# Clarifying Ambiguous Keywords with Personal Word Embeddings for Personalized Search

JING YAO, School of Information, Renmin University of China
ZHICHENG DOU, Gaoling School of Artificial Intelligence, Renmin University of China
JI-RONG WEN, Beijing Key Laboratory of Big Data Management and Analysis Methods, Key Laboratory of Data Engineering and Knowledge Engineering, MOE

Personalized search tailors document ranking lists for each individual user based on her interests and query intent to better satisfy the user's information need. Many personalized search models have been proposed. They first build a user interest profile from the user's search history, and then re-rank the documents based on the personalized matching scores between the created profile and candidate documents. In this article, we attempt to solve the personalized search problem from an alternative perspective of clarifying the user's intention of the current query. We know that there are many ambiguous words in natural language such as "Apple." People with different knowledge backgrounds and interests have personalized understandings of these words. Therefore, we propose a personalized search model with personal word embeddings for each individual user that mainly contain the word meanings that the user already knows and can reflect the user interests. To learn great personal word embeddings, we design a pre-training model that captures both the textual information of the query log and the information about user interests contained in the click-through data represented as a graph structure. With personal word embeddings, we obtain the personalized word and context-aware representations of the query and documents. Furthermore, we also employ the current session as the short-term search context to dynamically disambiguate the current query. Finally, we use a matching model to calculate the matching score between the personalized query and document representations for ranking. Experimental results on two large-scale query logs show that our designed model significantly outperforms state-of-the-art personalization models.

CCS Concepts: • **Information systems** → **Personalization**; • **Computing methodologies** → **Reinforcement learning;**

Additional Key Words and Phrases: Personalized search, personal word embedding, session graph, user interest

**ACM Reference format:**
Jing Yao, Zhicheng Dou, and Ji-Rong Wen. 2021. Clarifying Ambiguous Keywords with Personal Word Embeddings for Personalized Search. *ACM Trans. Inf. Syst.* 40, 3, Article 43 (November 2021), 29 pages.
https://doi.org/10.1145/3470564

This work was supported by National Natural Science Foundation of China No. 61872370 and No. 61832017, and Beijing Outstanding Young Scientist Program NO. BJJWZYJH012019100020098.
Authors' addresses: J. Yao, School of Information, Renmin University of China, Beijing, 100872; email: jing_yao@ruc.edu.cn; Z. Dou (corresponding author), Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, 100872; email: dou@ruc.edu.cn; J.-R. Wen, Beijing Key Laboratory of Big Data Management and Analysis Methods, Key Laboratory of Data Engineering and Knowledge Engineering, MOE, Beijing, 100872; email: jrwen@ruc.edu.cn.

1046-8188/2021/11-ART43 $15.00
https://doi.org/10.1145/3470564

## 1 INTRODUCTION

The search engine has become a common approach for us to obtain information from the Web in our daily life. Given a query, it generates a document ranking list in which documents are ordered by their relevance to the query. Obviously, it is difficult to meet the information needs of all users by returning them the same search result for the same query, because there are many ambiguous words in natural language and different users may have different query intents when entering these keywords. For example, the word "Apple" has two common meanings, i.e., the "Apple" fruit and the "Apple" company or products. An IT engineer may use this keyword to search for information about the "Apple" company or products, while a fruit farmer tends to seek information related to the "Apple" fruit. From this example, we can find that the IT engineer and the fruit farmer have their personalized understandings of the word "Apple" due to their different knowledge backgrounds and interests. To improve search result quality and user satisfaction, personalized search tries to model the user's preference and return more specific ranking results based on it.

Many personalized search models have been studied. Traditional personalization strategies [5, 9, 10, 17, 21, 43–45] mainly depend on click-based, topic-based, and other features extracted from the search history to analyze user interests. With the emergence of deep learning, new personalized search models [18, 29, 51] have achieved better personalization by learning representations of the user preferences with neural networks. The common idea of most existing personalization models is constructing a user interest profile on the search history at first and then adjusting the general document ranking list on account of the matching scores between the candidate documents and the created user profile. However, we argue that these models mainly consider the user interests reflected by the search history, but do not disambiguate the current query to figure out the user's real search intent. In this article, **we attempt to solve the personalized search problem from an alternative perspective of clarifying the ambiguous query keywords with personal word embeddings trained for each individual user.**

As shown in the previous "Apple" example, there may be a lot of ambiguous query keywords, and different users have personalized understandings of such words' meanings due to their individual knowledge backgrounds and interests. To clarify the specific meaning of such ambiguous keywords and disambiguate the current query, we claim that the same word for different users should be viewed as different words and own different semantic representations. Therefore, we propose a personalized search model that sets personal word embeddings for each user trained from her individual search history. In the user's query log, we consider that there are mainly two kinds of information contributing to user interests. One is the textual information of the issued queries and clicked documents, which reflects the user's already known word meanings and can be learned by a language model. The other is the user behavior information, including the query reformulation process and clicks. We employ a graph structure to better represent the user behavior in the query log, instead of a linear query sequence. Thus, we are able to train personal word embeddings that mainly contain the word meanings that the user already knows and user interests. Using such personal word embeddings for personalized search has several advantages. First, the user's personalized query intention can be clarified by representing each query keyword with personal word embedding. The documents that meet the user's query intent can be better matched. Second, we can directly re-rank the documents based on the matching scores between the personalized representation of the query and documents. The user interests reflected in the search history have been embedded in the personal word vectors after training the vectors on the user's search log. Hence it is not necessary for us to analyze the user's search history and build the user interest profile every time we re-rank the documents for a new query issued by the user, saving a large amount of time for maintaining the user profile.

Specifically, we design a **P**ersonalized **S**earch model **PEPS+** based on the idea of **P**ersonal word **E**mbeddings. The model consists of three stages. In the **first stage**, we construct a session graph to represent the user's search process and click behavior in the log data. And we design a pre-training model to train personal word embeddings for each user that preserve both the user's already known word meanings and the user interests encoded in the session behavior graph. In the **second stage**, we build a personalized ranking model to generate personalized document lists, with a personalized word embedding layer containing pre-trained embedding matrices for each user. This ranking model is composed of four modules. In the first module, we get personalized representations at the word level, and we calculate the contextual representations of the query and documents through a multi-head self-attention layer to fuse the contextual information for disambiguation. Second, we use a graph neural network [48] to combine the information of the past queries and clicked documents in the current session represented by a session graph and generate the representation of session-based query intent. With these personalized representations ready, we compute the matching score for document ranking with the KNRM component. Finally, we utilize two tasks, personalized document ranking and query reformulation, to jointly train the personal word embeddings and ranking model in an end-to-end way. In the third stage, to fit the user's dynamically changing interests, we further devise three online update approaches to track the new user interests along with the search process. To compare our PEPS+ model with state-of-the-art baselines, we conduct experiments on the publicly available AOL dataset and a large-scale commercial dataset. Experimental results show that our method can yield significant improvements in ranking results and higher performance over existing personalized search models.

To conclude, our main contributions are four-fold:

- We attempt to solve the problem of search results personalization from an alternative perspective of clarifying the ambiguous query keywords with personalized word representations to disambiguate the current query, without building and maintaining user interest profiles.
- Based on this idea, we design a pre-training model to train personal word embeddings for each user on their own search history, which captures not only the textual information in the query log but also the user interests reflected in the user's query reformulation process and click behavior.
- With the pre-trained personal word embeddings, we propose a ranking model, in which we first obtain the personalized representations of the query and documents, then compute their matching scores to generate the ranking list. Moreover, we also incorporate the information of the current session encoded in the session graph for further disambiguation.
- We design three approaches to update the personal word embeddings along with the search process to capture the changes of user interests reflected in the newly issued queries.

This article is an extended version of a paper presented at the SIGIR 2020. We expand the PEPS model in the original paper to the PEPS+, and the main extensions are: (1) As for the personal word embedding for each user, the original PEPS model simply initializes it with the Word2vec model trained from the corresponding user's query log, which only learns textual information but ignores others. To obtain better personal word embeddings that contain more accurate user interests, we represent the user's search behavior with a graph structure and build a new pre-training model to generate word embeddings that preserve both textual and structural information. (2) In PEPS, we calculate the personalized representations of the query and document for matching. However, the personal word embeddings are trained on the user's whole search history and may still contain more than the one used meaning. We know that the user's query intent in a session is usually similar. Thus, we add a session-based query intent representation module to dynamically disambiguate the query by employing GCN to incorporate the information in the current session.

The rest of the article is organized as follows: Related works are introduced in Section 2. In Section 3, we elaborate our proposed model. We describe the experimental settings in Section 4, present and analyze the experimental results in Section 5. The whole article is concluded in Section 6.

## 2 RELATED WORK

In general, this article is related to two parts of studies: (1) personalized search models and (2) graph-based methods in **information retrieval (IR)**. We separately introduce these two research fields in the next parts.

### 2.1 Personalized Search

Search results personalization has been proved to effectively improve the quality of search results by introducing user preferences, especially for those ambiguous queries [17, 18, 40, 41]. There is a wealth of related studies.

**Traditional Personalized Search Model.** Traditional personalized search models rely on some heuristic rules to analyze user interests. Motivated by the re-finding case where users sometimes issue the same query to search for the same documents, Dou et al. [17] proposed the P-Click model. This model evaluates the document relevance by counting how many times this document has been clicked by the same user under the same query in the search history. Topics of the clicked documents and issued queries are usually used to build user interest profiles. Some models [36, 47] applied a topic model such as **Latent Dirichlet Allocation (LDA)** [6, 47] to extract topic-based features, then build user interest profiles in the topic space. However, such topic models might suffer from huge labor costs and incomplete categories. Later models [10, 21, 43, 44] attempt to solve these problems by automatically learning topic representations and obtain the user interest vector. Some other studies [5, 45] realized personalization through feature engineering. They manually extract a series of features from the current query and the user's search history, including the original document rank position, query entropy, click-based features, topic-based features, and so on. Then, a learning to rank algorithm [7, 8] is used to combine these features in a non-linear way and train a ranking model. In addition to the features related to queries and clicks, location information and the user's reading level [4, 12] were also considered to help personalization. Traditional personalized models have achieved great progress, but most of them only focused on using some specific features to describe user interests, thus ignoring other information that is the same valuable.

**Deep Learning-based Personalized Search.** As deep learning becomes popular, many personalized search models based on learning have been studied. Due to the powerful representation learning ability of deep learning, we can capture potential user preferences. The problem that the representation ability of traditional manual-designed features is limited has been gradually relieved. These learning-based personalized search models mainly follow two kinds of approaches. One is the adaptation framework [37], which adapts the general ranking model to a personalized model by fine-tuning it with a few queries from that user. The other more common method is to learn an explicit representation of the user interest profile from the search history. Ge et al. [18] designed a **hierarchical RNN model with query-aware attention (HRNN)** to learn sequential information hidden in the search process and dynamically build user interest profiles according to the current query. Lu et al. [29] devised a personalized search framework PSGAN that applied the **generative adversarial network (GAN)** [19] to enforce the model pays more attention to those indistinguishable document pairs and learns more accurate user profile. Yao et al. [51] proposed a reinforcement learning-based personalized search model, employing a hierarchical **Markov Decision Process (MDP)** to track the users' interactions with the search engine and continuously update the ranking strategy. Zhou et al. [53] applied the memory network to help identify the user's more complex re-finding intentions. These models adjust the original document lists based on the created user interest profiles and achieve state-of-the-art performance. In this article, we propose a novel personalized search model from a different perspective, without building and maintaining explicit user interest profiles.

**Word Embedding for Personalization.** In recent years, there are a few models [3, 22, 28, 32, 35] attempting to apply word embeddings for personalization. Typically, Samarawickrama et al. [35] first trained a personalized neural language model on the user's search history and created a synonym table for the user based on the word similarity computed with the word embeddings. Then, they re-rank the documents based on the cosine similarity between the query keywords' synonyms and document terms. Amer et al. [3] applied word embeddings to find words similar with query keywords for query expansion and computed personalized relevance as the match score between the expanded query and document. These personalized search models used word embeddings to calculate word similarity and find some synonyms to indicate user interests and help document re-ranking. However, our model is different, which directly trains personal word embeddings containing user interests and personalized meanings for each user and computes relevance score by matching the personalized query and document representations. Furthermore, these existing models trained word vectors by merely unsupervised language models without user's click labels, which are the most credible indication of user preferences. Our designed PEPS+ is a novel personalized search model. It trained personal word vectors with both the user's behavior information and the textual information in the search log to embed the user interests, solving the problems faced by the above embedding-based models.

## 2.2 Graph-based Methods in IR

Graph-based methods have been widely studied and exploited to cope with different tasks in information retrieval. Based on the linking structure between web pages, several ranking algorithms, such as PageRank and HITS [27], have been proposed. A lot of existing studies utilized a graph structure to represent various user behavior in the search process and mine information from it. Ma et al. [30] constructed two bipartite graphs (user-query bipartite graph and query-URL bipartite graph) on the user's click-through data and then extracted information from the two graphs to infer the user's query intent for better query suggestion. Jiang et al. [24] designed a vector propagation algorithm that was applied on the click bipartite graph to enrich vector representations for queries and documents in the same semantic space. Zhang et al. [52] integrated the features about the click graph into a unified neural ranking model to improve product search performance by pre-training representations for the items and queries based on the query and products graph. For most existing models using graph structures, their key idea is to enrich the representation of nodes, which are usually queries and documents with structural information. Currently, the existing methods of learning node information from the graph structure can be categorized into two classes: network embeddings and graph neural networks [48]. The former mainly leverages the skip-grams [31] method in the language model to select positive and negative samples to capture the relationships between graph nodes. Typically, DeepWalk [33] and Node2vec [20] apply the skip-gram approach to learn node vectors with sequences generated by random walks. The latter one, graph neural networks, obtain node representation by recursively aggregating information from the neighbors. GCN [15, 26] is the most widely used deep technique for graphs, which generalizes the convolutional filters from images to graphs. In this article, we leverage the graph structure to represent complex user behavior in the search process, including query reformulation and document clicks. Then, we mine information in the behavior graph with GCN [26] to learn personal word embeddings that reflect the user interests better.

## 3 PEPS+, A PERSONAL WORD EMBEDDING-BASED PERSONALIZED SEARCH MODEL

As we stated in Section 1, most existing personalized search models either extract features related to user interests or learn a representation of user interest profile from the search history to help re-rank the document list. However, they did not pay attention to disambiguating the current query. Differently, we propose a personalized search model with personal word embeddings for each individual user to tackle the problem of search result personalization in an alternative way. Through training personal word embeddings on the corresponding user's query log with both the textual information and search behavior, we are able to obtain word vectors that mainly contain the word meanings the user already knows and user interests. Using such personal word embeddings to represent the current query, the user's real query intent can be clarified and the satisfied documents can be better matched.

To begin with, we formulate the problem with notations. Suppose that there are a lot of individual users, represented as $u_1, u_2, \ldots$, each user $u_i$ has her own search history $H_i$ at the current time. Following Reference [18], we split the whole search history $H_i$ into many sessions $H_i = \{S_1^i, \ldots, S_{M-1}^i, S_M^i\}$, where $M$ is the index of the current search session. Each session $S_m^i$ includes a sequence of queries $\{q_{m,1}^i, q_{m,2}^i, \ldots, q_{m,n_m}^i\}$ issued by the user $u_i$, with the clicked document set $D_{m,j}^{i,+}$ and un-clicked documents set $D_{m,j}^{i,-}$ under each query $q_{m,j}^i$. $n_m$ represents the total number of queries in the session $S_m^i$. Each query consists of a series of keywords, $q_{m,j}^i = \{w_1, w_2, \ldots, w_{|q_{m,1}^i|}\}$. Referring to Reference [18], the user's search behavior within a session is thought to show similar intents. Search sequences mainly reflect some sequential information about the user's behavior and interests, but might ignore some other potential relationships. To better explore the user's search process and capture user interests, we choose to use an undirected graph to represent user behavior in her query history. We represent the behavior graph of user $u_i$ as $\mathcal{G}_{u_i} = (V, E)$, where $V$ denotes the node set and $E$ is the edge set. The graph is created in a unit of sessions with independent search intent. We mainly focus on two kinds of behavior that can reflect user interests: the query reformulation process in a session during which the user revised her input query to find the required information, and the document click behavior under each query. Thus, the node set $V$ is composed of all queries and clicked documents; the edge set $E$ contains edges between consecutive queries in the same session and edges indicating clicked documents under an issued query. Following the above rules, we construct the behavior graph for a user $u_i$ as shown in Figure 1. Currently, the user $u_i$ enters a new query $q = w_1, w_2, \ldots, w_{|q|}$, and the underlying non-personalized search engine returns a candidate document list $D = \{d_1, d_2, \ldots\}$. Our proposed model is expected to re-rank the candidate documents based on the information contained in the user's search history $H_i$, giving higher priority to the documents that match the user's query intent and interest.

Our proposed personalized search model, referred to as PEPS+, mainly includes three stages: (1) The first is the pre-training stage. We design a pre-training model to generate personal word embeddings for each individual user based on her search history. (2) The second is the offline training stage to train a personalized ranking model. This model represents queries and documents with the pre-trained personal word embeddings, then computes interaction scores to rank candidate documents. (3) The third is the online update stage. We update each user's personal word embeddings with her newly generated search data to involve her latest interests.

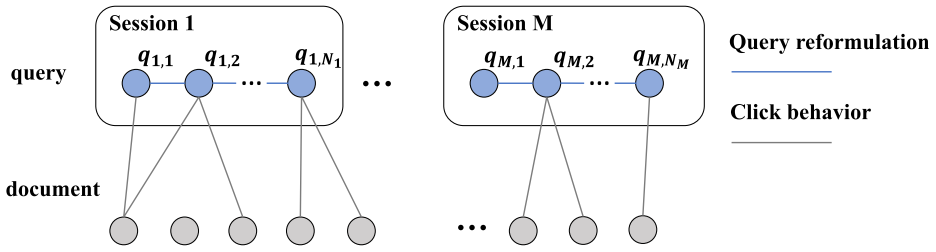

Fig. 1. The undirected graph structure of user behavior in the search process. The node set is composed of issued queries and clicked documents. The edge set contains edges between consecutive queries in the same session and edges between the issued query and clicked documents under this query.

In the next sections, we first describe the structure and tasks of the pre-training model. Then, we introduce each component of the personalized ranking model. In the final, we present three different approaches for online updates and make some discussions about our whole model.

## 3.1 Pre-training

In this article, we propose to train personal word embeddings that mainly contain the user-interested meanings to disambiguate the query keywords and achieve search results personalization. According to existing works [5, 17, 18], user interests are mainly reflected by the user's issued queries and clicked documents under each query. Therefore, the simplest approach to obtain personal word representations containing user interests is to train a personal language model on the user's corpus that consists of the user-issued queries and clicked documents [3, 35]. We implement this approach by training a personal Word2vec model [31] for each user with her own query log as one of our baselines, introduced as PPWE in Section 4.2. We know that the Word2vec model with the user's query log as the corpus is dedicated to capturing the textual and semantic information. However, it has difficulty in learning other structural information that is related to user interests, such as the sequential information hidden in the search history, how the query is reformulated by the user during the search process, and the document click behavior under each query. To cover more relationships, we represent the user behavior with a graph structure, as shown in Figure 1. We design a pre-training model based on the behavior graph to obtain personal word embeddings that contain the user interests and word meanings the user already knows. There are totally three pre-training tasks in this model, including the context prediction task to learn local textual and semantic information, relevance prediction task, and query sequence complementation task to learn user interests and structure proximity encoded in the user behavior graph. The architecture of the pre-training model is presented in Figure 2. In the following, we elaborate the settings of the pre-training architecture and the three pre-training tasks.

*3.1.1 Personal Word Embedding Layer.* In the pre-training model, there is a personal word embedding layer. In this layer, we keep personal word embedding matrices for each individual user and a shared global embedding matrix. We use two signs to identify a word, i.e., the word and the corresponding user ID, so the same word of different users are identified as different words. For example, the word "Apple" in the word embedding matrix of the user $u_i$ should be represented as "Apple + $u_i$," while "Apple + $u_j$" is for the user $u_j$. In this layer, the personal word embeddings are only pre-trained with the corresponding user's search log data. Thus, **the well-trained embedding of a specific word is not a general representation of various meanings of this word in the overall logs, but mainly the personalized meaning that the user already knows and can reflect the user interests.**

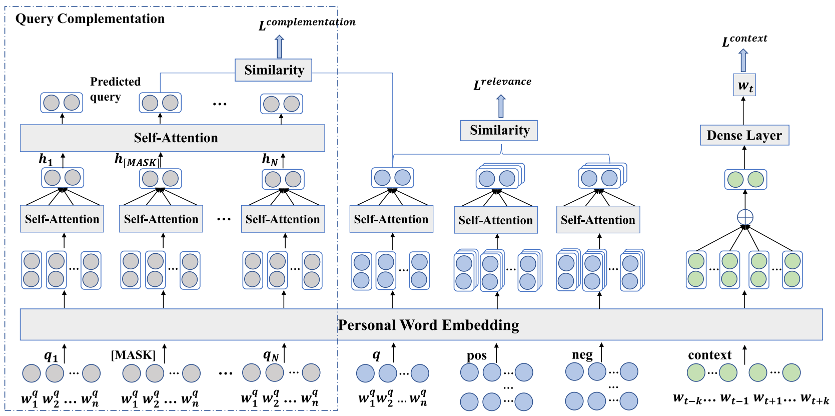

Fig. 2. Structure of the pre-training model. There is a well-designed personal word embedding layer and three pre-training tasks are employed for joint training, i.e., context prediction task, query sequence complementation task, and relevance prediction task.

We also have to decide the vocabulary of the personal word embedding matrix for each user. There is a global vocabulary on the whole query logs, but copying a global vocabulary for each user as the personal vocabulary yields several drawbacks: (1) Most words in the global vocabulary do not appear frequently in a user's query log, so it is unnecessary to maintain a complete global vocabulary for each user, which will take up a large amount of memory space. And (2) some words are not obviously ambiguous and informational, thus, we argue that it is better to train the embedding of these words on the whole logs. Based on these considerations, we keep a shared global word embedding matrix for all users and train it with the whole query logs. To construct personal vocabulary for each user, we filter the words in the global vocabulary according to several rules:

- words that are not stop words.
- words that occur in the user's log more than $c$ times.
- words with word entropy no less than *ent*.

$c$ and *ent* are hyper-parameters. In this article, we define the word entropy of a word $w$ as the average click entropy [17] of all queries containing that word, computed as:

$$\text{WordEntropy}(w) = \frac{\sum_{q \in Q(w)} \text{ClickEntropy}(q)}{|Q(w)|}, \tag{1}$$

$$\text{ClickEntropy}(q) = \sum_{d \in D(q)} -P(d|q) \log_2 P(d|q). \tag{2}$$

Here, $Q(w)$ represents the set of queries that contain the word $w$, and $D(q)$ is the collection of documents clicked under the query $q$. $P(d|q)$ is the probability of the clicks on document $d$ among all clicks of the query $q$, calculated as $P(d|q) = \frac{|\text{Clicks}(q,d)|}{|\text{Clicks}(q,\cdot)|}$.

With the filtered personal vocabulary, we can effectively control the storage space used by personal word vectors and the computation cost for updating the embeddings.

3.1.2 *Context Prediction.* To pre-train personal word embeddings, we adopt three pre-training tasks. The context prediction task is employed to learn semantic information. In a user's search log, we assume that the terms appeared in the issued queries and clicked documents express the meanings that the user already knows or is interested in. Thus, we refer to the classical language model CBOW [31] and perform the context prediction task on the user's personal search log to learn the personalized word embeddings. We regard each issued query or clicked document as a term sequence. Let $k$ denote the size of the context window. For each term $w_t$ in these sequences, the input is a total of $2k$ words before and after the target word $w_t$, i.e., $\{w_{t-k}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+k}\}$. They are passed through the personal word embedding layer, getting word vectors $\{v_{t-k}, \ldots, v_{t-1}, v_{t+1}, \ldots, v_{t+k}\}$. We compute the average of all these word vectors to obtain the context representation $c$. Then, the context representation $c$ is inputted into a dense layer with a parameter matrix $W$ to calculate the probability of the center word $p(w_t|\{w_{t-k}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+k}\})$. The whole calculation process is formulated as:

$$c = \frac{1}{2k} \sum_{i=-k}^{k} v_{t+k}, \tag{3}$$

$$u_t = W_{w_t} \cdot c, \tag{4}$$

$$p(w_t|\{w_{t-k}, \ldots, w_{t+k}\}) = \frac{\exp(u_t)}{\sum_i^{|V|} \exp(u_i)}. \tag{5}$$

$W_{w_t}$ are parameter vectors for the term $w_t$. Considering that there is a large number of words in the personal word embedding layer, we follow the negative sampling method [31] to select a small set of candidate words, denoted as $V$. $|V|$ is the size. The loss $L^{context}$ is computed as:

$$L^{context} = -\log p(w_t|\{w_{t-k}, \ldots, w_{t+k}\}). \tag{6}$$

*3.1.3 Query Sequence Complementation.* In addition to the textual information captured by the above context prediction task, we take the user interests reflected by the user's behavior into account. We illustrate the user's behavior in the whole search process with a graph structure, as shown in Figure 1. Observing the behavior graph, we mainly focus on two kinds of relationships. One is the transformation and reformulation between consecutive queries within the same session, located at the top of the graph. As stated in previous studies [18, 29], queries within a session usually express similar query intents. Thus, we expect the generated personal word embeddings to preserve the relationship between queries with different texts but similar intentions. We refer to BERT [16] model and design a pre-training task of query sequence complementation. The input is a sequence of $N$ consecutive queries under a session with one of the queries randomly masked, i.e., $\{q_1, [MASK], \ldots, q_N\}$. Each query consists of a series of terms, $q_i = \{w_1, w_2, \ldots, w_{|q_i|}\}$. Through the personal word embedding layer, the query $q_i$ is mapped into low-dimensional personalized word embeddings $P^{q_i} \in R^{dim \times |q_i|}$. To obtain the context-aware representation of each query $CR^{q_i}$, we first process the query word embeddings with a multi-head self-attention layer, and then sum up the representation of all the terms in the query.

$$CR^{q_i} = \sum_{j=1}^{|q_i|} CP_j^{q_i}, \tag{7}$$

$$CP^{q_i} = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h) W^A, \tag{8}$$

$$\text{head}_i = \text{softmax}\left(\frac{P^{q_i} W_i^Q (P^{q_i} W_i^K)^T}{\sqrt{d_k}}\right) \left(P^{q_i} W_i^V\right), \tag{9}$$

where $W_i^Q, W_i^K, W_i^V$, and $W^A$ are parameters of linear functions. With the context-aware representation of all queries in the sequence, $\{CR^{q_1}, \ldots, CR^{q_N}\}$, we pass them through another self-attention layer and get the output vector at the position of the masked query $PR^{[MASK]}$ as the predicted representation. We aim to maximize the similarity between the predicted representation and the vector of the true query $e_{q_i}$, which will be described in the next task, so the loss of this pre-training task is

$$L^{complementation} = 1 - sim(PR^{[MASK]}, e_{q_i}), \tag{10}$$

where $sim(\cdot)$ is the function of cosine similarity.

*3.1.4 Relevance Prediction.* In the behavior graph, besides the sequential relationship between queries in a session, the direct links between the query and query, query and document can also not be ignored. We believe these links also reflect information about user interests. To embed such structured proximity information, we implement a pre-training task of relevance prediction on the basis of the skip-gram representation method [31]. We sample a node in the graph and predict the nodes directly related to it. To be compatible with the above query sequence complementation task, we generate training samples with the masked query as the central node. Then, positive samples are all the nodes directly linked to the selected query, including the adjacent queries in the same session and the clicked documents of the query. In addition, we sample several negative nodes and adapt the negative sampling method for optimization. Negative samples mainly consist of the un-clicked documents of the query and the queries and documents randomly sampled under other sessions without links. The input of this pre-training task includes the central query, which is exactly the masked query in the query sequence complementation task, a positive set $S^+$ and a negative set $S^-$. For each inputted query or document, we calculate the contextual representation for them in the same way as described in the above task. Then, the loss function for training and optimization is as follows:

$$L^{relevance} = -\sum_{j \in S^+} \log \sigma\left(e_{q_i}^T e_j\right) - \sum_{k \in S^+} \log \sigma\left(-e_{q_i}^T e_k\right). \tag{11}$$

Here, $e_{q_i}$ denotes the contextual representation of the central query, and $e_j$, $e_k$ are for the positive samples and negative samples, respectively.

Finally, the loss of the pre-training model consists of three parts, i.e.,

$$L = \alpha_1 \cdot L^{context} + \alpha_2 \cdot L^{complementation} + \alpha_3 \cdot L^{relevance}. \tag{12}$$

$\alpha_1, \alpha_2, \alpha_3$ are weights to control the importance of each task. With the joint learning of the three pre-training tasks, we are able to obtain personal word representations that sufficiently contain the user interests reflected in the user behavior and semantic information in the query log.

Considering the query log of a single user is limited, we also suggest combining the query logs of similar users for pre-training. We refer to the user-based collaborative filtering algorithm [46] to find users with similar interests. We create a user similarity matrix $W$ and the interest similarity between two users $u_i$ and $u_j$ represented as $W_{ij}$ is calculated as:

$$W_{ij} = \frac{\sum_{d \in N(u_i) \cap N(u_j)} \frac{1}{\log(1 + |N(d)|)}}{\sqrt{|N(u_i)||N(u_j)|}}. \tag{13}$$

$N(u_i)$ and $N(u_j)$ represent the clicked document sets of user $u_i$ and $u_j$. $N(d)$ is the set of users who clicked the document $d$.

## 3.2 Personalized Ranking Model

After the pre-training stage, we obtain personal word embeddings for each user. The personal word embeddings mainly contain word meanings that the user knows and is interested in. Then, we design a personalized ranking model to promote the personalization results by clarifying the user's query intents with personal word embeddings. In the personalized ranking model, we are able to keep the personal word embeddings fixed or further fine-tune it with the corresponding

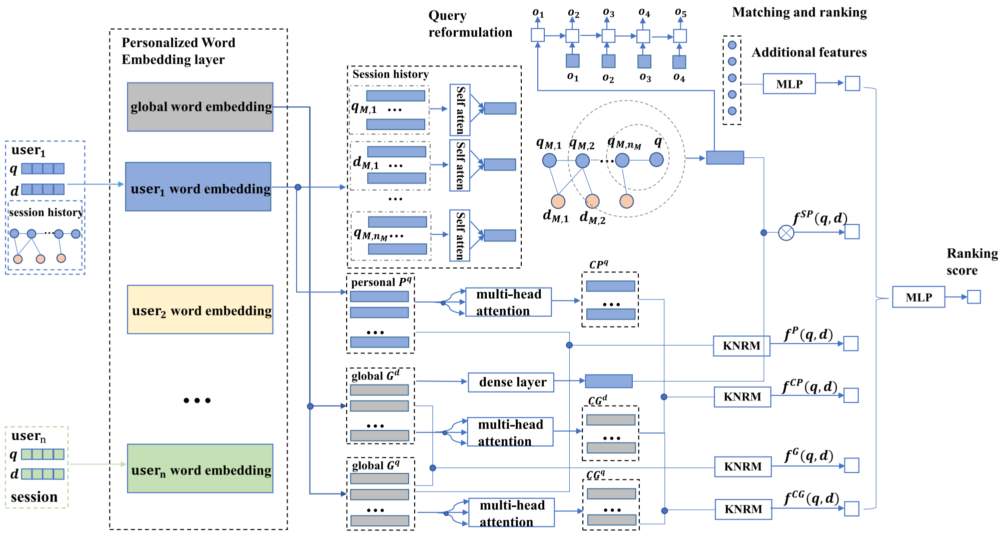

Fig. 3. Structure of the PEPS+. In the embedding layer, each user $u_i$ owns personal word embeddings pre-trained by the pre-training model on the user's data, and each word is identified by the word and user ID, such as "Apple + $u_i$." Given a query issued by $u_i$, a candidate document and current session queries, the embedding layer maps them to global and personal word representations, which are fed into the attention layer to obtain contextual representations and session representation. A GRU encodes the word representations to get query intent vectors used for query reformulation based on the Seq2Seq model. Finally, we compute the document relevance score with neural matching component KNRM and train the model through joint learning.

user's training samples. The whole architecture of the personalized ranking model is illustrated in Figure 3. We divide the model into four parts. In the first part, we get representations of the query and document from different granularities and perspectives. Second, we combine the short-term search history in the current session to calculate the session-based query intent representation. Third, we compute a personalized match score between the query and document through an interaction-based neural matching component KNRM [49]. We apply the pairwise LambdaRank [7] to train the ranking model. Finally, we design a query reformulation module and construct a multi-task framework to promote the personalized ranking. In the following, we describe details of each module in the personalized ranking model, respectively.

#### 3.2.1 *Query and Document Representation.*

With the pre-trained word embeddings, we are able to map the query $q = \{w_1^q, w_2^q, \ldots\}$ and document $d = \{w_1^d, w_2^d, \ldots\}$ into a low-dimensional vector space and obtain their basic text representations. In our model, the text representations are composed of four parts. Due to the personal word embeddings are used to reflect user interests, we merely consider representing the user-issued query with the personal word vectors but still representing the general document with the global word embedding.

(1) **Personalized word representation:** We obtain this part of representation by passing the query terms through the corresponding user's personal word embedding matrix in the personal embedding layer, getting $P^q \in R^{dim \times |q|}$ for the query. The word vectors mainly contain the meaning that the user already knows or is interested in, achieving personalization and disambiguation at the word level.

(2) **Personalized contextual representation:** To model the interactions between the query contexts and obtain personalized representation at the query level to further disambiguate the

query and clarify the personalized query intent, we exploit a multi-head self-attention layer [42] on the top of the personalized word representations outputted by the embedding layer, obtaining the matrix $CP^q \in R^{dim \times |q|}$. As for the calculation of $CP^q$, the input is the query word vectors $P^q$. We first process it with different linear functions to $d_q$, $d_k$, and $d_v$ dimension, where $q, k, v$ refer to the query, key, and value in the attention mechanism, respectively. Then, we conduct the attention function on the processed results of different heads in parallel, yielding several $d_v$ dimensional output values. These outputs are concatenated together and processed again with a dense layer to get the final attended result $CP^q$. The concrete computation formulas are listed as follows:

$$CP^q = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h) W^A, \tag{14}$$

$$\text{head}_i = \text{softmax}\left(\frac{P^q W_i^Q (P^q W_i^K)^T}{\sqrt{d_k}}\right)\left(P^q W_i^V\right), \tag{15}$$

where $W_i^Q$, $W_i^K$, $W_i^V$, and $W^A$ are parameters of linear functions. $CP^q$ fuse the contextual information to further clarify the ambiguous words.

(3) **Global word representation:** In actual situations, every user's interests are variable and knowledge is growing. Like the IT engineer in our previous example, in most cases, he issues the query "Apple" to seek for the Apple company or products. However, it is also inevitable that he would sometimes use other meanings of "Apple" to search for information that has never been searched before, such as the apple fruit. Therefore, in addition to the personalized representations, we also pay attention to the representations in the global vector space. We get global word representations $G^q \in R^{dim \times |q|}$ for the query and $G^d \in R^{dim \times |d|}$ for the document from the global word embedding matrix in the word embedding layer.

(4) **Global contextual representation:** Similar to the calculation of the personalized contextual representation, we obtain global contextual vectors $CG^q \in R^{dim \times |q|}$ for the query and $CG^d \in R^{dim \times |d|}$ for the document by applying a multi-head self-attention layer on the global word representations.

With these four parts, we obtain comprehensive query and document representations of different granularity and perspectives. They are helpful for matching the query and candidate documents more accurately.

### 3.2.2 *Session-based Query Intent Representation.*

In our model, we deploy personal word embeddings for each user that are pre-trained with the corresponding user's query log. However, these embeddings mainly reflect the user's long-term interests. Let us take the term "Java" as an example: If the user searched information about both the "Java Island" and the "Java Language" in the history, then the learned word vector of the term "Java" tends to contain both meanings. Therefore, we expect to further figure out the accurate meaning of the keywords in the current query. We have considered the contextual query representation, which employs the context in the current query for disambiguation. However, in many cases, the queries issued by users are very short, even one word, so disambiguation with only contextual terms in the current query is not enough. As stated in many existing studies [18, 50], a user's query intent in a search session is usually consistent, which is regarded as the user's short-term interests. Thus, we also intend to disambiguate the current query by incorporating the short-term history in the current session.

According to the problem definition, there are $n_M$ historical queries in the current session $S_M$. Each query corresponds to several clicked documents, i.e., $S_M = \{(q_{M,1}, d^+_{M,1}), \ldots, (q_{M,n_M}, d^+_{M,n_M})\}$. Their relationships are demonstrated with the graph structure in Figure 1. To better incorporate information of all the neighborhoods in the session graph, instead of only sequential information, we adopt GCN [15] to process the graph by direct propagation. At first, for each node in the session graph, we obtain their personalized contextual representation with a multi-head self-attention function and sum up the contextual vectors of all terms to get the representation of each node. Then, we use a GCN to recursively aggregate the semantic embeddings of the neighborhoods layer-by-layer. As for nodes $V$ in the $k$th layer, the node representations $H^k$ are computed in the following way:

$$H^{(k)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(k-1)}W^{(k-1)}\right). \tag{16}$$

Here, $H^{(0)}$ is the initial input of the graph convolutional network; $H^{(l)}$ is the output of the $l$th layer. $\tilde{A} = A + I_N, \tilde{A} \in R^{|V|\times|V|}$ is the adjacency matrix of the session graph with added self-connections, indicating connections between nodes. $I_N$ is the identity matrix and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ is the degree matrix. $W^{(k-1)}$ is a trainable matrix for the $k$th layer. $\sigma(\cdot)$ denotes an activation function, which is $ReLu(\cdot)$ in this article. Through the graph convolutional network, the information of other nodes in the session graph can be learned by propagation.

Finally, the output of the current query node is regarded as the representation of the session-based query intent, denoted as $SP^q$.

3.2.3 *Query-document Matching and Ranking.* With the personalized and global representations of the query and documents from different aspects, we are able to compute the personalized matching scores to re-rank the original document list. To capture fine-grained matching features between the query and document, we adopt the interaction-based neural matching model KNRM [49]. For the personalized word representations of the query, we compute the match score between it and the global word representations of a document as follows: We first construct a similarity matrix $S^P$ where $S^P_{ij}$ is the embedding similarity between the personalized query word $w^q_i$ and the global document word $w^d_j$ calculated by cosine similarity. Then, various RBF kernels are applied on the personalized word similarity matrix $S^P$ to extract multi-level soft-match features $\phi(S^P)$ between the query and document [49].

$$\phi(S^P) = \sum_{i=1}^{|q|} \log(\vec{K}(S^P_i)), \tag{17}$$

$$\vec{K}(S^P_i) = \{K_1(S^P_i), \ldots, K_K(S^P_i)\}, \tag{18}$$

$$K_k(S^P_i) = \sum_j \exp\left(-\frac{(S^P_{ij} - \mu_k)^2}{2\sigma^2_k}\right). \tag{19}$$

$|q|$ is the query length, and $K$ is the number of RBF kernels. $\mu_k$ and $\sigma_k$ are the center and variance of the $k$th kernel.

After obtaining a series of query-document ranking features $\phi(S^P)$, we use an MLP with the $\tanh(\cdot)$ activation function to combine all these features and compute the matching score $f^P(q,d)$ between the personalized word representations of the query and the global word representations of the document, i.e.,:

$$f^P(q,d) = \tanh\left(W^T_P \phi(S^P) + b^P\right). \tag{20}$$

Same as the calculation process above, we further use another three KNRM components with different parameters to compute the matching scores for other query and document representations described in Section 3.2.1, getting three scores $f^{CP}(q,d)$, $f^G(q,d)$, and $f^{CG}(q,d)$. Another matching score $f^{SP}(q,d)$ between the session-based query intent representation and document vector is computed by cosine similarity. The document vector is obtained by passing the personalized document representation through a dense layer.

In addition to the matching scores between the query and document, we also follow References [18, 29] to incorporate some click-based features, topic-based features, and relevance features to help ranking, totally 98 dimensions. These features are proved to be effective in Reference [5]. We input these features into an MLP layer with the $\tanh(\cdot)$ activation function to calculate a relevance score $f^r$. Finally, all these six scores are combined with an MLP to get the personalized score for the document $f(q, d, H)$, and we re-rank the original non-personalized search result with this score to get a personalized ranking list.

We apply a pairwise LTR algorithm LambdaRank [7] to train our personalized ranking model. First, we create training document pairs on the whole logs, with the clicked documents as positive samples and the skipped documents as negative ones. $\lambda$ for each document pair is the change value of the metric MAP when swapping the positions of the two documents. And the final loss function is the dot product of the $\lambda$ and cross entropy between the real probability distribution of the relative relationship and the predicted probabilities. We have:

$$loss = (-\hat{p}_{(i>j)} \log(p_{(i>j)}) - \hat{p}_{(i<j)} \log(p_{(i<j)}))|\lambda_{ij}|. \tag{21}$$

$\hat{p}_{(i>j)}$ is the true probability that $d_i$ is more relevant than $d_j$, and $p_{(i>j)}$ is the predicted probability computed as the score difference $(\text{score}(d_i) - \text{score}(d_j))$ between the two documents normalized by a logistic function.

*3.2.4 Query Reformulation.* In most situations, it is difficult for users to express their true query intents with accurate queries. In our model, we have personalized query representation so we are able to infer the user's real query intent and reformulate the query to promote the ranking. This can also help the learning of personal word embeddings in return. Based on this motivation, we design a query reformulation module and construct a multi-task framework to jointly train it with the personalized ranking task described above. Due to the lack of manually annotated reformulated queries that express the user's true query intents, we follow References [23, 38] to use the next query in the same session as the learning target of this task. The next query is usually thought to express the user's intent more accurately than the current query. Referring to existing query generation or suggestion models [23, 38], we implement the query reformulation task with reference to the sequence-to-sequence structure [39]. The input is the representation vector of the session-based query intent $SP^q$ outputted by the GCN, which is regarded as the representation of the user's real query intent. A common decoder GRU [11] is applied. Each token $o_t$ in the target sequence is predicted based on the current hidden state $s_t$ and the previous decoded tokens $\{o_1, \ldots, o_{t-1}\}$, i.e:

$$p(o_t|\{o_1, \ldots, o_{t-1}\}, h_{|q|}) = \text{softmax}(W \cdot s_t + b), \tag{22}$$

where $s_t = GRU(s_{t-1}, o_{t-1})$. Then, the probability of the target sequence $p(o)$ is defined as the joint probability of all the tokens:

$$p(o) = \prod_{t=1}^{T} p(o_t|\{o_1, \ldots, o_{t-1}\}, h_{|q|}). \tag{23}$$

We train the query reformulation module by minimizing the negative log likelihood of the target sequence. And the whole multi-task framework is optimized by minimizing the sum of the negative log likelihood and the pairwise loss.

## 3.3 Online Update

The personal word embeddings pre-trained from the user's current query log have contained the user interests reflected in the search history. In real-world application scenarios, users will continuously issue new queries and their interests are dynamically changing in the streaming setting. To ensure that our personal word embeddings contain the latest user interests, we should fine-tune the personal word embeddings according to the newly issued queries along with the search process, keeping the ranking model fixed. At the same time, we are also supposed to maintain the user's interests in the long-term history. Therefore, the data we utilize to update the personal word embeddings comes from two sources: the user's search history and the newly collected query behavior. As for the user's historical data, we randomly select a set of samples. In this article, considering the time interval for fine-tuning the word embeddings, we design three different approaches.

**Update by stage:** In the first step, we train personal word embeddings for each user and a personalized ranking model offline with all user's search logs before the current moment. In the second step, we set a fixed duration of a stage. During this stage, we collect all the query data but do not change the word embeddings. Third, at the end of each stage, we fine-tune the personal word embeddings with the newly collected data and data sampled from the previous search history, keeping other parameters of the ranking model static. Last, we repeat the second and the third steps to track the latest search process and user interests.

**Update by session:** In information retrieval, we usually view search sessions as search activities with independent query intents, which may reflect complete user interests. Therefore, based on the same update process described above, we propose to adjust the personal word embeddings at a session interval.

**Update by query:** Many personalized models [5, 18] divide the user interests into long-term and short-term user interests, where the short-term interests are defined as interests in a session. The method that updates the personal word embeddings by sessions has difficulty in capturing the impact of the short-term user interests on the subsequent queries in the same session. Thus, we also design an approach to update the word embeddings in units of queries to capture more fine-grained interests.

The three methods are applied to fine-tune the personal word embeddings with the latest queries issued by users online on the basis of the offline well-trained model. But it may be difficult for the model to achieve the global optimal state by such methods of incremental fine-tuning, and a long-term adjustment may make the model perform worse. With this consideration, we suggest fine-tuning the personal word embeddings by the update-by-query approach in a short time. Then, a large batch of new training samples can be added to the original dataset to retrain personal word embeddings for each user offline, achieving a balance between effectiveness and efficiency.

### 3.4 Discussion about PEPS+

Here, we discuss the performance and feasibility of our proposed personalized search model PEPS+. Although we set a personal word embedding matrix for each user in the model, we have stated in Section 3.1.1 that we filter the words in personal vocabulary strictly and merely some necessary ambiguous words should be maintained, which will not take up too much space and computing resources. If there are a huge number of users, then we can set a max number of users on a single model and distribute all users on multiple models. In addition, through embedding user interests into personal word representations, our model is not required to keep the user's long-term search history in memory and process the history to build the user interest profile. PEPS+ only needs to compute the query and document representations and their matching score with a shallow neural matching model. These calculations take little time compared to RNN, improving the performance and saving memory space. Thus, we can conclude that our proposed model provides several advantages for personalized search: First, the personal word embeddings contain user interests, so the ambiguous keywords can be disambiguated and personalized query intent can be clarified by representing the query with the personal word embeddings. Second, the PEPS+ model improves

Table 1. Statistics of the Datasets

| Dataset | AOL Dataset | | | Commercial Dataset | | |
|---|---|---|---|---|---|---|
| | Train | Valid | Test | Train | Valid | Test |
| #session | 187,615 | 26,386 | 23,040 | 71,731 | 13,919 | 12,208 |
| #query | 814,129 | 65,654 | 59,082 | 188,267 | 37,951 | 41,261 |
| avg session len | 3.945 | 3.298 | 2.602 | 2.548 | 2.727 | 2.697 |
| avg query len | 2.845 | 2.832 | 2.895 | 3.208 | 3.263 | 3.281 |
| avg query #click | 1.249 | 1.118 | 1.115 | 1.194 | 1.182 | 1.202 |

the efficiency of personalized search significantly, without obviously increasing pressure on the space occupation.

## 4 EXPERIMENTAL SETTINGS

### 4.1 Dataset and Evaluation Metrics

We evaluate our model on two non-personalized search logs, i.e., the AOL dataset and a log dataset from a commercial search. The basic statistics of the two datasets are listed in Table 1.

**AOL Dataset:** This a publicly available non-personalized query log dataset collected from 1st March 2006 to 31st May 2006. Each piece of data contains a user anonymous ID, a query text, the time when the query was issued, a clicked url, and the rank position of the url. Following References [1, 2], we segment the whole user query sequences into sessions with boundaries decided by the similarity between two consecutive queries. To ensure that every user has enough search history for mining preferences and building a user profile, we set the first five weeks log as the history, and the remaining data are used for model training, validation, and testing with the proportion 6:1:1. AOL dataset only records clicked documents that are regarded as relevant documents under each query, without unclicked documents, so we refer to References [1, 2] to find irrelevant documents and construct a general result list with the BM25 algorithm [34]. Following References [1, 2], we construct five candidate documents for each query in the training and validation set, while 50 candidates for each testing query. And each document corresponds to a title crawled from the web according to the recorded document url. After the process, the dataset includes 110,869 users. We count the number of queries issued by each user and find that most users only have a small amount of search data. Considering that we need enough individual query log to train the corresponding personal word embeddings, we sample about 30,000 users with the most training data and set personal word embeddings for each of them, global word embeddings for the remaining users.

**Commercial Dataset:** This is a large-scale query log collected from a non-personalized commercial search engine between 1st Jan. 2013 and 28th Feb. 2013. Each query record contains a user ID, a query text string, query issued time, the top 20 retrieved urls, click labels for each url, and their dwelling time. Each url corresponds to a document with text body. Following References [18, 29], we define SAT-click as the clicks with longer than 30 seconds of dwelling time, and documents with an SAT-click are viewed as relevant documents. With 30 minutes of user inactivity as the boundary [5], we segment the search process into sessions. Then, we divide the search logs in the first six weeks as the historical data and the last two weeks as the experimental data, which is further split into the training set, validate set, and testing set by sessions with 4:1:1 ratio. There are 5,998 users in the dataset. We select 4,000 users with the most training data and build personal word embeddings for each of them.

**Evaluation metrics:** Based on the assumption that the recorded clicked/SAT-Clicked documents are relevant and the other candidate documents are irrelevant [18], we apply the most widely used ranking metrics **Mean Average Precision (MAP), Mean Reciprocal Rank (MRR)**, and **Precision at 1 (P@1)** to evaluate our model and baselines. We also adopt the Avg.Click metric, which is the average ranking position of all clicked documents under each query. A lower value of Avg.Click indicates better ranking quality. Considering the fact that users' recorded click actions are inevitably influenced by the original order and some documents are not clicked may not be due to their irrelevance but their low rankings, we use a more credible metric P-Improve [29], based on reliable relevance preferences in this article. Following References [13, 25, 29], we construct inverse document pairs viewing only the documents skipped above the clicks and the non-clicked next document as irrelevant and compute P-Improve as the ratio of the correctly ranked inverse pairs. We only use the P-Improve metric on the commercial dataset whose recorded document lists were actually presented to users. The candidate lists of the AOL dataset are constructed by us with the BM25 algorithm but are not the lists shown to users, hence the P-Improve value calculated on the AOL dataset is unreliable. Because of this, we do not use the P-Improve metric for the AOL dataset.

## 4.2 Baselines

In addition to the original ranking (on the AOL dataset, it is generated by BM25; on the commercial dataset, it is returned by the search engine), we select several state-of-the-art ad hoc neural ranking models and personalized search models as baselines, listed as follows:

(1) **KNRM:** KNRM [49] is a kernel-based neural ranking model for ad hoc search. It conducts a kernel-pooling technique on the word similarity matrix to extract multi-level soft match features, which are combined with a pairwise LTR algorithm to get the final ranking score.

(2) **Conv-KNRM:** Conv-KNRM [14] is an upgrade of the KNRM model. It uses Convolutional Neural Networks to obtain the representations of n-grams and soft matches them. Then, the kernel pooling and learning-to-rank layers are conducted to calculate the final score.

(3) **P-Click:** Dou et al. [17] proposed P-Click to re-rank documents based on how many times it has been clicked by the same user under the same query in search history, satisfying the user's re-finding behavior.

(4) **SLTB:** It [5] extracts 102 features from the user's search log such as click-based features and topic-based features, and combines all the features with the LTR algorithm LambdaMart [8] to generate the personalized ranking list.

(5) **HRNN:** This model [18] utilizes a hierarchical RNN with a query-aware attention mechanism to dynamically build the short-term and long-term user interest profiles according to the current query. Documents are re-ranked based on the similarities with the user profiles, and the additional SLTB features are also considered.

(6) **PSGAN:** It [29] is a personalized framework that applies GAN to generate queries that match the user's query intent better and select document pairs more valuable for learning user interests. In this article, we take the variant PSGAN-D as our baseline.

(7) **RLPer:** It is a reinforcement learning-based personalization model [51], using hierarchical MDP to track the sequential interactions between the users and search engine, with document lists at the high level and document pairs at the low level. It continuously updates the underlying personalized ranking model with the users' real-time feedback.

(8) **RPMN:** Zhou et al. [53] made use of external memories to enhance re-finding behavior for search results personalization. They set query memory and document memory to analyze users' different re-finding intents, respectively.

(9) **PPWE:** This is a pipeline personalized word embedding-based model we implement as a baseline. To re-rank the documents for the current query, we first train personalized word vectors on the user's search history before this query by Word2vec model [31] to obtain the query and document representations, and then compute the relevance scores with the KNRM model and SLTB features.

(10) **PWEBA:** This is a personalization model for Twitter search [35]. It first trains personal word embeddings on the user's history and creates a word-synonym table based on word vector similarity, then it re-ranks the generic list with similarities between the query's synonyms and documents.

(11) **PEPS:** It is the personal word embedding-based personalized search model in the original conference paper. No pre-training model and the session-based query intent representation in the PEPS model, compared to the PEPS+.

### 4.3 Model Settings

In our model, the dimension of personal word embeddings is set as 100. For the AOL set, the words with less than five occurrences and entropy less than 0.7 are filtered from the personal vocabulary. The min occurrence is set as 8 and the word entropy is 0.65 for the commercial dataset. We set the max length of queries as 20 for both datasets, and the max length of document titles is 50 for the AOL dataset, and the max document length is set as 300 for the commercial dataset. The max number of queries in a session is set as 20. As for the multi-head self-attention, we use eight heads and the dimension of each head is 50. The graph convolutional network has two layers. The KNRM component has 11 kernels with $\mu \in \{-0.9, -0.7, \ldots, 0.9, 1\}$ and $\sigma$ is set to 0.1 [49]. The size of hidden states in GRU is 100. Model optimization uses the Adam optimizer, with batch size as 200, learning rate as 1e-3 and $\epsilon = 1e - 5$. We train the model for a total of 5–10 epochs and store the one performing best on the validation set.

## 5 EXPERIMENTAL RESULTS AND ANALYSIS

We conduct several groups of experiments on the two datasets to analyze the performance of the PEPS+ model thoroughly. The experimental results are reported as follows:

### 5.1 Overall Performance

First, we compare the overall performance of all baselines and our PEPS+. To be consistent, we train all models on the training set, and then evaluate them on the testing set without any dynamic update. Results are shown in Table 2 and Table 3. The reason for using the I-Imp metric only on the commercial dataset has been stated in Section 4.1. We find:

(1) **Compared to all baselines, the PEPS+ model achieves significant improvements in terms of all evaluation metrics on both datasets, with paired t-test at p < 0.01 level on the AOL query log.** Especially for the two state-of-the-art personalized search models RLPer and RPMN, our model outperforms them greatly on the AOL set. Our model improves RLPer by 20.1% in MAP and 19.8% in the MRR metric. In addition, it promotes 20.1% in MAP and 19.7% in MRR metric on the basis of RPMN. On the commercial dataset, our PEPS+ model also only performs a little worse than the RPMN model. We infer it may due to that there indeed exists some re-finding behavior in this query log. RPMN is great at handling this case. Both RLPer and RPMN re-rank the original document list by learning user interest profiles to predict their current user intents. These results prove that the personalized search model PEPS+ designed in an alternative way is also effective for personalization and achieves the best performance.

(2) **Comparing with the closer baseline PPWE and the original PEPS model, our expanded personalized search model PEPS+ performs much better whether it fixes the**

Table 2. Overall Performances of Models on the AOL Search Log

| Model | MAP | | MRR | | P@1 | | Avg.C | |
|---|---|---|---|---|---|---|---|---|
| Ad hoc search model | | | | | | | | |
| Ori. | .2504 | −58.0% | .2596 | −57.4% | .1534 | −71.2% | 17.11 | +100.6% |
| KNRM | .4291 | −28.1% | .4391 | −27.9% | .2704 | −49.2% | 8.17 | −4.22% |
| ConvK | .4738 | −20.6% | .4849 | −20.5% | .3266 | −38.6% | 7.92 | −7.2% |
| User profile based personalized search model | | | | | | | | |
| PClick | .4224 | −29.2% | .4298 | −29.5% | .3788 | −28.8% | 16.45 | +92.9% |
| SLTB | .5072 | −15.0% | .5194 | −14.8% | .4657 | −12.5% | 13.91 | +63.1% |
| HRNN | .5423 | −9.1% | .5545 | −9.1% | .4854 | −8.8% | 10.48 | +22.9% |
| PSGAN | .5480 | −8.2% | .5601 | −8.1% | .4892 | −8.1% | 10.21 | +19.7% |
| RLPer | .5965 | −0.1% | .6094 | −0.1% | .5319 | −0.1% | 8.67 | +1.6% |
| RPMN | .5968 | – | .6097 | – | .5323 | – | 8.53 | – |
| Embedding-based personalized search model | | | | | | | | |
| PWEBA | .4284 | −28.2% | .4368 | −28.4% | .2687 | −49.5% | 8.16 | −4.3% |
| PPWE | .6542 | 9.6% | .6668 | 9.4% | .5613 | 5.5% | 5.13 | −39.9% |
| PEPS | .7127 | +19.4% | .7258 | +19.0% | .6279 | 17.9% | 4.15 | −51.4% |
| PEPS+(Conv) | .6987 | +17.1% | .7122 | 16.8% | .6132 | 15.2% | 4.57 | −46.4% |
| PEPS+(fix) | .7048$^{\dagger}$ | +18.1% | .7186$^{\dagger}$ | +17.9% | .6224$^{\dagger}$ | 16.9% | 4.38$^{\dagger}$ | −48.7% |
| PEPS+ | **.7165**$^{\dagger\ddagger}$ | +20.1% | **.7302**$^{\dagger}$ | +19.7% | **.6335**$^{\dagger\ddagger}$ | +19.0% | **4.11**$^{\dagger}$ | −51.8% |

Relative performances compared with RPMN are in percentages. "†" indicates significant improvements over all baselines except PEPS with paired t-test at $p < 0.01$ level. "‡" indicates significant improvements over all baselines with paired t-test at $p < 0.05$ level. The best results are in bold. PEPS+(fix) means the personalized word embedding layer is initialized with the pre-training model and then fixed during training.

**personal word embeddings or not.** PPWE is a pipeline personalized baseline model proposed by us. It first trains the personal Word2vec model on the user's query log data, and then obtains static personalized representations of the query and document for ranking with the KNRM model, without supervised fine-tuning of the word embeddings. Compared to PPWE, the end-to-end PEPS model introduces the idea of fine-tuning the personal word embedding pre-trained by the Word2vec model with the click labels. It also uses the multi-head self-attention mechanism to capture the interactions between query contexts to clarify the personalized meaning of a specific keyword. The better performance of PEPS over PPWE confirms that supervised learning with the click labels can help train better personal word embeddings and contextual information of the current query is important for disambiguation. Our PEPS+ further improves the original PEPS model. In PEPS+, we represent user behavior during the search process with a graph structure and implement a separate pre-training model to obtain personal word embeddings that preserve both textual information of the query log and the structural information in the behavior graph. Furthermore, we also add session-based query intent representation to figure out the user's intent based on the short-term search history. The better performance of the new model demonstrates that PEPS+ captures the user interests better. To verify the effectiveness of the multi-head self-attention encoder, we replace it with a convolutional neural network used in Conv-KNRM [14] to capture local information. We find the multi-head self-attention performs better than the convolutional component. We infer that it may be because the multi-head self-attention layer can involve the context within the whole text sequence, while the convolutional neural network attends to the local context. In addition, the multi-head self-attention has the ability to capture relevant information in more than one space.

Table 3. Overall Performances of Models on the Commercial Query Log

| Model | MAP | | MRR | | P@1 | | P-Imp | |
|---|---|---|---|---|---|---|---|---|
| Ad hoc search model | | | | | | | | |
| Ori. | .7399 | −10.2% | .7506 | −10.0% | .6162 | −15.4% | – | – |
| KNRM | .4916 | −40.3% | .5001 | −40.1% | .2849 | −60.9% | .0655 | −75.3% |
| ConvK | .5872 | −28.7% | .5977 | −28.4% | .4188 | −42.5% | .1422 | −46.5% |
| User profile-based personalized search model | | | | | | | | |
| PClick | .7509 | −8.8% | .7634 | −8.5% | .6260 | −14.0% | .0611 | −77.0% |
| SLTB | .7921 | −3.8% | .7998 | −4.1% | .6901 | −5.3% | .1177 | −55.7% |
| HRNN | .8065 | −2.1% | .8191 | −1.8% | .7127 | −2.2 % | .2404 | −9.5% |
| PSGAN | .8135 | −1.3% | .8234 | −1.3% | .7174 | −1.5% | .2489 | −6.3 |
| RLPer | .8186 | −0.6% | .8302 | −0.5% | .7263 | −0.3% | .2502 | −4.7% |
| RPMN | .8238 | – | .8342 | – | .0.7294 | – | .2656 | – |
| Embedding-based personalized search model | | | | | | | | |
| PWEBA | .7415 | −10.0% | .7529 | −9.7% | .6201 | −14.9% | .0433 | −83.7% |
| PPWE | .8138 | −1.2% | .8249 | −1.1% | .7187 | −1.3% | .2338 | −11.9% |
| PEPS | .8221 | −0.2% | .8321 | −0.2% | .7251 | −0.4% | .2545 | −4.2% |
| PEPS+(Conv) | .8182 | −0.7% | .8283 | −0.7% | .7213 | −1.0% | .2485 | −6.4% |
| PEPS+(fix) | .8213 | −0.3% | .8313 | −0.3% | .7239 | −0.6% | .2531 | −4.7% |
| PEPS+ | .8226 | −0.1% | .8328 | −0.2% | .7255 | −0.4% | .2549 | −4.0% |

PEPS+(fix) means the personalized word embedding layer is initialized with the pre-training model and then fixed during training.

(3) **Generally, all personalized search models improve the original ranking results greatly, indicating that personalization is helpful for promoting users' search experience.** The increase of the P-Click model and RPMN model validates the effectiveness of re-finding behavior. The SLTB model realizes personalization by extracting various interest-related features from the search history. HRNN, PSGAN, and RLPer, which build user interest profiles with a hierarchical RNN, achieve great results. The great performance of PSGAN confirms the importance of high-quality data for the training process of personalized search models. The word-embedding-based models PWEBA, PPWE, PEPS, and PEPS+ also show great improvements in the ranking quality. However, we find that PClick and PWEBA perform worse than the neural ad hoc ranking models KNRM and Conv-KNRM. We analyze it may be because the ability of deep learning-based models is stronger than traditional models, and the word embedding in PWEBA trained on a single user's limited history data is unreliable.

In a word, the great overall performance strongly verifies that **our PEPS+ can obtain personal word embeddings that contain accurate user interests through the pre-training model and it clarifies the user's personalized query intents of ambiguous queries to improve search result personalization.**

## 5.2 Ablation Experiments of Ranking Model

The PEPS+ model includes several main components: the personal word embedding layer, text representations of different aspects, session-based query intent representation, and the query reformulation module. To figure out the role of each part for search result personalization, we perform several ablation experiments on the AOL dataset. We illustrate the experimental result in Table 4 and make some discussions about it.

Table 4. Results of Ablation Experiments

| PEPS+ Variant | MAP | | MRR | | P@1 | | Avg.C | |
|---|---|---|---|---|---|---|---|---|
| PEPS+ | .7165 | – | .7302 | – | .6335 | – | 4.11 | – |
| w/o Attn. | .6904 | −3.64% | .7044 | −3.53% | .6058 | −4.37% | 4.65 | +13.14% |
| w/o Attn, PWE | .6739 | −5.95% | .6877 | −5.82% | .5871 | −7.32% | 4.78 | +16.30% |
| w/o Attn, GWE | .6756 | −5.71% | .6897 | −5.55% | .5898 | −6.90% | 4.79 | +16.55% |
| w/o Session Intent | .7127 | −0.53% | .7258 | −0.60% | .6279 | −0.88% | 4.15 | +0.97% |
| Ablation on query reformulation | | | | | | | | |
| w/o Query Ref | .7147 | −0.25% | .7285 | −0.23% | .6305 | −0.47% | 4.14 | +0.73% |

Relative performances compared with complete PEPS+ are in percentages. PWE/GWE means personal/global word embeddings.

**Personal & global word embeddings.** In PEPS+, we design a special word embedding layer that includes a shared global word embedding matrix and personal word embedding matrices for each user. To confirm the respective effects of the global and personal word embeddings, we alternatively strip off the two parts to conduct experiments. We also turn off the multi-head self-attention mechanism to make the comparison results clearer. The results are presented at the 3rd and 4th rows in Table 4. We find that the model loses 5.95% in MAP and 7.32% in P@1 metric without the personal word vectors on the AOL dataset. After removing the global word embeddings, the impacts on the model are similar. This indicates that the personal word embeddings containing user interests is critical for personalization. But every user has changing interests and growing knowledge, and they are likely to use new meanings of a word that has never been involved in their own search history. In such cases, global embedding can be helpful to provide general great results.

**Contextual representation.** On the top of the word representations, we apply a multi-head self-attention layer to obtain the contextual representations taking the word interactions within the current query into account. We disable the self-attention layer to analyze the contribution of the term context, and report the results at the 2nd row in Table 4. Without the attention layer, the MAP, MRR, P@1 metrics drop 3.64%, 3.53%, 4.37%, respectively, on the AOL dataset. This demonstrates that the specific meaning of a keyword not only depends on itself but also on the contextual terms in the query. Thus, the multi-head attention contributes to clarifying the meaning of a word.

**Session-based query intent representation.** Based on the PEPS model in the original paper, we add a module of session-based query intent representation to further improve search results personalization in this version. This module applies a GCN to predict the current query intent by aggregating information from the short-term search history in the current session that is represented with a session graph. We conduct the ablation study to confirm the effectiveness of this module. The experimental results are shown in the 5th row in Table 4. We find that the model decreases 0.53% in MAP, 0.60% in MRR, and 0.88% in P@1 on the AOL dataset without the session-based query intent part. The results indicate that the user's short-term query intent in a session is helpful to clarify the user's current query intent, and the proposed model in this version outperforms the original model.

**Query reformulation.** We use a query reformulation task to help figure out the user's real query intent through a multi-task framework. To analyze the impacts of this multi-task framework carefully, we turn off the joint learning mechanism (i.e., the sequence decoder) and report the performance in Table 4. Observing the experimental result, the performance drops without the whole query reformulation module. It illustrates that the query reformulation task is helpful for clarifying the user's query intent, but the effectiveness is not so obvious. We conjecture it is because

Table 5. Results of Ablation Experiments about the Pre-training Model

| PEPS+ Variant | MAP | | MRR | | P@1 | | Avg.C | |
|---|---|---|---|---|---|---|---|---|
| PEPS(fixed) | .7048 | – | .7186 | – | .6224 | – | 4.38 | – |
| w/o context prediction | .6774 | −3.89% | .6914 | −3.79% | .5971 | −4.06% | 5.29 | +20.78% |
| w/o query comp | .7037 | −0.16% | .7171 | −0.21% | .6202 | −0.35% | 4.26 | −2.74% |
| w/o relevance prediction | .7028 | −0.28% | .7126 | −0.33% | .6194 | −0.48% | 4.35 | −0.68% |

Relative performances compared with PEPS+ (fixed) with personal word embeddings trained by complete pre-training model are in percentages.

the training of the personalized ranking model relies more on the click information but less on other labels.

Through the ablation studies above, we observe that every component in our model has certain effects for improving the personalized search.

## 5.3 Experiments on Pre-training Model

The target of this article is to train personal word embeddings for each individual user to clarify the personalized meaning of each entered keyword and eliminate the ambiguity of the current query. To this end, we design a pre-training model to train personal word embeddings that contain user interests based on the corresponding user's query log. This proposed pre-training model consists of three sub-tasks: context prediction, query sequence complementation, and relevance prediction. To analyze the contribution of each task for learning better personal word embeddings, we conduct an ablation analysis about the pre-training model. We turn off one pre-training task and train the word embeddings with the remaining tasks. Then, we employ the trained word embedding on the subsequent personalized ranking model to evaluate its performance based on the ranking quality. According to the experimental results shown in Table 5, we make some analysis about the pre-training model as follows:

It can be seen from the experimental results that removing any pre-training task has a certain impact on the performance of the trained personal word embeddings used in the ranking model. Specially, removing the sub-task of context prediction that is based on the classical language model CBOW [31] causes the most decline in the ranking results. We analyze that this task is the basis of the pre-training model. The words in the user-issued queries and clicked documents express the personalized meanings that the user already knows or is interested in. By training a language model on the user's individual query log data, this textual information can be directly captured. The damage to the ranking results caused by discarding the relevance prediction sub-task shows that the user's click behavior during the search process reflects the user's interest information. Clicks are indeed always used to indicate relevance between the document and the query in practical log data. Furthermore, the contributions of the query sequence complementation sub-task are less than the other two pre-training tasks, but it still has a certain effect. This also proves that the user's query reformulation behavior contains some user interest information, and the user's query intents are relatively similar in the same session. In Table 5, we find that the MAP metric decreases when removing some sub-tasks but the Avg.C metric shows a little increment. We analyze it may be because MAP is a weighted metric that gives higher weights to high-ranking positions and lower weights to low positions, but Avg.C is calculated through averaging. When results at high positions drop down and some results at low positions raise up, the MAP metric loses much while the Avg.C may increase. In summary, the results of this ablation experiment demonstrate that the defined pre-training tasks have certain effects on learning personal word embeddings containing user interests.

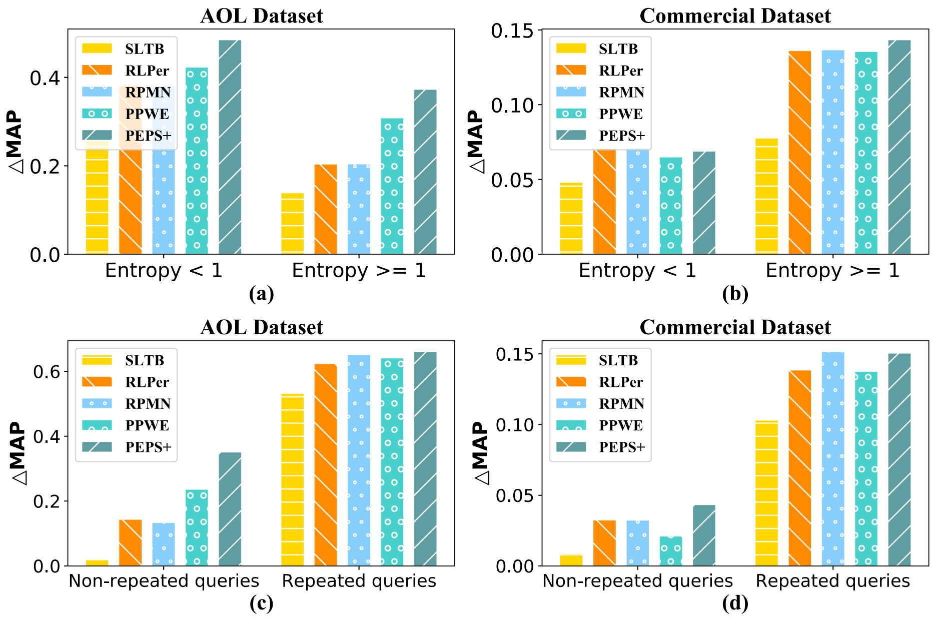

Fig. 4. Experimental results on different query sets. (a) is the results about queries with different entropies, (b) is results on repeated/non-repeated queries.

## 5.4 Experiments on Different Query Sets

To better analyze how our proposed model improves the personalized search, we divide all testing queries into different sets and compare the improvement of MAP based on the original ranking result of our PEPS+ model and several well-performed baselines. The whole results are illustrated in Figure 4.

**Informational & Navigational Queries.** Click entropy is an effective measure of whether a query is ambiguous. A query with a high click entropy indicates that users always have different query intents when issuing this query so their click behavior is greatly different. References [17, 40] have shown that it is more necessary to personalize the search results for queries with higher click entropy. Thus, we divide all queries into informational queries with click entropy >= 1 and navigational queries with entropy < 1 for evaluation. In Figure 4, (a) and (b) show the performance of our model and other baselines on the two query logs.

First, although the relative performance on the two different query sets of all models is opposite on the two datasets, which we think may due to the data distribution, we find our model consistently outperforms all baselines on both query groups of the two datasets. Specially, compared to the best baseline RPMN, our model also has obvious improvements no matter if the query is clear or ambiguous, especially on the informational query set. It confirms the ability of PEPS+ to perform well on queries that more require personalization.

**Repeated & Non-repeated Queries.** In personalized search, the user's behavior on the relevant queries in the search history is critical for analyzing the user's interests for the current query. Some studies [5, 17] even directly use the click features to promote document ranking, but such methods lack the ability of generalization. To further explore the generalization and learning abilities of our model, we categorize all testing queries into repeated or new queries according to whether they have appeared in the search history. The comparison results of PEPS and baselines are shown in Figures 4(c) and (d).

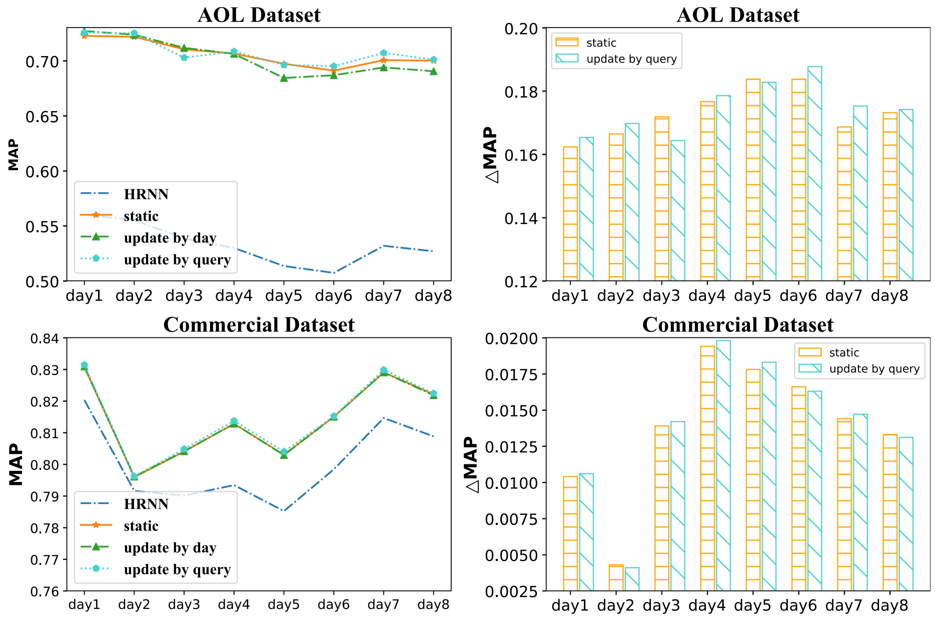

Fig. 5. Performance of different online update methods.

Consistently, we find all personalized search models achieve greater improvements on the repeated query set for the two datasets. This demonstrates that most personalized models can satisfy the user's re-finding needs well, especially the RPMN model, which performs the best on repeated queries on the commercial dataset. But some may fail on the non-repeated queries, such as the traditional feature-based SLTB model. Our PEPS+ shows great results on both query sets and the proportion of improvement on the non-repeated queries is larger. This verifies that our model can not only apply the click-based features to support the user's re-findings but also can learn the user's real interests to improve the results of newly issued queries.

## 5.5 Experiments with Online Update Methods

In real-world search scenarios, the user's interests are dynamically changing as the user issues new queries. We design three approaches to fine-tune the word embeddings along with the user's search process to capture the user interests reflected in the newly issued queries. To explore the effects, we perform simulation experiments on the last eight-day testing data. We set a day as a stage and adjust the word embeddings with two approaches. We calculate MAP on the data of each day, obtaining the performance curves and MAP improvements on HRNN shown in Figure 5.

Focusing on the left graph in Figure 5, we can find that all the performance curves show similar change trends, which should be determined by the original distribution of the testing data. And the curves of our model lie above that of HRNN. Comparing the two different adjustment approaches, the method of updating by query outperforms updating by day on both datasets. It indicates that the short-term user interests in the same session are very effective for improving the results of the subsequent queries. In general, both the update methods improve the MAP compared to the static test in the first several days, but only the update-by-query method performs better than the static test in the latter days on the AOL set, and the other method is worse. We analyze the possible reason is the incremental fine-tuning approach is unstable and is difficult to make the model achieve the global optimal state, and a long-term fine-tune might make the model perform worse. As for this

Table 6. Experimental Results about Model Efficiency

| Models | KNRM | Conv-KNRM | HRNN | PEPS+ |
|---|---|---|---|---|
| Time Cost | 50 item/s | 40 item/s | 1 item/s | 6 item/s |
| Space Cost | 1,000MB | 3,000MB | 8,000MB | 3,000MB |

"item/s" means the number of testing samples processed in every second, "MB" is used to measure memory space (GPU).

problem, we also propose a solution that we continuously update the personal word embeddings in a short time, and then a batch of new samples is added to train a global optimal model after a long time.

## 5.6 Experiments about Model Efficiency

In Section 3.4, we discuss the time and space complexity of the PEPS+ model proposed in this article. Here, we conduct an experiment to make some empirical analysis. The comparison results are shown in Table 6. "item/s" means the number of testing samples processed in every second. "MB" is the unit of memory space. Both the time and space cost are measured in the testing process (space cost corresponds to the required GPU memory).

From the row of time cost, we observe that KNRM and Conv-KNRM show high efficiency, because they have a simple structure and do not need to handle the user history. However, HRNN is very slow, because it is required to dynamically build a user interest profile from the user history every time a new query is submitted. Our personalized search model PEPS+ presents efficiency close to the ad-hoc ranking models. PEPS+ discards the process of constructing user profiles online but only needs personalized text matching, improving HRNN too much. As for space cost, our proposed PEPS+ requires memory less than HRNN. PEPS+ is supposed to keep a lot of personal word embedding matrices and the session history, without the long-term history. In addition, we filter the words in personal vocabulary strictly and only maintain some necessary terms. Whereas HRNN needs to process the user's whole long-term history to build the user profile and maintain a lot of intermediate results. Users' long-term history is large-scale, which is represented in vectors, even more than personal word embeddings. Thus, our PEPS+ model indeed improves the efficiency of personalized search without obviously increasing space cost.

## 5.7 Experiments about Amount of Query Log

Based on the consideration that training word embeddings usually requires a large corpus, we conduct an experiment on the amount of users' query logs. We sample a set of users from the AOL dataset. Each user has an eight-week search history. We keep the personalized ranking model trained on the whole query logs fixed. Then, we generate personal word embeddings by fine-tuning the global word embeddings with two/four/six/eight weeks of query logs. We evaluate the personal word embeddings on the test set and show the results in Figure 6.

According to the results in Figure 6, we find that personal word embeddings become more accurate with the increase of query logs. This verifies that a large corpus is helpful for generating high-quality word embeddings. However, our model also shows great results even with little query logs. We infer it may be due to us setting global word embeddings, which helps ensure the basic relevance of the results.

## 5.8 Visualization of Personal Word Embeddings

In this section, we conduct a study to intuitively analyze how personal word embeddings can reflect user interests. We compute the average of a user's personal word embeddings as her unique user

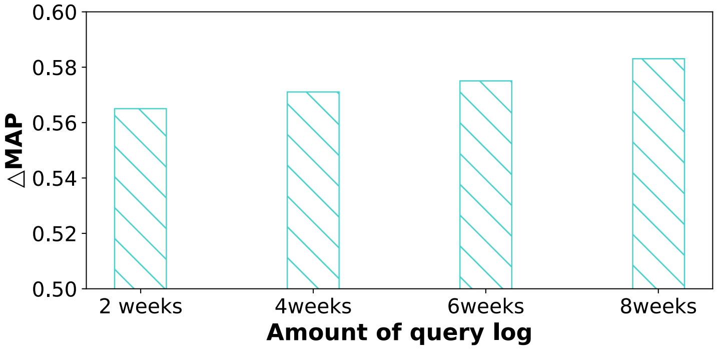

Fig. 6. Experimental results about different amount of query log.

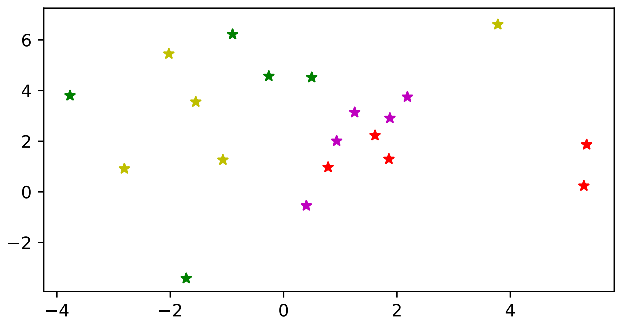

Fig. 7. Visualization of the distance between different users. Users with the same color are similar users.

embedding. We sample several users and their similar users found by the collaborative filtering algorithm introduced in Section 3.1. Then, we visualize the distance between different users by the PCA algorithm. We illustrate the results in Figure 7. Users with the same color are similar users. We can see that most similar users are relatively close, while the distance between dissimilar users is relatively far. Thus, we think that personal word embeddings reflect user interests to some extent.

## 5.9 Review of the Experiment

In the subsections above, we conduct experiments to compare the overall performance of our proposed PEPS+ model and baselines. We also explore the effects of each sub-module in the whole architecture. To make the findings clearer, we summarize all experiments and list the key observations as follows:

- In our model, we train personal word embeddings for each individual user with her search history, which indeed contains user interests. Using these word embeddings to represent the user's query, we can clarify her personalized search intent and generate personalized search results. This corresponds to the personal word embedding layer in our model.
- In a user's search log, the semantic information contained in the user-issued queries and clicked documents contributes the most to training personal word embeddings, followed by the user's click behavior. This corresponds to the context prediction task and relevance prediction task in the pre-training model.

- The information from the contexts is critical for eliminating the ambiguity of the query and figuring out the user's intent, including the local context within the text sequence and the context in the session history. This corresponds to the multi-head self-attention layer and session-based query intent representation in our model.

## 6 CONCLUSION

In this article, we solved the problem of search results personalization in an alternative way. Existing personalized search approaches mainly create user interest profiles and personalize search results based on the created profiles. Differently, we explore the idea that different users have personalized understandings of the same word. We proposed PEPS+, a personalization model in which we set personal word embeddings for each individual user to disambiguate the issued query. We represented the user behavior in the whole search process with a graph structure. Based on the graph, we designed a pre-training model to generate personal word embeddings that preserve the textual information, structural information, and user interests. Furthermore, we applied a self-attention mechanism to obtain personalized contextual query representations and used a graph convolutional network to get session-based query intent representation to clarify the user's query intent. Then, we designed a multi-task framework including personalized ranking and query reformulation to jointly train the personal word embeddings and ranking model. We also worked out three approaches for the online update to track the user's new interests. Experimental results on two large-scale query logs verified the effectiveness of our personalized search model. In the future, we will explore better user interest learning algorithms.

## REFERENCES

[1] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2018. Multi-task learning for document ranking and query suggestion. In *Proceedings of the 6th International Conference on Learning Representations.*

[2] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2019. Context attentive document ranking and query suggestion. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 385–394.

[3] Nawal Ould Amer, Philippe Mulhem, and Mathias Géry. 2016. Toward word embedding for personalized information retrieval. *CoRR* abs/1606.06991 (2016).

[4] Paul N. Bennett, Filip Radlinski, Ryen W. White, and Emine Yilmaz. 2011. Inferring and using location metadata to personalize web search. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval.* 135–144.

[5] Paul N. Bennett, Ryen W. White, Wei Chu, Susan T. Dumais, Peter Bailey, Fedor Borisyuk, and Xiaoyuan Cui. 2012. Modeling the impact of short- and long-term behavior on search personalization. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval.* 185–194.

[6] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2001. Latent Dirichlet allocation. In *Advances in Neural Information Processing Systems: Natural and Synthetic.* 601–608.

[7] Christopher J. C. 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.* 89–96.

[8] Chris J. C. Burges, Krysta M. Svore, Qiang Wu, and Jianfeng Gao. 2008. *Ranking, Boosting, and Model Adaptation.* Technical Report MSR-TR-2008-109.

[9] Fei Cai, Shangsong Liang, and Maarten de Rijke. 2014. Personalized document re-ranking based on Bayesian probabilistic matrix factorization. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval.* 835–838.

[10] Mark James Carman, Fabio Crestani, Morgan Harvey, and Mark Baillie. 2010. Towards query log based personalization using topic models. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management.* 1849–1852.

[11] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing.* 1724–1734.

[12] Kevyn Collins-Thompson, Paul N. Bennett, Ryen W. White, Sebastian de la Chica, and David Sontag. 2011. Personalizing web search results by reading level. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management.* 403–412.

[13] Nick Craswell, Onno Zoeter, Michael J. Taylor, and Bill Ramsey. 2008. An experimental comparison of click position-bias models. In *Proceedings of the International Conference on Web Search and Web Data Mining*. 87–94.

[14] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional neural networks for soft-matching N-grams in ad hoc search. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. 126–134.

[15] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5–10, 2016, Barcelona, Spain*, Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (Eds.). 3837–3845.

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

[17] Zhicheng Dou, Ruihua Song, and Ji-Rong Wen. 2007. A large-scale evaluation and analysis of personalized search strategies. In *Proceedings of the 16th International Conference on World Wide Web*. 581–590.

[18] 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*. 347–356.

[19] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative adversarial networks. *CoRR* abs/1406.2661 (2014).

[20] 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, San Francisco, CA, USA, August 13–17, 2016*, Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi (Eds.). ACM, 855–864.

[21] Morgan Harvey, Fabio Crestani, and Mark James Carman. 2013. Building user profiles from topic models for personalised search. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*. 2309–2314.

[22] Guangneng Hu. 2019. Personalized neural embeddings for collaborative filtering with text. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, June 2–7, 2019, Volume 1 (Long and Short Papers)*. 2082–2088.

[23] Jyun-Yu Jiang and Wei Wang. 2018. RIN: Reformulation inference network for context-aware query suggestion. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 197–206.

[24] Shan Jiang, Yuening Hu, Changsung Kang, Tim Daly Jr., Dawei Yin, Yi Chang, and ChengXiang Zhai. 2016. Learning query and document relevance from a web-scale click graph. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17–21, 2016*, Raffaele Perego, Fabrizio Sebastiani, Javed A. Aslam, Ian Ruthven, and Justin Zobel (Eds.). ACM, 185–194.

[25] Thorsten Joachims, Laura A. Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th Annual International ACM SIGIR, Conference on Research and Development in Information Retrieval*. 154–161.

[26] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations*.

[27] Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *J. ACM* 46, 5 (1999), 604–632.

[28] Cheng Li, Mingyang Zhang, Michael Bendersky, Hongbo Deng, Donald Metzler, and Marc Najork. 2019. Multi-view embedding-based synonyms for email search. In *Proceedings of the SIGIR Conference on Research and Development in Information Retrieval*. 575–584.

[29] 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*. 555–564.

[30] Hao Ma, Haixuan Yang, Irwin King, and Michael R. Lyu. 2008. Learning latent semantic relations from clickthrough data for query suggestion. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, USA, October 26–30, 2008*, James G. Shanahan, Sihem Amer-Yahia, Ioana Manolescu, Yi Zhang, David A. Evans, Aleksander Kolcz, Key-Sun Choi, and Abdur Chowdhury (Eds.). ACM, 709–718.

[31] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations*.

[32] Daisuke Oba, Naoki Yoshinaga, Shoetsu Sato, Satoshi Akasaki, and Masashi Toyoda. 2019. Modeling personal biases in language use by inducing personalized word embeddings. In *Proceedings of the 2019 Conference of the North*

*American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2–7, 2019, Volume 1 (Long and Short Papers)*. 2102–2108.

[33] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online learning of social representations. In *the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'14, New York, NY, USA - August 24–27, 2014*, Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani (Eds.). ACM, 701–710.

[34] Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.* 3, 4 (2009), 333–389.

[35] Sameendra Samarawickrama, Shanika Karunasekera, Aaron Harwood, and Ramamohanarao Kotagiri. 2017. Search result personalization in Twitter using neural word embeddings. In *Proceedings of the 19th International Conference on Big Data Analytics and Knowledge Discovery*. 244–258.

[36] Ahu Sieg, Bamshad Mobasher, and Robin D. Burke. 2007. Web search personalization with ontological user profiles. In *Proceedings of the Conference on Information and Knowledge Management*. 525–534.

[37] Yang Song, Hongning Wang, and Xiaodong He. 2014. Adapting deep RankNet for personalized search. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*. 83–92.

[38] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*. 553–562.

[39] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 3104–3112.

[40] Jaime Teevan, Susan T. Dumais, and Daniel J. Liebling. To personalize or not to personalize: Modeling queries with variation in user intent. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 163–170.

[41] Jaime Teevan, Daniel J. Liebling, and Gayathri Ravichandran Geetha. 2011. Understanding and predicting personal navigation. In *Proceedings of the 4th International Conference on Web Search and Web Data Mining*. 85–94.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Advances in Neural Information Processing Systems*. 5998–6008.

[43] Thanh Vu, Dat Quoc Nguyen, Mark Johnson, Dawei Song, and Alistair Willis. 2017. Search personalization with embeddings. In *Proceedings of the 39th European Conference on IR Research*. 598–604.

[44] Thanh Tien Vu, Alistair Willis, Son Ngoc Tran, and Dawei Song. 2015. Temporal latent topic user profiles for search personalisation. In *Proceedings of the European Conference on Information Retrieval*. 605–616.

[45] Hongning Wang, Xiaodong He, Ming-Wei Chang, Yang Song, Ryen W. White, and Wei Chu. 2013. Personalized ranking model adaptation for web search. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 323–332.

[46] Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. 2006. A user-item relevance model for log-based collaborative filtering. In *Proceedings of the 28th European Conference on Advances in Information Retrieval*. 37–48.

[47] Ryen W. White, Wei Chu, Ahmed Hassan Awadallah, Xiaodong He, Yang Song, and Hongning Wang. 2013. Enhancing personalized search by mining and modeling task behavior. In *Proceedings of the 22nd International World Wide Web Conference*. 1411–1420.

[48] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* 32, 1 (2021), 4–24.

[49] 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*. 55–64.

[50] Hui Yang, Dongyi Guan, and Sicong Zhang. 2015. The query change model: Modeling session search as a Markov decision process. *ACM Trans. Inf. Syst.* 33, 4 (2015), 20:1–20:33.

[51] Jing Yao, Zhicheng Dou, Jun Xu, and Ji-Rong Wen. 2020. RLPer: A reinforcement learning model for personalized search. In *Proceedings of the Web Conference*. 2298–2308.

[52] Yuan Zhang, Dong Wang, and Yan Zhang. 2019. Neural IR meets graph embedding: A ranking model for product search. In *the World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13–17, 2019*, Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia (Eds.). ACM, 2390–2400.

[53] Yujia Zhou, Zhicheng Dou, and Ji-Rong Wen. 2020. Enhancing re-finding behavior with external memories for personalized search. In *Proceedings of the 13th ACM International Conference on Web Search and Data Mining*. 789–797.

Received October 2020; revised June 2021; accepted June 2021