RLPS: A Reinforcement Learning–Based Framework for Personalized Search

JING YAO, School of Information, Renmin University of China
ZHICHENG DOU and JUN XU, 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, Ministry of Education of the People's Republic of China

Personalized search is a promising way to improve search qualities by taking user interests into consideration. Recently, machine learning and deep learning techniques have been successfully applied to search result personalization. Most existing models simply regard the personal search history as a static set of user behaviors and learn fixed ranking strategies based on all the recorded data. Though improvements have been achieved, the essence that the search process is a sequence of interactions between the search engine and user is ignored. The user's interests may dynamically change during the search process, therefore, it would be more helpful if a personalized search model could track the whole interaction process and adjust its ranking strategy continuously. In this article, we adapt reinforcement learning to personalized search and propose a framework, referred to as RLPS. It utilizes a **Markov Decision Process** (**MDP**) to track sequential interactions between the user and search engine, and continuously update the underlying personalized ranking model with the user's real-time feedback to learn the user's dynamic interests. Within this framework, we implement two models: the listwise RLPS-L and the hierarchical RLPS-H. RLPS-L interacts with users and trains the ranking model with document lists, while RLPS-H improves model training by designing a layered structure and introducing document pairs. In addition, we also design a feedback-aware personalized ranking component to capture the user's feedback, which impacts the user interest profile for the next query. Significant improvements over existing personalized search models are observed in the experiments on the public AOL search log and a commercial log.

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

Additional Key Words and Phrases: Personalized search, reinforcement learning, Markov decision process (MDP)

This work was supported by National Key R&D Program of China Grant No. 2018YFC0830703; National Natural Science Foundation of China Grants No. 61872370, No. 61832017, and No. 61872338; Beijing Outstanding Young Scientist Program Grant No. BJJWZYJH012019100020098; and Beijing Academy of Artificial Intelligence Grant No. BAAI2019ZD0305.
Authors' addresses: J. Yao, School of Information, Renmin University of China, No. 59 Zhongguancun Street, haidian District, Beijing 100872 P. R China; email: jing_yao@ruc.edu.cn; Z. Dou (corresponding author) and J. Xu, Gaoling School of Artificial Intelligence, Renmin University of China, No. 59 Zhongguancun Street, haidian District, Beijing 100872 P.R China; emails: {dou, junxu}@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, No. 59 Zhongguancun Street, haidian District, Beijing 100872 P.R China; email: jrwen@ruc.edu.cn.

1046-8188/2021/05-ART27 $15.00
https://doi.org/10.1145/3446617

ACM Reference format:
Jing Yao, Zhicheng Dou, Jun Xu, and Ji-Rong Wen. 2021. RLPS: A Reinforcement Learning–Based Framework for Personalized Search. *ACM Trans. Inf. Syst.* 39, 3, Article 27 (May 2021), 29 pages.
https://doi.org/10.1145/3446617

1 INTRODUCTION

The search engine has become a necessary tool to help people obtain information from the web in their daily lives. Users conduct a search with specific query intents, but studies have shown that their issued queries are usually short and ambiguous, which cannot express the intentions accurately [15, 35]. Let us take the query "*MAC*" as an example. A makeup artist may issue this query with the intent of searching for information about the cosmetic brand "*MAC*," while an IT engineer is likely to seek information about "*MAC*" computer using the same keyword. Returning general retrieval results about this query to both users may cause them to find unwanted documents that are relevant to "MAC" but are irrelevant to their individual information need. Personalized search provides a promising solution for this problem: it re-ranks the general document list for different users based on their user preferences and generates more accurate search results. Traditional strategies of personalized search employ user click behaviors, relevance-based features, and topic-based features to analyze user interests [4, 8, 10, 15, 19, 31, 39, 43]. In recent years, deep learning–based models were proposed to learn the representations of user interest profile for mining more potential preferences [17, 32, 38, 40].

Essentially, a user's personal search process can be viewed as a sequence of dynamic interactions between the user and the search engine: in each step of the interaction, the user issues a query and the search engine generates a document ranking list with the underlying ranking model. Then, the user browses the list, then clicks or skips each document, which we think implicitly reflects her current interests. During the sequential interaction process, the user's interests dynamically change. We expect the search engine to infer the user's current interests and return an accurate document list. Most existing personalized search models ignore the dynamic nature of the search process. They simply regard the historical search sequence as a static set of user issued queries as well as the retrieved and clicked documents, without sequential information, and they learn a fixed ranking strategy with all the recorded data. Then, the well-trained fixed ranking strategy is applied on the subsequent new queries without any update. Recently, some studies [17, 24] have considered sequential information hidden in the search process and user interests are dynamic. But they did not pay attention to how the interests change in the process and ignore that the ranking strategy should also be dynamically adjusted according to the change of user interests. In this article, we claim that *it would be more helpful for search results personalization if user interests can be modeled dynamically along with the search process and the ranking strategy can be updated continuously to adapt to the current user interests.*

To cope with this problem, we adapt reinforcement learning to personalized search and propose a novel **Reinforcement Learning**–based framework for **Personalized Search** (**RLPS**) in this article. Within the RLPS framework, we model the sequential interactions between the user and the search engine as a **Markov Decision Process** (**MDP**) [34]. RLPS provides several advantages for search results personalization. Firstly, by tracking the search process that contains the variations of the user's interests with an MDP, it is able to learn the user's dynamic interests better. Secondly, RLPS has the ability to dynamically adapt to the user's current interests as the personalized ranking strategy is updated continuously with real-time feedback from the user under the reinforcement learning framework. Thirdly, compared to existing learning-based personalized search models, RLPS can be trained with more training samples (annotated with rewards) through the

trial-and-error strategy in reinforcement learning. This is supposed to relieve the problem of the limited number of training samples in personalized search.

Reinforcement learning has been successfully applied to **information retrieval** (**IR**) [42, 45] and recommendation systems [49–51], but those specific models are not suitable for personalized search. In this article, we fully take the characteristics of personalized search into account and adapt reinforcement learning to design the RLPS framework. The search engine is set as the agent and the user as the environment. Because sessions are always viewed as search activities with independent user information need [17, 24], we set a session as an episode to capture the dynamic user interests. During a session episode, the sequential interactions in RLPS are as follows: the user issues a query, and the search engine exploits the underlying personalized ranking model to generate a personalized document list. Then, the user clicks on the returned document list to provide real-time feedback for the search engine to train the underlying ranking model. Records about this query are added to the user's search history for updating the user interest profile. Within this framework, we use an MDP to model the user's search process and propose two different implementations with different settings of the MDP, namely, RLPS-L and RLPS-H. RLPS-L is a listwise model designed to follow the previously described interaction process and train the personalized ranking model with the sampled document lists. However, studies [21] have shown that the user's click actions are biased and not suitable to be used as absolute relevant judgment. Thus, it would be better to introduce document pairs that indicate user preferences into the personalized ranking model training. To achieve interacting with document lists but training the model with document pairs, we improve the flatten RLPS-L with a hierarchical MDP to RLPS-H. In RLPS-H, we utilize the high-level MDP to track the interactions between the user and search engine in units of query and document list, and use the low-level MDP to follow all the document pairs constructed under each query. Both the document lists and document pairs are employed to train the personalized ranking model. To be consistent with the RLPS framework, we design a feedback-aware personalized ranking component that can capture the user's real-time click feedback. Furthermore, this model can also use the historical query sequence in the episode to predict the user's current real query intent. We adopt the widely used policy gradient algorithm REINFORCE to train RLPS, and two different approaches are provided for online evaluation and continuous updates. We conduct experiments on the public AOL search log [26] and log data from a commercial search engine to compare our proposed framework with state-of-the-art personalized search models. The results demonstrate that our models can achieve significant improvements over existing models.

Generally, the main contributions of this article can be summarized into four aspects: (1) To the best of our knowledge, it is the first time that reinforcement learning is being applied to personalize search. (2) We carefully adapt reinforcement learning to personalized search and propose a general reinforcement learning–based framework RLPS, which employs an MDP to track the user's sequential search process to learn the dynamically changing user interests and fits the personalized search scenario better. To be consistent, we also design a feedback-aware personalized ranking component. (3) Within the RLPS framework, we implement two specific models: the listwise RLPS-L and the hierarchical RLPS-H. RLPS-L follows the search process and trains the personalized ranking model with document lists. RLPS-H improves RLPS-L with a hierarchical MDP and introduces document pairs to train the ranking model better. (4) Experimental results on the public AOL logs and a query log from a commercial search engine verify that the proposed models significantly improve the quality of personalized search over state-of-the-art models, especially the RLPS-H.

This article is an extended version of a paper presented at the Web Conference 2020 (WWW 2020). The main extensions of the journal version are as follows: (1) Based on the proposed single model in the original paper, we generalize it into a reinforcement learning–based personalized

search framework RLPS, and implement two alternative personalized search models within this framework, namely, the listwise RLPS-L and hierarchical RLPS-H. (2) As for the original hierarchical MDP-based model, which is called RLPS-P in this article, we promote it to RLPS-H through refining the hierarchical MDP components and adding document lists to train the underlying ranking model better. (3) The underlying feedback-aware personalized ranking component is also further improved. Considering query reformulations in a session episode, we add a session-based query prediction module to help infer the user's query intent. (4) In addition to the publicly available AOL search log data, we further conduct experiments on a large-scale commercial query log to verify the effectiveness of our RLPS framework.

The rest of the article is organized as follows. Related works of this article, including personalized search models and applications of reinforcement learning, are briefly reviewed in Section 2. We introduce details of our reinforcement learning–based personalized search framework RLPS in Section 3. In Section 4, we describe the datasets, model settings, and selected baselines. And in Section 5, we compare and analyze the experimental results. Finally, we make a conclusion of the whole work in Section 6.

2 RELATED WORK

The related work to this article mainly concerns two fields: (1) search results personalization and (2) the applications of reinforcement learning. In the following, we separately introduce the details of the two parts of research works.

2.1 Search Results Personalization

Personalized search is a hot research field in information retrieval. Its target is to make the search results of ambiguous or broad queries more accurate for each user by incorporating their individual preferences. Numerous models have been proposed. In general, the basic idea of personalization is as follows: construct a user interest profile by analyzing the user's historical queries and click behaviors; then, re-rank the candidate documents based on the matching scores between the documents and the created user profile. According to the approaches to analyzing user preferences or building user interest profiles, we divide current studies into two categories: traditional personalization models and deep learning–based models.

Traditional personalized search models usually depend on mining the click-based features or document topics from the search history to analyze user interests. Motivated by the users' refinding behaviors [36] in search which imply users issue the same query to search for the same information, Dou et al. [15] proposed a simple but effective method, P-Click. This method gives priority to the documents clicked by the user under the same query in the history. Many personalization studies [4, 10, 19, 25, 31, 39] adopt topic models, such as the **Open Directory Project** (**ODP**), to analyze topic-based features from the clicked documents and build user profiles in the topic space. In addition, more fine-grained features about user interests are considered. The SLTB model [5] extracts a lot of features from the query log, including click-based features, original ranking position, query entropy, and so forth. Then, it employs a **learning to rank** (**LTR**) algorithm LambdaMART [7] to aggregate these features to calculate personalized scores. In addition to the click-based and topic-based features, the location and reading level of users are also demonstrated to have certain effects on search results personalization [3, 12]. These traditional methods manually extract information from the query logs to build user interest profiles, achieving certain improvements over personalized search. However, some drawbacks still exist since there are obvious limitations on manual feature design in that the covered information is incomplete.

Benefiting from the automatic learning ability of machine learning and deep learning, the drawbacks of traditional personalized models have been gradually relieved. Learning-based models

usually learn a representation of the user interest profile [38] or train a personalized search model [32] for each individual user. Song et al. [32] and Wang et al. [40] proposed frameworks that adapt a general ranking model to an individual search model with a small number of queries from the user. A hierarchical recurrent neural model with a query-aware attention mechanism (HRNN) [17] was proposed to capture the sequential information from the historical query logs and dynamically build long-term and short-term user profiles according to the current query. PS-GAN [24] is a **generative adversarial network** (**GAN**) –based personalized search framework, which enforces the personalized model to pay more attention to the training samples that are difficult to distinguish and learn a better representation of the user interest profile.

All the aforementioned approaches consider the sequential search process as a static set of user query behaviors. They learn fixed ranking strategies from all the recorded query logs, without continuous updates along with the change of the dynamic user interests. Differently, we track the entire search process and update the personalized ranking strategy continuously, obtaining a model suitable for the dynamically changing user interests best.

2.2 The Application of Reinforcement Learning

Reinforcement learning is usually used to solve problems that can be regarded as a process of sequential decisions or interactions [34]. It has been widely applied to IR [28, 42, 45, 47, 48] and recommendation systems [29, 30, 41, 50, 51]. For the ad-hoc search, Zeng et al. [42] initially proposed a LTR model based on MDP [34], called MDPRank. This model samples a document to rank at the current position in each step until constructing a ranking list. In other studies, MDP [45] and multi-armed bandits [28] were utilized to solve the problem of search results diversification. In addition, Zeng et al. [48] modeled the multi-page search process as an MDP and took the user's feedback of the previous pages to optimize the document list of the next page. However, all these RL-based ranking models proposed for the ad-hoc search task have not taken the user interests, the whole search history, and future search behaviors into account. Furthermore, the datasets used in these studies always have precise annotations of document relevance, which is different from the practical query log data with only noisy click labels used in personalized search.

In recommendation systems, the process during which the system recommends items and the user gives feedback can be naturally regarded as sequential interactions between the user and the recommender agent. Thus, there have been a lot of studies modeling the recommendation problem as an MDP and training models using the reinforcement learning framework. An MDP-based recommendation system [30] was proposed at an early time to consider the long-term effect of the current recommended item. Recently, several deep reinforcement learning models [49, 50, 51], trained by the **deep Q-network** (**DQN**) algorithm [34], were proposed to track the sequential interactions during the recommendation process. Furthermore, a reinforcement learning–based interaction interface [18] was designed to facilitate the users to express their interests. This interface utilizes reinforcement learning for exploration and exploitation. Both recommendation tasks and personalized search tasks are supposed to consider the user interests reflected in the interaction history. However, the user does not issue a specific query in recommendation systems, thus, it is not necessary to consider relevance with queries but only the user interests in recommendation. In this article, we fully consider the characteristics of personalized search to design our reinforcement learning–based framework RLPS.

3 RLPS—A REINFORCEMENT LEARNING–BASED FRAMEWORK FOR PERSONALIZED SEARCH

In this article, we focus on the essence that a user's personal search history can be regarded as a sequential interaction process between the user and the search engine. During the interaction

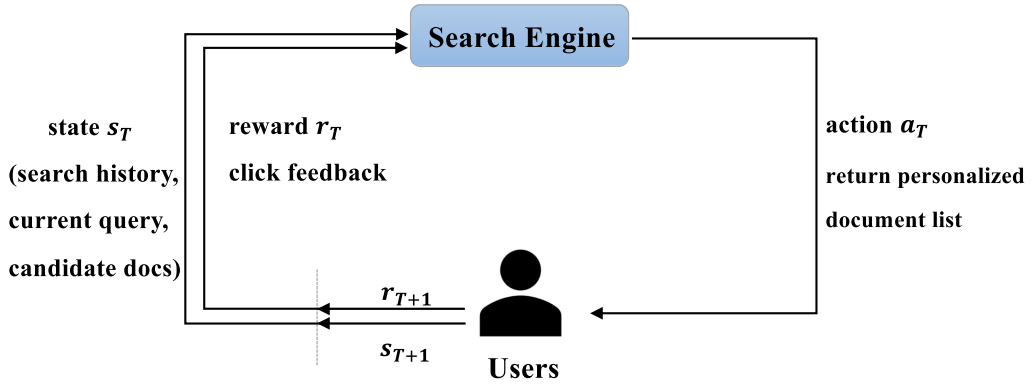


Fig. 1. Illustration of the interaction in RLPS framework. The search engine is the agent and the user is the environment.

process, the user's interests dynamically change. To learn a personalized ranking strategy fitting to the dynamically changing user interests best, we leverage the reinforcement learning on the problem of personalized search and propose a framework **RLPS**. In this framework, we utilize an MDP to model the user's entire interaction process and continuously update the personalized ranking strategy with the user's real-time feedback.

In this section, we first demonstrate the whole architecture of the proposed framework RLPS. Then, we introduce the implementations of two personalized search models within the framwork, namely, the listwise model RLPS-L and the hierarchical model RLPS-H. Finally, we describe the policy gradient training algorithm and the online test algorithm in detail.

3.1 RLPS—The General Framework

In our framework, we treat the personalized search engine as the agent, the user as the environment, and model the sequential interactions between the user and search engine as a reinforcement learning process, which is illustrated in Figure 1. This process and the personalization problem to be solved can be formulated with corresponding notations as follows. At each timestep T, the user u with search history H_T issues a query q_T. The underlying non-personalized search model returns a general ranking list D_T of candidate documents. Facing the current environment $\{H_T, q_T, D_T\}$ composed of the user's search history, issued query, and returned general document list, the personalized search engine is expected to take an action a_T. It analyzes the user's search history H_T to construct the user interest profile, then utilizes its current personalized ranking model M_T to re-rank the candidate documents in D_T based on the created user profile and generate a personalized document ranking list D'_T. Then, the user browses the document list D'_T and clicks or skips documents, based on which a reward r_T is calculated to indicate the quality of the ranking result. The personalized search engine (agent) updates the current ranking model M_T to M_{T+1} based on the search data and received reward r_T. The environment turns into a new state when the user enters a new query q_{T+1}. The new search history in the new state includes the last query q_T and search result D'_T, i.e., $H_{T+1} = H_T + \{q_T, D'_T\}$. The user interest profile will also be different from the last one due to the new search history. During the interaction process, the personalized ranking model is updated based on the user's real-time click feedback continuously until it converges to the optimal model.

The search process described above can be considered as a sequential decision process in which the personalized search engine determines the order of the documents in the document list returned to the user. We can mathematically formalize the process as an MDP, which is always

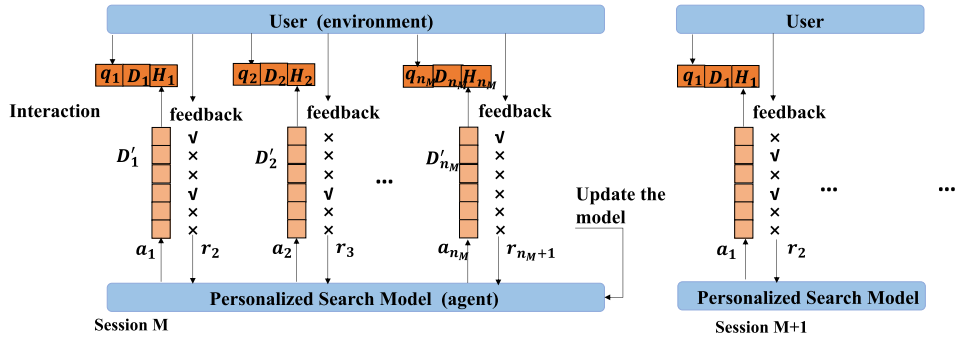


Fig. 2. Illustration of the RLPS-L model. During the sequential interaction process, each time the user issues a query and the search engine returns a document list.

represented as a tuple $\langle S, A, \mathcal{T}, \mathcal{R}, \pi \rangle$, including the state, action, transition, reward, and policy, and apply the policy gradient algorithm to train the underlying personalized ranking model. Therefore, there are two main parts in the RLPS framework: the components of the MDP tuple and the underlying personalized ranking model. We can design these modules in different ways to generate distinct personalized search models within this framework. In this article, we implement two typical models: the listwise RLPS-L and hierarchical RLPS-H. And we describe the two models in the next sections.

Through formulating the problem as a reinforcement learning process, our RLPS framework shows several advantages for personalized search:

(1) RLPS tracks the sequential interaction process and captures the user's interest variations during the search process to learn the dynamically changing user interests better.

(2) Along with the interaction process, the RLPS framework continuously updates the ranking strategy with the user's newly issued queries and real-time feedback to obtain a personalized ranking model fitting the user's current interests better.

3.2 RLPS-L: Listwise RLPS

During the sequential interaction process described in the RLPS framework, the user and the search engine interact in units of query and document list: each time the user issues a query and the search engine returns a document list. Thus, we naturally propose a listwise RLPS model, i.e., RLPS-L, in which we set each step of the reinforcement learning process to correspond to a query and an action of the agent is to generate a document list to be returned. In personalized search, we are committed to analyze the user interests reflected in the search process to help generate more accurate ranking results. Existing reinforcement learning models designed for ad-hoc retrieval [42, 48] mainly aim to optimize the search result of each single query, regardless of the impacts of other queries, which are not suitable for the personalization problem. Considering that users usually have consistent search intents in a search session, we set a session as an MDP episode and aim to maximize the long-term return of a whole session. The architecture of the listwise model RLPS-L is shown in Figure 2, and the details about each component of the MDP tuple are introduced as follows.

State S is a set of states describing the environment. In personalized search, the search engine (agent) is expected to re-rank the candidate documents based on both the user issued query and the user interests reflected in the search history. Therefore, we define the state at the step T as $s_T = \{H_T, q_T, D_T\}$. Consistent with the framework described in Section 3.1, H_T is the user's search

history before the current query q_T, and D_T is the list of candidate documents for q_T returned by the original non-personalized search model.

Action A is a set of actions the agent can select, which depends on the current state s_T and is also denoted as $A(s_T)$. In the listwise RLPS-L model, we are supposed to generate a personalized document list in each step. Given that we can regard the task of search results personalization as reconstructing a ranking list with the candidate documents based on user interests, we define the action set of the RLPS-L model as all possible permutations of documents in the list. The personalized search engine takes an action a_T to select a document ranking list to return to the user. Considering the size of the permutation space is too large and that valid clicks usually happen on the first few documents, we mainly focus on permutations of the first k documents (k is a hyper-parameter). For example, supposing we set the parameter k as 3 and there are five documents in D_T sorted by the original non-personalized search model, i.e., $D_T = \{d_1, \ldots, d_5\}$; the action set of the RLPS-L model should be $\{(d_1, d_2, d_3), (d_1, d_3, d_2), (d_2, d_1, d_3), (d_2, d_3, d_1), (d_3, d_1, d_2), (d_3, d_2, d_1)\} + (d_4, d_5)$. Through action sampling in reinforcement learning, every permutation is likely to become training data with reward as the label, increasing the number of training samples compared with supervised learning models.

Transition $\mathcal{T}(S, A)$ is a function $\mathcal{T} : S \times A \rightarrow S$ that maps the current state to the next state after an action was taken. The search engine takes an action a_T to return a documents list, which the user browses and clicks. Then, the user issues a new query q_{T+1}, and the behavior information about the current query is added to the user's search history. The transition is expressed as the following equations:

$$s_{t+1} = \mathcal{T}(s_t, a_t) = \{H_{t+1}, q_{t+1}, D_{t+1}\}, \tag{1}$$

$$H_{t+1} = H_t + \{q_t, D'_t\}. \tag{2}$$

Reward $\mathcal{R}(S, A)$ provides a supervision signal for the model training in reinforcement learning and is used to measure the quality of actions. After the search engine (agent) takes an action to return a document ranking list, the user will browse the list to click or skip the documents. This generates a relevance label for each document. Then, based on these labels, values of some evaluation metrics such as the **normalized discounted cumulative gain** (**NDCG**), **mean average precision** (MAP), the number of inverse document pairs, and so on can be calculated on this document list. Because we complete search result personalization by re-ranking the original non-personalized document list, we use the difference between the evaluation value of the document list selected by a_T and the original document list D_t as the reward. A higher reward means a greater improvement by our personalized ranking model. For example, we utilize the evaluation metric MAP as our reward function and calculate rewards as

$$r_T = MAP(a_T) - MAP(D_T). \tag{3}$$

Policy $\pi(a|s) : A \times S \rightarrow [0, 1]$ is a probabilistic distribution over the action set calculated with the current state, used as the policy to direct actions. From the descriptions above, we know that any action in $A(s_T)$ corresponds to selecting a possible permutation of the document list. We can calculate a personalized score for each document with a personalized ranking model, obtaining a list of scores $LSC_i = \{sc_{i1}, sc_{i2}, \ldots, sc_{in}\}$. Referring to the listwise LTR models ListNet [9] and ListMLE [23], we compute the probability of each document list L_i to obtain the final behavior policy $\pi(a|s)$ as follows.

$$score(L_i) = \prod_{j=1}^{k} \frac{\exp(sc_{ij})}{\sum_{l=j}^{n} exp(sc_{il})}, \tag{4}$$

$$\pi(a_i|s_T) = \frac{\exp(score(L_i))}{\sum_{L_j \in A(S_T)} exp(score(L_j))}. \quad (5)$$

Here, the first k documents in the original non-personalized document list D_T are arranged, and n is the total number of documents in D_T.

In RLPS-L, we can apply any learning-based personalized ranking models to compute the personalized scores for candidate documents, such as HRNN [17] or our proposed PHRNN model described in Section 3.3.2.

3.3 RLPS-H: Hierarchical RLPS

RLPS-L can successfully track the sequential interaction process to capture the user's dynamically changing interests and update the ranking strategy continuously. However, when we consider the characteristics of personalized search, we find several aspects to improve.

First, studies have shown that users' click behaviors are noisy and biased, and clicks cannot be used as absolute relevant judgments [21]. Because of this, the existing pointwise RL approaches [42, 48] and the simple listwise RLPS-L model may not perform adequately in the personalized search. Thus, we plan to exploit click preferences and adopt a pairwise learning to rank algorithm to train the underlying personalized ranking model in RLPS. The agent needs to determine the relative order of a document pair in each step.

Furthermore, during the actual interaction process, the search model is expected to return a document list to the user at each time, and we also hope it to sample all the document pairs under each query for model training. Therefore, in a session episode, we need to take account of both the document lists and document pairs. To implement interacting with the document lists and sampling all document pairs under each query for model training, we design a hierarchical RLPS-H model with a hierarchical MDP. In this method, we use the high-level MDP to model the interaction process with the user in units of query and document list, while in the low-level MDP, the model is required to process all document pairs under each query. Finally, both the document lists and document pairs are used for model training. We use T and t to represent the step number of the two levels, respectively.

In addition, we also design a feedback-aware personalized ranking component called PHRNN to capture the user's feedback. PHRNN is introduced in Section 3.3.2. The architecture of the proposed RLPS-H model including the hierarchical MDP structure and the underlying personalized ranking component PHRNN is shown in Figure 3. The details of each component are introduced as follows.

State. There are two levels of MDP in this model. As for the high-level MDP of interactions, the state at each step T is defined as $s_T = \{H_T, q_T, D_T\}$, which is the same as RLPS-L. In the low level, it is necessary for the search engine to sample all document pairs under the current query q_T to train the ranking model. We use $P_T = \{p_1^T, p_2^T, \ldots\}$ to represent the set of document pairs comprised of all documents in D_T. The model needs to determine the relative order of a document pair in each step t. Thus, we have the state $s_t^T = \{H_T, q_T, D_T, p_t^T\}$.

Action. In the high-level MDP, the search engine is required to return a personalized document list D'_T to the user at each step T. We define the action $A(s_T)$ as generating a document list based on the personalized scores of the documents calculated by the current ranking component M_T. In each step t of the low-level MDP, the agent compares the relevance of the two documents in the document pair $p_t^T = (d_i, d_j)$. Thus, the action set $A(s_t^T)$ can be defined as all possible relationships of the two documents, i.e., d_i is more relevant than d_j $\{(d_i > d_j)$, the two documents have the same relevance $(d_i = d_j)$, and d_i is less relevant than d_j $(d_i < d_j)\}$. The search engine samples an action a_t^T to determine their relative relationship.

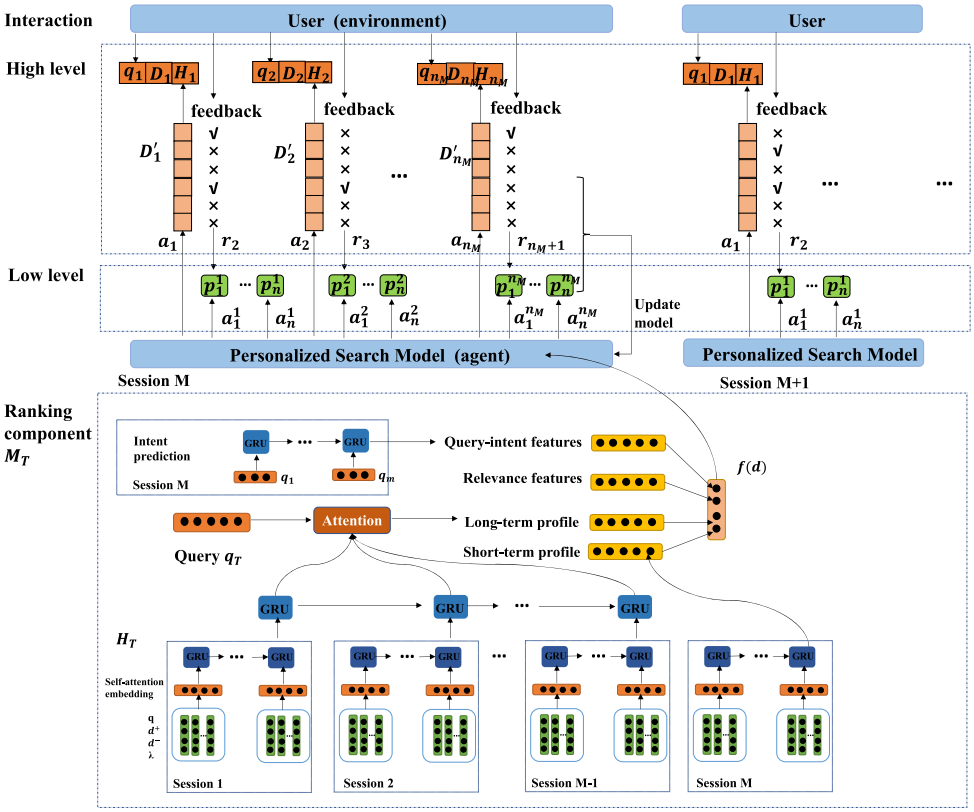


Fig. 3. Illustration of the RLPS-H model. The hierarchical MDP is on the top whose high level tracks the sequential interactions in units of query and document list, and low-level samples all document pairs under each query to train the ranking model. The proposed ranking component PHRNN used to compute the personalized scores for documents is at the bottom. Each query in the search history is represented by a series of document pairs.

Transition. In the high-level MDP of the interaction, the transition happens when the user inputs a new query q_{T+1}, and the information of the current query is added into the user's search history. As for the low-level MDP, with the clicked document list, we create a set of document pairs P_T, and the agent takes an action a_t^T to judge the relative relationship of the two documents in the pair p_t^T step by step. All the document pairs in P_T, the actions, and the corresponding rewards are collected to update the personalized ranking model from M_T to M_{T+1}. Consequently, the transition function $\mathcal{T}$ for the hierarchical MDP is expressed as the following equations:

$$s_{t+1}^T = \mathcal{T}(s_t^T, a_t^T) = \{h_T, q_T, D_T, p_{t+1}^T\}, \tag{6}$$

$$s_{T+1} = \mathcal{T}(s_T, a_T) = \{h_T + \{q_T, D_T'\}, q_{T+1}, D_{T+1}\}. \tag{7}$$

Reward. In the hierarchical RLPS-H model, we focus on using both document lists and document pairs as the training data. For document lists, we follow the RLPS-L model to use the evaluation metric difference as the reward. With regard to document pairs, we refer to the state-of-the-art

pairwise LTR algorithm LambdaRank [6] to calculate our reward. In LambdaRank, there is a matrix $\triangle$ where each element $\lambda_{i,j}$ denotes the difference between the metric values before and after exchanging the documents d_i and d_j in the ranking list. This matrix reflects the relative relationship of the documents. *Unlike those supervised learning models that calculate the matrix $\triangle$ based on the document list recorded in the query log, our model calculates it based on the currently returned personalized document list D'_T in the interaction. Such real-time feedback reflects the user's current interests, which can help RLPS train the personalized ranking model better.* We give a positive λ to the document pairs that are determined correctly by the model and a negative λ to the incorrectly determined pairs. The evaluation metrics can be MAP, the number of inverse document pairs, and so on.

Policy. Actions of the high-level MDP that is generating a personalized document list directly depends on the calculated personalized scores. Therefore, we only need to compute the policy for actions in the low-level MDP. From the descriptions above, we know that any action $a \in A(s_t^T)$ corresponds to a possible relative order of a document pair. Referring to the pairwise loss, we calculate the probability of any action as follows:

$$\pi(a_t^T|s_t^T) = \frac{\exp\left(f'\left(a_t^T\right)\right)}{\sum_{a \in A(s_t^T)} \exp\left(f'(a)\right)}, \tag{8}$$

$$f'(a) = \begin{cases} f(d_i|s_t^T) - f(d_j|s_t^T) & a = (d_i > d_j) \\ 0 & a = (d_i = d_j) \\ f(d_j|s_t^T) - f(d_i|s_t^T) & a = (d_i < d_j) \end{cases}, \tag{9}$$

where $f(d_i|s_t^T)$ and $f(d_j|s_t^T)$ are the personalized scores for documents d_i and d_j calculated by the current personalized ranking model M_T. We design a **feedback-aware personalized ranking component** specifically to adapt to our reinforcement learning–based RLPS framework, which is introduced in Section 3.3.2.

3.3.1 The Mixture Policy. Under standard reinforcement learning framework, the model is randomly initialized at the beginning. It may perform unstably and even converge to a worse state. To speed up the model learning process and ensure the model effects, we refer to the imitation learning and make some adjustments to the action policy in the RLPS framework. In addition to the action policy calculated by the agent, we also create an expert policy to direct the search engine how to interact with the user. The expert policy, denoted as $\tilde{\pi}$, is a deterministic policy built on the user's feedback, which merely gives probability to the action leading to the largest reward, defined as

$$\tilde{\pi}(a_T|s_T) = \begin{cases} 1, \, if \;\; a_T = \arg\max_{a_T \in A(s_T)} \mathcal{R}(a_T, s_T) \\ 0, \, \text{otherwise.} \end{cases} \tag{10}$$

The final behavior policy is obtained by linearly combining the expert policy $\tilde{\pi}(a_T|s_T)$ and the MDP calculated policy $\pi(a_T|s_T)$ as follows:

$$\hat{\pi}(a_T|s_T) = \epsilon * \tilde{\pi}(a_T|s_T) + (1 - \epsilon) * \pi(a_T|s_T), \tag{11}$$

where ϵ is a hyper-parameter used for balancing the two parts. The expert policy can be regarded as the annotation in supervised learning, and it can help the model to converge faster. But we still expect to apply the exploration of reinforcement learning. Thus, following [29], we set the

balancing parameter ϵ to decay exponentially at the rate of p, i.e.,

$$\epsilon \leftarrow \epsilon * p, 0 \leq p \leq 1. \tag{12}$$

In this way, the expert policy can direct the actions at the early stage to speed up the model learning process and keep the model stable. Then the action policy calculated by the agent becomes more important for directing behaviors, playing the advantages of reinforcement learning.

3.3.2 Feedback-Aware Personalized Ranking Component. In the previous descriptions about the RLPS framework and two implemented models, we mentioned a personalized ranking model that is used to build user interest profiles, calculate personalized scores for documents to generate ranking lists, and compute the action probability in Equation (4) and Equation (8). It can be any deep learning–based personalized model. In this article, to adapt to the reinforcement learning framework RLPS, we follow the state-of-the-art personalized model HRNN [17] and make some improvements on it to design our ranking model **PHRNN**. In HRNN, each query in the search history is represented by the concatenation of the query vector and the average vector of all clicked documents under this query, but those unclicked documents are totally ignored. In actual recorded query logs, users' click behaviors are usually noisy and cannot be used as absolute relevant judgment [21]. Some behaviors may be biased by document positions. Moreover, we also employ the relative relationships between documents to train our personalized ranking model with a pairwise learning to rank algorithm. Thus, it will be more beneficial for personalization and model training if information about the user's preferences can be mined from the historical data. Therefore, in our personalized ranking model PHRNN, we keep all document pairs for each historical query instead of merely clicked documents. Each query is represented by a set of document pairs constructed on the returned document list, and each pair corresponds to a concatenated vector (q, d^+, d^-, λ). q is the query vector, d^+ and d^- are vectors of the clicked and unclicked documents, respectively. λ is a weight for this document pair, which is calculated by LambdaRank [6]. The meaning of the weight λ has been expressed in Section 3.3. This weight can catch user's click feedback, getting a **feedback-aware personalized ranking model** fit to the RLPS framework with real-time interactions and user feedback. Note that the weights are not calculated on the document list recorded in the query log but on the current document list generated by the ranking model and the user's clicks in the interaction. This feedback is added to the user's history when it turns into the next query, and it will have impacts on the user interest profile for the subsequent queries. In addition, we set a search session as an episode in our RLPS framework, which also contains the variations and reformulation of query strings. Thus, we refer to some query suggestion models [20, 33] and implement a query intent prediction module in PHRNN. This module encodes the historical query sequence to obtain a representation vector as the real query intent. The concrete calculation for personalized document scores of our proposed personalized ranking model PHRNN is introduced as follows.

Recall that in the RLPS framework, the state faced by the search engine is $s_T = \{H_T, q_T, D_T\}$, and the ranking model is required to compute personalized scores for candidate documents. PHRNN splits the whole search history H_T into the long-term history and short-term history, i.e., $H_T = \{L_u^T, S_M^T\}$, and analyzes the short-term and long-term user interests from them, respectively. The short-term history refers to the past queries in the current session, which usually have consistent query intent with the current query, and queries in the previous sessions constitute the long-term history. The personalized score for documents is determined by the relevance from four aspects: the relevance with the currently issued query, the relevance with the short-term user interests, the relevance with the long-term user interests, and that with the predicted query intent. In particular,

the personalized ranking score $f(d|s_T)$ of the document d is calculated with the following formula:

$$\begin{aligned} f(d|s_T) &= f(d|\{H_T, q_T, D_T\}) \\ &= \mathcal{F}(\text{score}(d|q_T), \text{score}(d|S_M^T), \text{score}(d|L_u^T), \text{score}(d|q_P)). \end{aligned} \tag{13}$$

Here, $f(\cdot)$ denotes the score function, and $\mathcal{F}$ is an MLP layer used to combine the four relevance scores. $\text{score}(d|q_T)$, $\text{score}(d|S_M^T)$, $\text{score}(d|L_u^T)$, and $\text{score}(d|q_P)$ represent the relevance with the query, short-term interests, long-term user interests, and the predicted query intent, respectively. q_P represents the predicted query intent. The structure of PHRNN is shown at the bottom in Figure 3, as the underlying personalized ranking model of the agent. In the next, we briefly describe the key components of the PHRNN model, and more details about calculation, which is the same as HRNN, can be found in [17].

(1) For $\text{score}(d|q_T)$, we follow the SLTB model [5] to extract some click-based and topic-based features, such as original position of the document and click entropy of the current query, represented as $v_{q_T,d}$. Moreover, considering our previous idea that unclicked documents are important for us to capture user preferences, we also cover several additional features about the skipped documents. This relevance score is computed by aggregating these features with an MLP layer, $tanh(\cdot)$, as the activation function:

$$\text{score}(d|q_T) = tanh(\mathcal{F}(v_{q_T,d})). \tag{14}$$

(2) For $\text{score}(d|S_M^T)$, we use a low-level RNN^l to capture the sequential information in the current session and build the short-term interest profile. First, we represent each query as a matrix $DP_{M,i}$ composed of all document pair vectors, containing the user's preferences under this query. Each document pair is denoted as a concatenated vector (q, d^+, d^-, λ). Then, we apply a self-attention layer [37] and a dense layer to encode the document pair matrix into a single embedding of the query $q_{M,i}$, i.e.,

$$q_{M,i} = \mathcal{F}(\text{attention}(DP_{M,i}, DP_{M,i}, DP_{M,i})), \tag{15}$$

where $\text{attention}(\cdot)$ is the attention function and $\mathcal{F}$ represents the dense layer. The embedding sequence of all queries in the current session are fed into the low-level RNN^l step by step, $h_i^l = RNN^l(h_{i-1}^l, q_{M,i})$, and we take the last-step output $h_{n_M}^l$ as the short-term user profile. The relevance score between the short-term user interest and the document d is calculated by cosine similarity as

$$\text{score}(d|S_M^T) = \text{sim}(h_{n_M}^l, d). \tag{16}$$

(3) For $\text{score}(d|L_u^T)$, we use a high-level RNN^h and a query-aware attention mechanism to dynamically build the long-term user profile according to the current query, taking the short-term profile vectors of the past sessions $\{h_{n_1}^l, \ldots, h_{n_{M-1}}^l\}$ as the input. First, we compute the hidden state of the past sessions step by step, $h_m^h = RNN^h(h_{m-1}^h, h_{n_m}^l)$. Then, the attention weights for all the historical session vectors are calculated through a dense layer $wt_i = \sigma(\mathcal{F}(q_T, h_i^h))$, and normalized to α_i with a soft-max function. Finally, we obtain the long-term user profile as the weighted sum of the historical session vectors and compute the relevance score.

$$h_{M-1}^{h,q^T} = \sum_{i=1}^{M-1} \alpha_i h_i^h, \tag{17}$$

$$\text{score}(d|L_u^T) = \text{sim}(h_{M-1}^{h,q^T}, d). \tag{18}$$

(4) For score$(d|q^P)$, we first use an RNN to encode representations of all past queries in the current session into a vector h_M^q. Then, an MLP layer is built on the vector to process it and get the predicted query intent q_P. The relevance between the document and the predicted query intent is computed as the cosine similarity, i.e.,

$$\text{score}(d|q_P) = \text{sim}(q_P, d). \tag{19}$$

3.4 Training with Policy Gradient

In this article, we adopt the widely used policy gradient algorithm REINFORCE [34, 44] to train the underlying personalized ranking model of the agent. The parameters of the model to be learned mainly include the parameters for building user interest profiles, predicting the query intent and computing the personalized scores. Here, we take the hierarchical model RLPS-H as an example and describe its training approach as follows. As for the listwise model RLPS-L, the training process is the same, but the created training samples are different.

At first, we track each user's search process to sample interaction episodes and generate training data for the REINFORCE algorithm. In RLPS-H, we consider a search session S_m as an interaction episode and a hierarchical MDP is applied. The high-level MDP processes document lists while the low-level MDP is for document pairs constructed on the list. For the query q_T issued at the timestep T in the session, the agent first takes an action a_T in the high-level MDP. It computes personalized scores for the candidate documents with the current personalized ranking model to generate a document ranking list, and the user clicks on the list to give feedback r_T. Then, the agent processes all document pairs constructed on the generated document list in the low-level MDP. For each document pair p_t in step t, we compute the mixture action policy $\hat{\pi}(a|s_t^T)$ and sample an action a_t^T to determine the relative relationship of the two documents in the pair. The expert policy is calculated according to the user's click feedback. And a reward r_{t+1}^{T+1} is given to the action based on the λ calculated with the user's click feedback on the returned document list. After all the document pairs of the current query q_T are processed, it turns to the next query q_{T+1}. Until the end of this session, an interaction episode is obtained $E = (s_1, a_1, r_2, s_1^1, a_1^1, r_2^2, s_2^1, \ldots, s_{n_m}, a_{n_m}, r_{n_m+1}, s_n^{n_m}, a_n^{n_m}, r_{n+1}^{n_m+1})$. n is the number of document pairs under each query, and n_m is the total number of queries in this session S_m. We separate the high-level and low-level MDPs into $E_h = (s_1, a_1, r_2, \ldots, s_{n_m}, a_{n_m}, r_{n_m+1})$ and $E_l = (s_1^1, a_1^1, r_2^2, s_2^1, \ldots, s_n^{n_m}, a_n^{n_m}, r_{n+1}^{n_m+1})$, and use both the document lists and pairs sampled in the whole episode to update the model at the same time. In order to facilitate the description of the policy gradient training algorithm below, we merge the episodes of the two levels together and simplify the representation as $E = (s_1, a_1, r_2, s_2, \ldots, s_N, a_N, r_{N+1})$, where N represents the total number of steps in the episode.

For each sampled episode E, we are able to incorporate the future performance and calculate the discounted cumulative reward starting from each step t as G_t.

$$G_t = \sum_{k=1}^{N-t+1} \gamma^{k-1} r_{t+k}, \tag{20}$$

where γ is the discounted factor. Then we can split the whole episode E into many transitions (s_t, a_t, G_t) as training samples to train the underlying personalized ranking model. In order to make more effective use of these sampled data, we follow another reinforcement learning algorithm DQN [34] to apply memory replay technology. This model maintains all the sampled transitions in the memory space and randomly samples a mini-batch of transitions from the memory to update the model every time.

In the policy gradient algorithm, the discounted cumulative long-term return of the start state s_1 is usually regarded as a valid evaluation of the model's performance. Therefore, we define the

ALGORITHM 1: Training with REINFORCE

Input: training set D, learning rate η, discount factor γ, reward function $\mathcal{R}$, batchsize B
Output: well-trained parameters w
initialize w randomly, a replay memory $RM = []$
repeat
for all $S_m \in D$ **do**
sample an episode $E = (s_1, a_1, r_2, s_2, \ldots, s_N, a_N, r_{N+1})$
compute and store the N transitions (s_t, a_t, G_t) into RM
sample B transitions (s_t, a_t, G_t) from RM
compute gradient $\triangle w \leftarrow \frac{1}{B}(\sum_{i=1}^{B} G_t \bigtriangledown_w \log \pi(a_t|s_t; w))$
update the parameters $w \leftarrow w + \eta \triangle w$
end for
until converge
return w

expectation of the long-term value starting from the first step as the optimization of our model, represented as $J(w)$. w is the parameters in the ranking model to be trained.

$$J(w) = E_{\hat{\pi}}([G_1]). \tag{21}$$

G_1 in Equation (21) is the value of G_t when t is equal to 1.

Deduced from the above two formulas, the gradient $\bigtriangledown_w J(w)$ in the REINFORCE algorithm can be calculated as

$$\bigtriangledown_w(J(w)) = G_t \bigtriangledown_w \log \hat{\pi}_(a_t|s_t; w), \tag{22}$$

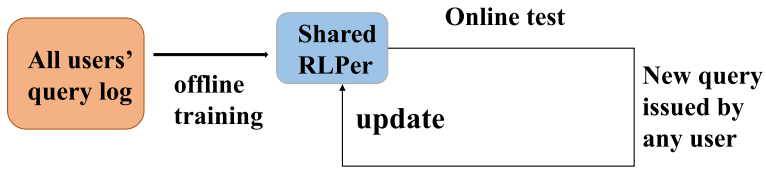
where a_t is the action sampled under the state s_t and the mixture behavior policy $\hat{\pi}_(a|s_t, w)$.

We train the underlying personalized ranking model of our RLPS framework, i.e., the PHRNN model described in Section 3.3.2, with a mini-batch of samples every time and update the parameters according to the gradients calculated in Equation (22). The complete procedure is formulated in Algorithm 1.

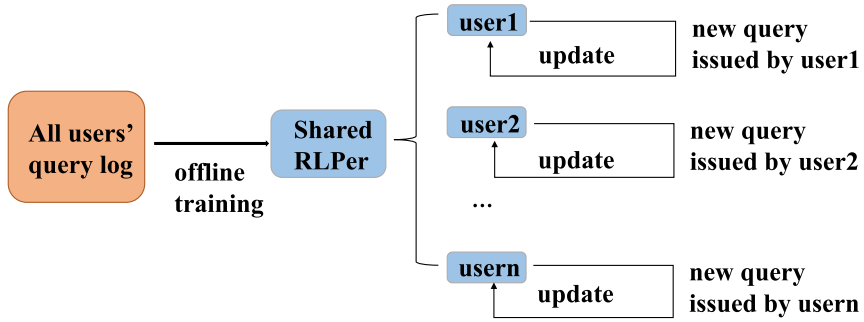
Here, we make a brief discussion about the efficiency of our RLPS framework. Concerning the model training process, we know that a model trained under the reinforcement learning framework usually converges more slowly than those supervised models and may fall into a worse state. We have introduced a deterministic expert policy as the behavior direction and form a mixture policy to speed up and stabilize the model training process. After the training process of the underlying personalized ranking model is completed, the RLPS framework costs about the same time as other deep learning–based personalized search models to build the user interest profile and re-rank the documents. Thus, compared with existing personalized models, our RLPS does not obviously impact efficiency.

3.5 Testing Online

Different from the existing supervised models for personalized search, RLPS proposed in this article is an interactive framework based on reinforcement learning. It tracks the user's entire search process as an interaction sequence with the search engine, and continuously updates the underlying personalized ranking strategy with the user's real-time feedback. With this kind of setting, the RLPS framework can be directly applied to an actual search situation, in which it interacts with users. In this section, we elaborate the algorithm about how to do an online test with our trained models, imitating the actual search situation.



(a) Online test approach1



(b) Online test approach2

Fig. 4. Illustration of the two online test approaches. Approach 1 trains a shared personalization model offline, and applies the same model for all users online, updating it with new queries issued by any user. Approach 2 clones a separate model from the shared model for each individual user, and updates the personal model with queries from the corresponding user.

Suppose we have trained a model offline, and it is online for users to use from now. A search session is viewed as an episode. In each round of interaction at the timestep T in a session, the user uploads a query q_T to the personalized search engine. Facing the state $s_T = \{H_T, q_T, D_T\}$, the search engine takes an action to return a personalized document list to the user. Then, the user provides real-time feedback to the agent by clicking or skipping the returned document. With the feedback as annotations, the agent generated several training samples for the underlying personalized ranking models. The user will continuously issue the next query q_{T+1}, and the information about the current query is added to the user's search history to update the user interest profile. Until the end of this search session, the agent updates the personalized ranking model with the annotated training samples generated during the whole episode. Above is the online update process. Following the same procedure but using a little different settings, we design two approaches for online usage. We illustrate the operating process of the two online test approaches in Figure 4.

(1) We first train a shared personalized ranking model offline based on all users' query logs under the RLPS framework, and then launch the only model online for all users to use. During the process of online test, we update the ranking model continuously with newly issued queries by all users according to the online test algorithm described above.

(2) Ideally, each user can have its own personalized ranking model enhanced from her individual query data. But it is impractical to train a separate ranking model from scratch for each individual user due to the limited amount and sparsity of personal query log data. To ensure the performance of the personalized model for each user, we still first train a shared model with all users' query logs offline. Then, we create a separate model cloned from the shared model for each individual

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 query len	2.845	2.832	2.895	3.208	3.263	3.281
avg #click	1.249	1.118	1.115	1.194	1.182	1.202

user and launch it online. Each cloned individual model is updated only when the corresponding user issues new queries.

We experiment to evaluate the two online testing strategies in Section 5.

4 EXPERIMENTAL SETTINGS

4.1 Dataset and Evalution Metrics

We select two non-personalized search log datasets to evaluate our model, which avoid the biases caused by other personalized models. Their statistics are presented in Table 1.

AOL Dataset. This is a publicly available dataset containing 3 months of search log from March 1, 2006 to May 31, 2006. Each record includes a user ID (anonymous), a session ID, a query, the time when the query was issued, a clicked document, and its ranking position in the original returned ranking list. There are 657,426 users and 16,946,938 queries in this log. We conduct some pre-processing on the dataset, including filtering all non-alphanumeric characters in the queries, segmenting sentences into words and lowercasing all the texts. Then, we compute the query vector by averaging the embeddings of all the terms in the query, and the document representation is calculated as the weighted average of each word embedding multiplied by its TF-IDF weight [17]. Furthermore, we also split the search sequence into sessions in the same way as [2, 22], with boundaries decided by the similarity between two consecutive queries.

Considering that most personalized search models achieve search results personalization by analyzing user interests from their individual search history, as well as our proposed PHRNN, we separate the whole query log into the historical data and experimental data. The query logs before April 3, 2006 are designated as the historical data, which is used as the background to indicate user interests. The last 8 weeks' search data is regarded as the experimental data, which are further divided into the training set, validation set, and testing set with a 6:1:1 ratio. To ensure that all users have sufficient personal search history for building user interest profiles, we remove those users with search history less than three sessions.

In the AOL search log, only clicked documents are recorded for each query, without unclicked documents. But we have claimed that users' click behaviors are noisy and preference information reflected by document pairs composed of a positive document and negative document is more reliable. Thus, we follow [1, 2] to sample unclicked documents and construct the candidate document list for each query. For a given query, we first rank all recorded documents in the query log with the BM25 algorithm. Then, we navigate to the positions of the clicked documents under this query and sample a fixed number of candidate documents centered at these positions. We sample a total of five candidate documents per query for the training set and validation set, while 50 candidate documents for each testing query.

Commercial Dataset. This search log is collected from a commercial search engine during the time period from January 4, 2013 to February 28, 2013. The original commercial search engine has not employed personalization techniques so that our experimental results on this dataset are

guaranteed not to be influenced by other personalized search models. There are a total of 354,063 users and 3,503,934 queries. Each query record in this log contains a user id, a query string, query issued time, the top 20 URLs retrieved by the original search engine, click labels, and their dwelling time. We follow [5, 15, 17] to segment each user's entire query log into sessions by longer than 30 minutes of inactivity time, obtaining 654,774 sessions in total.

We follow [5, 17, 39] to annotate the relevance for documents. Clicks with longer than 30 seconds of dwelling time or the last click of a search session can be viewed as clicks that the user is really satisfied with, i.e., SAT-click. For a URL without SAT-click under the current query, if a SAT-click occurs on it in the following two queries that have at least one common URL with the current query, we also give a SAT-click to this URL. Such processing can help us augment effective data. Finally, documents with SAT-click are regarded as relevant documents while the others are irrelevant documents. To ensure that all users have enough personal search history for analyzing the users' interests, we first divide the query log into experimental data and search history data which is only for mining user interests not for training or testing. Logs in the first 6 weeks are used as the historical data and the last 2 weeks logs as the experimental data. The experimental data are further split into the training set, validate set, and testing set by session with 4:1:1 ratio.

Evaluation. Following [42, 50], we use the users' click behaviors recorded in the query logs to simulate the real-time feedback in the interaction process. Supposing that the SAT-clicked documents are relevant, and the other documents are irrelevant, we utilize the widely used ranking metrics MAP, **mean reciprocal rank** (**MRR**), and precision at 1 (P@1) to evaluate our model. And we also utilize the average ranking position of the documents with SAT-click to measure the model's quality, denoted as Avg.Click [17]. A lower value of this metric indicates a better ranking model. In addition, considering the fact that users' click actions on the original document lists are influenced by the original order, some documents are not clicked may not be because they are irrelevant but because of their low rankings. Consequently, the generic evaluation metrics calculating with the SAT-click labels and positions may be somewhat problematic. Stated in [13, 21], it is more reasonable to take the documents skipped above the SAT-click and the non-clicked next documents as irrelevant. In this work, we use this approach to collect inverse document pairs from the original ranking, and follow [17, 24] to define a more credible metrics P-Improve. P-Improve is the percentage of the increased correct pairs compared with the original search results. Detailed calculations can be referred to in [24]. Note that we only compute the P-Improve metric on the commercial query log, because its recorded document lists are really displayed to the users. However, the original ranking lists of the AOL dataset are constructed by BM25, without returning to users.

4.2 Baselines

With regard to the AOL dataset, the original rankings are generated with the classical BM25 algorithm. The original document lists of the commercial dataset are directly returned by the corresponding search engine. In addition to the original rankings, we select several state-of-the-art neural ranking models, personalized search models, and a reinforcement learning–based learning to rank model as the experimental baselines. The details about all these models are listed as follows:

(1) **KNRM+User:** KNRM [46] is a kernel-based neural model for document ranking. It constructs a word similarity matrix between the query and document, and then employs a kernel-pooling technique on this matrix to extract multi-level soft match features. All these match features are combined with a learning-to-rank function to calculate the final ranking score. To introduce user interests into KNRM, we add the user profile construction module in the HRNN model.

(2) **Conv-KNRM+User:** This model [14] is an upgrade of KNRM. Instead of considering only single terms, it first utilizes convolutional neural networks to represent n-grams of various lengths and soft matches them. Then, the kernel-pooling technique and learning-to-rank layer are applied to those n-gram soft matches. Furthermore, the component of constructing user profiles is also combined.

(3) **P-Click:** Inspired by the users' re-finding behaviors that users often issue the same query to search for the same results, Dou et al. [15] proposed a basic personalization strategy P-Click. This model re-ranks documents based on the number of clicks the user made under the same query in the search history with Borda Count ranking fusion method [16]. P-Click brings great benefits to repeated queries.

(4) **SLTB:** Bennett et al. [5] extracts diverse features from the search history to analyze user interests, including click-based features, topic-based features, query entropy, and so on. All these features are aggregated by the pairwise LTR algorithm LambdaMart [7] to generate the final personalized ranking list. SLTB was regarded as the best before applying deep learning models.

(5) **HRNN:** This study [17] uses a hierarchical recurrent neural network with query-aware attention mechanism to dynamically build the short-term and long-term user interest profiles. It focuses on the sequential information hidden in the search process and highlights the historical queries relevant to the current query. Then, documents are re-ranked based on the relevance with the short-term and long-term user profiles.

(6) **PSGAN:** [24] is a personalized search framework aiming to overcome the problem of noisy training data on the basis of **generative adversarial network** (**GAN**). It can generate queries that express the user's query intent better and enforces the model to pay more attention to the document pairs that are difficult to distinguish. We implement the document selection–based model PSGAN-D in this paper.

(7) **MDPRank:** MDPRank [42] is a learning-to-rank model based on the reinforcement learning framework. It employs a MDP to model the construction process of a document ranking list. Each step corresponds to sampling a document from the candidate set to rank at the current position, and the promotion of DCG [11] after appending the document is defined as the reward. To adapt to personalized search, we add the module of user interest profile construction in HRNN [17] to the original MDPRank model.

(8) **RLPer:** This is the original model proposed in the WWW 2020 paper. It can be regarded as a pairwise implementation within our framework.

4.3 Model Settings

Parameter settings for our personalized ranking component PHRNN are referred to HRNN [17]. We train a 50-dimension Glove model [27] with the whole query log as the corpus, and obtain representations for the queries and documents by averaging the vectors of their terms. The short-term interest representation and the long-term interest representation are 300-dim. and 600-dim., respectively; the dimension of the hidden state is 300. The number of hidden units in the attention layer is 300, and that of the MLP is 512. More details can be found in HRNN [17].

To determine the parameters of our RLPS framework, we conduct multiple sets of experiments under the supervision of the validation set. Finally, the parameters are set as follows. The learning rate is $1e-4$ and the reward discount factor is 0.8. As for the mixture policy, the weight of the expert policy ϵ initialized as 1 and the decay rate p as $\{0, 0.9\}$. In the query logs, we find about 80% of the SAT-clicks occur on the first three documents. Thus, in the listwise model RLPS-L, we consider the permutation of the first three candidate documents, i.e., $k = 3$.

Table 2. Overall Performances of Models on the AOL Set

Model	MAP		MRR		P@1		Avg.Clk	
Random Rank	0.0924	−82.96%	0.09611	−82.67%	0.0233	−95.41%	25.53	141.9%
BM25	0.2504	−53.83%	0.2596	−53.18%	0.1534	−68.4%	17.53	66.08%
P-Click	0.4224	−22.11%	0.4298	−22.49%	0.3788	−21.96%	16.52	56.61%
SLTB	0.5072	−6.47%	0.5194	−6.33%	0.4657	−4.06%	13.92	31.98%
HRNN(list)	0.4116	−24.10%	0.4227	−23.77%	0.3425	−29.44%	15.64	48.25%
HRNN(pair)	0.5423	-	0.5545	-	0.4854	-	10.55	-
PSGAN-D	0.5480	+1.05%	0.5600	+0.99%	0.4892	0.78%	10.26	−2.70%
MDPRank	0.2728	−49.70%	0.2826	−49.04%	0.1727	−64.42%	15.84	50.16%
KNRM+User	0.4722	−12.93%	0.4836	−12.79%	0.3243	−33.19%	7.55	−28.45%
ConvK+User	0.4901	−9.63%	0.5018	−9.50%	0.3456	−28.80%	7.32	−30.56%
PHRNN	0.5509	+1.59%	0.5638	+1.68%	0.4911	+1.17%	10.06	−4.65%
RLPS-L	0.5182	−4.44%	0.5309	−4.26%	0.4545	−6.37%	10.82	−4.65%
RLPS-P	0.5981*	+10.29%	0.6127*	+10.50%	0.5368*	+10.59%	8.29*	−21.41%
RLPS-H	**0.6032***	**+11.23%**	**0.6185***	**+11.54%**	**0.5402***	**+11.29%**	**8.22***	**−22.09%**

Relative performances compared with HRNN are in percentages. RLPS-P and RLPS-H models significantly outperform all the personalized search baselines with paired test at $p < 0.01$ level. $*$ is used to indicate the significant improvements and the best results are in bold.

5 EXPERIMENTAL RESULTS AND ANALYSIS

In order to analyze and verify the effectiveness of our proposed framework comprehensively, we evaluate our framework and other baselines on the processed AOL search log and the commercial dataset. We report the experimental results in the following subsections and make some analysis.

5.1 Overall Performance

At first, we test all selected baselines and our proposed models PHRNN, RLPS-L, and RLPS-H on both search logs, and compare their overall performance on the whole dataset to measure the effectiveness of our models. To be consistent, we test all models offline without any update. Experimental results of the AOL search log are shown in Table 2 and those of the commercial dataset are in Table 3. From the two tables, we can find the following:

(1) **With regard to all evaluation metrics, our proposed RLPS framework shows significant improvements over the corresponding baselines with paired t-test at p<0.05 level.** Pay attention to the listwise RLPS-L; it outperforms our implemented listwise HRNN a lot, with 25.89% improvement on MAP and 32.70% on MRR on the AOL dataset. As for our proposed hierarchical RLPS-H, it promotes the state-of-the-art HRNN model greatly, with 11.23% improvement on the metric MAP, 11.54% on MRR, and 22.09% decrease on the Avg.Click metric. In addition, the RLPS-H model also performs much better than PSGAN, which exploits the generative adversarial network to enhance the training data of the personalized search model HRNN and achieves certain effects. With our reinforcement learning–based RLPS framework, more samples can also be obtained to train the underlying ranking model through the trial-and-error strategy. Thus, the promotion of our RLPS-H model over PSGAN demonstrates that not only the enhancement of training samples but also other advantages of our framework claimed in the former parts play a role in improving personalized search: First, our framework can capture the dynamic user interests better by training the user's search process. Second, the personalized ranking model is updated continuously along with the interactions based on the user's real-time feedback to maintain the

Table 3. Overall Performances of Models on the Commercial Dataset

Model	MAP		MRR		P@1		Avg.Clk	
Ori.Rank	0.7399	−8.26%	.7506	−8.36%	0.6162	-13.54%	2.211	16.25%
P-Click	0.7509	−6.89%	0.7634	−6.80%	0.6260	-12.17%	2.189	15.09%
SLTB	0.7921	−1.79%	0.7998	−2.36%	0.6901	−3.17%	1.959	3.00%
HRNN(list)	0.7969	−1.19%	0.8093	−1.20%	0.6961	−2.33%	1.915	0.68%
HRNN(pair)	0.8065	-	0.8191	-	0.7127	-	1.902	-
PSGAN-D	0.8135	+0.87%	0.8234	+0.52%	0.7174	0.66%	1.815	−4.57%
MDPRank	0.7654	−5.10%	0.7786	−4.94%	0.6421	−9.91%	2.059	8.25%
KNRM+User	0.5316	−34.09%	0.5401	−34.06%	0.3149	−55.82%	2.431	27.81%
ConvK+User	0.6272	−22.23%	0.6377	−22.15%	0.4488	−37.03%	2.385	25.39%
PHRNN	0.8085	+0.25%	0.8213	0.27%	0.7149	0.31%	1.891	−0.58%
RLPS-L	0.8102	0.46%	0.8223	0.39%	0.7154	0.38%	1.853	−2.58%
RLPS-P	0.8186*	+1.50%	0.8302*	+1.36%	0.7263*	+1.91%	1.797*	−5.52%
RLPS-H	**0.8202***	**+1.70%**	**0.8322***	**+1.60%**	**0.7277***	**+2.10%**	**1.783***	**−6.26%**

Relative performances compared with HRNN are in percentages. RLPS-P and RLPS-H models significantly outperform all the personalized search baselines with paired test at $p < 0.05$ level. $*$ is used to indicate the significant improvements and the best results are in bold.

user's latest interests. To summarize, the great performance of RLPS confirms the effectiveness of adapting the reinforcement learning framework to search results personalization.

(2) **Comparison of the two implementations within the framework shows that the hierarchical RLPS-H model further improves the results a lot on the basis of the listwise RLPS-L model.** We know that the RLPS-H model tracks the user's sequential interaction process from a more fine granularity than RLPS-L, and further samples document pairs which indicate user preferences to train the underlying personalized ranking model. We stated in the previous section that document pairs reflect more valid user interests. And the better performance of RLPS-H indicates the effectiveness of training the personalized ranking models in a pairwise way.

(3) Focusing on our proposed feedback-aware PHRNN, **we find that the PHRNN model performs slightly better than the original HRNN model.** We believe that one reason for the better performance of PHRNN is that taking the user's preferences reflected in the document pairs into consideration is more beneficial for learning the user's interests than using merely the clicked documents, especially for the noisy dataset.

(4) **All the personalized search models promote the original rank results a lot, and this confirms that personalized search has the potential to satisfy the user's information need better and enhance her search experience.** The improvement of the P-Click model [15] proves that there exist some re-finding behaviors in search. The diverse features about users, queries, and documents collected in SLTB [5] can be used to effectively analyze users' interests. Due to the outstanding representation learning ability of deep learning, models based on deep learning achieve the best personalization, including HRNN, PSGAN, our improved PHRNN, and the reinforcement learning–based framework RLPS. The great performance of PSGAN and RLPS also confirms the validity of enhancing the training samples for personalized models.

However, the adapted reinforcement learning–based pointwise LTR model MDPRank does not perform so well on personalized search. We analyze it may be that it is a ranking model specially designed for the ad-hoc search and it does not consider the search process, the long-term reward, and user interests. Furthermore, MDPRank was only proved to perform well on the datasets with precise annotations of document relevance, while the actual AOL query log and commercial dataset used in our experiment are noisy and biased. The worse performance of MDPRank also indicates

Table 4. Results of the Ablation Experiments on the AOL Set

Model Variation	MAP		P@1		Avg.Click	
RLPS-H(off)	0.603	-	0.540	-	8.22	-
-Session	0.584	−3.2%	0.523	−3.1%	8.79	+6.9%
-RL	0.565	−6.3%	0.508	−5.9%	9.36	+13.8%
Online test1	0.605	+0.3%	0.543	+0.6%	8.20	−0.3%
Online test2	0.606	+0.5%	0.546	+1.1%	8.17	−0.6%

"-Session" means setting a query as an episode and "-RL" means training the RLPS-H model in a supervised way. Online test1 is the method that maintains a shared updated model, and online test2 means maintaining a separate model for each user during the test.

the necessity of designing an RL-based model specifically for personalized search. In addition, the two neural IR models combined with user profiles also do not achieve great performance on the commercial dataset. We infer it is because the neural IR model performs worse than the original rankings.

In conclusion, the overall experimental results verify that *our proposed framework RLPS can learn dynamic user interests better than the other evaluated baselines and is able to obtain better personalized ranking model through updating it continuously with the user's real-time feedback.*

5.2 Ablation Analysis and Discussion

In order to analyze how our proposed RLPS framework improves search result personalization and the contribution of each component or setting, we conduct an ablation study about the RLPS-H model on the public AOL dataset. According to the experimental results presented in Table 4, we present some discussions in the following.

Session vs. Query. In reinforcement learning, we usually consider the long-term return of the current action to adjust the policy. Thus, we set a session, which is a search process with independent user intent, as an episode to employ the user's interaction process to help learn their interests better. To confirm the effectiveness of this configuration, we compare it with setting a single query as an episode, ignoring the user's feedback from interactions before and after the single query. The result is shown in the second line in Table 4; we find the model loses 2.3% in MAP, 2.5% in P@1, and increases Avg.Click by 6.06%. These results clearly suggest that *our RLPS framework tracking the user's interaction process helps capture user interests.* We think it may be because the latter queries in the same session can help clarify the user's current query intent.

RL vs. Supervised learning. We adapt the reinforcement learning framework to personalized search in this article. Compared to supervised learning, it can produce more training samples with the trial-and-error strategy and update the model with the user's real-time feedback. To verify the validity of reinforcement learning, we conduct an experiment to train the RLPS-H model in a supervised way. As presented in Table 4, the model trained with the ground-truth drops by 5.5% in MAP, 5.3% in P@1, and increases 12.8% in Avg.Click. The results prove that *the reinforcement learning framework helps train the personalized ranking model better than supervised learning.* More training samples play a big role.

Static vs. Continuous update. In most cases, we train a personalized ranking model offline and test it without updates in a short time. Instead, our proposed RLPS framework can interact with users to update the underlying ranking model continuously with real-time feedback when applied in an actual search situation. Two different approaches are presented in Figure 4. We conduct an online test and show the experimental results in the last two lines in Table 4. We find both approaches further improve the offline results in terms of all metrics. The second approach, which

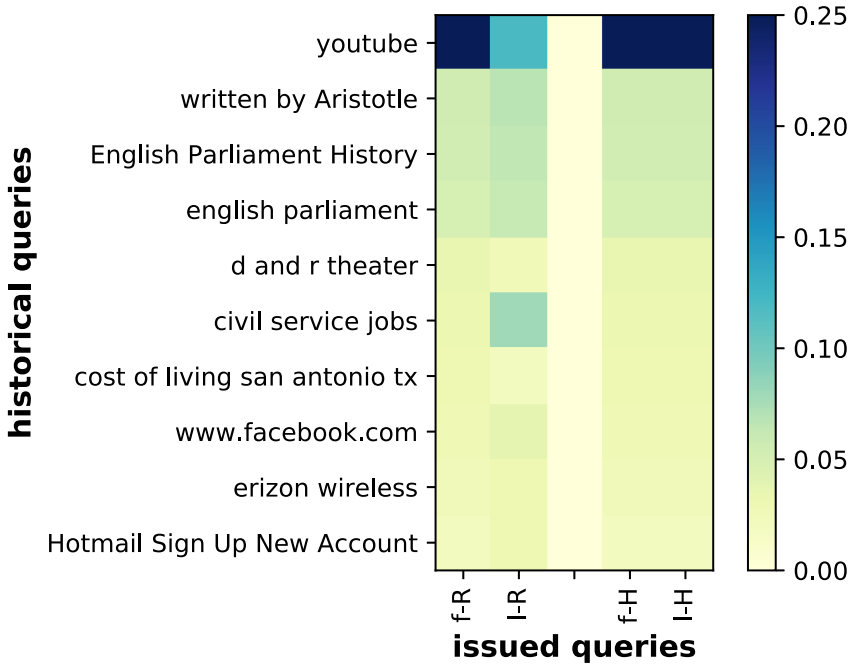


Fig. 5. The weights in the attention layer of HRNN ("H") and RLPS-H ("R") for two same queries issued by a user. In the figure, "f" indicates the former query, and "l" means the latter.

maintains a separate model for each user, demonstrates superior performance. This improvement indicates that *the continuous update mechanism generates a better ranking strategy fitting to the dynamic user interest.* And the excellent performance of the second online test approach inspires us that it can be more effective to keep a unique model for each individual user.

To display the continuous update of our framework more intuitively, we visualize the weights in the attention layer for two same test queries issued at different times. We compare the weights with the baseline HRNN [17] and show the results in Figure 5. In HRNN [17], the attention weights are calculated based on the current query and historical queries. We can find the attention weights of the two queries are different in our RLPS-H model though the two queries are the same, while the weights in HRNN are the same. It proves that our model has been updated between the two test queries to fit the current user interests better.

5.3 Effects of Different Reward Functions

In our reinforcement learning–based RLPS framework, rewards are calculated with an evaluation metric based on the returned document list. We describe an example of the MAP metric. Actually, there are some other available evaluation metrics, such as MRR, P1, and the number of inverse document pairs in the returned document list. In this experiment, we train the personalized ranking model with rewards computed on MAP, MRR, and #Inverse pair, respectively, and illustrate the model performances in Table 5.

Focusing on Table 5, we find that the models trained with different reward functions show similar performance. Although different evaluation metrics are employed, we follow the LambdaRank algorithm to compute the rewards so that their relative values are consistent. In reinforcement

Table 5. The Performance of RLPS-H with Different Reward Functions, Including MAP, MRR, and #Inverse Pair (the Average Number of Inverse Document Pairs in the Returned Document Lists)

Reward Function	MAP		MRR		P@1		Avg.Clk	
MAP	0.6032	-	0.6185	-	0.5402	-	8.22	-
MRR	0.6011	−0.35%	0.6168	−0.27%	0.5388	−0.26%	8.56	4.14%
#Inverse Pair	0.6028	−0.07%	0.6178	−0.11	0.5398	−0.07%	8.51	3.53%

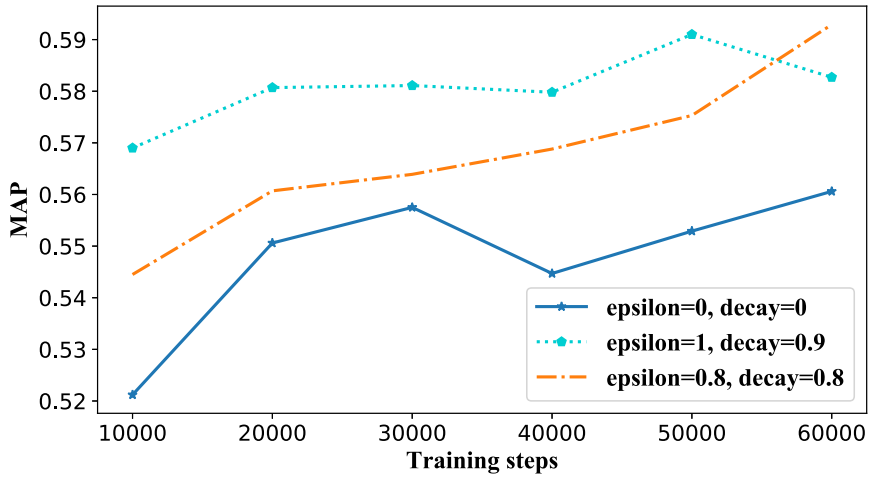


Fig. 6. Performance of RLPS-H with different values of epsilon and decay rate on the AOL dataset.

learning, rewards are used to measure the quality of actions, and the relative value of rewards should be more important than the absolute value for model training.

5.4 Effects of the Mixture Policy

Under the RLPS framework, we do not apply the standard reinforcement learning to train the underlying personalized ranking model but a mixture policy, which creates an expert policy to direct the agent actions and speed up the model training process at an early stage. At the same time, we also set two hyper-parameters, epsilon ϵ and decay rate p, to control the ratio of expert policy and calculated policy. In order to confirm the effects of the mixture policy, we experiment with several groups of epsilon and decay rate. The learning curves of RLPS-H are shown in Figure 6.

From Figure 6, we observe that the model trained with standard reinforcement learning converges much slower than those with a mixture policy. This result proves that the mixture policy has the ability to speed up model training. Comparing the two mixture policies, i.e., $\epsilon = 1, p = 0.9$ and $\epsilon = 0.8, p = 0.8$, we find the former converges faster, but the latter has the potential to achieve better performance in the later stage. We infer that it is because a too large ϵ might limit the exploration of reinforcement learning. Thus, we suggest that a large ϵ in the early stage and a proper decay rate p are helpful for producing a better model.

5.5 Experiments on Different Query Sets

For analyzing the specific contribution of our model on search result personalization, we split all queries into different subsets. Then, we evaluate RLPS-H on different query sets of the AOL log and report the results as follows.

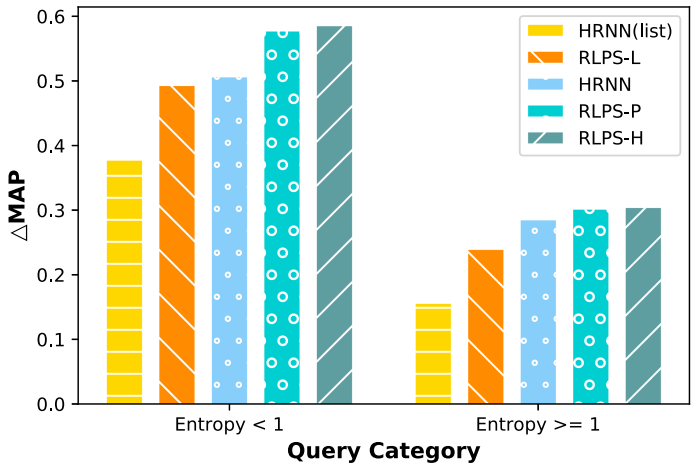


Fig. 7. Results on queries with different click entropy on the AOL dataset.

Ambiguous and non-ambiguous queries. First, we consider the model performance on ambiguous and non-ambiguous queries. The former mainly refers to queries that can be interpreted into several meanings by different users and need more personalization, while the latter tends to express a single intention. As stated in [15], click entropy can reflect how ambiguous a query is. Thus, we categorize all queries into two groups with the cutoff of click entropy at 1.0. Improvement on the MAP metric over the original BM25 rankings is viewed as the evaluation metric. Figure 7 shows the results on the two query groups, respectively.

From Figure 7, we observe that all personalized search models improve the search results on both the clear and ambiguous queries greatly. And all models consistently perform better on the non-ambiguous queries. As for our RLPS framework, it shows the best results on both query groups. Compared to the closest baseline HRNN, both our listwise RLPS-L and hierarchical RLPS-H presents significant improvements. In addition, the RLPS-H promotes the original RLPS-P to some extent, which verifies the effectiveness of document list samples.

Repeated and non-repeated queries. Second, we divide the testing query set into repeated and non-repeated queries according to whether they have appeared in history. For those repeated queries, many traditional models exploit the click features in the search history to help ranking, such as the P-Click algorithm based on the user's re-finding behaviors, SLTB, and so forth. However, it is difficult for them to cope with non-repeated queries due to a lack of information. In terms of the learning-based models, we expect they have the ability to predict document relevance for queries even without direct click-based features. We compare all models' performance on the two query groups, and results are shown in Figure 8.

From Figure 8, we observe that all the personalized search models perform much better on the repeated queries than non-repeated queries. The promotion on the repeated queries confirms that most personalized search models have the ability to take advantage of the click-based features to improve the ranking results. Our proposed models consistently perform the best on both the two groups. Though the RLPS-L model performs a little worse than HRNN on repeated queries, it shows greater effects on non-repeated queries. And the RLPS-P and RLPS-H models make a relatively greater promotion on non-repeated queries. This further demonstrates the effectiveness of our framework to capture the user interests from the search history to better cope with queries issued by the user even for the first time.

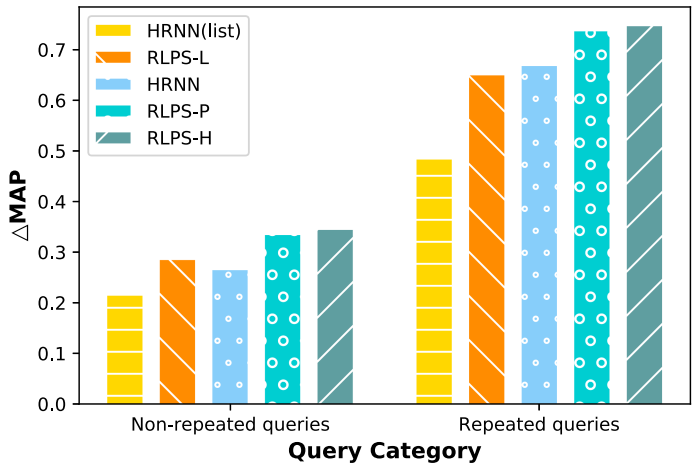


Fig. 8. Results on repeated/non-repeated queries on the AOL dataset.

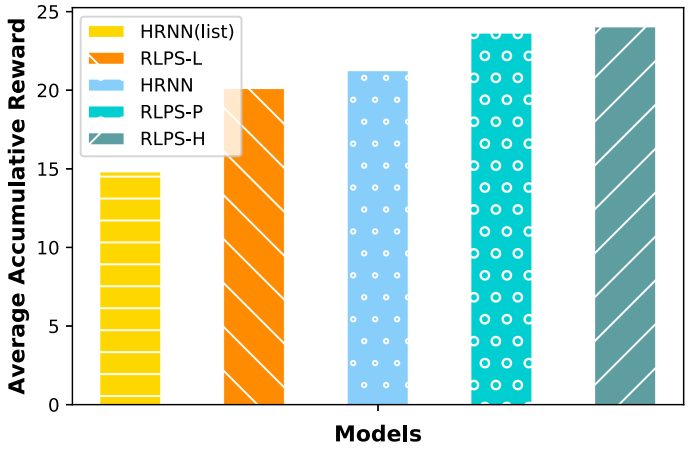


Fig. 9. Experimental results about accumulative reward on the AOL dataset.

5.6 Experiments on Accumulative Reward

In reinforcement learning, the model's optimization target is to maximize the expected long-term return of the initial state. In the RLPS framework, we view all query behaviors in a session as an episode, so that the model is expected to maximize the discounted sum of rewards in a session. To verify the validity of our proposed models from this perspective, we design the metric **average accumulative reward** to evaluate all the baselines and RLPS. We only consider the rewards given to the returned document lists in the sequential interactions, and calculate the average of the accumulative rewards of all sessions. Finally, we show the performance of all models in Figure 9.

Compared with all baselines, we observe that RLPS achieves the highest value of **average accumulative reward**, which meets our expectation and further confirms the correctness of our framework. In addition, a larger accumulative reward in a session indicates that we can satisfy the user's query intent in a session better, and our personalized search models seem more effective in the long run.

6 CONCLUSION

The personal search process can be regarded as sequential interactions between the user and search engine, during which the user interests dynamically change. The existing personalized search models train a fixed ranking model based on all recorded log data and do not update it in a short time. In this article, we propose a reinforcement learning–based framework RLPS, applying an MDP to track the user's interactive search process and continuously update the underlying personalized ranking model with the user's real-time feedback. We set the search engine as the agent and the user as the environment. Each time when the user inputs a query and clicks on the result, RLPS is able to get rewards from the user's feedback, update the ranking model, and create the user profile with new history. Specificaly, we implement the framework in two different ways, namely, RLPS-L and RLPS-H. In RLPS-L, we track the interactions and train the personalized ranking model in units of document lists. As for RLPS-H, we consider a hierarchical structure in which the high-level processes document lists and the low-level captures document pairs with user preferences for model training. Experiments on the public AOL search log and the commercial log data verified the effectiveness of our proposed models. Search is a complex interactive process between users and search engines. In this article, we give a preliminary attempt to model the sequential interactions with a reinforcement learning framework and exploit the most direct user behaviors, i.e., click actions. In the future, we plan to further explore the user's interaction patterns with better models.

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 (ICLR'18)*.

[2] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2019. Context attentive document ranking and query suggestion. In *Proceedings of SIGIR 2019*.

[3] Paul N. Bennett, Filip Radlinski, Ryen W. White, and Emine Yilmaz. 2011. Inferring and using location metadata to personalize web search. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'11)*. 135–144.

[4] Paul N. Bennett, Krysta Marie Svore, and Susan T. Dumais. 2010. Classification-enhanced ranking. In *Proceedings of the 19th International Conference on World Wide Web (WWW'10)*. 111–120.

[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 (SIGIR'12)*. 185–194.

[6] 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 on Machine Learning (ICML'05)*. 89–96.

[7] Chris J. C. Burges, Krysta M. Svore, Qiang Wu, and Jianfeng Gao. 2008. *Ranking, Boosting, and Model Adaptation.* Technical Report MSR-TR-2008-109. 18 pages.

[8] 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 (SIGIR'14)*. 835–838.

[9] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning (ICML'07)*. 129–136.

[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 (CIKM 2'10)*. 1849–1852.

[11] Charles L. A. Clarke, MaheedharKolla, Gordon V. Cormack, OlgaVechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. 2008. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 659–666.

[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 (CIKM'11)*. 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 (WSDM'08)*. 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 WSDM 2018*. 126–134.

[15] 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(WWW'07)*.

[16] Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. 2001. Rank aggregation methods for the Web. In *Proceedings of the 10th International World Wide Web Conference (WWW'10)*. 613–622.

[17] Songwei Ge, Zhicheng Dou, Zhengbao Jiang, Jian-Yun Nie, and Ji-Rong Wen. 2018. Personalizing search results using hierarchical RNN with query-aware attention. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM'18)*.

[18] Dorota Glowacka, Tuukka Ruotsalo, Ksenia Konuyshkova, Athukorala, and Giulio Jacucci. 2013. Directing exploratory search: Reinforcement learning from user interactions with keywords. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces*.

[19] 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 (CIKM'13)*. 2309–2314.

[20] 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 (CIKM'18)*. 197–206.

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

[22] Rosie Jones and Kristina Lisa Klinkner. 2008. Beyond the session timeout: Automatic hierarchical segmentation of search topics in query logs. In *Proceedings of CIKM 2008*. 699–708.

[23] Tie-Yan Liu. 2011. *Learning to Rank for Information Retrieval*. Springer. https://doi.org/10.1007/978-3-642-14267-3

[24] 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 SIGIR 2019*. 555–564.

[25] Nicolaas Matthijs and Filip Radlinski. 2011. Personalizing web search using long term browsing history. In *Proceedings of the 4th International Conference on Web Search and Web Data Mining (WSDM'11)*.

[26] Greg Pass, Abdur Chowdhury, and Cayley Torgeson. 2006. A picture of search. In *Proceedings of the 1st International Conference on Scalable Information Systems (Infoscale'06)*. 1.

[27] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*, a meeting of SIGDAT, a Special Interest Group of the ACL.

[28] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. 2008. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th International Conference on Machine Learning (ICML'08)*. 784–791.

[29] Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS'11)*. 627–635.

[30] Guy Shani, Ronen I. Brafman, and David Heckerman. 2013. An MDP-based recommender system. *CoRR* abs/1301.0600 (2013).

[31] Ahu Sieg, Bamshad Mobasher, and Robin D. Burke. 2007. Web search personalization with ontological user profiles. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM'07)*. 525–534.

[32] 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 (WSDM'14)*. 83–92.

[33] 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 (CIKM'15)*. 553–562.

[34] Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning—An Introduction*. MIT Press. http://www.worldcat.org/oclc/37293240

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

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

[37] 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 *Advances in Neural Information Processing Systems 30*.

[38] Thanh Vu, Dat Quoc Nguyen, Mark Johnson, Dawei Song, and Alistair Willis. 2017. Search personalization with embeddings. In *Advances in Information Retrieval—Proceeedings of the 39th European Conference on IR Research (ECIR'17)*. 598–604.

[39] Thanh Tien Vu, Alistair Willis, Son Ngoc Tran, and Dawei Song. 2015. Temporal latent topic user profiles for search personalisation. In *Advances in Information Retrieval—Proceedings of the 37th European Conference on IR Research (ECIR'15)*. 605–616.

[40] 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 (SIGIR'13)*. 323–332.

[41] Lu Wang, Wei Zhang, Xiaofeng He, and Hongyuan Zha. 2018. Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'18)*. 2447–2456.

[42] Zheng Wei, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2017. Reinforcement learning to rank with Markov decision process. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 945–948.

[43] 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 (WWW'13)*. 1411–1420.

[44] Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8 (1992), 229–256.

[45] Long Xia, Jun Xu, Yanyan Lan, Jiafeng Guo, Wei Zeng, and Xueqi Cheng. 2017. Adapting Markov decision process for search result diversification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 535–544.

[46] 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 SIGIR 2017*. 55–64.

[47] Hui Yang, Dongyi Guan, and Sicong Zhang. 2015. The query change model: Modeling session search as a Markov decision process. *ACM Transactions on Information Systems* 33, 4 (2015), 20:1–20:33.

[48] Wei Zeng, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2018. Multi page search with reinforcement learning to rank. In *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR'18)*. 175–178.

[49] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018. Deep reinforcement learning for page-wise recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys'18)*. 95–103.

[50] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with negative feedback via pairwise deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'18)*. 1040–1048.

[51] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Dawei Yin, Yihong Zhao, and Jiliang Tang. 2018. Deep reinforcement learning for list-wise recommendations. *CoRR* abs/1801.00209 (2018).

Received April 2020; revised November 2020; accepted December 2020