_{n_I}^I\}$. These two sets from intent memory are essential in calculating 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 3.1.1. (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:

$$p(d|U^q) = \phi(p(d|U^{+,q}), p(d|U^{-,q})). \tag{13}$$

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 refined document d' together as the target:

$$\begin{aligned} p(d|U^{+,q}) &= F_k([d, d'], [D^{+,q}, V^q]), \\ p(d|U^{-,q}) &= F_k([d, d'], D^{-,q}), \end{aligned} \tag{14}$$

where F_k is the matching function which follows the idea of the previous model K-NRM [37]. 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):

$$\begin{aligned} F_k(M_{ij}^+) &= \phi(f_1(M_{ij}^+), ..., f_o(M_{ij}^+), , ..., f_k(M_{ij}^+)), \\ f_o(M_{ij}^+) &= \sum_i log\left(\sum_j exp\left(-\frac{(M_{ij}^+ - \mu_o)^2}{2\sigma_o^2}\right)\right), \end{aligned} \tag{15}$$

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 3.1.2. (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':

$$p(q|U^d) = F_k([q, q'], [Q^d, K^d]). \tag{16}$$

(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:

$$p(q|d) = \phi(f_{q,d}). \tag{17}$$

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_{ij}^t and the predicted distance dis_{ij}^p as loss function, and we have:

$$loss = -|\lambda_{ij}| \left(dis_{ij}^t log(dis_{ij}^p) + (1 - dis_{ij}^t) log(1 - dis_{ij}^p)\right), \tag{18}$$

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

In summary, we propose a method to enhance the re-finding behavior in personalized search with memory networks. For each user, we store his historical behaviors into the memories from query, document, and intent. Given a new query and its candidate documents, we take the personalized information associated with them from our memories to predict the probability of each document being clicked. The personalized ranking is based on it. This new interaction is recorded to update the memories.

4 EXPERIMENTS

4.1 Dataset and Evaluation Metrics

We experiment with a large-scale query log of a commercial search engine, which 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 [35]. Finally, the dataset contains 738,731 queries issued by 5,998 users. These queries belong to 276,047 different sessions.

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 [20] for

the method based on word embedding, and utilize node2vec [11] to learn the representations of graph embedding.

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 [15]. Based on the consideration that a satisfied clicked document is more relevant than the skipped documents and the next unclicked document, following [18], we construct the inverse document pairs by them and take three metrics #Better, #Worse, and P-Imp. to evaluate the results.

4.2 Baselines and Our Models

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

P-Click [8]: This method counts the click number on the same document under the same query in user's history, and generates the personalized results by fusing the original ranking.

URP [28]: It extracts three types of information (Query Change, Personalized and shared features) to identify the re-finding and utilizes it to predict the user behavior 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 [18]: 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.

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

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

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.

We experiment with multiple sets of parameters, including GRU hidden state size in {200, **400**, 600}, the number of MLP hidden units in {64, 128, **256**, 512}, word embedding size in **300**, 1000}, graph embedding size in {**300**, 1000}, the number of kernel in {5, 7, 9, **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.

Table 2: Overall performances of models. "†" indicates the model outperforms all baselines significantly with paired t-test at $p < 0.05$ level. The best results are shown in bold.

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-QM	.8184†	.8298†	1.772†	13449	614	.2462
RPMN-DM	.8196†	.8301†	1.765†	13409	**588**	.2460
RPMN-IM	.8215†	.8312†	1.759†	14085	718	.2545†
RPMN	**.8238**†	**.8342**†	**1.745**†	**14735**	890	**.2656**†

4.3 Overall Results and Analysis

We evaluate the results of the different methods on the test set. The overall results are shown in Table 2. We can observe that:

(1) Personalized baselines vs. original ranking. All personalized strategies outperform the original ranking generated by search engine. 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 vs. baselines. Our proposed methods outperform baseline models in all evaluation metrics. 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 complete model RPMN has increased by 1.27% on MAP and 6.7% on P-Imp. As can be seen from the reduction of #Worse, our models have a lower risk of mistakes and their improvements are mostly reliable. These results show that it is feasible to enhance user re-finding behavior with external memories.

(3) RPMN vs. other methods we designed. The complete model outperforms other models that lack a memory. Specifically, removing the query memory causes a decline of 0.7% on MAP, whose impact is greater than others. 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 has less influence on the model. A possible reason is that most of the session-based re-finding can be explored by the other two memories. And the intent memory is used to discover re-finding behavior that is more implicit.

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

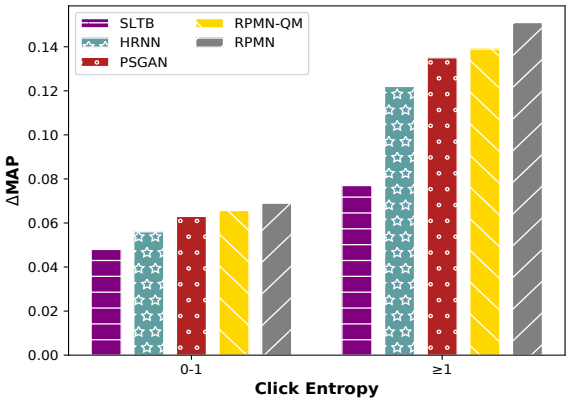

Figure 2: The results on queries with different click entropies

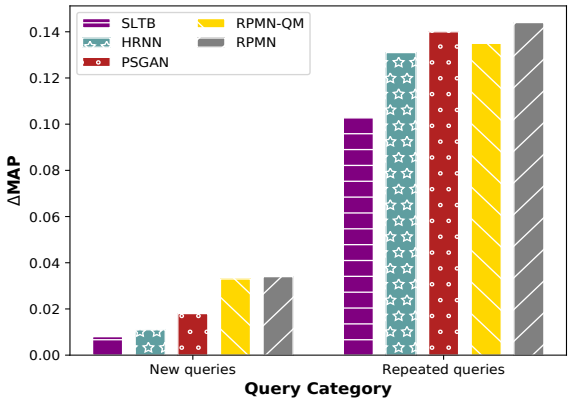

Figure 3: The results on repeat queries and new queries

our model improves on, we test the performance of our model on the different query sets in the remaining parts of this section.

4.4 Results on Different Query Sets

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

Informational queries vs. navigational queries. Previous studies have shown that user queries can be divided into navigation queries and informational queries according to the intent [8, 10, 25]. 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 Figure 2, all the personalized methods contribute more on informational queries (with larger click entropy) than navigational queries (with lower click entropy). Our models outperform the baselines on both query sets. 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.

Repeated queries vs. 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 Figure 3, we find that all personalized models have improved search quality on both query sets, while the improvement

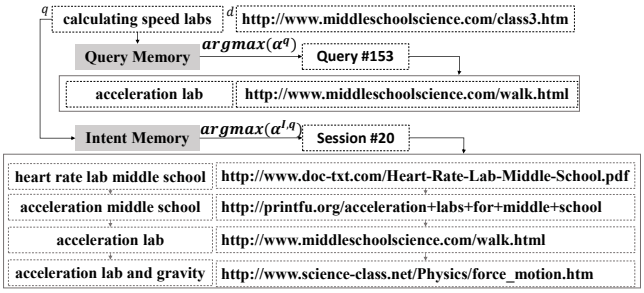

Figure 4: The case study for interpretability of RPMN

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 traditional 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.

4.5 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. In order to demonstrate the ability to explore potential re-finding behavior, we choose a new query-document pair to predict.

As shown in Figure 4, 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 effectively explain the personalized results.

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. 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. Experimental results confirmed the effectiveness and interpretability of our proposed model. In the future, we plan to design more ways to read from memory.

ACKNOWLEDGMENTS

We thank the reviewers' helpful comments. Zhicheng Dou is the corresponding author. This work was supported by National Natural Science Foundation of China No. 61872370, Beijing Outstanding Young Scientist Program NO. BJJWZYJH012019100020098, and the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China No. 2112018391.

REFERENCES

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

[2] Paul N Bennett, Krysta Svore, and Susan T Dumais. 2010. Classification-enhanced ranking. In *Proceedings of the WWW'2010*. ACM, 111–120.

[3] Paul N Bennett, Ryen W White, Wei Chu, Susan T Dumais, Peter Bailey, Fedor Borisyuk, and Xiaoyuan Cui. 2012. Modeling the impact of short-and long-term behavior on search personalization. In *Proceedings of the SIGIR'2012*. ACM, 185–194.

[4] Christopher Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine learning (ICML-05)*. 89–96.

[5] Fei Cai, Shangsong Liang, and Maarten De Rijke. 2014. Personalized document re-ranking based on bayesian probabilistic matrix factorization. In *Proceedings of the SIGIR'2014*. ACM, 835–838.

[6] Mark J. Carman, Fabio Crestani, Morgan Harvey, and Mark Baillie. 2010. Towards query log based personalization using topic models. In *Proceedings of the CIKM'2010*. 1849–1852.

[7] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP'2014*. 1724–1734.

[8] Zhicheng Dou, Ruihua Song, and Ji-Rong Wen. 2007. A large-scale evaluation and analysis of personalized search strategies. In *WWW'2007*. ACM, 581–590.

[9] David Elsweiler and Ian Ruthven. 2007. Towards task-based personal information management evaluations. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 23–30.

[10] Songwei Ge, Zhicheng Dou, Zhengbao Jiang, Jian-Yun Nie, and Ji-Rong Wen. 2018. Personalizing Search Results Using Hierarchical RNN with Query-aware Attention. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM '18)*. ACM, New York, NY, USA, 347–356. https://doi.org/10.1145/3269206.3271728

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

[12] Morgan Harvey, Fabio Crestani, and Mark J Carman. 2013. Building user profiles from topic models for personalised search. In *CIKM'2013*. ACM, 2309–2314.

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

[14] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 505–514.

[15] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR'2005*. 154–161.

[16] Alexander Kotov, Paul N Bennett, Ryen W White, Susan T Dumais, and Jaime Teevan. 2011. Modeling and analysis of cross-session search tasks. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 5–14.

[17] Xiujun Li, Chenlei Guo, Wei Chu, Ye-Yi Wang, and Jude Shavlik. 2014. Deep learning powered in-session contextual ranking using clickthrough data. In *NIPS'2014*.

[18] Shuqi Lu, Zhicheng Dou, Xu Jun, Jian-Yun Nie, and Ji-Rong Wen. 2019. PSGAN: A Minimax Game for Personalized Search with Limited and Noisy Click Data. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. ACM, New York, NY, USA, 555–564. https://doi.org/10.1145/3331184.3331218

[19] Nicolaas Matthijs and Filip Radlinski. 2011. Personalizing web search using long term browsing history. In *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 25–34.

[20] Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168* (2013).

[21] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126* (2016).

[22] Ahu Sieg, Bamshad Mobasher, and Robin Burke. 2007. Web search personalization with ontological user profiles. In *CIKM'2007*. ACM, 525–534.

[23] Yang Song, Hongning Wang, and Xiaodong He. 2014. Adapting deep ranknet for personalized search. In *WSDM'2014*. ACM, 83–92.

[24] Jaime Teevan, Eytan Adar, Rosie Jones, and Michael AS Potts. 2007. Information re-retrieval: repeat queries in Yahoo's logs. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 151–158.

[25] Jaime Teevan, Susan T Dumais, and Daniel J Liebling. 2008. To personalize or not to personalize: modeling queries with variation in user intent. In *SIGIR'2008*. ACM, 163–170.

[26] Jaime Teevan, Daniel J Liebling, and Gayathri Ravichandran Geetha. 2011. Understanding and predicting personal navigation. In *WSDM'2011*. ACM, 85–94.

[27] Sarah K Tyler and Jaime Teevan. 2010. Large scale query log analysis of re-finding. In *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 191–200.

[28] Sarah K Tyler, Jian Wang, and Yi Zhang. 2010. Utilizing re-finding for personalized information retrieval. In *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM, 1469–1472.

[29] Maksims Volkovs. 2015. Context models for web search personalization. *arXiv preprint arXiv:1502.00527* (2015).

[30] Thanh Vu, Dat Quoc Nguyen, Mark Johnson, Dawei Song, and Alistair Willis. 2017. Search personalization with embeddings. In *ECIR'2017*. Springer, 598–604.

[31] Thanh Vu, Dawei Song, Alistair Willis, Son Ngoc Tran, and Jingfei Li. 2014. Improving search personalisation with dynamic group formation. In *SIGIR'2014*. 951–954.

[32] Thanh Vu, Alistair Willis, Son N Tran, and Dawei Song. 2015. Temporal latent topic user profiles for search personalisation. In *ECIR'2015*. Springer, 605–616.

[33] Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916* (2014).

[34] Ryen W White, Wei Chu, Ahmed Hassan, Xiaodong He, Yang Song, and Hongning Wang. 2013. Enhancing personalized search by mining and modeling task behavior. In *WWW'2013*. ACM, 1411–1420.

[35] Ryen W White and Steven M Drucker. 2007. Investigating behavioral variability in web search. In *Proceedings of the 16th international conference on World Wide Web*. ACM, 21–30.

[36] Qiang Wu, Chris JC Burges, Krysta M Svore, and Jianfeng Gao. 2008. *Ranking, boosting, and model adaptation. Tecnical Report.* Technical Report. MSR-TR-2008-109.

[37] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR conference on research and development in information retrieval*. ACM, 55–64.