RLPer: A Reinforcement Learning Model for Personalized Search

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

ABSTRACT

Personalized search improves generic ranking models by taking user interests into consideration and returning more accurate search results to individual users. In recent years, machine learning and deep learning techniques have been successfully applied in personalized search. Most existing personalization models simply regard the search history as a static set of user behaviours and learn fixed ranking strategies based on the recorded data. Though improvements have been observed, it is obvious that these methods ignore the dynamic nature of the search process: search is a sequence of interactions between the search engine and the user. During the search process, the user interests may dynamically change. It would be more helpful if a personalized search model could track the whole interaction process and update its ranking strategy continuously. In this paper, we propose a reinforcement learning based personalization model, referred to as RLPer, to track the sequential interactions between the users and search engine with a hierarchical Markov Decision Process (MDP). In RLPer, the search engine interacts with the user to update the underlying ranking model continuously with real-time feedback. And we design a feedback-aware personalized ranking component to catch the user's feedback which has impacts on the user interest profile for the next query. Experimental results on the publicly available AOL search log verify that our proposed model can significantly outperform state-of-the-art personalized search models.

CCS CONCEPTS

• **Information systems → Personalization**; • **Computing methodologies → Reinforcement learning**;

KEYWORDS

Personalized Search, Reinforcement Learning, MDP

ACM Reference Format:
Jing Yao[2], Zhicheng Dou[1], Jun Xu[1], and Ji-Rong Wen[3,4]. 2020. RLPer: A Reinforcement Learning Model for Personalized Search. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380294

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380294

1 INTRODUCTION

Search is one of the major approaches for us to obtain information in our daily life. When users enter a query in search engines, they usually have a specific query intent. However, studies have shown that the intent cannot be expressed accurately by the issued keyword query which is usually short and ambiguous [12, 29]. Let us take the query *'MAC'* as an example. A makeup artist may use this query to search for information about the cosmetic brand *'MAC'*, while an IT engineer is likely to seek information about the *'MAC'* computer using the same query. In a current search engine, both users may find unwanted documents which are about 'MAC' but are irrelevant to their real information need. Personalized search is a way to solve this problem by taking user interests into account and returning different results to individual users. So far, there have been many research achievements on this task, including the traditional personalization models [4, 8, 9, 12, 15, 26, 33, 37] and learning based models [13, 27, 32, 34] proposed in recent years.

Typically, a user's personal search process can be regarded as a series of interactions between the user and the search engine: the user inputs a query and the search engine ranks the candidate documents with the ranking model. Then, the user clicks or skips documents and implicates her current interests. During the sequential interactions, the user's interests may dynamically change, and the search engine is expected to generate document lists fitting the user's current interests best. Most existing personalized search models simply regard the search process as a static set of user issued queries as well as the retrieved and clicked documents, and learn fixed ranking strategies with all recorded data. Then, they apply the fixed ranking strategies on new queries without continuous update. They ignore the fact that the user's interests are dynamically changing during the interactions. Though some studies [13, 18] have considered user interests are dynamic, they don't pay attention to the change process and ignore that ranking strategies should also change accordingly. In this paper, we argue that **it would be more helpful for personalization if we model user interests dynamically and update the ranking strategy continuously**.

To tackle this problem, we propose a novel **R**einforcement **L**earning based **Per**sonalized search model **RLPer** in this paper. Within RLPer, we utilize a Markov Decision Process (MDP) [28] to track the interactions between the user and the search engine, and update the personalized ranking model continuously. RLPer provides several advantages for search results personalization. Firstly, it is able to learn the user's dynamic interests better by tracking the search process which contains the variations of user interests. Secondly, RLPer has the ability to dynamically adjust to the user's

current interests as the personalized ranking strategy is updated continuously with the user's feedback in the reinforcement learning framework. Thirdly, compared with existing learning based personalized search models, RLPer 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 limited training samples in personalized search.

Efforts have been made on applying reinforcement learning for information retrieval (IR) [36, 39] and recommendation [42–44], but those models are not suitable for personalized search. In this paper, we design our model RLPer fully based on the characteristics of personalized search to better track the interaction process and train the personalized model. We set the search engine as the agent, the user as the environment and a session as an episode to track the dynamic user interests. During a session episode, the search engine interacts with the user in units of query and document list. The user inputs a query and the search engine returns a personalized document list, then the user provides real-time clicks as rewards to the search engine to train the ranking model and the user's interest profile is updated with these new click behaviors. But it would be better to train our personalized ranking model with the document pairs indicating user preferences, because studies [16] have shown the user's click actions are biased and cannot be used as absolute relevant judgement. To interact with document lists but train the model with document pairs, we formulate the user's search process as a hierarchical Markov Decision Process (MDP). The high level MDP tracks the interaction process by queries and document lists, while all document pairs under each query are sampled as training data to update the ranking model in the low level MDP. In addition, we also design a feedback-aware underlying personalized ranking component which can catch the user's click feedback. Policy gradient algorithm REINFORCE is applied to train RLPer. We experiment with the public AOL search log to compare our proposed model with state-of-the-art personalized search models. The results show that our model can significantly improve the effectiveness of personalized search over existing models.

In summary, our main contributions are three-fold: (1) To the best of our knowledge, it is the first time that reinforcement learning being applied to personalized search. (2) We carefully adapt reinforcement learning to personalized search and implement a hierarchical MDP model with feedback-aware personalized ranking component, which fits the personalized search scenario better than existing reinforcement learning models for IR. (3) Experimental results on the public AOL logs verified that the proposed RLPer model significantly improves the quality of personalized search over state-of-the-art models.

The rest of the paper is organized as follows. Related works are reviewed in Section 2. We introduce our model RLPer in Section 3. In Section 4 and Section 5, we discuss the experimental setups and analyze the results. Finally, we conclude the work in Section 6.

2 RELATED WORK

The related works of this paper concern two fields: (1) Search Results Personalization and (2) The Application of Reinforcement Learning.

Search Results Personalization. Personalized search is used to make the search results of ambiguous/broad queries more accurate for each user. Numerous models have been proposed to solve this problem and their basic idea is: First, build user interest profiles by analyzing their query logs. Second, re-rank the candidate documents according to the user profile and the query. Based on the approaches of building user profiles, we divide existing studies into traditional personalized search models and learning based models.

Traditional personalized search models usually define some heuristic rules to analyze the search history and obtain user interests. Focusing on users' re-finding behaviors [30] in search, Dou et al. [12] proposed an effective method P-Click, which thinks the documents clicked under the same query in the history to be more relevant. Many personalized models [4, 9, 15, 19, 26, 33] adopt a topic model to extract topics from the clicked documents and build user profiles in the topic space. In addition, SLTB [5] manually extracts a lot of features from the query logs to realize search results personalization, including click-based features, query entropy and so on. The user's location and reading level [3, 11] were also applied to achieve personalization. These traditional methods manually extract information from the query logs to build user interest profiles, achieving certain improvements. However, there still exist some drawbacks that they are only able to get user interests covered in these limited features but miss other important information.

As machine learning and deep learning become popular, these drawbacks have been gradually relieved. The learning based models usually learn a representation of the user interest profile [32] or train a personalized search model [27]. Song et al. [27] and Wang et al. [34] proposed frameworks for adaptation, which adapt the general ranking model to an individual personalized model with a small amount of queries from that user. A hierarchical RNN model (HRNN) [13] was proposed to capture the sequential information in the historical query logs and generate long-term and short-term user profiles related to the current query. PSGAN [18] enhanced the data for personalized model training.

All the aforementioned approaches consider the search process as a static set of user query behaviors and learn fixed ranking strategies from the recorded query logs, without continuous change. Differently, we track the user's entire search process and update the personalized ranking strategy continuously, obtaining a model fitting the dynamically changing interests best.

The Application of Reinforcement Learning. Reinforcement learning is usually used to solve problems which can be regarded as a process of sequential decisions or interactions [28]. And it has been widely applied on information retrieval (IR) [22, 36, 39–41] and recommendation [24, 25, 35, 43, 44]. As for the ad-hoc search, Zeng et al. [36] initially proposed a learning to rank (LTR) model based on MDP [28], called MDPRank. This model samples a document to rank at the current position in each step until constructing a ranking list. MDP [39] and multi-armed bandits [22] were utilized to solve the problem of search results diversification respectively. In addition, Zeng et al. [41] modeled the multi-page search process as a MDP and took the user's feedback in the former pages to optimize the document list of the next page. However, all these RL based ranking models proposed for ad-hoc search have not taken the user interests, historical and future search behaviors into account. And their datasets always have precise annotations for document relevance, different from the noisy data in personalized search.

In recommendation, the process can be naturally regarded as a sequential interactions between the user and the recommend agent. Thus, many studies model the recommendation problem as a MDP and train models in reinforcement learning framework. A MDP-based recommendation system [25] was proposed at an early time to consider the long-term effect of the current recommended item. Recently, several deep reinforcement learning models [42–44] were proposed to track the sequential interactions during the recommendation process, trained with the DQN algorithm [28]. Furthermore, a reinforcement learning based interaction interface [14] was designed to facilitate the users to indicate their interests. These tasks share some commonalities with personalized search that all of them are supposed to consider the user interests and history. But the recommend agent provides a single item at a time, different from the search engine which needs to sort a document list. And there is no need to think of the relevance with queries in recommendation. In this paper, we fully consider the characteristics of personalized search to design our reinforcement learning model.

3 RLPER - A REINFORCEMENT LEARNING MODEL FOR PERSONALIZED SEARCH

In this paper, we focus on the essence that personal search history is an interaction process between the user and search engine. During this process, the user interests dynamically change. To learn an optimal personalized ranking strategy fitting the dynamically changing user interests best, we propose a reinforcement learning based model RLPer to track the user's entire search process and update the personalized ranking strategy continuously.

In this section, we firstly formulate the problem of personalized search as a reinforcement learning process. Then, we introduce our personalization model RLPer and its policy gradient training algorithm in detail. Finally, we describe the online test algorithms.

3.1 Problem Statement

To start with, let us formulate the problem of personalized search as a reinforcement learning process with notations. We use the personalized search engine as the agent and treat the user as the environment. At each time T, the user u which has search history H_T, inputs a query q_T. And the underlying non-personalized search engine returns a candidate document list D_T. Facing the current environment composed of $\{H_T, q_T, D_T\}$, the personalized search engine takes an action a_T. It utilizes its current personalized ranking model M_T to generate a personalized document list $D_T^{'}$ with candidate documents in D_T, according to the issued query q_T and the user profile built on the search history H_T. Then, the user browses the document list $D_T^{'}$ to click or skip documents, providing a reward r_T to indicate the quality of the ranking result. The agent then updates the current ranking model M_T to M_{T+1} based on the received reward r_T. And the environment turns into a new state when the user issues a new query q_{T+1}. The new search history 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 updated based on the new search history. During this reinforcement learning process, the personalized ranking model is updated based on the user's click feedback continuously until converging to the optimal model.

3.2 RLPer - The proposed Model

The search process described above can be regarded as a sequential decision process during which the personalized search engine decides the order of the documents in the returned document list. Therefore, we mathematically formalize the search process as a MDP, which is always represented as a tuple $\langle S, A, \mathcal{T}, \mathcal{R}, \pi \rangle$ including the state, action, transition, reward and policy. We design each component of the MDP tuple fully based on the characteristics of personalized search.

(1) In personalized search, we are committed to capturing the user interests throughout the whole search process. Existing reinforcement learning models designed for ad-hoc retrieval [36, 41], which only aim to maximize the immediate return of a single query, are not suitable for this problem. Considering that search sessions are viewed as search activities with independent user intents, we use a session as a MDP episode. We aim to maximize the long-term return of a whole session.

(2) Studies have shown that users' click behaviours are noisy and biased, and clicks cannot be used as absolute relevant judgment [16]. Because of this, the existing pointwise RL approaches [36, 41] may not work well in the personalized search. Instead, we exploit click preferences and adopt pairwise learning to rank algorithm to train the personalized ranking model in RLPer. The agent needs to judge the relative order of a document pair in each step in RL framework.

(3) In the actual interaction process, the search model is expected to return a document list to the user at each time. But it needs to sample all the document pairs under a query as the training data. Therefore, we need to take both the list and document pair levels into consideration in a session episode. To implement interacting with document lists but training the model with document pairs under each query, we design a **hierarchical MDP**. In the hierarchical MDP, we use the high level MDP to track the interaction process in units of queries and document lists, while the low level MDP processes all document pairs under each query. We use T and t as the serial number for the two levels respectively.

In addition, we also design a feedback-aware personalized ranking component called **PHRNN** to capture the user's feedback which will be introduced in Section 3.2.2. The architecture of our proposed RLPer model including the hierarchical MDP structure and the underlying personalized ranking component is shown in Figure 1. The details of each component are introduced as follows.

State S is a set of states describing the environment. As for the high level MDP of interactions, the user inputs a query q_T at each time step T. The search engine (agent) is expected to re-rank the documents based on both the inputted 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 problem statement in Section 3.1, H_T is the user's search history before the query q_T, and D_T is the list of candidate documents for q_T returned by the original non-personalized search engine. In the low level, the search engine needs to train the ranking model with all document pairs under the current query q_T. 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, and the model needs to determine the relative order of a document pair in each step t. So we have $s_t^T = \{H_T, q_T, D_T, p_t^T\}$.

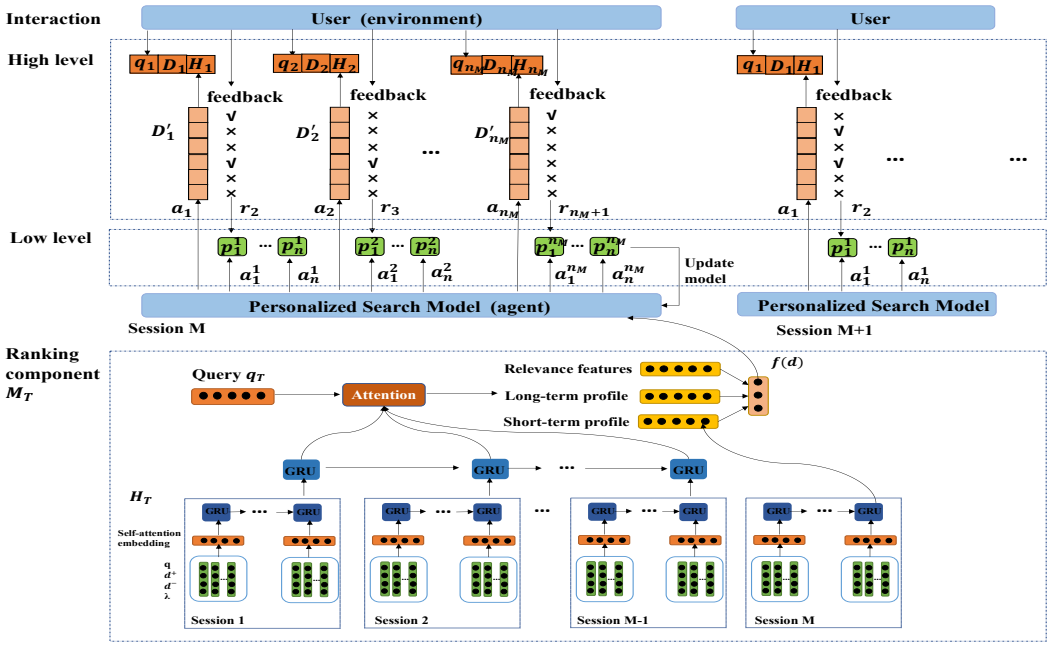

Figure 1: Illustration of the RLPer model. The hierarchical MDP is on the top whose high level tracks the sequential interactions in units of query and document list, and low level trains the ranking model with document pairs under each query. The proposed ranking component PHRNN used to compute the personalized scores for documents is at the bottom. Each query in the search history is represented by a series of document pairs.

Action A is a set of actions for the agent to select, which depends on the current state s_t and is also denoted as $A(s_t)$. In the high level, the search engine is required to return a personalized document list D'_T to the user at each step T, according to the personalized score of the documents calculated by the current ranking component M_T. In each step t of the low level MDP, we consider the agent to compare 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 $\{(d_i > d_j), (d_i = d_j), (d_i < d_j)\}$, and the search engine samples an action a_t^T to determine their relative relationship.

Transition $\mathcal{T}(S, A)$ is a function $\mathcal{T} : S \times A \rightarrow S$ which maps the current state to the next state after taking an action. As for the high level MDP of the interaction, the search engine takes an action to return a personalized document list D'_T to the user, and the user browses the ranking list and clicks to provide real-time feedback. Then, this query and document list with real-time clicks are added to the user's search history for building new user profile. With the clicked document list, we create a set of document pairs P_T, and the agent takes action a_t^T to judge the relative relationship of the two documents in the pair p_t^T step by step in the low level MDP. 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}. When the user inputs a new query q_{T+1}, the high level MDP turns into the next state with new historical data. Consequently, the transition function $\mathcal{T}$ for the hierarchical MDP works as the following equations:

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

$$s_{T+1} = \mathcal{T}(s_T, a_T) = \{h_T + \{q_T, D'_T\}, q_{T+1}, D_{T+1}\}. \tag{2}$$

Reward $\mathcal{R}(S, A)$ provides supervision signals for the model training in reinforcement learning, used to measure the influence of actions. Due to we focus on using document pairs as the training data, we refer to the state-of-the-art pairwise LTR algorithm LambdaRank [6] to design our rewards. In LambdaRank, there is a matrix Δ where each element $\lambda_{i,j}$ means 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. **Different from those supervised learning models which calculate the matrix Δ on the document list recorded in the query log, we calculate 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 RLPer train the personalized ranking model better.** We give a positive λ to the document pairs that are judged correctly by the model and a negative λ to wrong pairs.

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 behaviors. Actions of the high level MDP directly depends on the personalized scores. 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 the document pair. Referring to the pairwise loss, we calculate the probability of any action as follows:

$$\pi(a_t^T|s_t^T) = \frac{\exp\left(f'\left(a_t^T\right)\right)}{\sum_{a\in A(s_t^T)}\exp\left(f'(a)\right)}, \tag{3}$$

$$f'(a) = \begin{cases} f(d_i|s_t^T) - f(d_j|s_t^T) & a = (d_i > d_j) \\ 0 & a = (d_i = d_j) \\ f(d_j|s_t^T) - f(d_i|s_t^T) & a = (d_i < d_j) \end{cases}, \tag{4}$$

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 specially design a **feedback-aware personalized ranking component** to adapt to our reinforcement learning based RLPer model, which will be introduced in Section 3.2.2.

Through formulating the problem as a RL process and designing the MDP components above, our RLPer shows several advantages:

(1) RLPer tracks the interaction process, captures the user's interest feedback and variations during a session to learn the dynamic user interests better.

(2) Along with the interaction process, the RLPer model can continuously update the ranking strategy with the newly received queries and the user's feedback to obtain a model fitting the current user interests best.

3.2.1 The Mixture Policy. In offline training process, we make a few adjustments to the behavior policy based on imitation learning [24]. Because the initialized model in reinforcement learning is random and unstable, we create an expert policy to teach the search engine how to interact with the user at the early stage, speeding up the learning process. The expert policy is a deterministic policy which only gives probability to the actions resulting in the largest reward, denoted as $\tilde{\pi}$. We define $\tilde{\pi}$ as:

$$\tilde{\pi}(a_t^T|s_t^T) = \begin{cases} 1, & if\ a_t^T = \arg\max_{a_t^T\in A(s_t^T)} \mathcal{R}(a_t^T, s_t^T) \\ 0, & \text{otherwise} \end{cases}. \tag{5}$$

We build the final behavior policy as a linear combination of the expert policy $\tilde{\pi}(a_t^T|s_t^T)$ and the MDP calculated policy $\pi(a_t^T|s_t^T)$:

$$\hat{\pi}(a_t^T|s_t^T) = \epsilon * \tilde{\pi}(a_t^T|s_t^T) + (1-\epsilon) * \pi(a_t^T|s_t^T), \tag{6}$$

where ϵ is a hyper-parameter used for balancing the two parts. Following [24], we set ϵ to decay exponentially at the rate of p, i.e:

$$\epsilon \leftarrow \epsilon * p, 0 \le p \le 1. \tag{7}$$

Thus, the expert policy guides the behaviors at the initial stage to help the model learn much faster. Then the MDP calculated policy becomes more and more important for directing behaviors, playing advantages of reinforcement learning.

3.2.2 Feedback-aware Personalized Ranking Component. In our RLPer model, we need a personalized ranking component to build user interest profiles and calculate personalized scores for documents used to generate ranking list and compute the action probability in Eq. (3). It can be any deep personalized model. In this paper, we follow the personalized model HRNN [13] and make some improvements on it to adapt to our RLPer. 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, but those irrelevant documents are totally ignored. Actually, users' click behaviours are noisy and cannot be used as absolute relevant judgment [16]. And we apply pairwise LTR algorithm to train our personalized ranking model in this paper. It can be more beneficial for personalization if information about the user's preferences can be learned from the historical data. Therefore, we propose a model to improve HRNN by taking all document pairs under each query into account instead of merely clicked documents, called **PHRNN**. We represent each query as a batch of document pairs, each pair corresponding 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. Considering RLPer is a model with real-time interactions and user feedback, we add a weight λ for each document pair like LambdaRank [6] to catch the feedback, getting a **feedback-aware ranking model**. Noting that **the weights are calculated based on the currently returned document list and the user's clicks in the interaction. The feedback is added to the user's history which will have influences on the user interest profile for the next query.** Our personalized ranking component PHRNN calculates the personalized score as follows.

Recall that $s_t^T = \{H_T, q_T, D_T, p_t^T\}$ in RLPer. PHRNN splits the whole search history H_T into the long-term and short-term history, i.e., $H_T = \{L_u^T, S_M^T\}$. The short-term history includes past queries in the current session, reflecting the current query intent, while the other queries constitute the long-term history. Then, the model calculates the personalized score for documents with relevance from three aspects: the relevance with the query, the relevance with the short-term user interests and that with the long-term user interests. Specifically, the personalized ranking score $f(d|s_t^T)$ of the document d is calculated by:

$$\begin{aligned} f(d|s_t^T) &= f(d|\{H_T, q_T, D_T, p_t^T\}) \\ &= \mathcal{F}(\text{score}(d|q_T), \text{score}(d|S_M^T), \text{score}(d|L_u^T)), \end{aligned} \tag{8}$$

where f is the score function and $\mathcal{F}$ is a dense layer to combine the three parts. $\text{score}(d|q_T)$, $\text{score}(d|S_M^T)$ and $\text{score}(d|L_u^T)$ represent the relevance with the query, short-term and long-term user interests respectively. The structure of PHRNN is shown at the bottom of Figure 1. We briefly introduce the key components of the model next and more details about calculation can be found in [13].

(1) For $\text{score}(d|q_T)$, we extract some relevance and click features as those in SLTB [5] as well as several additional features about irrelevant documents following our previous idea, represented as $v_{q_T,d}$, then we use a dense layer to aggregate these features:

$$\text{score}(d|q_T) = tanh(\mathcal{F}(v_{q_T,d})). \tag{9}$$

(2) For $\text{score}(d|S_M^T)$, we use the low-level RNN^l to build the short-term interest profile in a session. Here, we represent the i^{th} query as a matrix $DP_{M,i}$ composed of all document pair vectors. Then, a self-attention layer [31] and a dense layer are applied to encode the matrix into a query embedding $q_{M,i}$, i.e: $q_{M,i} = \mathcal{F}(\text{atten}(DP_{M,i}, DP_{M,i}, DP_{M,i}))$, where atten() is the attention function. All query embeddings in a session are fed into the RNN^l and we take the last-step output h_M^l as the short-term user profile. The short-term interests relevance score is calculated as:

$$\text{score}(d|S_M^T) = \text{sim}(h_M^l, d). \tag{10}$$

(3) For score$(d|L_u^T)$, we use the high-level RNN^h and a query-aware attention mechanism to build the long-term user profile, taking the short-term interest profile vectors of the past sessions $\{h_1^l, \ldots, h_{M-1}^l\}$ as the input. First, $h_m^h = RNN^h(h_{m-1}^h, h_m^l)$. Then, the weights for all the historical sessions are calculated through a dense layer $wt_i = \sigma(\mathcal{F}(q_T, h_i^h))$, and normalized to α_i with a softmax function. Finally, the long-term user profile is obtained by $h_{M-1}^{h,q^T} = \sum_{i=1}^{M-1} \alpha_i h_i^h$, and the relevance score is computed as:

$$\text{score}(d|L_u^T) = \text{sim}(h_{M-1}^{h,q^T}, d). \tag{11}$$

3.3 Training with Policy Gradient

In this paper, we adopt the widely used policy gradient algorithm REINFORCE [28, 38] to train our model. The parameters to learn in our model are denoted as w, including the parameters for building user interest profiles and computing personalized scores.

At first, we need to sample episodes as the training data of the REINFORCE algorithm. In the hierarchical MDP of RLPer, we consider a search session S_m as a sampling episode. For each query q_T of the high level MDP issued at time step T in the session, the personalized ranking model need to judge all document pairs in the returned document list in the low level MDP. At each step t, we compute the mixture policy $\hat{\pi}(a|s_t^T)$ on the action set $A(s_t^T)$ and sample a_t^T to determine the relative order of the document pair p_t. A reward r_{t+1}^T is obtained according to the λ calculated with the user's click feedback r_T. After all the document pairs of this query q_T are judged, it converts to the next query q_{T+1}. Until the end of this session, we get an episode $E = (s_1^1, a_1^1, r_2^1, s_2^1, \ldots, s_n^{n_m}, a_n^{n_m}, r_{n+1}^{n_m})$, where n is the number of the document pairs under each query, and n_m is the total number of queries in this session S_m. We use the document pairs sampled in the whole episode to update the model. In order to facilitate the description of the policy gradient training algorithm below, we simplify the representation of the episode as $E = (s_1, a_1, r_2, s_2, \ldots, s_N, a_N, r_{N+1})$, where N represents the length of the whole episode.

With episodes sampled, we can calculate the discounted cumulative reward starting from each step t as G_t:

$$G_t = \sum_{k=1}^{N-t+1} \gamma^{k-1} r_{t+k}, \tag{12}$$

where γ is the discounted factor. Then we can split every episode into many transitions (s_t, a_t, G_t) as training samples to train the model. In order to make more effective use of these sampled data, we follow another reinforcement learning algorithm DQN [28] to apply memory replay technology. This method stores all transitions in the memory and samples a mini-batch of transitions for model training every time.

In the policy gradient algorithm, we use the discounted cumulative long-term return of the start state s_1 to evaluate the model. Therefore, the optimization target of our model, denoted as $J(w)$ where w is parameters in the RLPer model, can be defined as the expectation of the long-term value starting from the first step:

$$J(w) = E_{\hat{\pi}}([G_1]). \tag{13}$$

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

$$\nabla_w(J(w)) = G_t \ \nabla_w \log \hat{\pi}_(a_t|s_t; w), \tag{14}$$

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

In this paper, we train the personalized ranking model in RLPer, i.e, the model described in Section 3.2.2, with a mini-batch of samples every time and update the parameters according to the gradients calculated in Eq.(14). The complete procedure is formulated in algorithm 1.

As for the training process, we know the model trained with the reinforcement learning method converges more slowly than those supervised models, but we have introduced the mixture policy in this paper which combines a deterministic expert policy to stimulate the training process. With the well-trained RLPer, it costs about the same time as HRNN to build the user interest profile and re-rank the documents. Thus, compared with existing personalized models, our RLPer doesn't obviously decrease the efficiency.

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 $\triangle w \leftarrow \frac{1}{B}(\sum_{i=1}^{B} G_t \ \nabla_w \log \pi(a_t|s_t; w))$
    update the parameters $w \leftarrow w + \eta \triangle w$
  end for
until converge
return w

3.4 Testing Online

The RLPer model in this paper is expected to track the user's entire search process, and continuously update the personalized ranking strategy with real-time feedback during the sequential interactions. It can be directly applied in the actual search situation. Here, we analyze the online test algorithm about our trained model.

In each interaction at the time step T in a session, the user inputs a query q_T, and the personalized search engine gets the state $s_T = \{H_T, q_T, D_T\}$. Based on the state, the search engine returns a personalized document list to the user. And the user clicks or skips the documents to give feedback. Then, the search engine takes a series of actions a_t^T to determine the relative order of all document pairs in P_T and gets rewards calculated with the user's feedback. When the user issues the next query q_{T+1}, the information about the last query is added to user's search history to build the new interest profile. Until the end of the session, we update the personalized ranking model with document pairs generated in the whole episode. Following the same process but a little different settings, we propose two approaches for the online test:

(1) We train a shared personalized search model based on all users' query logs offline, and then apply the same model for all users online. Different from the offline test during which the trained model does not change when a test query is received, the online model changes continuously – it is updated each time when a new test query is received.

(2) Ideally, each user can have its own personalized ranking model. Due to the limited amount of personal query log data, it is impractical to train a separate ranking model from start for each individual user. Alternatively, we first obtain a shared model with all training query logs. Then, we create a separate model cloned from the shared model for each individual user. Each cloned individual model is updated when the corresponding user issues a new query.

We experiment with the two online testing strategies in Section 5.

4 EXPERIMENTAL SETTINGS

4.1 Dataset and Evaluation

Dataset We evaluate our model on the public AOL search log [20] collected from 1^{st} March 2006 to 31^{th} May 2006. Following [36, 43], we use the click information in the query logs to simulate the real-time clicks in the interaction. There are totally 657,426 users and 16,946,938 queries in this log. And each record contains an anonymous user id, a query string, query issued time, a clicked url and its original ranking position. We filter all non-alphanumeric characters in the queries, apply word segmentation and lowercasing. We follow [2, 17] to segment the query sequences into sessions based on the similarity between two consecutive queries. In this paper, a query is represented by averaging the embeddings of all its keywords, while a document representation is the weighted average of each word embedding multiplied by its TF-IDF weight.

We need to analyze the user interests from their search history for personalization. To ensure that all users have enough personal search history, we divide the query logs before 3^{rd} April 2006 as the history data which is only for mining the user interests, and the remaining as the experimental data. We filter out those users whose history is less than three sessions. As for the experimental data, we set the first six weeks as the training set and the others are equally divided into validation and testing sets.

The AOL search log records only clicked documents without unclicked documents, both of which are required to train our model in a pairwise way, so we follow [1, 2] to construct the candidate document lists. For a given query, Ahmad et al. [1] aggregated a candidate list with the top documents ranked by BM25 [23], appending the recorded clicked documents. But they [2] observed that many recorded clicked urls' content crawled in 2017 has no lexical overlap with the issued queries, may be because that the content of some urls has been updated since 2006 when the AOL log was sampled. To avoid the influence of such biases on model training, Ahmad et al [2] first collected the top 1000 retrieved documents for each query and filtered out those queries whose recorded clicked documents were not ranked in the top 1000. Then, they found the positions where BM25 ranks the recorded clicks and sampled a fixed number of candidate documents centered at these positions. Finally, 50 candidate documents are selected for each testing query, and 5 candidates per query for the training and validation sets. The statistics of the constructed dataset are shown in Table 1.

Table 1: Statistics of the constructed dataset.

Items	Traininig	Validation	Testing
#session	187,615	26,386	23,040
#query	814,129	65,654	59,082
average session length	3.945	3.298	2.602
average query length	2.845	2.832	2.895
average query #click	1.249	1.118	1.115

Evaluation Consistent with existing works [13, 18], we utilize the widely used IR metrics MAP, MRR and P@1 to evaluate our model. And we also use the average ranking position of the clicked documents [13], denoted as Avg.Click. A lower value of this metric indicates a better model.

4.2 Baselines

We rank candidate documents with BM25 algorithm to generate the original ranking. And we select several state-of-the-art personalized search models and a RL based LTR model as the experimental baselines. All the details are listed as follows:

(1) **P-Click**: Dou et al. [12] proposed the algorithm P-Click, which re-ranks the documents based on the number of clicks the user made under the same query, benefiting repeated queries.

(2) **SLTB**: Bennett et al. [5] analyzed user interests by extracting diverse features from the short-term and long-term search history, 102 features in total. All the features are inputted to the LambdaMart [7] to generate the final personalized ranking list. SLTB was regarded as the best before applying deep learning models.

(3) **HRNN**: It [13] uses a hierarchical RNN with query-aware attention to dynamically build the short-term and long-term user profiles according to the current query. Then, documents are re-ranked based on the user profiles and some relevance features.

(4) **PSGAN**: PSGAN [18] is a generative adversarial network (GAN) framework for personalized search enhanced from HRNN. It generates queries that match the users' query intents better and applies the model composed of a discriminator and a generator to select document pairs more valuable for model training. We reproduce the variant PSGAN-D.

(5) **MDPRank**: MDPRank [36] is a RL based pointwise LTR model. It formalizes the construction process of a document list as a MDP, samples a document from the candidate set to rank at the current position in each step, and defines the promotion of DCG [10] as the reward. We adapt this ad-hoc search model to personalized search by adding the relevance between the document and user interests when computing the relevance score.

4.3 Model Settings

We follow [13] to select hyper-parameters for our personalized ranking component PHRNN. First, we train a 50-dimension glove model [21] on the whole query log to obtain representations of all queries and documents. Dimensions of the short-term and long-term interest vectors are set as 300 and 600 respectively, and dimensions of the hidden state in both the two RNNs are 300. The number of hidden units in the attention layer is 300, and that of the MLP is 512. More setting details are described in HRNN [13].

Table 2: Overall performances of the models. RLPer model significantly outperforms 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. RLPer(off) means testing the model offline without update, the same as other baselines.

Model	MAP		MRR		P@1		Avg.Clk	
Ramdom Rank	.0924	-82.96%	.09611	-82.67%	.0233	-95.41%	25.53	141.9%
BM25	.2504	-53.83%	.2596	-53.18%	.1534	-68.4%	17.53	66.08%
P-Click	.4224	-22.11%	.4298	-22.49%	.3788	-21.96%	16.52	56.61%
SLTB	.5072	-6.47%	.5194	-6.33%	.4657	-4.06%	13.92	31.98%
HRNN	.5423	-	.5545	-	.4854	-	10.55	-
PSGAN-D	.5480	+1.05%	.5600	+0.99%	.4892	0.78%	10.26	-2.70%
MDPRank	.2728	-49.70%	.2826	-49.04%	.1727	-64.42%	15.84	50.16%
PHRNN	.5509	+1.59%	.5638	+1.68%	.4911	+1.17%	10.06	-4.65%
RLPer(off)	**.5981***	**+10.29%**	**.6127***	**+10.50%**	**.5368***	**+10.59%**	**8.29***	**-21.41%**

Parameters of our reinforcement learning based model RLPer are determined as follows. The learning rate is $1e-4$ and the reward discount factor is 0.8. We adopt the mixture policy to train the offline model with ϵ initialized as 1 and the decay rate p as $\{0, 0.9\}$. The parameters are selected under the supervision of the validation set, and we select the model with the best performance on the validation set for testing.

5 EXPERIMENTAL RESULTS AND ANALYSIS

To verify and analyze the effectiveness of our proposed model comprehensively, we take experiments on the processed AOL search log. The experimental results are reported and discussed as follows.

5.1 Overall Performance

The overall performance is a credible measurement of how effective a model is. We test all the baselines and our proposed models PHRNN, RLPer on the whole dataset offline, without any update. The results are presented in Table 2. After observation, we find:

(1) **In terms of all evaluation metrics, our RLPer model shows significant improvements on all baselines with paired test at p<0.01 level**. Pay attention to our RLPer (off), it outperforms state-of-the-art HRNN model greatly, with 10.29% improvement on the metric MAP, 10.50% on MRR and 21.41% improvement on the Avg.Click metric. In addition, our RLPer also outperforms PSGAN a lot. It improves 9.14% on the MAP and 9.73% on the P@1 metric. PSGAN is a model to enhance the training data for HRNN based on GAN and achieves certain effects as presented in Table 2. Recall that our RLPer also has more training samples. Thus, RLPer's promotion on PSGAN highlights that not only more training samples play role in our model, but also tracking the whole interaction process and the continuous update mechanism.

(2) Focusing on our proposed PHRNN and RLPer, **we find that PHRNN model performs a little better than the original HRNN model, and RLPer further improves the results on the basis of PHRNN.** We think one reason for PHRNN's better performance than HRNN is that taking the user's preferences reflected in the document pairs into consideration is more beneficial for learning the user's interests compared to merely the clicked documents, especially for the noisy dataset. In terms of RLPer and PHRNN, we know RLPer adapts the reinforcement learning framework on PHRNN, so the great promotion of RLPer confirms the effectiveness of applying reinforcement learning framework to search results personalization. We analyze the improvements may because of the RLPer's advantages claimed in the former parts: first, it can capture the dynamic user interests better by tracking the search process and obtaining the user's real-time feedback. Second, it updates the personalized ranking model continuously along with the interactions. And third, it can generate more training samples with the trial-and-error strategy.

(3) **All the personalized search models improve the basic ad-hoc ranking results a lot**, which indicates search results personalization is helpful for promoting users' search experience. The improvement of P-Click [12] model proves that users usually perform re-finding behaviors in search. The diverse features about users, queries and documents designed in SLTB [5] can effectively analyze users' interests and query intents. Models based on learning achieve state-of-the-art results, including HRNN and PSGAN which both build user interest profiles with a hierarchical RNN, our improved PHRNN and the reinforcement learning based model RLPer. The great performance of PSGAN also confirms its validity of enhancing the training data for personalized models.

However, the reinforcement learning based pointwise LTR model MDPRank doesn't perform so well on personalized search. The possible reason may be: it is a model designed for the ad-hoc search, lacking the ability to consider the long-term history information and capture the user's interests. Furthermore, MDPRank performs well on the datasets with precise annotations for document relevance, while the actual query logs for personalized search is noisy and biased. Its worse performance also indicates the necessity of designing a RL based model specially for personalized search.

In conclusion, the overall experimental results verify that **our proposed model RLPer can capture user interests better and is able to learn better personalized ranking model**.

5.2 Ablation Analysis and Discussion

In this paper, we claim that the RLPer model provides several advantages in improving personalized search. To analyze how our designed model and which component contributes to personalization, we conduct several ablation studies. Results are shown in Table 3 and we make some discussions as follows.

Table 3: Results of the ablation experiments. '-Session' means setting a query as an episode and '-RL' means train RLPer model in a supervised way. Online test1 is the online test method which maintains a shared updated model, and online test2 means maintaining a separate model for each user during the test.

Model Variation	MAP		P@1		Avg.Click	
RLPer(off)	.598	-	.536	-	8.29	-
-Session	.584	-2.3%	.5234	-2.5%	8.79	+6.06%
-RL	.565	-5.5%	.508	-5.3%	9.36	+12.8%
Online test1	.609	+1.86%	.543	+2.01%	8.18	-1.4%
Online test2	.613	+2.57%	.549	+2.68%	8.08	-2.6%

session v.s. query In our model, we set a session, which is viewed as search activity with independent user intent, as an episode to track the user's interaction process and capture the interests. To confirm the effectiveness of this configuration, we conduct an experiment with a single query as an episode, ignoring the feedback from interactions before and after the single query. From the second line in Table 3, we find the model loses 2.3% in MAP, 2.5% in P@1 and increases 6.06% in Avg.Click without the session episode. This results clearly suggest that **our RLPer tracking the user's interaction process benefits to personalization**. We think it may be because the context interactions can help clarify the user's dynamically changing interests and current query intent.

RL v.s. supervise learning We apply the reinforcement learning framework to deal with the process of interactions in search. Its trial-and-error strategy produces more training samples than supervise learning. We do experiment to train the RLPer model in a supervised way to verify the validity of the reinforcement learning framework. As presented in Table 3, the model trained with the ground-truth drops 5.5% in MAP, 5.3% in P@1 and increases 12.8% in Avg.Click. The results prove that **reinforcement learning framework helps train the personalized ranking model better**. It may due to the RL based model catches the interactive essence of search. In addition to more training samples, one more possible reason may be that whether and how well the query intent is satisfied in the later search process helps to update the current ranking model.

static v.s. continuous update Most of the existing personalized search models are trained offline and tested statically without update in a short time. Instead, we argue that our RLPer model can be updated continuously along with interactions when applied in the actual search situation, and we provide two approaches for online test. Focusing on the online test results shown in the last two lines in Table 3, we find both approaches perform greatly and improve the offline results a lot in terms of all metrics. The second approach, which maintains a separate model for each user, performs the best. This improvement demonstrates that **the continuous update mechanism of the ranking strategy fits the current user interests better.** 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 model more intuitively, we visualize the weights in the attention layer for two same test queries issued at different time. We compare the weights with the baseline HRNN [13] and show the results in figure 2. In HRNN [13], 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 RLPer 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.

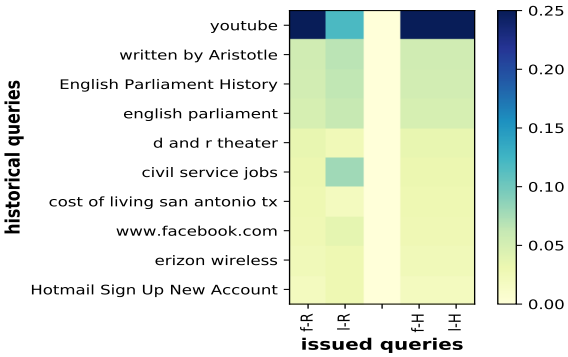

Figure 2: The weights in the attention layer of HRNN ('H') and RLPer ('R') for two same queries issued by a user. In the figure, 'f' indicates the former query, and 'l' means the latter.

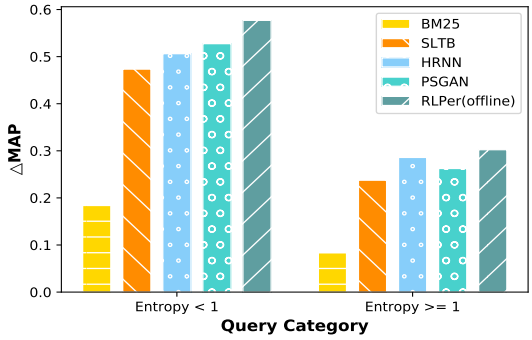

Figure 3: Results on queries with different click entropies.

5.3 Experiments on Queries with Different Click Entropies

As stated in [12], click entropy can reflect how ambiguous a query is. A query with a high click entropy indicates that users have different query intents when inputting this query, and search results personalization should be more necessary for such query. Therefore, we experiment on query collections with different click entropy to evaluate our model more specifically. We categorize all queries into two groups with the cutoff of click entropy at 1.0. And we take the models' improvement on the MAP compared with the original random ranking as the evaluation metric. Figure 3 shows the results on the two query groups respectively.

From the figure, we can find that all personalized search models improve both the clear and ambiguous queries greatly compared

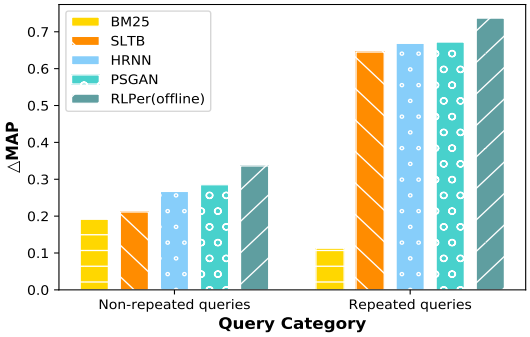

Figure 4: Results on repeated/non-repeated queries.

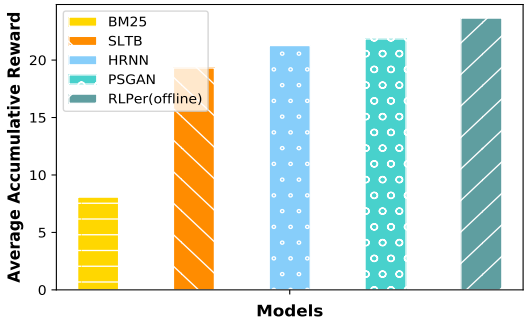

Figure 5: Experimental results about accumulative reward.

with the ad-hoc ranking algorithm. Consistently, our RLPer model performs the best among all the personalized model on both query groups. Compared with the closese baseline HRNN, our model also presents significant improvement.

5.4 Experiments with Repeated/Non-repeated Queries

Studies [12, 30] have shown that it is a common fact that users may issue the same query for the same information in search engines. Inspired by this phenomenon, Dou et al [12] proposed the algorithm P-Click. A large number of features in SLTB [5] are also designed based on the re-finding mechanism, achieving great effects. However, these models based on click features may fail on queries that have never been asked by the user. 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. With this consideration, we divide the testing set into repeated/non-repeated queries and evaluate on the two sets to investigate the effectiveness of our model. Results are shown in Figure 4.

From Figure 4, we find that all the personalized search models perform much better on the repeated queries than non-repeated queries, except for the ad-hoc search method BM25 which is even worse on the repeated queries. The promotion on the repeated queries illustrates that most personalized search models have the ability to take advantage of the click-based features to improve the ranking results. Our proposed model consistently performs the best on both the two groups. Though the absolute improvements of our model on the two query groups are similar, it makes a relatively greater promotion on non-repeated queries. This further demonstrates the effectiveness of our model to capture the user interests from the search history to better cope with queries issued by the user even at the first time.

5.5 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 hierarchical MDP of RLPer, 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 model from this dimension, we design the metric **average accumulative reward** to evaluate all the baselines and RLPer. Recall that we set the λ calculated on the document ranking list as the rewards for document pairs. Due to the baseline models are tested offline, we use the personalized document scores to decide the relative order of the documents in pairs and calculate the average of the accumulative rewards of all sessions. Finally, we show all models' performance in Figure 5.

Compared with all baselines, we observe RLPer achieves the highest value of **average accumulative reward**, which meets our expectation and confirms the correctness of our model. 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 model seems more effective in the long run.

6 CONCLUSION

Personal search process can be regarded as sequential interactions between the user and search engine, during which the user interests dynamically change. Different from the existing personalized search models which train a fixed ranking model at one time, in this paper, we proposed a reinforcement learning based model RLPer to track the user's interactive search process and continuously update the personalized ranking model. Specifically, we set the search engine as the agent and model the interactions as a hierarchical MDP. Each time when the user inputs a query and clicks on the result, RLPer is able to get rewards from the user's feedback, update the ranking model and create the user profile with new history. Experiments on the public AOL search log verified the effectiveness of our proposed model. Search is a complex interactive process between users and search engines. In this paper, we give a preliminary attempt to model this process with reinforcement learning. We modeled and exploited the simplest user behaviour, i.e. clicks for training adaptive personalized models. In the future, we plan to explore richer user behaviours and design better interaction models.

ACKNOWLEDGEMENTS

Zhicheng Dou is the corresponding author. This work was supported by National Natural Science Foundation of China No. 61872370, No. 61832017 and No. 61872338, Beijing Outstanding Young Scientist Program NO. BJJWZYJH012019100020098, and Beijing Academy of Artificial Intelligence No. BAAI2019ZD0305.

REFERENCES

[1] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2018. Multi-Task Learning for Document Ranking and Query Suggestion. In *6th International Conference on Learning Representations, ICLR 2018,*.

[2] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2019. Context Attentive Document Ranking and Query Suggestion. In *Proceedings of SIGIR 2019, Paris*.

[3] Paul N. Bennett, Filip Radlinski, Ryen W. White, and Emine Yilmaz. 2011. Inferring and using location metadata to personalize web search. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*. 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 2010, Raleigh, North Carolina, USA, April 26-30, 2010*. 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 *The 35th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '12, Portland, OR, USA, August 12-16, 2012*. 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 *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*. 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 *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014*. 835–838.

[9] 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 2010, Toronto, Ontario, Canada, October 26-30, 2010*. 1849–1852.

[10] Clarke, L. A Charles, Kolla, Maheedhar, Cormack, V Gordon, Vechtomova, Olga, Ashkan, and Azin. 2008. Novelty and diversity in information retrieval evaluation.

[11] 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 2011, Glasgow, United Kingdom, October 24-28, 2011*. 403–412.

[12] 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 2007*.

[13] 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 2018, Torino, Italy, October 22-26, 2018*.

[14] 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*.

[15] Morgan Harvey, Fabio Crestani, and Mark James Carman. 2013. Building user profiles from topic models for personalised search. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*. 2309–2314.

[16] 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*.

[17] 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.

[18] Shuqi Lu, Zhicheng Dou, Xu Jun, Jian-Yun Nie, and Ji-Rong Wen. 2019. PSGAN: A Minimax Game for Personalized Search with Limited and Noisy Click Data. In *Proceedings of SIGIR 2019*. 555–564.

[19] Nicolaas Matthijs and Filip Radlinski. 2011. Personalizing web search using long term browsing history. In *Proceedings of the Forth International Conference on Web Search and Web Data Mining, WSDM 2011*.

[20] Greg Pass, Abdur Chowdhury, and Cayley Torgeson. 2006. A picture of search. In *Proceedings of the 1st International Conference on Scalable Information Systems, Infoscale 2006, Hong Kong, May 30-June 1, 2006*. 1.

[21] 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 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*.

[22] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. 2008. Learning diverse rankings with multi-armed bandits. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*. 784–791.

[23] Stephen E. Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval* 3, 4 (2009), 333–389.

[24] 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 Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*. 627–635.

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

[26] Ahu Sieg, Bamshad Mobasher, and Robin D. Burke. 2007. Web search personalization with ontological user profiles. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, November 6-10, 2007*. 525–534.

[27] Yang Song, Hongning Wang, and Xiaodong He. 2014. Adapting deep RankNet for personalized search. In *Seventh ACM International Conference on Web Search and Data Mining, WSDM 2014, New York, NY, USA, February 24-28, 2014*. 83–92.

[28] Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement learning - an introduction*. MIT Press. http://www.worldcat.org/oclc/37293240

[29] 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, Singapore, July 20-24, 2008*. 163–170.

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

[31] 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*.

[32] Thanh Vu, Dat Quoc Nguyen, Mark Johnson, Dawei Song, and Alistair Willis. 2017. Search Personalization with Embeddings. In *Advances in Information Retrieval - 39th European Conference on IR Research, ECIR 2017, Aberdeen, UK, April 8-13, 2017, Proceedings*. 598–604.

[33] 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 - 37th European Conference on IR Research, ECIR 2015, Vienna, Austria, March 29 - April 2, 2015. Proceedings*. 605–616.

[34] Hongning Wang, Xiaodong He, Ming-Wei Chang, Yang Song, Ryen W. White, and Wei Chu. 2013. Personalized ranking model adaptation for web search. In *The 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013*. 323–332.

[35] 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 2018, London, UK, August 19-23, 2018*. 2447–2456.

[36] 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, Shinjuku, Tokyo, Japan, August 7-11, 2017*. 945–948.

[37] 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 *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*. 1411–1420.

[38] Ronald J. Williams. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning* 8 (1992), 229–256.

[39] 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, Shinjuku, Tokyo, Japan, August 7-11, 2017*. 535–544.

[40] Hui Yang, Dongyi Guan, and Sicong Zhang. 2015. The Query Change Model: Modeling Session Search as a Markov Decision Process. *ACM Trans. Inf. Syst.* 33, 4 (2015), 20:1–20:33.

[41] 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 2018, Tianjin, China, September 14-17, 2018*. 175–178.

[42] 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 2018, Vancouver, BC, Canada, October 2-7, 2018*. 95–103.

[43] 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 2018, London, UK, August 19-23, 2018*. 1040–1048.

[44] 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).