# Knowledge-guided Deep Reinforcement Learning for Interactive Recommendation

Xiaocong Chen[1], Chaoran Huang[1], Lina Yao[1], Xianzhi Wang[2], Wei liu[1], Wenjie Zhang[1]
[1]School of Computer Science and Engineering, University of New South Wales, Australia
[2]School of Computer Science, University of Technology Sydney, Australia
[1]{xiaocong.chen, chaoran.huang, lina.yao, wei.liu, wenjie.zhang}@unsw.edu.au
[2]xianzhi.wang@uts.edu.au

***Abstract*—Interactive recommendation aims to learn from dynamic interactions between items and users to achieve responsiveness and accuracy. Reinforcement learning is inherently advantageous for coping with dynamic environments and thus has attracted increasing attention in interactive recommendation research. Inspired by knowledge-aware recommendation, we proposed Knowledge-Guided deep Reinforcement learning (KGRL) to harness the advantages of both reinforcement learning and knowledge graphs for interactive recommendation. This model is implemented upon the actor-critic network framework. It maintains a local knowledge network to guide decision-making and employs the attention mechanism to capture long-term semantics between items. We have conducted comprehensive experiments in a simulated online environment with six public real-world datasets and demonstrated the superiority of our model over several state-of-the-art methods.**

***Index Terms*—Recommender System, Reinforcement Learning, Deep Neural Network**

## I. Introduction

Recommendation systems have been widely used by industry giants such as Amazon, YouTube, and Netflix to identify relevant, personalized content from large information spaces. Modern recommendation systems are facing severe pressures for coping with emerging new users, ever-changing pools of recommendation candidates, and context-dependent interests [**?**]. In contrast, traditional recommendation methods focus on modeling user's consistent preferences and may not reflect high dynamics in user interest and environments. In such situations, interactive recommendation rises as an effective solution that incorporates dynamic recommendation processes to improve the recommendation performance. An interactive recommendation system would recommend items to an individual user and then receive the feedback to adjust its policies during the iterations [1]. Many studies model interaction recommendation as a Multi-Armed Bandit (MAB) problem [2]–[4]. Such methods generally assume a user's preference is consistent during the recommendation and focus on the trade-off between immediate and future rewards. Therefore, they face challenges for handling environments with dynamically changing user preference or interest. Reinforcement learning (RL) is a promising approach to interactive recommendation. Considerable efforts have shown the outstanding performance of RL methods in recommendation systems [5]–[7], thanks to its ability to learn from user's instant feedback. Given its potential to handle dynamic interactions, RL has been widely regarded to be a possible better solution for interactive recommendation. However, most existing RL techniques in interactive recommendation focus on the usefulness instead of performance. For example, Liu et al. [8] employ the RL to increase the recommendation diversity, but not focus on the efficacy. The primary reason is the agent only provides limited and partial information, making it difficult to control the decision-making process properly. Besides, interactive recommendation systems usually contain a large number of discrete candidate actions, leading to high time complexity and low accuracy of RL-based techniques. Moreover, all the Deep Q-Networks (DQN)-based work [**?**], [6], [9], [10] gets struggled with a large number of discrete actions because DQN contains a maximise operation, which considers all actions. When the size of action increasing, the maximise operation will come to extremely slow, or even get stuck. The policy gradient based methods will get stuck in this case as well because it may converge in the local maximum instead of the global maximum.

Recently, knowledge-aware recommendation systems have become popular as the knowledge graph can transfer the relation to contextual information and boost the recommendation performance [**?**], [11]. Inspired by the above research, we propose a framework named knowledge-guided deep reinforcement learning (KGRL) for interactive recommendation. We use the actor-critic framework to formulate the whole process. Specially, we design a knowledge graph to represent relations between items so that the recommendation system can make recommendations based on the relations, and the critic network employs the knowledge graph as the guideline to improve the performance.

The critic network is used to evaluate the performance of the actor so as to let the actor optimize itself to the correct direction. Besides, we apply graph convolutional network (GCN) inside the critic network capture the high-level structural information inside the knowledge graph and Deep Deterministic Policy Gradients (DDPG) to train our model. In summary, we make the following contributions in this work:

- We proposed a novel model KGRL where the knowledge graph is introduced into the reinforcement learning pro-

TABLE I: Main notations

| Symbols | Meaning |
|---|---|
| $\mathcal{U}$ | Set of users |
| $\mathcal{I}$ | Set of items |
| $\mathcal{R}$ | Set of relations |
| $\mathcal{E}$ | Set of entities |
| $\lvert\cdot\rvert$ | Number of unique elements in $\cdot$ |
| $\mathcal{S}_{u,t}$ | User $u$'s recent actions before timestamp $t$ |
| $\mathcal{G} = (\mathcal{E}, \mathcal{R})$ | Constructed Knowledge Graph |
| $\mathbb{E}$ | Item embedding |
| $W$ | parameter matrices |
| $S_t$ | state space at timestamp $t$ |
| $a_t$ | action space at timestamp $t$ |
| $d$ | dimension of the latent space |

cess to help the agent make decisions.

- To improve the efficiency, we maintain a local knowledge network which is based on the knowledge graph, to fasten the process while keeping the performance;
- Comprehensive experiments in the simulated online environment with six real-world datasets prove the performance of our propose approach.

## II. Problem Definition

An interactive recommendation system features incorporating user's feedback dynamically during the training process. Given a set of users $\mathcal{U} = \{u, u_1, u_2, u_3, ...\}$ and a set of items $\mathcal{I} = \{i, i_1, i_2, i_3, ...\}$, the system first recommends item $i_1$ to user $u_1$ and then gets a feedback $x$. The system aims to incorporate feedback to improve future recommendations. To this end, it needs to figure out an optimal policy $\pi^*$ regarding which item to recommend to the user to achieve positive feedback. We can formulate the problem as a Markov Decision Process (MDP) by treating the user as the environment and the system as the agent. We define the key components of the MDP as follows (Table I summarizes the main notations used in this paper):

- State: A state $S_t$ is determined by the recent $l$ items in which the user was interested before time $t$.
- Action: Action $a_t$ represents a user's dynamic preference at time $t$ as predicted by the agent.
- Reward: Once the agent chooses a suitable action $a_t$ based on the current state $S_t$ at time $t$, the user will receive the item recommended by the agent. The user's feedback on the recommended item (i.e., clicking the item, ignoring it) accounts for the reward $r(S_t, a_t)$, which will be considered to improve the recommendation policy $\pi$.
- Discount Factor $\gamma$: The discount factor $\gamma \in [0, 1]$ is used to balance between the future and immediate rewards—the agent will fully focus on the immediate reward when $\gamma = 0$ and take into account all the (immediate and future) rewards otherwise.

## III. Methodology

Our approach involves two steps: knowledge preparation and deep reinforcement recommendation

### A. Knowledge Preparation

We construct the knowledge graph based on entity-relation-entity tuples $\{(i, r, j)|i, j \in \mathcal{E}, r \in \mathcal{R}\}$. For example, the tuple *(The Elements of Style, book.author, William Strunk Jr.)* means that *William Strunk Jr* authored the book *The Elements of Style*. We consider every item (e.g., *The Elements of Style*) as an entity in the knowledge graph $\mathcal{G}$ and transform the knowledge graph to represent user's preference more precisely [12]. Given a user $u \in \mathcal{U}$ and an item $q \in \mathcal{I}$, suppose $\mathcal{D}(i)$ is the set of items that has direct relationship with item $i$ and $r_{ij}$ denotes the relation between items $i$ and $j$. We calculate the user-specific relation scores as follows:

$$f_u^{r_{ij}} = g(u, r_{ij}) \text{ where } g : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$$

where $g$ is a scoring function (e.g., inner product) to compute the score between user and relation; $d$ is the dimension of user representation and relation representation; $u \in \mathbb{R}^d, r_{ij} \in \mathbb{R}^d$; $f_u^{r_{ij}}$ measures the importance of $r_{ij}$ to user $u$.

Let $\mathcal{D}(i)$ be the set of candidates to recommend, we normalize the user-specific relation scores as follows:

$$\overline{f_u^{r_{ij}}} = \frac{f_u^{r_{id}}}{\sum_{d\in\mathcal{D}(i)} f_u^{r_{id}}} \in [0, 1]$$

Inspired by [13], we transform the knowledge graph into a user-specific graph $A_u$, which is an adjacency matrix of $\mathbb{R}^{|\mathcal{I}|\times|\mathcal{I}|}$. In this matrix, each position $(i, j)$ corresponds to a score $\overline{f_u^{r_{ij}}}$, and a higher score indicates a stronger relation between two items $i$ and $j$.

### B. Deep Reinforced Recommendation

We develop our recommendation model (Figure 1) based on the Actor-Critic reinforcement learning framework [14], where the actor generates actions, the critic evaluates actions, and the actor network updates the policy based on the suggestion made by the critic.

*1) Actor Network $\phi$:* Given a current state $S_t$, the actor network employs a neural network to infer an optimal policy $\pi^*$ to work out an action $a_t$. Given $S_t$, which consists of user's recent interests (shown in Figure 1, we first obtain vector representation of user's recent interest via embedding. Suppose we have a set of user'srecently interested items before time $t$, $\mathcal{S}_{u,t} = \{\mathcal{S}_u^1, \mathcal{S}_u^2, ..., \mathcal{S}_u^l\}$. The actor network takes an input sequence $\mathcal{S}_{u,t}$ and the corresponding feedback sequence $\{\mathcal{F}_u^1, \mathcal{F}_u^2, ..., \mathcal{F}_u^l\}$ to deliver an output sequence $\{\mathcal{S}_u^2, \mathcal{S}_u^3, ..., \mathcal{S}_u^{l+1}\}$. Given an original item embedding matrix $\mathcal{M} \in \mathbb{R}^{|\mathcal{I}|\times d}$ ($d$ is the dimension of the latent space), we apply positional embedding [15], $P \in \mathbb{R}^{n\times l}$, to preserve the order of user's previously interested items, which updates the item embedding into the following:

$$\mathbb{E} = \begin{bmatrix} \mathcal{M}_1 + P_1 \\ \mathcal{M}_2 + P_2 \\ \cdots \\ \mathcal{M}_l + P_l \end{bmatrix}$$

We then fed this embedding into a self-attention layer to reduce impurity in the embedding [16]. The layer uses the

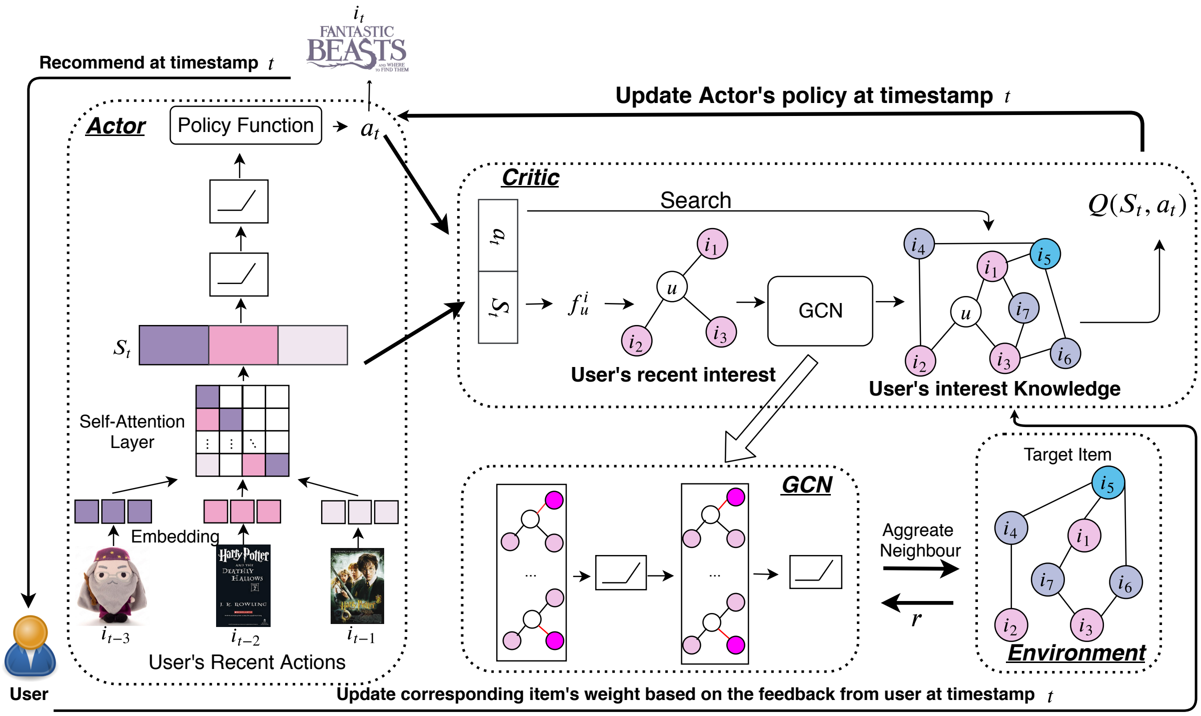


Fig. 1: The KGRL structure. The left and right parts describe the actor network and the critic network, respectively, at time $t$. The model takes user's recent actions (regarding toys, books, and movies) as the input and recommends new items as the output. Those actions will be represented as the latent factor in this model. The user, in turn, provides feedback for the model to update user's interest knowledge's weights.

scaled dot-product attention [17], which is originally defined as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

where $Q, K, V$ denotes queries, keys, and values, respectively; $\sqrt{d_k}$ is the scaling factor to regulate the value range of $QK^T$. After applying the embedding $\mathbb{E}$ as the input, the attention turns into the following:

$$\text{Attention}(\mathbb{E}W^Q, \mathbb{E}W^K, \mathbb{E}W^V)$$

where $\mathbb{E}W^Q, \mathbb{E}W^K, \mathbb{E}W^V \in \mathbb{R}^{d \times d}$. We fed this embedding into two fully connected layers, which use ReLU and tanh as the activation functions, respectively as described in [15]. The output of the attention layer is the state $S_t$ at time $t$.

*2) Critic Network $\psi$:* We design the critic network to estimate the Q-value function $Q(S_t, a_t)$ to evaluate actor's policy. The critic network takes state representation $S_t$ and action representation $a_t$ as the input (shown in Figure 1). We design a local knowledge network within the critic network to capture the high-order structural proximity among the items in the knowledge graph using graph convolutional network (GCN). Specifically, given a user-specific graph $g_i^u$ generated from the current state $S_t$, we feed it into a two-layer GCN that applies the following layer-wide propagation rule:

$$H^{l+1} = \sigma(D^{-\frac{1}{2}} \hat{A}_u D^{-\frac{1}{2}} H^l W^l) \tag{1}$$

where $H^{l+1}$ is the representation of entities at layer $l+1$; $A_u$ is the input matrix that aggregates the neighbour's entities; $\hat{A}_u$ is set to $A_u + I$, where the $I$ is an identity matrix used to avoid negligence of the old representation via self-connection; $D_u$ is the diagonal degree matrix for $\hat{A}_u$ where $D_u^{ii} = \sum_j \hat{A}_u^{ij}$ (the symmetric normalization was applied to keep the representation $H^l$ stable, as denoted by $D^{-\frac{1}{2}} \hat{A}_u D^{-\frac{1}{2}}$); $W^l$ is the weight matrix for layer $l$; and $\sigma(\cdot)$ denotes the non-linear activation function.

Recent research shows the feasibility of searching in graphs processed by GCN [18]. Since GCN capture's all the structural information in the knowledge graph, it will not affect the search results. In this study, we assume an unweighted graph where a user is equally interested in every item. Then, we start searching with the actor predicted action $a_t$ (i.e., predicted item $i_p$) to the real target $i_t$, based on the user's personalized interest knowledge (i.e., trained graph with all parameters $\theta_{kg}$). Finally, we calculate the Q value by estimating the reward $r$ based on the distance between the predicted item and the target item:

$$r = \frac{100}{\sqrt{\text{Distance}(i_p, i_t) + \epsilon}} * W_{pt}$$

where $W_{pt}$ is the sum of weight of the shortest path from $i_p$ to $i_t$; $\epsilon$ is the parameter to avoid the denominator becoming 0. We calculate the distance using the Dijkstra's algorithm with MinHeap.

### C. Complexity Analysis

We analyze the time and space complexity of the critic network, especially the search part, in this section. We consider a vector composition (i.e., the combination of the state vector and action vector) and assume the transmission time as a constant $c$. Given a user interested in $I_u$ items, we consider the worst case—a complete graph and each item $i$ having $M$ nearest non-duplicate neighbours. Thus, we get a graph with $I_u + I_uM$ nodes (exclude the centralised user node) and $(I_u + I_uM)(I_u + I_uM - 1)/2$ edges. We then calculate the time and space complexity as $\mathcal{O}((|I_u + I_uM)^2 + |I_u + I_uM| \log |I_u + I_uM|) \sim \mathcal{O}(|I_u + I_uM|^2)$ and $\mathcal{O}(2|I_u + I_uM|) \sim \mathcal{O}(|I_u + I_uM|)$. In comparison, if we feed the environment knowledge graph to the critic network directly, the time and space complexity would be $\mathcal{O}(|I + IM|^2)$ and $\mathcal{O}(|I + IM|)$. Apparently, the local knowledge network significantly improves the performance and saves the memory space in our model ($I_u \ll I$). Moreover, the local knowledge network is easier to converge as it has fewer nodes than the environment knowledge graph.

### D. Training Strategy

Training the actor-critic network requires train two parts of the neural network simultaneously. We apply the Deep Deterministic Policy Gradient (DDPG) (Algorithm 1) to train our model [19], where we train the critic by minimising a loss function:

$$l(\theta_\psi) = \frac{1}{N}\sum_{j=1}^{N}((r + \gamma\xi) - \psi_{\theta_\psi}(S_t, a_t))^2$$
$$\text{where } \xi = \psi_{\theta'_\psi}(S_{t+1}, \phi_{\theta'_\phi}(S_{t+1}))$$

where $\theta_\psi$ is the parameter in critic; $\theta_\phi$ is the parameter in actor; $N$ is the size of mini-batch from the replay buffer; $\psi_{\theta'_\psi}$ and $\phi_{\theta'_\phi}$ are the target critic and target actor network, respectively.

Algorithm 2 describes the training of the local knowledge network, where we define the same loss function for all users for the local knowledge network :

$$l_k = \sum_{u \in \mathcal{U}} (\sum_{i:y_{ui}} J(y_{ui}, \hat{y}_{ui}))$$

where $J$ is the cross-entropy; $y_{ui}$ is a piece-wise function to reflect the interest/action (defined below):

$$y_{ui} = \begin{cases} 1 & \text{if u interested in i} \\ 0 & \text{otherwise} \end{cases}$$

## IV. Experiments

In this section, we report our experimental evaluation of our model in comparison with several state-of-the-art models using real-world datasets.

**Algorithm 1:** DDPG algorithm for our model

```
Initialize actor network $\phi$ with parameter $\theta_\phi$ and critic
  network $\psi$ with parameter $\theta_\psi$ randomly;
Initialize target network $\phi'$ and $\psi'$ with weight
  $\theta'_\phi \leftarrow \theta_\phi$, $\theta'_\psi \leftarrow \theta_\psi$ ;
Initialize the local knowledge network ;
Initialize Replay Buffer $\mathcal{B}$ ;
for $i = 0$ to $n$ do
    Receive the initial state $S_i$ ;
    for $t = 1$ to $T$ do
        Infer a action $a_t$ according to the $\phi(\cdot)$ ;
        Execute the action $a_t$ to receive a reward $r_t$
          and observe a new state $S_{t+1}$;
        $\mathcal{B}$.append($S_t, a_t, r_t, S_{t+1}$) ;
        Sample a random mini-batch of $\mathcal{N}$ transitions
          $(S_k, a_k, r_k, S_{k+1})$ from $\mathcal{B}$ ;
        Set $y_i = r_t + \gamma\xi$ ;
        Update Critic by minimise the loss $l(\theta_\psi)$ ;
        Update local knowledge net by Algorithm 2 ;
        Update the Actor policy by using the sampled
          policy gradient:
        $\nabla_{\theta_\phi}\phi =$
          $\frac{1}{N}\sum_{j=1}^{N} \nabla_a \psi(S_k, a)|_{a=\phi(S_k)} \nabla_{\theta_\phi}\phi(S_k)$ ;
        Update target network:
        $\theta'_\phi \leftarrow \tau\theta_\phi + (1 - \tau)\theta'_\phi$;
        $\theta'_\psi \leftarrow \tau\theta_\psi + (1 - \tau)\theta'_\psi$;
    end
end
```

**Algorithm 2:** Training the local knowledge network

```
input: The user specific graph $g_i^u$, environment KG $\mathcal{G}_e$
Initialize the parameters for GCN $\theta$ ;
Initialize the depth of graph $d_g$ ;
Initialize the reward storage $P$;
for $i$ in $g_i^u$ do
    Receive the reward $r$ from $\mathcal{G}_e$ ;
    P.append(r);
end
$r$ = min(P);
while GCN is not converge do
    if $d_g < r$ then
        aggregate next level's neighbours into $g_i^u$
          $d_g \leftarrow d_g + 1$;
    end
    Update the GCN and its corresponding $\theta$;
end
```

### A. Datasets

We conducted experiments on six public real-world datasets (Table II shows the statistics). All these datasets provide the necessary information for building the respective knowledge graphs.

TABLE II: Statistics of our experimental datasets

| Dataset | # of users | # of items | # of interactions |
|---|---|---|---|
| Amazon CD | 75,258 | 64,443 | 3,749,004 |
| Librarything | 73,882 | 337,561 | 979,053 |
| Book-Crossing | 278,858 | 271,379 | 1,149,780 |
| GoodReads | 808,749 | 1,561,465 | 225,394,930 |
| MovieLens-20M | 138,493 | 27,278 | 20,000,263 |
| Netflix | 480,189 | 17,770 | 100,498,277 |

**Book-Crossing**[1]: This dataset contains user's demographic information and book information from the Book-Crossing community. It is extremely sparse with a density of 0.0041%.

**MovieLens-20M**[2]: This is a well-known benchmark dataset that contains 20 million ratings from around 140 thousand users on the MovieLens website. It also provides movie tags, which can be used to build relations in the knowledge graph.

**Librarything**[3]: This dataset contains book review information collected from the librarything website.

**Amazon CDs and Vinyl**[4]: This is a highly sparse dataset that contains the product metadata, user reviews, ratings, and item relations, as part of the Amazon e-commence dataset.

**Netflix Prize**[5]: This dataset contains 100 million ratings from 480 thousand users and item information for yearly open competition to improve Netflix's recommendation performance.

**Goodreads**[6]: This dataset contains user's ratings and reviews to books on the Goodreads book review website.

### *B. Evaluation Metrics*

We evaluate the performance of recommendation using three metrics: precision, recall, and normalized Discounted Cumulative Gain (nDCG). All the metrics were calculated based on the top-10 recommendations to each user for each test case. To ease processing, we removed users who have fewer than ten interactions and scaled the ratings from all datasets to the range of $[0, 5]$. Only the items with a rating score higher than three were considered a relevant item.

### *C. Experimental Setup*

We evaluated our model in a simulated online environment built upon offline public datasets, using the algorithm proposed in [8] and the aforementioned reward function. This way, we avoided collecting private user information and expensive online training [20]. Specifically, the simulator generated feedback based on logistic matrix factorization (LMF) [21]. We randomly split each dataset into a training set (70%), a validation set (10%), and a testing set (20%) to conduct 10-fold cross-validation. The discount factor $\gamma$ was initialized to 0.99.

### *D. Compared Methods*

We compared out model with several competitive baselines:

**Policy-Guided Path Reasoning (PGPR) [11]**: A state-of-the-art knowledge-aware model that employs reinforcement learning for explainable recommendation.

**Tree-structured Policy Gradient Recommendation (TPGR) [22]**: A state-of-the-art model that uses reinforcement learning and binary tree for large-scale interactive recommendation.

**HLinearUCB [2]**: A contextual-bandit approach that learns extra hidden features for each arm to model the reward for interactive recommendation.

**Wolpertinger [5]**: A deep reinforcement learning framework that uses DDPG and KNN for recommendations in large discrete action spaces.

**DeepPage [6]**: A DDPG-based reinforcement learning model that learns a ranking vector for page-wise recommendation.

**DRN [7]**: A DQN-based recommendation method that employ deep Q learning to estimate Q-value for news recommendation.

**FactorUCB [23]**: A matrix factorization-based bandit algorithm for interactive recommendation .

**ICTRUCB [4]**: A MAB approach that uses a depend arm for online interactive collaborative filtering.

### *E. Results*

Table III shows our evaluation results of recommendation models. We observed that our model outperformed all the baselines in all metrics almost on all the datasets—it performed only slightly worse than TPGR on the Book-Crossing dataset. This may be attributed to the specifical design of TPGR to deal with large-scale datasets. None of the MAB-based methods (HLinearUCB, FactorUCB and ICTRUCB) performed well on those datasets because they all assume static user interest and may not give up-to-date recommendations We also observed that PGPR performed worse than DRN on the Amazon CD and Book-Crossing datasets—these sparse datasets might not provide sufficient relation for PGPR to infer the recommendation path. Finally, all the models achieved their best results on the MovieLens-20M dataset, given the rich information and dense relation in the dataset.

### *F. Ablation and Complexity Studies*

We conducted ablation studies to explore the impact of the attention mechanism and local knowledge network on the performance of our model on the above six datasets. We selectively choose MovieLens-20M and the Book-Crossing as the example because the Book-Crossing dataset is the most sparse one and the MovieLens-20M is the most dense one; they can show the capability of our model in the normal case and extreme case. Due to the exponential increase in time usage, we only show the first five level of neighbours. The results (Figure 2(a,d)) show that our model's performance dropped slightly (by 1% in precision, 2% in recall, and 1% in nDCG) without the attention mechanism while elevated

[1] http://www2.informatik.uni-freiburg.de/~cziegler/BX/

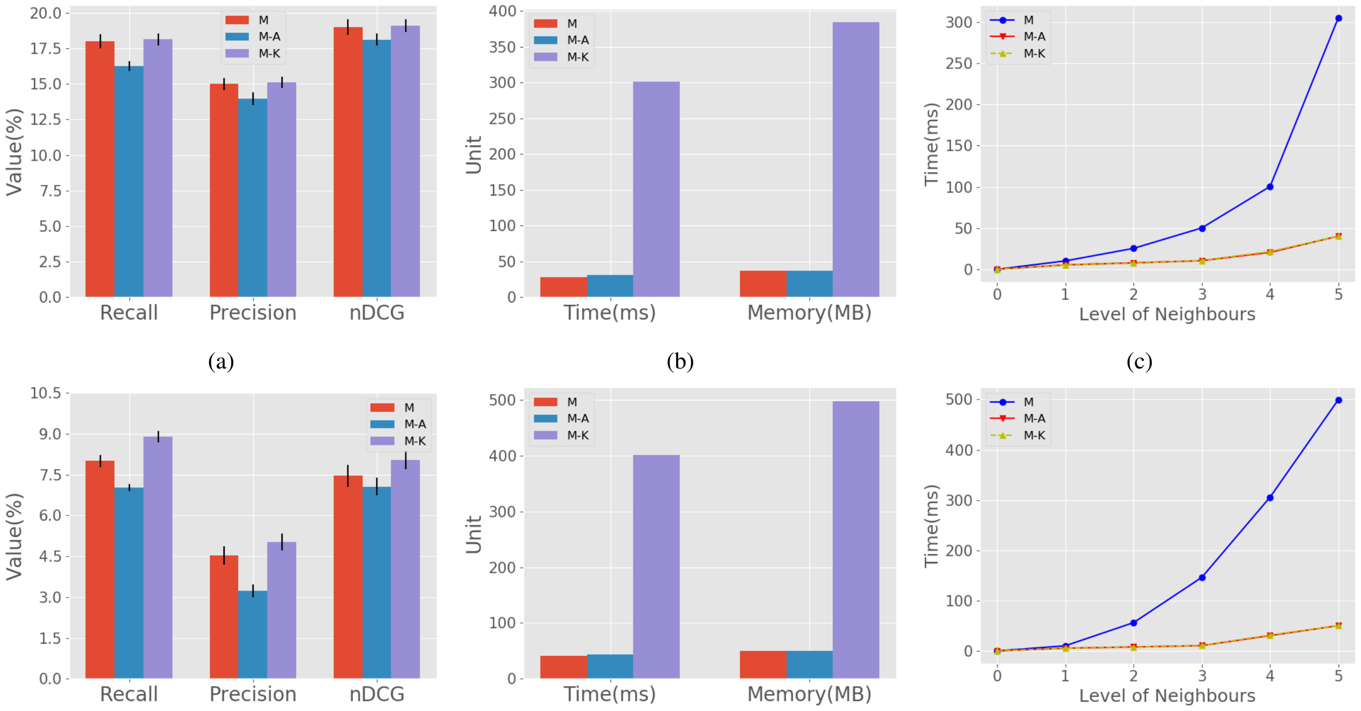
[2] https://grouplens.org/datasets/movielens/

[3] http://cseweb.ucsd.edu/~jmcauley/datasets.html#social_data

[4] http://jmcauley.ucsd.edu/data/amazon/

[5] https://www.kaggle.com/netflix-inc/netflix-prize-data

[6] http://cseweb.ucsd.edu/~jmcauley/datasets.html#goodreads



(a) (b) (c)

(d) (e) (f)

Fig. 2: Ablation and complexity studies on MovieLens-20M(a,b,c) and Book-Crossing(d,e,f): (a,d) Three models' performance in Recall, Precision, and nDCG; (b,e) Three models' time and memory consumption in conducting search for a target item located among fifth level neighbours; (c,f) Three models' time consumption along with an increasing level of the target item. *M* denotes our original model, *M-A* the model without the attention layer, meaning the item embedding will directly goes to state, and *M-K* the model deprived of the local knowledge network—in this case, the model uses GCN to learn the whole environment inside itself. The level of neighbours represents the geographical location indicative of the shortest distance. For example, first-level neighbours represent the items which have a distance of 1 to the current item $i$.

slightly without the local knowledge network because the model already contains all the information, including abundant relation between items to support the decision making.

We also used valgrind[7] to monitor the memory usage, which, on the other hand, reveals the huge advantages of using a local knowledge network in reducing both the time and space complexity (also see Figure 2(b)). We mentioned that in figure 2 (c,f), the model $M - K$ have an incredible increase in time consumption when the level goes over 2. One possible reason is that as the level goes higher, the graph comes more and more complex, which will affect the search critically.

## V. Related Work

Most existing work models interactive recommendation as a Multi-Armed Bandit (MAB) problem. And the primary solution lies in finding an Upper Confidence Bound (UCB). Li et al. [24] employ the first linear model to calculate the UCB for each arm. Since then, many researchers combine other techniques such as matrix factorization, to find the UCB [23]. For example, Wang et al. [4] proposed a new approach by choosing a dependent arm to calculate the UCB; Shen et al. [25], instead, use deep learning-based methods to solve MAB.

Recent studies have shown the effectiveness of reinforcement learning in modeling interactions-related recommendation processes, where the recommendation problems are usually formulated as Markov Decision Processes. One approach is based on Deep Q-learning (DQN) [26], which maximizes the Q-value from the predicted item and the target item. Zheng et al. [7] combine the DQN with the Dueling Bandit Gradient Decent (DBGD) [27] policy to recommend news. Another thread of methods is DDPG-based [19]. Such methods aim to let the agent learn a proper policy instead of using the Q-value. For example, Liu et al. [8] adopt DDPG to promote the diversity in interactive recommendation; Zhao et al. [6] use DDPG for page-wise recommendation. It is also worth mentioning that knowledge graphs can be useful for providing guidance in explainable recommendation [11]. Knowledge-aware recommendation systems heavily rely on the use of relation inference to generate paths for recommendations [28]. Wang et al. [29] show graph convolutional network can help learn neighbour representations and thus boost the recommen-

[7] http://www.valgrind.org/

TABLE III: The overall results of our model comparison with several state-of-arts models in different datasets. The result was reported by using the percentage and based on top-10 recommendation as mentioned before. The highlighted result in bold is the best result.

| Dataset | Amazon CD | | | Librarything | | |
|---|---|---|---|---|---|---|
| Measure (%) | Recall | Precision | nDCG | Recall | Precision | nDCG |
| Wolpertinger | 1.542 ± 0.192 | 1.521 ± 0.145 | 3.331 ± 0.201 | 3.441 ± 0.313 | 3.673 ± 0.221 | 4.115± 0.251 |
| HLinearUCB | 3.112 ± 0.331 | 2.647 ± 0.171 | 4.005 ± 0.341 | 8.102 ± 0.396 | 7.431 ± 0.204 | 8.157 ± 0.241 |
| FactorUCB | 3.531 ± 0.232 | 4.512 ± 0.242 | 6.012 ± 0.251 | 8.541 ± 0.241 | 8.162 ± 0.355 | 8.653 ± 0.351 |
| ICTRUCB | 4.124 ± 0.293 | 3.110 ± 0.395 | 5.982 ± 0.602 | 9.201 ± 0.241 | 7.980 ± 0.151 | 8.012 ± 0.466 |
| DeepPage | 7.124 ± 0.181 | 4.127 ± 0.134 | 7.245 ± 0.154 | 10.342 ± 0.422 | 9.012 ± 0.241 | 9.124 ± 0.673 |
| DRN | 8.006 ± 0.232 | 4.234 ± 0.241 | 6.112 ± 0.241 | 10.841 ± 0.112 | 9.412 ± 0.242 | 9.527 ± 0.455 |
| TPGR | 7.294 ± 0.312 | 2.872 ± 0.531 | 6.128 ± 0.541 | 14.713 ± 0.644 | 12.410 ± 0.612 | 13.225 ± 0.722 |
| PGPR | 6.619 ± 0.123 | 1.892 ± 0.143 | 5.970 ± 0.131 | 11.531 ± 0.241 | 10.333 ± 0.341 | 12.641 ± 0.442 |
| Ours | **8.208 ± 0.241** | **4.782 ± 0.341** | **7.876 ± 0.511** | **15.128 ± 0.241** | **12.451 ± 0.242** | **14.985± 0.252** |
| Dataset | Book-Crossing | | | GoodReads | | |
| Measure (%) | Recall | Precision | nDCG | Recall | Precision | nDCG |
| Wolpertinger | 0.782 ± 0.121 | 1.235 ± 0.131 | 0.976 ± 0.242 | 6.245 ± 0.122 | 3.415 ± 0.207 | 5.315 ± 0.321 |
| HLinearUCB | 2.421 ± 0.131 | 1.724 ± 0.141 | 2.865 ± 0.322 | 7.917 ± 0.303 | 5.151 ± 0.214 | 6.561 ± 0.351 |
| FactorUCB | 3.123 ± 0.141 | 2.976 ± 0.223 | 3.536 ± 0.241 | 5.643 ± 0.441 | 4.129 ± 0.221 | 6.122 ± 0.395 |
| ICTRUCB | 3.441 ± 0.121 | 3.421 ± 0.333 | 4.001 ± 0.321 | 8.415 ± 0.132 | 6.432 ± 0.221 | 7.124 ± 0.241 |
| DeepPage | 5.124 ± 0.323 | 3.245 ± 0.142 | 6.976 ± 0.142 | 10.071 ± 0.212 | 7.961 ± 0.232 | 8.329 ± 0.232 |
| DRN | 7.124 ± 0.122 | 4.123 ± 0.112 | 7.433 ± 0.142 | 10.620 ± 0.123 | 8.432 ± 0.241 | 9.461 ± 0.442 |
| TPGR | 7.246 ± 0.321 | **4.523 ± 0.442** | **7.870 ± 0.412** | 13.219 ± 0.323 | 10.322 ± 0.442 | 9.825 ± 0.642 |
| PGPR | 6.998 ± 0.112 | 3.932 ± 0.121 | 7.333 ± 0.133 | 11.421 ± 0.223 | 10.042 ± 0.212 | 9.234 ± 0.242 |
| Ours | **8.004 ± 0.223** | 4.521 ± 0.332 | 7.459 ± 0.401 | **13.444 ± 0.321** | **10.331 ± 0.331** | **11.641 ± 0.446** |
| Dataset | MovieLens-20M | | | Netflix | | |
| Measure (%) | Recall | Precision | nDCG | Recall | Precision | nDCG |
| Wolpertinger | 7.821 ± 0.171 | 2.341 ± 0.142 | 4.002 ± 0.151 | 3.924 ± 0.222 | 2.911 ± 0.141 | 3.425 ± 0.261 |
| HLinearUCB | 13.591 ± 0.281 | 10.601 ± 0.132 | 12.537 ± 0.285 | 5.142 ± 0.314 | 5.052 ± 0.362 | 6.007 ± 0.425 |
| FactorUCB | 14.421 ± 0.412 | 11.229 ± 0.365 | 11.422 ± 0.611 | 5.643 ± 0.432 | 4.129 ± 0.233 | 6.122 ± 0.442 |
| ICTRUCB | 14.345 ± 0.212 | 9.923 ± 0.222 | 11.051 ± 0.423 | 7.00 1± 0.312 | 6.212 ± 0.432 | 9.112 ± 0.523 |
| DeepPage | 12.472 ± 0.312 | 10.161 ± 0.332 | 13.129 ± 0.322 | 8.431 ± 0.212 | 7.324 ± 0.133 | 9.872 ± 0.223 |
| DRN | 14.742 ± 0.223 | 14.092 ± 0.342 | 16.245 ± 0.242 | 12.310 ± 0.144 | 10.213 ± 0.142 | 16.562 ± 0.153 |
| TPGR | 16.431 ± 0.369 | 13.421 ± 0.257 | 18.512 ± 0.484 | 12.512 ± 0.556 | 11.512 ± 0.595 | 17.425 ± 0.602 |
| PGPR | 14.234 ± 0.207 | 9.531 ± 0.219 | 11.561 ± 0.228 | 10.982 ± 0.181 | 10.123 ± 0.227 | 17.134 ± 0.243 |
| Ours | **18.021 ± 0.498** | **14.989 ± 0.432** | **19.007 ± 0.543** | **13.009 ± 0.343** | **11.874 ± 0.232** | **19.082 ± 0.348** |

dation performance. Another approach for knowledge aware recommendation is the embedding based [30], [31].

## VI. Conclusion and Future Work

In this paper, we have proposed a knowledge-guided deep reinforcement learning framework (KGRL) for interactive recommendation. KGRL uses the critic-actor learning framework to harness the interaction between users and the recommendation system and employs a local knowledge network to improve the stability and quality of the critic network for better decision-making. Extensive experiments over an online simulator with six public real-world datasets demonstrate its superior performance over state-of-the-art models. To verify the effectiveness for each component, we conduct the ablation study for the local knowledge network and attention mechanism and selectively present the performance both in normal case and extreme case. We are planning to introduce various types of user information (e.g., user's thought when browsing items) to enrich the interaction and deploy our model in online business platforms to further test the performance in the future. In addition, the cold-start problem is another big challenge to be focused on. Besides, the algorithm 1 used to train the model still lacks the knowledge about how to update step size will affect the training time and the convergence which can be solved in the future work.

## References

[1] X. Zhao, W. Zhang, and J. Wang, "Interactive collaborative filtering," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM, 2013.

[2] H. Wang, Q. Wu, and H. Wang, "Learning hidden features for contextual bandits," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2016, pp. 1633–1642.

[3] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[4] Q. Wang, C. Zeng, W. Zhou, T. Li, S. S. Iyengar, L. Shwartz, and G. Grabarnik, "Online interactive collaborative filtering using multi-armed bandit with dependent arms," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[5] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin, "Deep reinforcement learning in large discrete action spaces," *arXiv preprint arXiv:1512.07679*, 2015.

[6] X. Zhao, L. Xia, L. Zhang, Z. Ding, D. Yin, and J. Tang, "Deep reinforcement learning for page-wise recommendations," in *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 2018, pp. 95–103.

[7] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie, and Z. Li, "Drn: A deep reinforcement learning framework for news recommendation," in *Proceedings of the 2018 World Wide Web Conference*. IW3C2, 2018, pp. 167–176.

[8] Y. Liu, Y. Zhang, Q. Wu, C. Miao, L. Cui, B. Zhao, Y. Zhao, and L. Guan, "Diversity-promoting deep reinforcement learning for interactive recommendation," *arXiv preprint arXiv:1903.07826*, 2019.

[9] X. Zhao, L. Zhang, Z. Ding, L. Xia, J. Tang, and D. Yin, "Recommendations with negative feedback via pairwise deep reinforcement learning," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 1040–1048.

[10] S.-Y. Chen, Y. Yu, Q. Da, J. Tan, H.-K. Huang, and H.-H. Tang, "Stabilizing reinforcement learning in dynamic environment with application to online recommendation," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 1187–1196.

[11] Y. Xian, Z. Fu, S. Muthukrishnan, G. de Melo, and Y. Zhang, "Reinforcement knowledge graph reasoning for explainable recommendation," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2019, pp. 285–294.

[12] Y. Cao, X. Wang, X. He, Z. Hu, and T.-S. Chua, "Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences," in *The World Wide Web Conference*. ACM, 2019, pp. 151–161.

[13] H. Wang, M. Zhao, X. Xie, W. Li, and M. Guo, "Knowledge graph convolutional networks for recommender systems," in *The World Wide Web Conference*. ACM, 2019, pp. 3307–3313.

[14] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[15] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018.

[16] C. Zhou, J. Bai, J. Song, X. Liu, Z. Zhao, X. Chen, and J. Gao, "Atrank: An attention-based user behavior modeling framework for recommendation," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[18] Z. Li, Q. Chen, and V. Koltun, "Combinatorial optimization with graph convolutional networks and guided tree search," in *Advances in Neural Information Processing Systems*, 2018.

[19] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[20] W. Zhang, U. Paquet, and K. Hofmann, "Collective noise contrastive estimation for policy transfer learning," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[21] C. C. Johnson, "Logistic matrix factorization for implicit feedback data," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[22] H. Chen, X. Dai, H. Cai, W. Zhang, X. Wang, R. Tang, Y. Zhang, and Y. Yu, "Large-scale interactive recommendation with tree-structured policy gradient," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3312–3320.

[23] H. Wang, Q. Wu, and H. Wang, "Factorization bandits for interactive recommendation," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[24] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 661–670.

[25] Y. Shen, Y. Deng, A. Ray, and H. Jin, "Interactive recommendation via deep neural memory augmented contextual bandits," in *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 2018, pp. 122–130.

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[27] A. Grotov and M. de Rijke, "Online learning to rank for information retrieval: Sigir 2016 tutorial," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2016.

[28] H. Zhao, Q. Yao, J. Li, Y. Song, and D. L. Lee, "Meta-graph based recommendation fusion over heterogeneous information networks," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 635–644.

[29] H. Wang, F. Zhang, M. Zhang, J. Leskovec, M. Zhao, W. Li, and Z. Wang, "Knowledge-aware graph neural networks with label smoothness regularization for recommender systems," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019.

[30] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2016, pp. 353–362.

[31] J. Huang, W. X. Zhao, H. Dou, J.-R. Wen, and E. Y. Chang, "Improving sequential recommendation with knowledge-enhanced memory networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 2018, pp. 505–514.