Context-Aware Ranking by Constructing a Virtual Environment for Reinforcement Learning

Junqi Zhang[1], Jiaxin Mao[1], Yiqun Liu[1*], Ruizhe Zhang[1], Min Zhang[1], Shaoping Ma[1]
Jun Xu[2], Qi Tian[3]
[1]Department of Computer Science and Technology, Institute for Artificial Intelligence, Beijing National Research Center for Information Science and Technology, Tsinghua University
[2]School of Information, Renmin University of China
[3]Huawei Noah's Ark Lab
yiqunliu@tsinghua.edu.cn,

ABSTRACT

Result ranking is one of the major concerns for Web search technologies. Most existing methodologies rank search results in descending order according to pointwise relevance estimation of single results. However, the dependency relationship between different search results are not taken into account. While search engine result pages contain more and more heterogenous components, a better ranking strategy should be a context-aware process and optimize result ranking globally. In this paper, we propose a novel framework which aims to improve context-aware listwise ranking performance by optimizing online evaluation metrics. The ranking problem is formalized as a Markov Decision Process (MDP) and solved with the reinforcement learning paradigm. To avoid the great cost to online systems during the training of the ranking model, we construct a virtual environment with millions of historical click logs to simulate the behavior of real users. Extensive experiments on both simulated and real datasets show that: 1) constructing a virtual environment can effectively leverage the large scale click logs and capture some important properties of real users. 2) the proposed framework can improve search ranking performance by a large margin.

KEYWORDS

Context, Heterogeneous Result, Ranking, Reinforcement Learning

ACM Reference Format:
Junqi Zhang[1], Jiaxin Mao[1], Yiqun Liu[1*], Ruizhe Zhang[1], Min Zhang[1], Shaoping Ma[1], Jun Xu[2], Qi Tian[3]. 2019. Context-Aware Ranking by Constructing a Virtual Environment for Reinforcement Learning. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3357384.3357945

1 INTRODUCTION

Web search engines usually rank results according to relevance scores in descending order. Assuming that users browse search results sequentially from top to bottom on search engine result pages (SERPs), ranking relevant results at the top positions will reduce users' efforts in locating useful information. To obtain relevance estimation for search results without manual annotations, users' click logs have been used as implicit relevance feedback. While the click signal is vulnerable to some behavior biases, such as the position bias [13, 14] and attention bias [28], click models [6, 11, 12, 15, 21] have been proposed to derive an unbiased relevance estimation from the large scale click logs. To our best knowledge, most of these existing methods estimate the relevance of each query-document pair in a pointwise manner. Although this framework gains much success in improving search ranking performance, it faces two important challenges with the development of recent search techniques.

First, the sequential browsing hypothesis no longer holds in today's heterogeneous search scenarios where apart from the ten blue hyperlinks, images, videos, news, and even applications are aggregated in a unified result list. Different types of vertical results have distinct presentation styles. Multimedia contents are also incorporated in the snippets of vertical results [40]. Previous studies have revealed that users' attention may be attracted by non-textual information items [28]. Meanwhile, many works have observed the vertical bias [7, 34] that users may examine vertical results first, which alters users' behavior and leads to a non-sequential examination sequence. Therefore, ranking search results in descending order of relevance may be suboptimal in heterogeneous search scenarios.

Second, the interaction effect between search results is usually ignored by existing methods. The selection and ranking of the search result depends not only on its own relevance but also on the context. For example, a user searching for the official website of a university may also be interested in its wikipedia page. Putting them nearby may facilitate the user's search process. Besides, when a user seeks for a video, putting a number of similar video sites at the top may be considered as redundant, although they are all relevant. Thus, beside relevance, the dependency relationship between search results also needs to be considered in a listwise optimization process. Although diversified ranking or novelty-based ranking [5, 38, 41] techniques take the interaction effect into consideration, they usually treat the effect as another ranking factor and try to incorporate it into existing ranking methods. A better solution should be an end-to-end process which directly optimize the final ranking list considering both the relevance and interaction effect.

*Corresponding author

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3357945

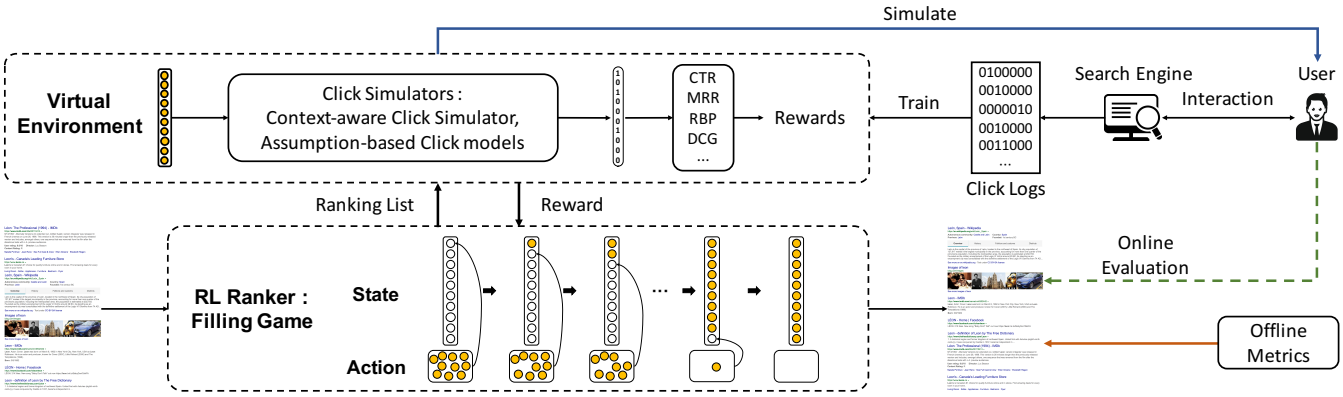

Figure 1: The RL ranking framework. The yellow circles represent search results. The RL Ranker acts as the agent while the virtual environment adopts click simulators to return rewards.

Regarding above issues, a context-aware listwise ranking process is needed to take the heterogeneity and dependency relationship of search results into account. Wei et al. has formulated the ranking problem as a Markov Decision Process (MDP) in [36]. However, they aim to improve the learning to rank (LTR) algorithms by leveraging the evaluation measures calculated at all the ranking positions. In this work, we adopt the MDP formulation and propose a novel reinforcement learning [18, 19, 32] (RL) framework to improve the listwise ranking performance by optimizing online evaluation metrics. As shown in the lower part of Figure 1, at each step t, the ranking agent will choose a result from the candidates (*actions*) to place at position t in the ranking list. By regarding the already placed results as the current *state* and setting a proper immediate *reward*, the MDP ranking model can take both the dependency relationship between search results and the listwise evaluation metrics into consideration. A unified representation for the heterogeneous verticals is also derived from the multimodal contents displayed on SERPs. By training the RL model with the unified representation, the heterogeneity of search results is considered in the ranking process.

While the RL framework is suitable for modeling the context-aware ranking problem, training the model can be a great challenge. Previous works [36] train the RL model based on relevance labels in an offline manner, which requires much human efforts in generating the labels. However, when utilizing users' click feedback to train reinforcement learning in online search systems, there are some serious problems. First, to learn an optimal policy, the RL model usually requires a large number of trials of the agent and rewards from the environment. Failed trials will seriously affect search engine users' experience. Second, it takes time to collect enough trials and rewards from users. For example, a reranked list for a query may not be searched for a long time and we can not know whether this reranking strategy is good or not. To avoid harming the performance of online systems and train the RL model more efficiently, we propose to construct a virtual environment with historical click logs to simulate the behavior of real users (as shown in Figure 1). The RL model is then trained in the virtual environment to optimize multiple online evaluation metrics. Previous studies have shown that online evaluation metrics have stronger correlations with user satisfaction in heterogeneous search environment [8, 9]. Building a virtual environment not only alleviates the training cost of RL models, but may also help to optimize user satisfaction more directly. The advantages of the proposed framework can be summarized as three-folds:

- The heterogeneity and dependency relationship of search results are taken into consideration for ranking. The context-aware listwise ranking optimization helps to model users' browsing process in heterogeneous search scenarios and considers the interaction effects between search results.
- A virtual environment is constructed with historical click logs for costless and efficient training of RL models in search systems.
- By optimizing online evaluation metrics in the virtual environment, users' feedbacks are exploited to improve ranking performance. The RL ranking framework can be further extended to online training for a wide range of optimization tasks for commercial search engines.

The rest of the paper is organized as follows. We describe related work in Section 2. Section 3 formally introduces the RL ranking framework. The experiment settings and results are presented in Section 4 and 5. Finally, we conclude this paper and discuss future work in Section 6.

2 RELATED WORK

2.1 Heterogeneous Search Scenarios

Modern search engines aggregate different information items such as images, news and videos to a unified result page. The aggregated vertical results contain multimedia contents and vary a lot in presentation styles. Different from ten blue hyperlinks, vertical results alter users' browsing behavior on SERPs to a large extend. Users may be attracted by multimedia contents or vertical results and the examining process is no longer from the top to the bottom. As shown in [20, 40], users can judge the relevance of search results directly from the contents displayed on SERPs as they provide valuable signals for users' information needs. Users will not necessarily click the url to browse the landing page in some cases, such as the knowledge graph results and the direct answer results. It is essential to take the heterogeneity of search results into account for ranking algorithm design and user behavior modeling.

2.2 Diversified Ranking and Novelty Retrieval

Diversified ranking aims to provide search results that can cover a wide range of users' information needs. One important goal is to rank documents according to the relevance as well as novelty scores [5, 16, 37, 41]. Based on the assumption that users browse search results in a top-down manner, typical methods regard diversified ranking as a sequential result selection process. The interaction effect of search results is considered to maximize the diversity of the ranking list. However, most of the methods treat relevance and novelty as two independent factors, and rank search results according to a linear combination of relevance and novelty features [16, 37]. To explicitly model the dynamic utility that the user perceives from the preceding results, Xia et al. proposes to model diversified ranking as a Markov Decision Process (MDP). However, the diversity measures of search results are calculated based on relevance annotations, which is not adaptable for large-scale online training.

2.3 Click Models

Click logs are valuable sources of users' implicit relevance feedback. A lot of click models have been proposed to utilize click logs for user behavior simulation and relevance estimation, such as User Browsing Model (UBM) [12], Dynamic Bayesian Networks model (DBN) [6], and Dependent Click Model (DCM)[15]. As the heterogeneous search results account for more and more in search engines, some works [7, 34] find that users may examine vertical results first, which alters users' behavior and leads to a non-sequential examination process. Some advanced click models such as UBM-Layout [11] and Mobile Click Model (MCM) [21] are proposed to take the vertical bias into consideration in heterogeneous search scenarios.

As deep neural networks have shown astonishing advantages in broad applications, some works have tried to model user behavior with neural networks. The Neural Click Model (NCM) [2] learns to represent concepts that are useful for modeling user behavior with Recurrent Neural Networks. The Click Sequence Model (CSM) [3] is implemented as an encoder-decoder to predict the order in which a user will interact with search results. Different from these models, we propose a hierarchical neural architecture to model the dependency relationship of search results. The result feature is also derived from the multimodal contents displayed on SERPs to leverage the heterogeneity of search results.

2.4 Reinforcement Learning

Reinforcement learning [18, 19] is originated from psychology and neuroscience understanding of how humans learn to take actions in the environment. It can be formulated as a Markov Decision Process (MDP). Recently, combining reinforcement learning and deep learning techniques has shown great success in broad applications [23], such as games [30, 33], robotics [39], computer vision [4] and natural language processing [31] tasks.

Recently, many works have applied reinforcement learning methods for ranking tasks [25, 29, 35, 36]. Oosterhuis and De Rijke adopts the RL approach to deal with complex ranking settings, which learns both the user preferred document order and display position order for result presentation. The page presentation optimization of SERPs has also been recast as a reinforcement learning problem in [35]. Wei et al. formulates the ranking problem as a Markov Decision Process (MDP) to optimize the evaluation measure calculated at all positions. Some work has adapted RL for complex online system optimization [29]. They construct a virtual retail platform to train the ranking policy offline. Assuming that the environment is static, the inverse reinforcement learning (IRL) methods [24] can learn a reward function from the data and train the policy according to the reward function.

Different from these works [25, 35, 36], we aim to improve the ranking performance of search engines by optimizing the listwise online metrics through the deep reinforcement learning approach. To solve the offline training problem, we construct a static virtual environment to simulate search engine users and provide rewards, which is similar with that in [29].

Table 1: Notations of the RL ranking framework.

Notation	Description
q, x_i	Query, the ith search result in the session
c_i, p_i	Click label of x_i, Predicted click probability of x_i
f_q, f_{ses}	Query feature, Session feature
$f_{res,i}, f_{click,i}$	Feature of x_i, Feature of c_i
$f_{state,t}$	State feature at time step t
$s, \mathcal{S}$	State, State space
$a, \mathcal{A}$	Action, Action space
$\mathcal{T}$	Transition function
γ	Discount factor
$\mathcal{R}, r$	Reward function, Step reward
R	Discounted cumulative reward
π, π^*	Policy, Optimal policy

3 RL RANKING FRAMEWORK

Problem Formulation. In search systems, the users act as the environment, while the ranking strategy of the system is the agent. As shown in Figure 1, the RL ranking framework consists of two major components. The first one is the click simulator, which is learned from historical click logs and serves as the virtual environment. The second one is the RL Ranker. The RL ranking framework is model-based [32] as the RL Ranker interacts with the learned virtual environment, which can be formulated as follows:

Let $S = \{q, x_1, x_2, \ldots, x_N\}$ denote a query session, where q is the query and x_j is the jth search result in the original ranking list of the search engine, N is the number of results in the session. The RL Ranker reranks the result list to $\hat{S} = \{q, x_{i_1}, x_{i_2}, \ldots, x_{i_N}\}$, where $\{i_1, i_2, \ldots, i_N\}$ is the permutation of $\{1, 2, \ldots, N\}$. The click simulator samples clicks $C = \{c_{i_1}, c_{i_2}, \ldots, c_{i_N}\}$ on the reranked list $\hat{S}$. The reward at each time step is given by $reward = \mathcal{R}(C)$. When training the RL Ranker, the parameters in the click simulator are fixed and the parameters in the RL Ranker are updated by the reward signals. The notations used in this paper are summarized in Table 1.

Result Representation. As heterogeneous search results account for more and more in search engines, the contents displayed on SERPs provide valuable signals for relevance judgement [20, 40]. To provide a unified representation of heterogeneous search results for the click simulator and RL Ranker, we train a feature extractor based on the approach proposed in Tree-based Deep Neural Network (TreeNN) [20]. We adopt their approach by embedding the visual

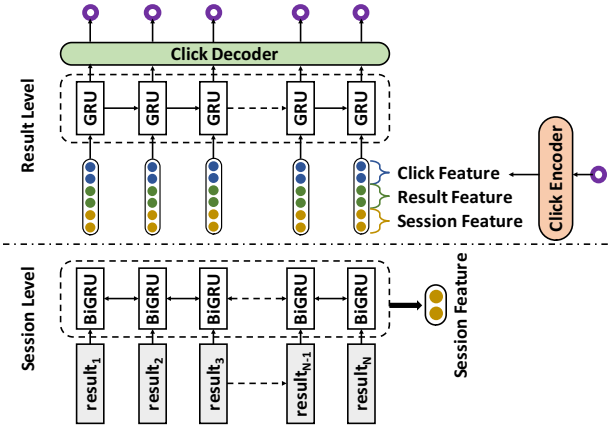

Figure 2: Context-aware Click Simulator. The hollow purple circles represent user clicks.

and textual information into the HTML tree, which can better leverage the structure information to represent the heterogeneous results. The SRR[1] dataset is adopted to train the feature extractor to obtain a good representation for search results. Limited by the space, we do not describe the feature extractor network in details. The implementation of the network is published online [2].

3.1 Environment: Context-aware Click Simulator

Click models can be adopted to simulate click feedbacks. However, there are two main limitations. First, the contents of search results can not be incorporated in click models, which are valuable signals for ranking especially in heterogeneous search scenarios. Second, click models can only deal with query-document pairs seen before. However, search results for a query are always changing rapidly. The click model trained at one time can not work effectively as time goes by.

To solve the two issues, in this work, we propose a novel neural Context-aware Click Simulator (CCS) as the virtual environment for the RL Ranker. CCS learns from the historical click logs to simulate the real users, which predicts click probability not only according to click signals, but also the multimodal contents of search results. As users' click behavior is affected not only by the relevance of a single search result but also its context, the dependency relationship of search results can not be ignored. To model the context, a hierarchical structure is proposed. The framework of CCS is shown in Figure 2.

3.1.1 Session Level. The lower level of CCS is the session based module. It is implemented as a bi-direction GRU [10] framework. The concatenation of the last hidden states of the two directions in BiGRU is adopted as the session feature. From top to bottom and bottom to top, the session feature captures the global context information of the result list.

$$h_t = [u_{1:t}, u_{N:N-t+1}] \tag{1}$$

$$h_{t+1} = \text{BiGRU}(h_t, f_{res,t+1}) \tag{2}$$

$$f_{ses} = h_N = [u_{1:N}, u_{N:1}] \tag{3}$$

[1] http://www.thuir.cn/data-srr/
[2] https://github.com/IR-Ranker/RLRank

where N is the number of search results in a query session, $u_{1:i}$ or $u_{N:j}$ is the hidden state of BiGRU at time step t in two directions, $[\cdot]$ denotes concatenation of different features, $f_{res,t}$ is the tth search result in the session, f_{ses} is the session feature. The initial hidden state h_0 for BiGRU is the query feature f_q.

3.1.2 Result Level. The higher result level module is designed to predict click probability of each search result sequentially based on the session context. At each time step, three different kinds of features are aggregated as the input to a single-direction GRU. The first one is the session feature encoding the global context. The second one is the respective result feature at each step. A common assumption in click models is that previous clicks will influence the click behavior in the following browsing process of the user. Following this assumption, we take the click behavior at previous step as the third input feature.

The click behavior can be *click*, *skip* or *unknown* (for the initial step), which is embedded into vectorial representation through the *click encoder* (Equation 4). At each step of GRU, the hidden state is projected into a click probability score between 0 and 1 through the *click decoder* (Equation 9). The prediction of the result level module can be formulated as follows:

$$f_{click,t} = \text{embedding}(c_{t-1}) \tag{4}$$

$$\hat{f}_{res,t} = W_{res} f_{res,t} + b_{res} \tag{5}$$

$$\hat{f}_{ses} = W_{ses} f_{ses} + b_{ses} \tag{6}$$

$$f_t = [f_{click,t},\ \hat{f}_{res,t},\ \hat{f}_{ses}] \tag{7}$$

$$h_t = \text{GRU}(h_{t-1}, f_t) \tag{8}$$

$$p_t = \text{sigmoid}(W_{click} h_t + b_{click}) \tag{9}$$

where $c_{t-1} \in \{click, skip, unknown\}$ denotes the click behavior in step $t-1$, $W_{res}, W_{ses}, W_{click}$ and $b_{res}, b_{ses}, b_{click}$ are weights and biases for each kind of features respectively, p_t is the final predicted click probability of the tth search result in the ranking list, $p_t \in [0, 1]$.

The loss function is *CrossEntropy*, which is defined as:

$$\mathcal{L}(p, c; \theta) = \frac{1}{M} \sum_{j=1}^{M} \sum_{i=1}^{N} (-c_{j,i} \log p_{j,i} - (1 - c_{j,i}) \log (1 - p_{j,i})) + \lambda ||\theta||_2^2 \tag{10}$$

where M is the number of training sessions, N is the number of results in the query session, $c_{j,i}$ and $p_{j,i}$ denote the click label and predicted click probability of the ith result in the jth training sample, θ includes all the parameters in the neural network, λ denotes the $L2$ regularizer coefficient.

When training the Context-aware Click Simulator, the feature representations of search results are fixed and we focus on the optimization of the click simulator.

3.2 Agent: RL Ranker

To model the dependency and context of search results, we recast the context-aware listwise ranking problem as a sequential filling game (as shown in Figure 1). At the initial time step, the result list is empty, while all the results are in the candidate set. Based on the query information, one search result is chosen to the first position. Regarding the results already placed in the ranking list, we compare between the candidate results and choose the most proper one to the next position.

The ranking problem can be formulated as a Markov Decision Process (MDP), which is described by a tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma\}$. $\mathcal{S}$ denotes the state space, and $\mathcal{A}$ denotes the action space. $\mathcal{T}: \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the transition function $\mathcal{T}(s_{t+1}|s_t, a_t)$ to generate the next state s_{t+1} from current state s_t and action a_t. $\mathcal{R}: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, while the reward at tth time step $r_t = \mathcal{R}(s_t, a_t)$. $\gamma \in [0, 1]$ is the discounting factor for future rewards. Formally, the MDP components are specified with the following definitions:

State s is the global information of the search results already ranked in the list. At the initial time, s_0 is the query information.

Action a is a search result that the RL Ranker chooses to the next position in the ranking list.

Transition $\mathcal{T}$ changes the state of the ranking list, adding one search result at each time step.

Reward $\mathcal{R}$ is the reward function which can be the online users' feedbacks. In this work, the reward is given by the virtual environment based on simulated clicks.

In this paper, the MDP problem is solved with the policy gradient algorithm of REINFORCE [32]. At each time step t, the policy $\pi(a_t|s_t)$ defines the probability of sampling action $a_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$. The sampling strategy can be randomized according to the probability or just choose the action with the highest one. These two strategies are denoted as "RandomSample" and "MaxSample" respectively. The aim of RL is to learn an optimal policy π^* by maximize the expected cumulative reward $R_t = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k r_{t+k}]$.

3.2.1 Policy Network. The structure of the policy network is shown in Figure 3. The deep neural network architecture can learn policies from high-dimensional raw input data in complex RL environment. Specifically, the state feature is extracted by a single-direction GRU. The search results which have been placed at proper positions are input to the GRU sequentially according to the ranking order. The last hidden state of GRU is adopted as the state feature for RL. The state feature is then concatenated with the candidate result features and input to a multi-layer perceptron to assess the probability of each candidate to be chosen as the action at this time step. The action is then sampled according to different strategies (RandomSample or MaxSample) as the next result to be filled into the ranking list. At time step t, there are t and $N - t$ results in the ranking list and candidate set respectively, where N is the number of results in the query session. The policy network can be formulated as follows:

$$h_k = \text{GRU}(h_{k-1}, f_{res,i_k}), k = 1, 2, \ldots, t \tag{11}$$

$$f_{state,t} = h_t \tag{12}$$

$$f_{i_j} = \text{MLP}([f_{state,t}, f_{res,i_j}]), j = t+1, t+2, \ldots, N \tag{13}$$

$$\pi_t = \text{softmax}(f_{i_{t+1}}, f_{i_{t+2}}, \ldots, f_{i_N}) \tag{14}$$

$$a_t = \text{sample}(\pi_t) \tag{15}$$

where $\{i_1, i_2, \ldots, i_N\}$ is a permutation of $\{1, 2, \ldots, N\}$, MLP denotes the multi-layer perceptron, the sampling strategy can be "RandomSample" or "MaxSample" as described above, a_t is the chosen action in time step t, h_0 is the query feature f_q.

3.2.2 Reward Design. The virtual environment for RL is implemented as the click simulator. Given the reranked list of search results, the click simulator predicts the click feedbacks. A lot of online evaluation metrics can be designed as the rewards based on the click sequence. In this work, we try two kinds of typical online evaluation metrics CTR (Click Through Rate) and MRR(Mean Reciprocal Rank), and two kinds of typical offline metrics DCG (Discounted Cumulative Gain) [27] and RBP (Rank-Biased Precision) [22] for reward design. All these metrics are modified to utilize the sequential clicks as reward signals, and be capable to give reward at each RL step. The clicks at top K positions are accumulated as the reward for the Kth time step as we find that it works better in training than just giving the click signal at the Kth position. The definition of the four modified rewards are as follows ("AC" means "accumulated"). p is set to 0.8 in RBP-AC.

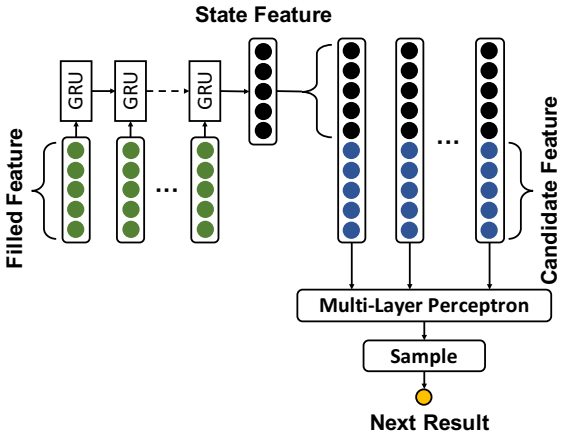

Figure 3: The policy network for RL Ranker.

$$\textbf{CTR-AC:} \quad \text{CTR-AC@K} = \frac{1}{K}\sum_{i=1}^{K} c_i \tag{16}$$

$$\textbf{MRR-AC:} \quad \text{MRR-AC@K} = \sum_{i=1}^{K} \frac{c_i}{i} / \sum_{i=1}^{K} \frac{1}{i} \tag{17}$$

$$\textbf{RBP-AC:} \quad \text{RBP-AC@K} = \sum_{i=1}^{K} p^{i-1} c_i / \sum_{i=1}^{K} p^{i-1} \tag{18}$$

$$\textbf{DCG-AC:} \quad \text{DCG-AC@K} = \sum_{i=1}^{K} \frac{c_i}{log_2(i+1)} / \sum_{i=1}^{K} log_2(i+1) \tag{19}$$

3.2.3 RL Training. A trajectory $\tau = s_0, a_0, s_1, a_1, \ldots, s_N, a_N$ is sampled according to the policy π in each episode. The episode terminates when the ranking list is filled with results. The objective of the training process is to maximum $J(\theta)$.

$$J(\theta) = \mathbb{E}_{\pi_\theta}[R(\tau)] \tag{20}$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta log \pi_\theta(\tau) R(\tau)] \tag{21}$$

Equation 21 can be approximated by a Monte Carlo estimator [19]:

$$\nabla_\theta J(\theta) \propto \frac{1}{M} \sum_{i=1}^{M} \sum_{j=1}^{N} \nabla_\theta log \pi_\theta(s_{i,j}, a_{i,j}) R_{i,j} \tag{22}$$

where θ is the parameters of the policy network, M is the number of samples, N is the number of results in a session.

It is known that training a RL model from scratch is not effective. As the original ranking list has already achieved a generally satisfactory performance, we add the original ranking list to the training samples. This may introduce bias into the Monte Carlo estimator and deriving an approximately unbiased policy gradient estimator is left for future work. Specifically, M is set to 21 in the experiments, including one sample of the original ranking list and the others reranked by π_θ.

When training the RL Ranker, the feature representation of search results and the click simulator are all fixed as a stable environment. In this manner, we can make sure the training of the RL Ranker will not change the virtual environment or result representations. This is important because if we do not fix the virtual environment, the performance improvement may be caused by the click simulator rather than the RL Ranker.

4 EXPERIMENT SETUP

4.1 Datasets and Training Process

The datasets used in our experiments are sampled from SRR published in [40]. Each query has corresponding top 10 search results. Each search result is represented by the parse tree and annotated with a 4-graded relevance label. The parse tree of the search result incorporates the images and texts into the HTML codes. The two real datasets "Real 2017" and "Real 2018" are constructed in September, 2017 and December, 2018 respectively with click logs recorded by a popular commercial search engine in China. The queries of Real 2018 are subset of Real 2017. Search results of the same query in the two datasets are not exactly the same as time has changed. There are 3.24 same results for each query in the two datasets on average. The number of same results is from 0 to 8.

For the simulation experiments in Section 5.2, we sample simulated click logs according to the rule in Equation 28. The queries and results in Simulation dataset are the same with Real 2017. We split the 1,971 queries and corresponding results as well as simulated click logs in Simulation dataset into two parts at a ratio of 4:1 (1,569 : 402), which are denoted as "Seen Set" and "Unseen Set".

When training the click simulators, the click logs are split into training, validation and testing sets at a ratio of 3 : 1 : 1. The training process of the whole RL ranking framework is shown in Figure 5. For simulation study, X is the Seen Set and Y is either the Seen Set or Unseen Set. For real data study, X is the Real 2017 dataset and Y is either Real 2017 or Real 2018 dataset.

Table 2: Statistics of real and simulated datasets.

Dataset		Real Data		Simulation Data	
		Real 2017	Real 2018	Seen	Unseen
#Queries		1,971	1,239	1,569	402
#Logs	Training	1,498,227	-	940,854	241,745
	Validation	499,671	-	315,131	80,270
	Testing	500,470	533,254	313,015	79,985

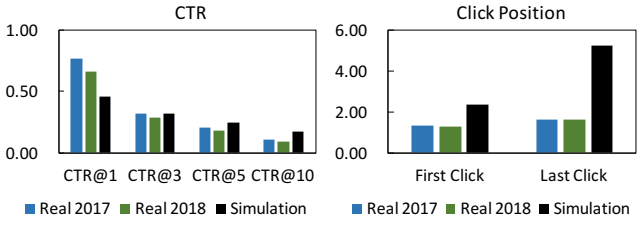

Figure 4: Statistics of click logs in different datasets.

4.2 Experiment Settings

The dimension of result features is 1,000. For the Context-aware Click Simulator (CCS), the dimensions of hidden states in the session level and result level are all set to 100. The dimension of hidden state of the policy network in the RL Ranker is 1,000. The discounting

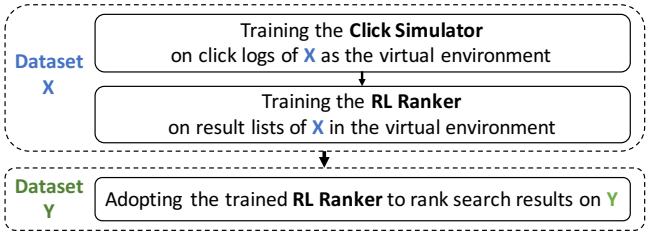

Figure 5: Training and utilization of the RL ranking framework.

factor γ is set to 0.9. The five click models are all trained for 1,000 epochs. The RL Ranker under each reward function are trained for 500 epochs as the loss begins to converge. When training the RL Ranker, we adopt the "RandomSample" strategy for π_θ, while "MaxSample" is adopted for evaluation as we fully trust the learned policy.

5 EXPERIMENTS

The proposed framework aims to improve the ranking performance by using a click simulator to construct a virtual environment for the RL Ranker. To demonstrate the effectiveness of this framework, in this section, we conduct a series of experiments to address the following research questions:

- **RQ1:** Can click simulators capture the important properties of real search engine users?
- **RQ2:** Can RL Ranker optimize the online evaluation metrics effectively in the virtual environment?
- **RQ3:** How does the whole RL ranking framework perform on real click logs?

Online and offline metrics measure users' search experience from different perspectives. They both significantly correlate with actual user satisfaction [9]. Click logs are utilized to train the RL ranking framework for online metric optimization. To the best of our knowledge, there is no ranking algorithm directly optimizing online evaluation metrics. Thus, we evaluate how the RL ranking framework can improve the online metrics compared with the original ranking lists by a simulated user in Section 5.2. To demonstrate the effectiveness of the RL ranking framework on offline metrics, we compare it with some state-of-the-art ranking methods in Section 5.3 (note that the RL ranking framework is not directly optimized for the offline metrics based on relevance judgements).

5.1 Evaluating Click Simulators (RQ1)

To show that the Context-aware Click Simulator (CCS) can accurately predict users' click behavior, especially in the heterogeneous search environment, we compare it with five existing click models. The five conventional click models can be divided into two categories. DCM [15], UBM [12] and DBN [6] do not take the vertical bias into account, while UBM-Layout [11] and MCM [21] incorporate the influence of heterogeneous verticals on users' click behavior into the assumptions.

5.1.1 Training and Evaluation. All these models are trained on the training logs of Real 2017 and tested on the testing logs of Real 2017 and Real 2018 respectively. The models are evaluated in terms of Perplexity and LogLikelihood. A lower Perplexity and a higher LogLikelihood indicate the click simulator can predict users' clicks

more accurately. The definitions of the two metrics are as follows:

$$\text{Perplexity}_i = 2^{-\frac{1}{M}\sum_{j=1}^{M}(c_{ji}\log p_{ji}+(1-c_{ji})\log(1-p_{ji}))} \tag{23}$$

$$\text{LL} = \frac{1}{MN}\sum_{j=1}^{M}\sum_{i=1}^{N}(c_{ji}\log p_{ji} + (1-c_{ji})\log(1-p_{ji})) \tag{24}$$

where c_{ji} and p_{ji} refer to the binary click and predicted click probability of the ith result in the jth query session respectively. N is the number of results in a session. M is the number of sessions in the dataset.

5.1.2 Analysis. The performance of different click models and the CCS is shown in Table 3. From the results, we can see that the five click models and CCS perform well on Real 2017 dataset, where the query-document pairs in the testing set are all seen in the training set. This indicates that the click simulators can accurately model the user click by exploiting large-scale click logs. However, on Real 2018 dataset, where some new search results that are not included in training set occur, the performance of conventional click models declines a lot. As the proposed CCS takes the content of search results into account, it has a stronger generalization ability and performs the best on Real 2018 dataset.

Because results in search engines are changing rapidly and conventional click models can not work effectively on new search results, we will adopt the Context-aware Click Simulator as the virtual environment in the following experiments.

Table 3: Performance of different click simulators. The perplexity score is the average of Perplexity$_i$ over all positions (top 10).

Model		Real 2017		Real 2018	
		Perplexity	LL	Perplexity	LL
Basic	DCM	1.1489	-0.1301	1.2666	-0.2230
	UBM	1.1435	-0.1253	1.2164	-0.1779
	DBN	1.1401	-0.1225	1.2012	-0.1653
Vertical-Aware	UBM-Layout	1.1433	-0.1252	1.2246	-0.1856
	MCM	**1.1401**	**-0.1224**	1.2001	-0.1647
CCS		1.1409	-0.1230	**1.1994**	**-0.1625**

5.2 Simulation Study (RQ2)

To verify that the proposed framework can optimize a range of online evaluation metrics, we conduct a simulation study with simulated users, which is similar to those conducted in [1, 35]. A simulation rule is designed according to some prior knowledge and assumptions to generate simulated click behavior based on the relevance and vertical type annotations. In this way, we create a simulated user that can: 1) generate click logs on original result lists for training; 2) produce *ground truth* user behavior on the reranked list to test the effectiveness of the RL ranker.

5.2.1 Simulation Rule. The simulation rule in [1] is extended to the heterogeneous search scenario by incorporating the vertical bias. The extended assumption is that, users click a search result x belonging to query q ($c_q^x = 1$) only when it is both observed ($o_q^x = 1$) and perceived as relevant ($r_q^x = 1$), and at the same time, the vertical type is preferred to click ($v_q^x = 1$) by users. o_q^x, r_q^x, v_q^x and c_q^x are all Bernoulli random variables. We sample these variables by the following formulations:

o_q^x: The position bias ρ estimated through eye-tracking experiments in [17] is adopted to sample o_q^x. $\nu \in [0, +\infty)$ controls the severity of position biases, which is set to 2.0 in our experiment. i denotes the ranked position of the search result.

$$P(o_q^x = 1) = \rho_i^\nu \tag{25}$$

r_q^x: The relevance annotations are utilized to sample r_q^x. $y \in [0, 3]$ is the 4-level relevance label for result x and y_{max} is 3 in our dataset. Parameter ϵ introduces click noise into the click decision process. A search result not very relevant to the query also has a small but positive probability to be clicked. ϵ is set to 0.2 in our experiment.

$$P(r_q^x = 1) = \epsilon + (1-\epsilon)\frac{2^y - 1}{2^{y_{max}} - 1} \tag{26}$$

v_q^x: The vertical bias ω is estimated on a large number of real click logs. There are 19 different result types in SRR. The average click through rate (CTR) for each result type is computed as $\omega = AverageCTR(v)$, where v denotes the vertical type of the search result. The statistical results show that the variances of CTR of results belonging to the same type across different queries are small (most of them are less than 0.01). This indicates that the vertical bias holds for a wide range of search intends. $\mu \in [0, +\infty)$ controls the severity of the vertical bias, which is set to 0.5 in our experiment.

$$P(v_q^x = 1) = \omega_v^\mu \tag{27}$$

c_q^x: The final clicks are sampled according to the multiplication of the probabilities of o_q^x, r_q^x, v_q^x. To control the severity of position bias and vertical bias, we need to adjust the hyper-parameters ν and μ, which can cause $P(o_q^x = 1)$ and $P(v_q^x = 1)$ to be really small. Thus, $P(o_q^x = 1)$ and $P(v_q^x = 1)$ are linearly mapped to $[0.3, 1]$ in our experiment. The click probability of result x is given by:

$$P(c_q^x = 1) = P(o_q^x = 1)P(r_q^x = 1)P(v_q^x = 1) \tag{28}$$

5.2.2 Simulation Data. The rule designed in Equation 28 is regarded as a simulated user, who clicks on the result lists and produces click logs. 1,000 click sessions are produced by the simulated user for each query, resulting in 1,971,000 sessions. As shown in Figure 4, the simulated click logs are close to the two real log datasets in terms of CTR@3,5,10. This shows the designed rule has actually captured some important properties of real users. As there is strong first-click bias for real users, CTR@1 for real log data is higher than the simulated data. The first click and last click position are also a little higher than the real click logs. The properties of the simulated logs can be adjusted by the severity parameters.

5.2.3 Training of RL Ranker. Figure 6 shows the performance curves during training. The clicks for training and validation are all sampled by CCS. The RL Ranker has close performance on the Seen Set and Unseen Set. We adopt the "MaxSample" strategy for validation and "RandomSample" strategy for training. There is stable improvement during training and the performance begins to converge after 500 epochs. The performance of the validation curves are higher than the training curve, which shows that the learned policy is reliable by "MaxSample".

5.2.4 Evaluation of RL Ranker. As the search engine needs to handle both the repeated queries and new queries that has never appeared in the search logs, we evaluate the performance of the proposed framework on both the previously *seen* queries (Seen Set) and *unseen* queries (Unseen Set). The trained RL Ranker is adopted to rerank result lists of the testing queries. The rule-based simulated user described in Section 5.2.1 interacts with the reranked list to

Table 4: Performance of RL Ranker with different rewards on Seen Set. The improvement of reranked lists under different reward functions is significant on all evaluation metrics compared to the original ranking ("Label") with p-value< 0.001.

	MRR	CTR				cRBP			cDCG			Click Position	
		@1	@3	@5	@10	@3	@5	@10	@3	@5	@10	First Click	Last CLick
Label	0.5884	0.4548	0.3187	0.2461	0.1702	0.1647	0.1902	0.2113	0.7445	0.8570	1.0081	2.3766	5.2122
CCS	0.5890	0.4562	0.3187	0.2462	0.1702	0.1647	0.1903	0.2114	0.7450	0.8580	1.0088	2.3731	5.2105
CTR-AC	**0.6104**	0.4742	**0.3454**	**0.2647**	**0.1758**	**0.1776**	**0.2044**	**0.2243**	**0.7995**	**0.9177**	**1.0574**	**2.2279**	**5.0033**
MRR-AC	0.6086	0.4738	0.3413	0.2610	0.1749	0.1759	0.2021	0.2223	0.7928	0.9085	1.0510	2.2473	5.0552
RBP-AC	0.6100	**0.4743**	0.3440	0.2641	0.1756	0.1771	0.2040	0.2239	0.7975	0.9161	1.0561	2.2320	5.0130
DCG-AC	0.6086	0.4729	0.3423	0.2624	0.1751	0.1763	0.2029	0.2229	0.7940	0.9113	1.0525	2.2405	5.0247

Table 5: Performance of RL Ranker with different rewards on Unseen Set. The improvement of reranked lists under different reward functions is significant on all evaluation metrics compared to the original ranking ("Label") with p-value< 0.001.

	MRR	CTR				cRBP			cDCG			Click Position	
		@1	@3	@5	@10	@3	@5	@10	@3	@5	@10	First Click	Last CLick
Label	0.5908	0.4572	0.3224	0.2494	0.1728	0.1663	0.1924	0.2140	0.7516	0.8665	1.0206	2.3697	5.2513
CCS	0.5907	0.4567	0.3232	0.2500	0.1733	0.1667	0.1928	0.2145	0.7530	0.8682	1.0229	2.3723	5.2583
CTR-AC	0.6156	0.4797	**0.3536**	**0.2689**	**0.1788**	**0.1816**	**0.2081**	**0.2283**	**0.8165**	**0.9331**	1.0758	2.2064	**5.0333**
MRR-AC	0.6142	**0.4805**	0.3485	0.2652	0.1776	0.1795	0.2057	0.2261	0.8085	0.9239	1.0685	2.2265	5.0777
RBP-AC	**0.6163**	0.4801	0.3532	0.2687	**0.1788**	**0.1816**	0.2080	**0.2283**	0.8164	**0.9331**	**1.0760**	**2.2037**	5.0349
DCG-AC	0.6118	0.4754	0.3481	0.2654	0.1777	0.1791	0.2055	0.2260	0.8058	0.9219	1.0667	2.2258	5.0641

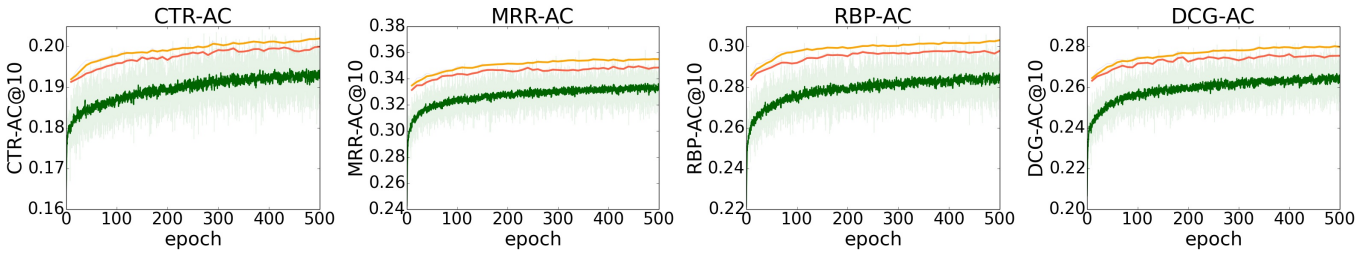

Figure 6: The performance curves during the training process. The line in green denotes the rewards of training samples. The lines in orange and red refer to the validation performance on the Seen Set and Unseen Set, respectively.

simulate online testing. Besides the intuitive CTR and the first/last click position, we use the following online evaluation metrics, MRR, cRBP, and cDCG (RBP and DCG computed with click labels):

$$\text{cRBP@K} = (1-p)\sum_{i=1}^{K} p^{i-1}c_i,\ \text{cDCG@K} = \sum_{i=1}^{K}\frac{c_i}{log_2(i+1)} \quad (29)$$

where p is set to 0.8 in cRBP, c_i is the click label at position i. As cRBP@1 = $(1-p)$ CTR@1 and cDCG@1 = CTR@1, we do not show cRBP@1 and cDCG@1 for simplicity.

Table 4 and 5 show the performance of the RL Ranker under different rewards on the Seen and Unseen datasets. The first and second row (denoted as "Label" and "CCS") show the performance of the original ranking lists when the clicks are sampled by the rule-based simulated user (the labels of the Simulation dataset) and CCS respectively.

From the results we can find that, first, the RL Ranker improves the online evaluation metrics by a large margin over the original rankings (Line "Label" in Table 4 and Table 5) under different reward functions. The average position of the first and last click also moves forward. As the rule-based simulated user can imitate real users' behavior to some extent and is independent of the proposed model and the training process, the improvements suggest that the proposed framework can potentially improve the ranking performance effectively.

Second, all the online metrics can be optimized through the training process. This shows the potential capability of the RL Ranker to optimize a wide range of objectives for commercial search engines as long as a proper reward is designed. It is also an interesting finding that the specific reward may not be the best choice to optimize the corresponding evaluation metric. For example, the reward "CTR-AC" performs the best to optimize all the MRR, CTR, cRBP and cDCG evaluation metrics on the Seen Set in Table 4. We will leave finding the best reward function for a specific online evaluation metric for future work.

Third, the RL Ranker performs well on both the seen and unseen results. This reveals the strong generalization ability of the RL ranking framework in the changing situations, which is important for real-world online systems.

5.3 Real Data Study (RQ3)

As we have demonstrated that the proposed framework works well in the simulation study to optimize online evaluation metrics, we would further investigate how it performs on real click logs recorded by the search engine, and whether the RL Ranker trained in virtual environment actually helps to improve the ranking performance. We evaluate the RL ranking framework on datasets constructed at different time to show the generalizability of the framework, which is an important concern for online search systems.

Table 6: Offline performance of different models on Real 2017 dataset. The improvement of the RL Ranker compared to the best baseline models (DCM and UBM-Layout) is not significant with p-value< 0.05.

	NDCG @1	NDCG @3	NDCG @5	NDCG @10	MRR	RBP
DLA	0.6822	0.7282	0.7821	0.8942	0.9435	0.6224
JRE	0.7553	0.7631	0.8056	0.9090	0.9302	0.6264
DCM	**0.7987**	0.7926	**0.8318**	**0.9219**	**0.9491**	0.6345
UBM	0.7900	0.7885	0.8259	0.9196	0.9462	0.6334
DBN	0.7792	0.7819	0.8195	0.9161	0.9472	0.6314
UBM-Layout	0.7947	**0.7934**	0.8312	0.9217	0.9466	**0.6348**
MCM	0.7628	0.7725	0.8151	0.9128	0.9463	0.6304
CTR-AC	0.7694	0.7725	0.8188	0.9141	0.9462	0.6300
MRR-AC	0.7961	0.7879	0.8247	0.9192	0.9489	0.6315
RBP-AC	0.7830	0.7756	0.8178	0.9160	0.9451	0.6301
DCG-AC	**0.8024**	**0.7950**	**0.8341**	**0.9226**	**0.9498**	**0.6340**

Table 7: Offline performance of different models on Real 2018 dataset. * and ** denote the significant improvement compared to the best baseline model (UBM-Layout) with p-value< 0.05 and < 0.01 respectively.

	NDCG @1	NDCG @3	NDCG @5	NDCG @10	MRR	RBP
DLA	0.5422	0.6211	0.6930	0.8416	0.7577	0.5316
JRE	0.6508	0.6851	0.7320	0.8673	0.8038	0.5444
DCM	0.7084	0.7011	0.7374	0.8766	0.8364	0.5460
UBM	0.7090	0.7006	0.7379	0.8764	0.8402	0.5457
DBN	0.6897	0.6903	0.7292	0.8715	0.8283	0.5433
UBM-Layout	**0.7173**	**0.7066**	**0.7426**	**0.8791**	**0.8419**	**0.5472**
MCM	0.6923	0.6937	0.7323	0.8729	0.8314	0.5440
CTR-AC	0.7328	0.7468	0.7825	0.8941	0.8538	0.5596
MRR-AC	0.7389	0.7655	0.7945	0.8996	0.8597	0.5626
RBP-AC	**0.7459***	**0.7667****	**0.7986****	**0.9018****	**0.8606***	**0.5634****
DCG-AC	0.7307	0.7591	0.7926	0.8975	0.8553	0.5613

5.3.1 Baselines. Different from the simulation experiments, there is no rule-based simulated user to judge the reranked result lists. We need to evaluate the ranking performance by the relevance-based offline metrics. The RL ranking framework is compared with the click models, the unbiased learning to rank approach Dual Learning Algorithm (DLA) [1], and the state-of-the-art method on SRR dataset Joint Relevance Estimation (JRE) [40] model. All the baselines are adapted to utilize click logs for training.

- **Click Models**: The estimated query-document relevance scores (default to 0.5 for unseen pairs) are utilized to rank results.
- **JRE**: As UBM-Layout performs well on both Real 2017 and Real 2018 datasets, the estimated query-document relevance scores on Real 2017 by UBM-Layout is adopted as weak labels to train JRE.
- **DLA**: DLA learns an automatic unbiased learning-to-rank model from biased click logs. We extract 33 features based on the texts of the landing pages corresponding to each search result as in [1, 26].

5.3.2 Analysis. The experiment results on the Real 2017 and Real 2018 datasets are shown in Table 6 and 7. From the results, we can see that:

First, click models perform the best among the baselines which directly utilize the debiased query-document relevance for ranking. Although unifying the learning of propensity models and ranking models, it is hard for the unbiased learning to rank method DLA to beat the click models. JRE achieves close performance with UBM-Layout which provides the weak relevance labels.

Second, although the RL Ranker is trained to optimize the online evaluation metrics, the learned policy also performs well on the offline metrics. The RL Ranker trained with DCG-AC achieves comparable and slightly better performance than the best click models DCM and UBM-Layout on Real 2017. However, as the RL Ranker is not optimized for the offline metrics, the improvements are not so significant. On Real 2018 dataset, the RL Ranker under all reward functions achieves much better performance than the baselines. This shows the RL ranking framework can effectively leverage the implicit relevance signals in click logs, even without making any explicit assumptions on user behavior. The virtual environment captures important properties of real users and serves as a reliable reward provider and performance judger for the RL Ranker. The versatility of the RL ranking framework is potentially important in the heterogeneous search scenarios.

Third, the RL ranking framework has stronger generalizability than other methods based on click logs. As shown in Table 7, the RL Ranker performs much better than DLA, JRE and click models on Real 2018 dataset. The RL ranking framework trained on historical data performs well in the new environment, while a sharp decline in performance is spotted for the baselines. Generalizability is an important concern for search engines. A large amount of Web documents or multimedia contents come to the online system at every moment. It is infeasible to train the model constantly for new contents, which is required by the click models. The proposed RL ranking framework is more adaptable for practical search engines.

6 CONCLUSION

Ranking the heterogeneous search results with complex dependency relationship is an emerging but critical problem for modern search engines. In this work, we formulate the context-aware ranking as a listwise optimization problem and try to solve it in a reinforcement learning paradigm. For offline training, a virtual environment is constructed to simulate real search engine users. By extensive experiments on both simulation data and real data, we demonstrate the effectiveness of the click simulator in modeling user behavior and the strong capability of RL Ranker in optimizing online as well as offline evaluation metrics. We also show that the RL ranking framework has a strong generalizability over time, which is valuable for the search engines dealing with ever-changing information on the Web. The framework also shows potential in optimizing a wide range of ranking objectives for online systems, which can be exploited to enhance users' search experience in a more direct manner.

Finally, we discuss some issues about applying the proposed framework to online systems. The online training can be built upon the policy learned offline with the log data and the virtual environment, which is more efficient and brings little harm to the online system. Besides, different components of the RL ranking framework can utilize different features in practical applications. For example, the click simulator can be trained offline with highly computational deep neural features to achieve better performance, while the RL Ranker, which will be used to rank the results online, can utilize some more cost-efficient handcrafted features.

ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China (2018YFC0831700) and Natural Science Foundation of China (Grant No. 61622208, 61732008, 61532011).

REFERENCES

[1] Qingyao Ai, Keping Bi, Luo Cheng, Jiafeng Guo, and W. Bruce Croft. 2018. Unbiased Learning to Rank with Unbiased Propensity Estimation. (2018).

[2] Alexey Borisov, Ilya Markov, Maarten De Rijke, and Pavel Serdyukov. 2016. A Neural Click Model for Web Search. In *International Conference on World Wide Web*.

[3] Alexey Borisov, Martijn Wardenaar, Ilya Markov, and Maarten De Rijke. 2018. A Click Sequence Model for Web Search. (2018).

[4] Juan C. Caicedo and Svetlana Lazebnik. 2015. Active Object Localization with Deep Reinforcement Learning. In *IEEE International Conference on Computer Vision*.

[5] Jaime Carbonell and Jade Goldstein. 1999. *The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries*.

[6] Olivier Chapelle and Ya Zhang. 2009. A dynamic bayesian network click model for web search ranking. In *Proceedings of the 18th international conference on World wide web*. ACM, 1–10.

[7] Danqi Chen, Weizhu Chen, Haixun Wang, Zheng Chen, and Qiang Yang. 2012. Beyond ten blue links: enabling user click modeling in federated web search. In *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM, 463–472.

[8] Ye Chen, Yiqun Liu, Ke Zhou, Meng Wang, Min Zhang, and Shaoping Ma. 2015. Does Vertical Bring more Satisfaction?:Predicting Search Satisfaction in a Heterogeneous Environment. (2015), 1581–1590.

[9] Ye Chen, Ke Zhou, Yiqun Liu, Min Zhang, and Shaoping Ma. 2017. Meta-evaluation of Online and Offline Web Search Evaluation Metrics. In *International Acm Sigir Conference*.

[10] Kyunghyun Cho, Bart Van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *Computer Science* (2014).

[11] Aleksandr Chuklin, Pavel Serdyukov, and Maarten De Rijke. 2013. Using Intent Information to Model User Behavior in Diversified Search. In *European Conference on Information Retrieval*.

[12] Georges E Dupret and Benjamin Piwowarski. 2008. A user browsing model to predict search engine click data from past observations.. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 331–338.

[13] Laura Granka, Matthew Feusner, and Lori Lorigo. 2008. Eyetracking in Online Search. *Bericht, Google and University of California and Cornell University* (2008).

[14] Zhiwei Guan and Edward Cutrell. 2007. An eye tracking study of the effect of target rank on web search. In *Sigchi Conference on Human Factors in Computing Systems*. 417–420.

[15] Fan Guo, Chao Liu, and Yi Min Wang. 2009. Efficient multiple-click models in web search. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*. ACM, 124–131.

[16] Shengbo Guo and Scott Sanner. 2010. Probabilistic latent maximal marginal relevance. (2010).

[17] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately Interpreting Clickthrough Data as Implicit Feedback. 4–11.

[18] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. 1998. An Introduction to Reinforcement Learning. (1998).

[19] Levente Kocsis and Csaba Szepesvari. 2006. Bandit Based Monte-Carlo Planning. *Lecture Notes in Computer Science* 4212 (2006), 282–293.

[20] Yiqun Liu, Junqi Zhang, Jiaxin Mao, Min Zhang, Shaoping Ma, Qi Tian, Yanxiong Lu, and Leyu Lin. 2019. Search Result Reranking with Visual and Structure Information Sources. *ACM Trans. Inf. Syst.* 37, 3, Article 38 (June 2019), 38 pages. https://doi.org/10.1145/3329188

[21] Jiaxin Mao, Cheng Luo, Min Zhang, and Shaoping Ma. 2018. Constructing Click Models for Mobile Search. In *International Acm Sigir Conference*.

[22] Alistair Moffat and Justin Zobel. 2008. Rank-biased precision for measurement of retrieval. *Acm Transactions on Information Systems* 27, 1 (2008), 1–27.

[23] Seyed Sajad Mousavi, Michael Schukat, and Enda Howley. 2018. Deep Reinforcement Learning: An Overview. (2018).

[24] Andrew Y. Ng and Stuart Russell. 2000. Algorithms for Inverse Reinforcement Learning. In *International Conference on Machine Learning*.

[25] Harrie Oosterhuis and Maarten De Rijke. 2018. Ranking for Relevance and Display Preferences in Complex Presentation Layouts. (2018), 845–854.

[26] Tao Qin, Tie Yan Liu, Jun Xu, and Hang Li. 2010. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval* 13, 4 (2010), 346–374.

[27] Kalervo Rvelin, Kek, and Jaana Inen. 2002. Cumulated gain-based evaluation of IR techniques. *Acm Transactions on Information Systems* 20, 4 (2002), 422–446.

[28] Chengyao Shen and Qi Zhao. 2014. Webpage saliency. In *European Conference on Computer Vision*. Springer, 33–46.

[29] Jing-Cheng Shi, Yang Yu, Qing Da, Shi-Yong Chen, and An-Xiang Zeng. 2018. Virtual-taobao: virtualizing real-world online retail environment for reinforcement learning. (2018), 845–854.

[30] Huang A. Maddison C. J. Guez A. Silver, D. and D Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587, 484–489.

[31] Pei Hao Su, Milica Gasic, Nikola Mrksic, Lina M. Rojas Barahona, Stefan Ultes, David Vandyke, Tsung Hsien Wen, and Steve Young. 2016. On-line Active Reward Learning for Policy Optimisation in Spoken Dialogue Systems. (2016).

[32] Richard S Sutton and Andrew G Barto. 2011. Reinforcement learning: An introduction. (2011).

[33] Mnih Volodymyr, Kavukcuoglu Koray, Silver David, Andrei A Rusu, Veness Joel, Marc G Bellemare, Graves Alex, Riedmiller Martin, Andreas K Fidjeland, and Ostrovski Georg. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[34] Chao Wang, Yiqun Liu, Min Zhang, Shaoping Ma, Meihong Zheng, Jing Qian, and Kuo Zhang. 2013. Incorporating vertical results into search click models. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 503–512.

[35] Yue Wang, Dawei Yin, Luo Jie, Pengyuan Wang, Makoto Yamada, Yi Chang, and Qiaozhu Mei. 2018. Optimizing Whole-Page Presentation for Web Search. *ACM Transactions on the Web (TWEB)* 12, 3 (2018), 19.

[36] Zeng Wei, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2017. Reinforcement Learning to Rank with Markov Decision Process. In *ACM SIGIR 2017*.

[37] Long Xia, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2015. Learning Maximal Marginal Relevance Model via Directly Optimizing Diversity Evaluation Measures. (2015).

[38] Long Xia, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2017. Adapting Markov Decision Process for Search Result Diversification. In *International Acm Sigir Conference on Research & Development in Information Retrieval*.

[39] Fangyi Zhang, Juergen Leitner, Michael Milford, Ben Upcroft, and Peter Corke. 2015. Towards Vision-Based Deep Reinforcement Learning for Robotic Motion Control. *Computer Science* (2015).

[40] Junqi Zhang, Yiqun Liu, Shaoping Ma, and Qi Tian. 2018. Relevance Estimation with Multiple Information Sources on Search Engine Result Pages. In *Proceedings of the 2018 ACM on Conference on Information and Knowledge Management*. ACM, 10.

[41] Yadong Zhu, Yanyan Lan, Jiafeng Guo, Xueqi Cheng, and Shuzi Niu. 2014. Learning for Search Result Diversification. In *International Acm Sigir Conference on Research & Development in Information Retrieval*.