KERL: A Knowledge-Guided Reinforcement Learning Model for Sequential Recommendation

Pengfei Wang
Beijing University of Posts and Telecommunications
wangpengfei@bupt.edu.cn

Yu Fan
Beijing University of Posts and Telecommunications
fanyubupt@gmail.com

Long Xia
School of Information Technology, York University
longxia@yorku.ca

Wayne Xin Zhao[*][†]
Gaoling School of Artificial Intelligence
Renmin University of China
batmanfly@gmail.com

Shaozhang Niu
Beijing University of Posts and Telecommunications
szniu@bupt.edu.cn

Jimmy Huang
School of Information Technology, York University
jhuang@yorku.ca

ABSTRACT

For sequential recommendation, it is essential to capture and predict future or long-term user preference for generating accurate recommendation over time. To improve the predictive capacity, we adopt reinforcement learning (RL) for developing effective sequential recommenders. However, user-item interaction data is likely to be sparse, complicated and time-varying. It is not easy to directly apply RL techniques to improve the performance of sequential recommendation.

Inspired by the availability of knowledge graph (KG), we propose a novel ***K***nowledge-guid***E***d ***R***einforcement ***L***earning model (KERL for short) for fusing KG information into a RL framework for sequential recommendation. Specifically, we formalize the sequential recommendation task as a Markov Decision Process (MDP), and make three major technical extensions in this framework, including state representation, reward function and learning algorithm. First, we propose to enhance the state representations with KG information considering both exploitation and exploration. Second, we carefully design a composite reward function that is able to compute both sequence- and knowledge-level rewards. Third, we propose a new algorithm for more effectively learning the proposed model. To the best of our knowledge, it is the first time that knowledge information has been explicitly discussed and utilized in RL-based sequential recommenders, especially for the exploration process. Extensive experiment results on both next-item and next-session recommendation tasks show that our model can significantly outperform the baselines on four real-world datasets.

CCS CONCEPTS

• **Information systems → Personalization**; **Recommender systems**.

KEYWORDS

Sequential Recommendation, Reinforcement Learning, Knowledge Graph

ACM Reference Format:
Pengfei Wang, Yu Fan, Long Xia, Wayne Xin Zhao, Shaozhang Niu, and Jimmy Huang. 2020. KERL: A Knowledge-Guided Reinforcement Learning Model for Sequential Recommendation. In *Proceedings of Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401134

1 INTRODUCTION

Sequential recommendation aims to recommend the next item or next few items successively based on users' sequential interaction behaviors [10, 27]. Various methods have been proposed to address this task, such as classic matrix factorization techniques [15] and popular recurrent neural network approaches [5, 9, 14]. Typically, these methods are trained with Maximum Likelihood Estimation (MLE) for fitting observed interaction sequence step by step. However, long-term or overall effectiveness has not been well characterized in the optimization objectives by previous studies. The remarkable recent progress on reinforcement learning (RL) [24] provides a promising solution to this problem by considering maximizing long-term performance.

Although it is appealing in theory, it is a non-trivial problem to optimize long-term reward for sequential recommendation in practice. First, user-item interaction data is likely to be sparse or limited. It is not easy to directly learn towards a more difficult optimization objective. Second, a core concept or mechanism for RL models is the *exploration* process. It may not be reliable to adopt a blind or random exploration strategy for capturing the evolvement of user interests. In essence, user behaviors are complicated and varying, and a more controllable learning process is preferred for applying RL algorithms to sequential recommendation. Inspired by the availability of knowledge graph (KG) and its applicability in various fields [13, 32], we would like to utilize the informative

[*] Corresponding author.
[†] Also with, School of Information, Renmin University of China.

SIGIR '20, July 25–30, 2020, Virtual Event, China
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401134

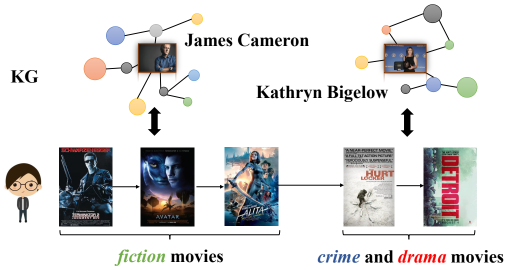

Figure 1: An example to illustrate the drift of user interests. At the first stage, the user focuses on fiction movies, and then switches to the crime and drama movies at the second stage.

KG data to guide the RL-based learning method for sequential recommendation.

Indeed, KG data has been widely utilized in recommendation tasks [12, 26, 28]. Previous studies mainly utilize KG data for *exploitation*, while they seldom consider the effect of knowledge information in the *exploration* process. As such, they cannot well capture the potential drift of user preference in the coming time. To see this, we present an illustrative example in Fig. 1, where a sequence of five movies watched by a user has been given. Interestingly, the overall sequence can be split into two consecutive stages (*i.e.,* subsequences) reflecting different movie preferences. The first three movies are directed by "*James Cameron*", and the rest two movies are directed by his former wife "*Kathryn Bigelow*". Intuitively, KG data can be useful to improve the recommendation performance within each individual stage, since the movies in the same stage share similar KG characteristics. However, the movies across the two stages are essentially different in many aspects, *e.g.,* genre, style, and story. Existing knowledge-aware sequential recommenders are likely to be "trapped" in the first stage, and cannot effectively capture the preference drift between the two stages. Such a problem is essentially about the trade-off between *exploitation* and *exploration* in RL. Considering the two factors, it needs to develop a more principled approach to sequential recommendation for better utilizing knowledge information.

To address the above issues, in this paper, we propose a novel ***K***nowledge guid***E***d ***R***einforncement ***L***earning model (KERL for short) for fusing KG information into a RL framework for sequential recommendation. Specifically, we formalize the sequential recommendation task as a Markov Decision Process (MDP), and make three major technical extensions in this framework. First, we propose to enhance the state representations with KG information. By learning both sequence-level and knowledge-level state representations, our model is able to capture user preference more accurately. Especially, we argue that it is important to utilize KG information in the exploration process. To achieve this, we construct an induction network that aims to predict future knowledge characteristics of user preference. In this way, we can learn knowledge-based user preference, considering both exploitation and exploration. Second, we carefully design a composite reward function that is able to compute both sequence-level and knowledge-level reward signals. For sequence-level reward, we borrow the BLEU metric [21] from machine translation, and measure the overall quality of the recommendation sequence. For knowledge-level reward, we force the knowledge characteristics of the actual and the recommended sequences to be similar. Third, we propose a truncated policy gradient strategy to train our model. Concerning the sparsity and instability in training induction network, we further incorporate a pairwise learning mechanism with simulated subsequences to improve the learning of the induction network. To evaluate the proposed model, we construct extensive experiments on four datasets by comparing it with several competitive baselines. Experiment results on both next-item and next-session recommendation tasks show that our model can significantly outperform all the baselines in sequential recommendation tasks.

In summary, the contributions of our work are as follows:

- We formalize the sequential recommendation task into a Markov Decision Process (MDP), and fuse KG information to enhance the recommendation performance. To our knowledge, it is the first time that knowledge graph data has been explicitly discussed and utilized in *RL-based sequential recommenders*, especially for the exploration process.
- We make three novel extensions in the MDP framework for sequential recommendation, including state representation, reward function and learning strategy. With the three major extensions, KG information has been effectively utilized and integrated into the RL-based sequential recommenders.
- Empirical results on four real-world datasets show that our model can consistently outperform state-of-the-art baselines on both next-item and next-session recommendation tasks under different metrics.

2 RELATED WORK

In this section we briefly review three research areas related to our work, namely sequential recommendation, knowledge-based recommendation, and reinforcement learning respectively.

Sequential Recommendation. Sequential recommendation aims to predict users' future behaviors given their historical interaction data. Early work usually utilized Markov Chains to capture single-step dependence of sequential behaviors. For example, Rendle et al. [22] designed a personalized markov chain to provide recommendations. Furthermore, Wang et al. [27] utilized representation learning metric to model complex interactions between users and items, while Pasricha et al. [19] combined translation and metric-based approaches for sequential recommendation. Another line is to model multistep sequential behaviors, which are proved to be a more effective way for sequential recommendation, and Recurrent Neural Networks (RNN) based models are widely applied in this area [5, 20, 29]. Comparing with the previous MC-based models, RNN based models can well capture longer sequential behaviors for recommendation. For example, Quadrana et al. [20] utilized Gated Recurrent Units (GRU) to model click sequences for session-based recommendation. Li et al. [16] further fed attention mechanism into RNN to capture both the user's sequential behavior and main purpose for session-based recommendation. Kang et al. [14] proposed a novel self-attentive approach to modeling the pairwise item interaction in user sequences.

Knowledge-based Recommendation. With the development of knowledge graph (KG) technology [2, 7, 17, 18], researchers also tried to incorporate KG to improve the performance of recommender systems. For example, Huang et al. [12] utilized Memory Network to store and represent knowledge base information to enhance the effectiveness and interpretability of sequential recommendation. Huang et al. [11] designed a multi-hop reasoning architecture to utilize the taxonomy information for improving item recommendation. Wang et al. [26] introduced preference propagation to automatically propagate users' potential preferences in KG. Wang et al. [28] exploited high-order relations by extracting paths from KG to capture more connectives between users and items. Though it is effective to incorporate KG to improve the performance, these works do not model users' long-term benefits, thus the performance might be limited.

RL-based Recommendation Reinforcement learning has been introduced into recommender systems as its advnatange of considering users' long-term feedbacks [34, 36]. For example, Zhang et.al. [30] proposed a policy-gradient approach to searching paths in KG to interpret the recommendation process. Zou et.al. [37] formulated the ranking process as a multi-agent Markov Decision Process, where mutual interactions among documents are incorporated to compute the ranking list. Bai et.al. [1] explored adversarial training over a model-based RL framework for recommendation. To our knowledge, it is the first time that KG information has been explicitly discussed and utilized in RL-based sequential recommenders, especially for the exploration process.

3 PRELIMINARY

Notations. We consider a sequential interaction scenario in recommender systems, where a user selects interested items at different time steps. Let $\mathcal{U}$ denote a set of users and $\mathcal{I}$ denote a set of items. For each user $u \in \mathcal{U}$, we use $i^u_{1:n} = i^u_1 \rightarrow i^u_2 \rightarrow \cdots \rightarrow i^u_n$ to denote the interaction sequence for user u, where i^u_t represents the item that u has interacted with at t-th time step and n is the sequence length. Similarly, we use $i^u_{j:k} = i^u_j \rightarrow i^u_{j+1} \rightarrow \cdots \rightarrow i^u_k$ to represent a subsequence of $i^u_{1:n}$ ranging from index j to index k, where $1 \leq j < k \leq n$. In addition to users' interaction histories, a knowledge graph (KG) $\mathcal{G}$ is available for our task, where each record is a knowledge triple involving two entities with some relation. We assume that the item set can be aligned to the KG [33]. Via the aligned KG, we can obtain the associated knowledge information of the items, *e.g.,* the author of a book or the director of a movie.

Task Definition. Based on these notations, our task of *sequential recommendation* aims to predict the next item that a user u will interact with given both the interaction history $i^u_{1:n}$ and KG $\mathcal{G}$. As shown in [12], it will be similar to define the sequential recommendation task in a basket- or session-based setting. For simplicity, we only present the next-item setting, and will focus on next-item recommendation in model description (Section 4).

Markov Decision Process. We consider a reinforcement learning (RL) approach to sequential recommendation. We first briefly introduce Markov Decision Process (MDP) [25]. Generally, a MDP can be described by a quintuple $\langle \mathcal{S}, \mathcal{A}, T, R, \pi \rangle$: (1) $\mathcal{S}$ is a set of states, and each $s \in \mathcal{S}$ denotes the information state of the environment; (2) $\mathcal{A}$ is a set of actions, and each $a \in \mathcal{A}$ denotes an action that the agent that is able to perform; (3) T is the state transition function for updating the state according to the action and current state, *i.e.,* $s_{t+1} = T(s_t, a_t)$; (4) R is the reward function and $r = R(s, a)$ giving the immediate reward of performing action a at state s; and (5) $\pi(a|s)$ describes the behavior of an agent, usually modeled by a probability distribution over the possible actions.

4 OUR APPROACH

In this section, we introduce the proposed Knowledge-Guided Reinforcement Learning Model (KERL) in detail, and the overall architecture of KERL is presented in Fig. 2. Our approach is able to effectively fuse knowledge graph (KG) information into the RL framework for sequential recommendation. In what follows, we start with a Markov Decision Process (MDP) formulation for our task, and then present our extensions on state representation, reward function and learning algorithms.

For simplicity, we describe the approach for a single user u, and it is straightforward to extend the formulas to a set of users. We drop the superscript of u in the notations (Section 3) for ease of reading.

4.1 A MDP Formulation for Our Task

First, we use MDP to frame our task. In an MDP, there is an agent to interact with the environment at discrete time steps. At each time step t, the process is in some state $s_t \in S$. In our task, the environment's state can be considered to contain all useful information for sequential recommendation, including interaction history and KG. In our case, we consider two major elements:

$$s_t = [i_{1:t}, \mathcal{G}], \tag{1}$$

where $i_{1:t}$ denotes the previous interaction sequence generated by user u and $\mathcal{G}$ denotes the KG information. The initial state is set as: $s_0 = [\emptyset, \mathcal{G}]$. Following [6], we can use an embedding vector $\mathbf{v}_{s_t} \in \mathbb{R}^{L_S}$ to encode the information of state s_t. $\mathbf{v}_{s_t}$ is expected to encode useful information for representing state s_t.

According to state s_t, the agent performs an action $a_t \in \mathcal{A}$, which selects an item i_{t+1} from the item set $\mathcal{I}$ for recommendation. The action behavior is modeled by a policy $\pi(s_t)$, which defines a function that takes the state s_t as input and outputs a distribution over all the possible items. In this paper, we use a softmax function to compute the probability of selecting an item:

$$\pi(a_t|s_t) = \frac{\exp\{\mathbf{q}_{i_{j(a_t)}} \mathbf{W}_1 \mathbf{v}_{s_t}\}}{\sum_{i_j \in \mathcal{I}} \exp\{\mathbf{q}_{i_j} \mathbf{W}_1 \mathbf{v}_{s_t}\}} \tag{2}$$

where $\mathbf{q}_j \in \mathbb{R}^{L_I}$ denotes the embedding vector for the j-th item, $\mathbf{W}_1$ is the parameter in the bilinear product and $\mathbf{v}_{s_t}$ is the embedding vector for state s_t.

After each action, the agent receives a numerical intermediate reward, *i.e.,* $r_{t+1} = R(s_t, a_t)$. The reward function can be set to reflect the recommendation performance as needed in our task. Furthermore, it utilizes the transition function T ($T : \mathcal{S} \times T \rightarrow \mathcal{S}$) to update the state:

$$s_{t+1} = T(s_t, a_t) = T\Big([u, i_{1:t}, \mathcal{G}], i_{j(a_t)}\Big). \tag{3}$$

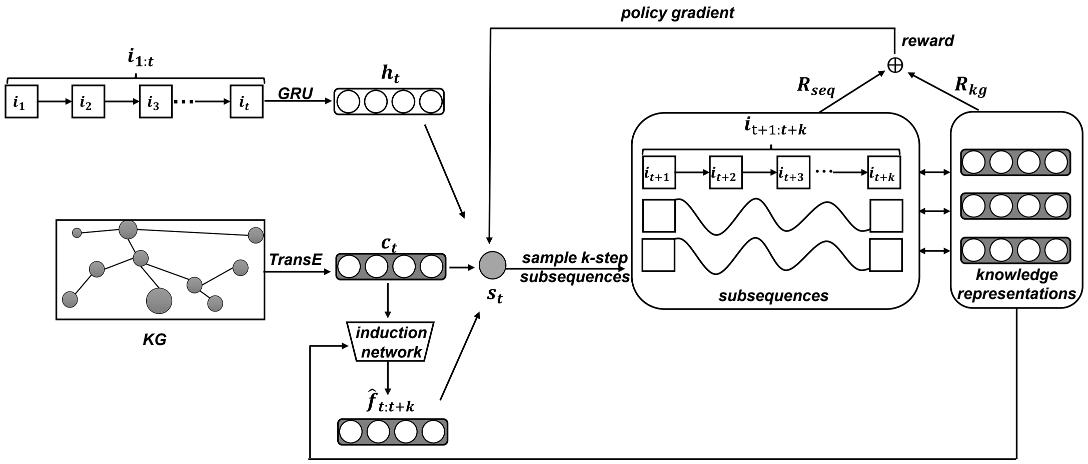

Figure 2: The overall architecture of Knowledge-Guided Reinforcement Learning Model (KERL). KERL formalizes the sequential recommendation by a MDP, and three novel extensions are designed to fuse KG information to enhance the recommendation performance.

The new state s_{t+1} can be written as $[i_{1:t+1}, \mathcal{G}]$, and is also associated with an embedding vector $\mathbf{v}_{s_{t+1}}$.

4.2 Learning Knowledge-Enhanced State Representation

A key point for MDP-based algorithms is to model and learn a good state representation, since it encodes necessary information characteristics for a state. Although many RL methods have been proposed in sequence-based tasks [21, 35], they mainly focus on the learning algorithm and lack the utilization of external knowledge information. Here, we propose to incorporate knowledge information to enhance the state representations, and set two kinds of state representations, namely sequence-level and knowledge-level state representations. In this way, it is expected to guide the sequence-level RL learning algorithm using the informative KG data.

4.2.1 Sequence-level State Representation. For the first kind of state representations, we adopt a standard recurrent neural network for encoding previous interaction sequence:

$$\mathbf{h}_t = \mathrm{GRU}(\mathbf{h}_{t-1}, \mathbf{q}_{i_t}; \Phi_{gru}) \tag{4}$$

where GRU(·) is the Gated Recurrent Unit [4], $\mathbf{q}_{i_t}$ is the embedding vector for item i_t, and Φ_{gru} denotes all the related parameters of the GRU network. Such a representation mainly captures sequential characteristics of user preference, and it does not utilize knowledge information for deriving state representations.

4.2.2 Knowledge-level State Representation. As shown in [12], KG data is useful to improve the performance of sequential recommendation algorithms. However, previous methods mainly consider enhancing item or user representations with KG data for fitting short-term behaviors with MLE [35]. They seldom study how KG data can be utilized for *exploration* that optimizes long-term objective. To make a good trade-off between exploitation and exploration, we consider modeling two kinds of knowledge-based preference for a user, namely current knowledge-based preference (short as *current preference*) and future knowledge-based preference (short as *future preference*).

Learning Current Preference. It is relatively easy to derive the current knowledge characteristics based on historical data. Recall that, each item i is associated with an entity e_i in KG $\mathcal{G}$. We utilize the widely used KG embedding method TransE [3] to derive an embedding vector for an item i_t, denoted by $\mathbf{v}_{e_{i_t}} \in \mathbb{R}^{L_E}$. Furthermore, we use a simple average pooling method to aggregate all the KG embeddings of the historical items that a user has interacted with:

$$\mathbf{c}_t = \sum_{i=1}^{t} \mathrm{Average}(\mathbf{v}_{e_{i_t}}). \tag{5}$$

Note that here we do not consider temporal information or attention mechanism in the above equation, since it does not show significant performance improvement than the above simple method.

Predicting Future Preference. As the key point to achieve effective exploration, we incorporate *future preference* for capturing the possible interest evolving of a user at upcoming time steps. Intuitively, knowing future preference is useful for sequential recommendation, especially in a RL setting. Based on current preference, our idea is to develop an induction network to directly predict the future preference. Specially, we construct a neural network using a Multi-Layer Perception. At time step t, we predict a k-step future preference representation taking as input the current preference representation $\mathbf{c}_t$ (Eq. 5):

$$\mathbf{f}_{t:t+k} = \mathrm{MLP}(\mathbf{c}_t; \Phi_{mlp}), \tag{6}$$

where $\mathbf{f}_{t:t+k}$ denotes the k-step future preference at time t, and we use Φ_{mlp} to represent parameters used in the induction network. We assume that knowledge-based preference should not change too much at consecutive time steps. Hence, we aim to predict future preference based on existing information. Learning future preference from KG data is particularly useful to a RL-based algorithm, since it is key to develop a reasonable and informative exploration process. By using a MLP component, the induction network is expected to better capture the evolvement of user interests, especially preference drift.

4.2.3 Deriving the Final State Representation. Based on the above discussions, we are ready to give the final state representation in our model. For a state s_t, its representation $\mathbf{v}_{s_t}$ is the combination of three representation vectors:

$$\mathbf{v}_{s_t} = \mathbf{h}_t \oplus \mathbf{c}_t \oplus \mathbf{f}_{t:t+k}, \tag{7}$$

where "$\oplus$" is the vector concatenation operator, $\mathbf{h}_t$ is the sequence-level state vector (Eq. 4), $\mathbf{c}_t$ is the current knowledge-based preference (Eq. 5) and $\mathbf{f}_{t:t+k}$ is the future knowledge-based preference (Eq. 6). The first factor $\mathbf{h}_t$ mainly characterizes sequence-level information, the second factor $\mathbf{c}_t$ summarizes the existing knowledge characteristics for achieving knowledge-based exploitation, and the third factor $\mathbf{f}_{t:t+k}$ predicts possible future preference for achieving knowledge-based exploration. By combining the three factors, our approach naturally incorporates KG data for state representation in the MDP framework, making a good trade-off between exploitation and exploration.

4.3 Setting the Reward with Knowledge Information

Defining an appropriate reward function is especially important for RL algorithms. In sequential recommendation, the final performance is usually measured based on the exact match for item IDs. While, the interaction sequence is generated by a user according to her/his preference over item attributes or profiles (which can be obtained from KG). Hence, besides item-level performance, it is also important to measure the quality of the inferred knowledge-level preference.

4.3.1 Reward Decomposition. Based on the above motivation, at time step t, we define the k-step reward function by integrating two different reward functions:

$$R(s_t, a_t) = R_{seq}(i_{t:t+k}, \hat{i}_{t:t+k}) + R_{kg}(i_{t:t+k}, \hat{i}_{t:t+k}), \tag{8}$$

where $R_{seq}(\cdot,\cdot)$ and $R_{kg}(\cdot,\cdot)$ measure the sequence-level and knowledge-level rewards, respectively, taking as input the information of the ground-truth subsequence $i_{t:t+k}$ and the recommended subsequence $\hat{i}_{t:t+k}$. Note that we consider a k-step measurement for approximating the overall performance. Next we discuss how to set both $R_{seq}(\cdot,\cdot)$ and $R_{kg}(\cdot,\cdot)$ for our task.

4.3.2 Sequence-level Reward. In sequence recommendation, a good reward function should not only consider the performance at individual steps, but also need to measure the overall performance in terms of a recommendation sequence. Inspired by the works in the evaluation of machine translation [21], we borrow the metric of *BLEU* for sequence recommendation. Formally, given actual interaction subsequence $i_{t:t+k}$ and the recommended subsequence $\hat{i}_{t:t+k}$, we define the reward function as

$$R_{seq}(i_{t:t+k}, \hat{i}_{t:t+k}) = \exp\left(\frac{1}{M}\sum_{m=1}^{M} \log prec_m\right) \tag{9}$$

where $prec_m$ is the modified precision and calculated as:

$$prec_m = \frac{\sum_{p_m \in i_{t:t+k}} \min\left(\#(p_m, i_{t:t+k}), \#(p_m, \hat{i}_{t:t+k})\right)}{\sum_{p_m \in i_{t:t+k}} \#(p_m, i_{t:t+k})}$$

where p_m is an m-gram subsequence of $i_{t:t+k}$, $\#(p_m, i_{t:t+k})$ is the number of times that p_m appears in $i_{t:t+k}$, and M determines how many m-gram precision-scores to use. As we can see, such a reward function advocates the recommendation algorithm to generate more consistent m-grams from the actual sequence. It naturally measures the sequence-level performance for our task.

4.3.3 Knowledge-level Reward. In the second reward function, we do not focus on the exact match with item IDs. Instead, we consider measuring the quality of knowledge-level characteristics reflected in the sequences. Given an actual and predicted subsequences, namely $i_{t:t+k}$ and $\hat{i}_{t:t+k}$, we still use the simple average method in Eq. 5 to aggregate the TransE embeddings of items to derive subsequence-level knowledge representations, denoted by $\mathbf{c}_{t:t+k}$ and $\hat{\mathbf{c}}_{t:t+k}$, respectively. These two knowledge-level representations reflect the user preference over item attributes or profiles. To measure the difference between the two vectors, we adopt the cosine similarity as the reward function:

$$R_{kg}(i_{t:t+k}, \hat{i}_{t:t+k}) = \frac{\mathbf{c}_{t:t+k} \cdot \hat{\mathbf{c}}_{t:t+k}^{\top}}{\parallel \mathbf{c}_{t,t+k} \parallel \cdot \parallel \hat{\mathbf{c}}_{t:t+k} \parallel}. \tag{10}$$

We can flexibly replace the cosine function with other forms of similarity measurements. We compute the reward by considering both the sequence-level and knowledge-level match. To our knowledge, such an idea is seldom considered for deriving the cost function or reward function in sequential recommendation.

By plugging Eq. 9 and Eq. 10 into Eq. 8, we can derive the final reward function. By providing these two kinds of reward signals, we expect the RL algorithm to yield a better recommendation performance.

4.4 Learning and Discussion

In our model, we leverage external knowledge graph (KG) to improve the RL algorithm for sequential recommendation. A key point lies in how to learn an effective knowledge-level state representation for sequential recommendation. For this purpose, we first utilize the Monte Carlo algorithm to sample a set of truncated subsequences to train MDP. Based on these subsequences, a pairwise ranking mechanism is then proposed to learn the induction network. The overall algorithm is given in Alg. 1. We next introduce the training details.

4.4.1 Training with Truncated Policy Gradient. In our task, our goal is to learn a stochastic policy π that maximizes the expected cumulative reward $J(\Theta)$ for all uses. The derivative of $J(\Theta)$ can be given below:

$$\nabla J(\Theta) = E_{\pi}\left[\sum_{u}\sum_{j=t}^{n} \gamma^{j-t} R_j \frac{\nabla \pi(a_t|s_t;\Theta)}{\pi(a_t|s_t;\Theta)}\right] \tag{11}$$

where γ is the discount factor to balance the importance of the current reward and future reward, and "Θ" represents all the involved

Algorithm 1 Learning algorithm for KERL

```
Input: policy $\pi$, number of sample times $L$, reward function $R$, knowledge graph $\mathcal{G}$, and learning rate $\alpha$.
Initialize $\Theta \leftarrow$ random values
Obtain KG embedding according to TransE
for each $u$ in $\mathcal{U}$ do
    for $t = 1$ to $n$ do
        Obtain $\mathbf{h}_t$ and $\mathbf{c}_t$ according to Eq. 4 and Eq. 5
        $\mathbf{v}_{s_t} = \mathbf{h}_t \oplus \mathbf{c}_t \oplus MLP(\mathbf{c}_t)$
        for $l = 1$ to $L$ do
            Sample a $k$-step subsequence $\hat{i}^{(l)}_{t:t+k}$ according to Eq. 2
            $\Theta \leftarrow \Theta + \alpha \sum_{j=t}^{t+k} \gamma^{j-t} R_j \frac{\nabla \pi(\hat{i}^{(l)}_t | s_t; \Theta)}{\pi(\hat{i}^{(l)}_t | s_t; \Theta)}$
            Update $\Phi_{mlp}$ of induction network with pairwise ranking
            $\mathbf{v}_{s_t} = \mathbf{h}_t \oplus \mathbf{c}_t \oplus MLP(\mathbf{c}_t)$
        end for
    end for
end for
return $\Theta$
```

parameters to learn. We employ a truncated policy gradient strategy to learn the parameters. Specifically, for each state s_t, KERL simulates a truncated k-step subsequence $\hat{i}^{(l)}_{t:t+k}$ according to policy function Eq. 2. Given $\hat{i}^{(l)}_{t:t+k}$, the learning process is written as follows:

$$\nabla\Theta = \sum_{j=t}^{t+k} \gamma^{j-t} R_j \frac{\nabla \pi(\hat{i}^{(l)}_t | s_t; \Theta)}{\pi(\hat{i}^{(l)}_t | s_t; \Theta)} \qquad (12)$$

KERL then repeats this process L times to estimate a better value.

4.4.2 Training the Induction Network. An important component of our model is the induction network (Eq. 6), which predicts the future preference based on current preference. To train such a neural network, a straightforward method is to fit it with ground-truth preference vector (*i.e.,* computing the actual preference using Eq. 6) using regression loss, *e.g.,* Mean-Squared Error (MSE). However, the KG information is likely to contain noise or irrelevant information for the recommendation task. Besides, the supervision signal for real-vector regression is sparse in our scenario, since only a single ground-truth preference vector $\mathbf{f}_{t:t+k}$ is available for a given subsequence. To better learn the induction network, we propose a pairwise ranking strategy to improve the training process.

Based on the simulated subsequences for state s_t, we can derive their corresponding knowledge representations using the method in Eq. 6, denoted by $\hat{\mathbf{f}}^{(1)}_{t:t+k}, \hat{\mathbf{f}}^{(2)}_{t:t+k}, \cdots, \hat{\mathbf{f}}^{(L)}_{t:t+k}$. By calculating their reward values according Eq. 10, we can form pairwise comparisons as additional constraints to improve the learning. Specifically, we first utilize their reward values to construct the preference order over L subsequences, given $\hat{\mathbf{f}}^{(l)}_{t:t+k}$ and $\hat{\mathbf{f}}^{(l')}_{t:t+k}$, and then add a pairwise constraint to train the induction network, where $\text{MLP}(\hat{\mathbf{f}}^{(l)}_{t:t+k}) > \text{MLP}(\hat{\mathbf{f}}^{(l')}_{t:t+k})$ if $R_{kg}(i_{t:t+k}, \hat{i}^{(l)}_{t:t+k}) > R_{kg}(i_{t:t+k}, \hat{i}^{(l')}_{t:t+k})$ for $1 \le l, l' \le L$.

4.4.3 Discussion. The major novelty of the KERL model lies in the incorporation of future knowledge-based preference $\mathbf{f}_{t:t+k}$ (Eq. 6). Such a factor has been missing in previous knowledge-aware recommendation models [11, 12, 28], which makes it difficult to capture the drift or evolvement of user interests. Although previous RL-based recommendation models [30, 35, 37] force the model to maximize the long-term reward, they either rely on the reward function or adopt a random exploration strategy. Therefore, a high-level, informative exploration strategy has not been well studied by these RL-based models. Furthermore, we utilize both sequence-level and knowledge-level reward functions to enhance such an informative exploration process.

With the learned KERL model, given a user and his/her interaction sequence, we can calculate the probability of an action for some an item according to Eq. 2. Then, we rank the items according to their probabilities, and select the top-N results as the final recommendations.

Table 1: Statistics of datasets for experiments (a.v.l=average sequence length).

Dataset	#interactions	#users	#items	#a.v.l	#entities	#relations
Beauty	198,502	22,363	12,101	9.87	42,355	20
CD	472,265	17,052	35,118	23.69	151,484	27
Books	1,856,747	52,406	41,264	28.43	313,956	49
LastFM	203,975	7,694	30,658	21.11	214,524	19

5 EXPERIMENT

In this section, we evaluate our proposed models by comparing with several state-of-the-art methods. We begin with introducing the experimental setup, and then analyze the experimental results.

5.1 Experimental Setup

Dataset. We evaluate different recommendation algorithms over four datasets, including three e-commerce recommendation datasets, and a music recommendation dataset.

- Amazon[1]: Amazon [8] comprises a large corpus of reviews and timestamps on various products. We adopt three categories of diverse sizes and sparsity levels, which are *Books*, *Beauty*, and *CD*.
- LastFM[2]: LastFM [23] is a music listening dataset released from Last.fm online music system. We take the subset of the dataset where the timestamp is from Jan, 2015 to June, 2015.

For all datasets, we remove users and items with less than five interaction records. In our work, we need to obtain the knowledge graph information for items in each dataset. For the lastFM and Book dataset, we use the KB4Rec dataset [12, 33] to link the items with Freebase entities to obtain KB triples. For the rest Amazon datasets, we use the same relations by following [30] as knowledge information. The statistics of four datasets are shown in Table 1.

For each user, we sort her/his records according to the timestamp to form the interaction sequence. Based on the sorted sequences, for next-item recommendation task, we hold out the last item of each sequence as the test data; for next-session recommendation task, we hold out the items from the last session in the interaction

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

[2] http://www.cp.jku.at/datasets/LFM-1b/

Table 2: Performance comparison on sequential recommendation between the baselines and our model (all the values in the table are percentage numbers with % omitted). The best performance in each row is in bold font, and the starred numbers represent best baseline performance. The last column shows the absolute improvement of our results against the best baseline, which is significant at p-value≤0.05. We do not present results of Ripple and KGAT on session-based task as they are not suitable for this task.

Task	Dataset	Evaluation metric	Sequential-based Models			Knowledge-based Models		Hybrid Models			Improvement
			FPMC	DREAM	GRU4Rec	Ripple	KGAT	GRU_F	KSR	KERL	
Next-item recommendation	Beauty	Hit-Ratio@10	30.6	32.1	39.4	42.2	44.0	49.2	51.0*	**54.1**	3.1%
		NDCG@10	18.1	23.3	29.5	21.4	27.6	32.8*	32.2	**36.5**	3.7%
	CD	Hit-Ratio@10	21.3	24.5	50.5	58.4	63.4	64.2	68.3*	**73.7**	5.4%
		NDCG@10	11.6	15.6	32.9	37.6	41.7	40.6	45.0*	**50.8**	5.8%
	Books	Hit-Ratio@10	19.6	21.3	56.2	63.8	70.2	68.3	75.1*	**80.0**	4.9%
		NDCG@10	11.0	20.9	38.5	42.8	45.8	43.1	52.4*	**57.1**	4.7%
	LastFM	Hit-Ratio@10	31.0	35.5	52.8	52.5	55.8	58.2	62.7*	**64.2**	1.5%
		NDCG@10	19.7	22.6	40.7	38.2	42.1	44.1	48.1*	**50.1**	2.0%
Next-session recommendation	Beauty	Hit-Ratio@10	18.8	20.3	24.2	—	—	43.2	45.9*	**48.2**	2.3%
		NDCG@10	10.1	11.2	14.0	—	—	28.0	29.3*	**31.4**	2.1%
	CD	Hit-Ratio@10	17.0	20.5	31.3	—	—	54.5	57.2*	**60.5**	3.3%
		NDCG@10	8.71	10.2	16.7	—	—	31.4	32.2*	**36.8**	4.6%
	Books	Hit-Ratio@10	15.5	18.9	28.9	—	—	52.8	54.6*	**58.2**	3.6%
		NDCG@10	7.94	11.3	17.8	—	—	29.1	30.0*	**35.2**	5.2%
	LastFM	Hit-Ratio@10	19.1	20.2	22.0	—	—	31.9	41.5*	**44.1**	2.6%
		NDCG@10	11.0	11.4	12.7	—	—	18.8	22.8*	**25.0**	2.2%

sequence as the test data, in which a day is considered as a session. The rest data of the two tasks is treated as the training data, while 5% of data from testing sets are further randomly selected as the validation sets. To accelerate the evaluation process, for each user u, we randomly sample 100 negative items, and rank these items with the ground-truth item.

Baselines. We adopt three types of baselines for comparison, including sequential-based models, knowledge-based models, and hybrid models. For sequential based models, we consider both shallow models and deep models:

- *FPMC* [22] is a shallow model that combines matrix factorization and factorizes first-order Markov chains for sequential recommendation.
- *DREAM* [31] leverages recurrent neural network to model the dynamic representations of users and the sequential relationships between items.
- *GRU4Rec* [9] is a session-based recommendation, which utilizes GRU to capture users' long-term sequential behaviors.

The knowledge-based models include:

- *KGAT* [28] explores the high-order connectivity with semantic relations in collaborative knowledge graph for knowledge-aware recommendation.
- *Ripple* [26] is an embedding-based method that models users' potential interests along links in the knowledge graph for recommendation.

The hybrid methods include:

- *KSR* [12] is a novel GRU-based sequential recommender by integrating it with knowledge-enhanced KV-MNs.
- *GRU_F* [10] is an extension of GRU4Rec that incorporates auxiliary features for recommendation. Similar to [12], we concatenate the pre-trained BPR item vector and the KG embedding as the input of GRU.

For GRU4Rec, KGAT, RippleNet, and KSR, we use the codes released by their authors. The rest models are implemented in PyTorch.

Evaluation Metrics. For each a user, a method will produce a top-N recommendation list for evaluation. Following [12], we employ the commonly used Hit-Ratio@k and NDCG@k as evaluation metrics. k is set to 10 in our experiments. For next-session recommendation task, we have multiple ground-truth items for each user. We average the results of all the items as the final performance. With paired t-test, performance differences are considered statistically significant when the p-value is lower than 0.05.

Parameter Settings. We optimize all models with the Adam optimizer by setting the batch size as 1024. For BLEU metric, we consider up to 3-grams. For baselines, we optimize each of them according to the validation sets. For our model, the hidden layer size of GRU and item embedding size are set to 64, and the KG embedding size with TransE is set to 50. For the hyper-parameters, the discount factor γ is set to 0.9, the episode length k is set to 3 and 5 on next-item recommendation and next-session recommendation task respectively.

5.2 Performance Comparison

We compare our KERL model against several competitive baseline methods on both next-item and next-session recommendation task.

Table 3: Performance comparison of KERL$_h$, KERL$_{h+c}$, KERL$_{h+f}$ and KERL over four datasets. Best performance is in bold font.

Task	Dateset	Metric	KERL$_h$	KERL$_{h+c}$	KERL$_{h+f}$	KERL
Next Item	Beauty	Hit-Ratio@10	46.5	52.2	50.9	**54.1**
		NDCG@10	31.0	34.2	33.2	**36.5**
	CD	Hit-Ratio@10	64.1	71.6	70.2	**73.7**
		NDCG@10	40.1	48.8	48.2	**50.8**
	Books	Hit-Ratio@10	74.7	78.1	77.1	**80.0**
		NDCG@10	41.3	57.0	56.4	**58.1**
	LastFM	Hit-Ratio@10	57.3	61.4	59.8	**63.7**
		NDCG@10	41.3	47.9	47.0	**50.1**
Next Session	Beauty	Hit-Ratio@10	41.3	46.5	45.0	**48.2**
		NDCG@10	24.5	30.2	29.7	**31.4**
	CD	Hit-Ratio@10	54.2	57.5	56.1	**60.5**
		NDCG@10	30.4	35.0	34.3	**36.8**
	Books	Hit-Ratio@10	51.9	56.3	55.2	**58.2**
		NDCG@10	30.6	33.7	33.1	**35.2**
	LastFM	Hit-Ratio@10	32.3	42.1	40.8	**43.1**
		NDCG@10	17.5	22.6	21.8	**24.0**

We present the comparison results in Table 2. From this table, we can observe that:

For sequential-based models, comparing with the shallow model FPMC, the deep models DREAM and GRU4Rec obtain a better performance on all evaluation metrics. The major reason is that FPMC only captures very short dependency in sequences (*i.e.,* one step), while deep models can model a longer sequence context. Besides, both DREAM and GRU4Rec adopt more powerful neural architecture than the FPMC which adopts the matrix factorization.

By incorporating KG into recommender systems, the knowledge-based methods perform better than sequential-based methods on all evaluation metrics. As will be shown latter, this observation is also consistent with our previous findings in Table 3 (*i.e.,* KERL$_c$ and KERL$_f$ perform better than KERL$_h$). Specially, based on graph neural network, KGAT is able to model higher-order connectives than other knowledge-based methods, and thus achieves a better performance.

For hybrid models (*i.e., knowledge+sequential*), both GRU$_F$ and KSR perform better than the above two classes of methods. It shows that both sequence- and knowledge-level characteristics are useful to consider for sequential recommendation. Specially, KSR adopts a knowledge-enhanced key-value memory networks for tracing both sequential and knowledge-aware user preference, and outperforms GRU$_F$ that uses simple knowledge integration in most cases.

Finally, our proposed approach KERL achieves the best performance among all the methods on four datasets. Although the baseline KSR [12] has also utilized knowledge information for enhancing sequential recommendation, it cannot predict and model future knowledge characteristics of user preference. The major novelty of KERL is that it formalizes the sequential recommendation task in a MDP setting. We incorporate knowledge information into both state representations and reward functions. With such meaningful extensions, our model is able to better utilize knowledge information for sequential recommendation.

5.3 Ablation Study

KERL makes several important extensions to fuse KG information into RL for sequential recommendation. In this section we conduct ablation study experiments to analyze their impact.

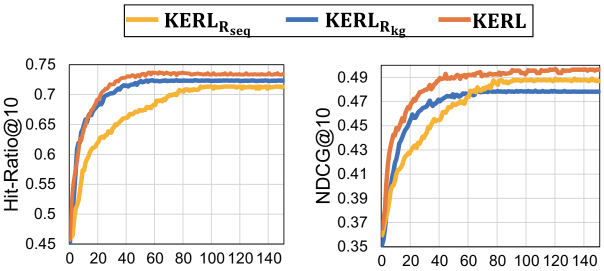

Figure 3: Performance curves of KERL and its two variants with the varying iterations on CD dataset.

5.3.1 *Analysis on Knowledge-Enhanced State Representation.*

In this part we analyze the impact of fusing KG information into the state for sequential recommendation. Recall that we have three representations in Eq. 7, namely $\mathbf{h}_t$ (Eq. 4), $\mathbf{c}_t$ (Eq. 5) and $\mathbf{f}_{t:t+k}$ (Eq. 6). Hence, we consider three variants for comparison by examining the effect of each part for sequential recommendation, including:

- *KERL$_h$* using only the sequential representation $\mathbf{h}_t$;
- *KERL$_{h+c}$* using the the sequential representation $\mathbf{h}_t$ and the current knowledge representation $\mathbf{c}_t$;
- *KERL$_{h+f}$* using the sequential representation $\mathbf{h}_t$ and the future knowledge representation $\mathbf{f}_{t:t+k}$.

The results of KERL and its three variants on four datasets are shown in Table 3. We can see that KERL$_h$ performs worst on all evaluation metrics over four datasets. It indicates the necessity of fusing KG information into recommendation. Though incorporating future knowledge can improve the recommendation performance (*e.g.,* KERL$_{h+f}$ performs better than KERL$_h$), KERL$_{h+c}$ still outperforms KERL$_{h+f}$ on all evaluation metrics. Finally, by combining the three parts, the complete model KERL outperforms its three variations in term of all evaluation measures.

5.3.2 *Analysis on Reward Function.*

In KERL, a hybrid reward function is introduced to guide the agent for constructing effective recommendation, which considers the impact of both sequence- and knowledge-level reward. In this section we analyze the effect of each individual reward on the final performance. Specifically, according to Eq. 8, we introduce two variants:

- *KERL$_{R_{seq}}$* using only the sequence-level reward;
- *KERL$_{R_{kg}}$* using only the knowledge-level reward.

Figure 3 shows the test performance curves of KERL and its two variants on CD dataset. Since similar conclusions can also be drawn on other datasets, we omit these results due to space limit. As we can see, after convergence, KERL$_{R_{seq}}$ performs better than KERL$_{R_{kg}}$ on NDCG@10 metric, while KERL$_{R_{kg}}$ shows a better performance on Hit-Ratio@10 metric. Such a difference is likely to be incurred by the nature of the two reward functions. The sequence-level reward optimizes the BLEU loss, tending to rank the correct items on higher positions, while the knowledge-level reward optimizes a knowledge-based loss, tending to recall more correct items. Combining their merits, the complete approach KERL achieves a better performance with both reward functions.

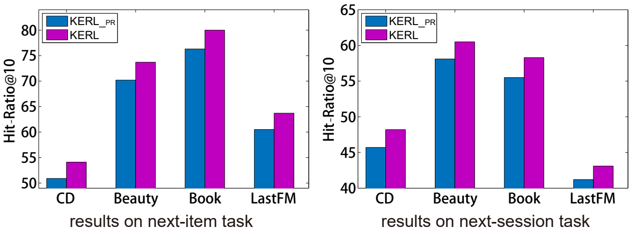

Figure 4: Performance comparison of two different learning strategies of KERL on next-item and next-session recommendation tasks.

5.3.3 Analysis on Learning Strategy for Induction Network. As shown in Table 3, the future knowledge-based representation is important to the final recommendation performance, which is derived using the induction network (Eq. 6). Recall that KERL first pretrains the induction network with the MSE loss, then pairwise ranking constraints are incorporated to improve the learning. In this part, we analyze the impact of pairwise learning strategy on KERL. Specifically, we remove the pairwise ranking constraints for learning the induction network, denoted by $KERL_{-PR}$. Figure 4 presents the performance comparison *with* and *without* the pairwise learning strategy. As we can see, KERL consistently outperforms KERL_{-PR} on four datasets with the Hit-Ratio@10 metric. It indicates the effectiveness of the proposed learning strategy for our approach.

5.3.4 Analysis on the Subsequence Length. To train the induction network, we also apply the k-step truncated learning strategy for stimulating sample sequences in Section 4.4.2. Since KERL look aheads k steps to generate sample sequences, we examine the impact of the exploration length k on the final performance. Specifically we vary the value of k in the set $k \in \{1, 3, 5, 7, 9\}$. We report the test performance of KERL in terms of Hit-Ratio@10 against the length k on the four datasets in Table 4. We have the following observations. First, as the length k increases, the test performance in terms of Hit-Ratio@10 increases accordingly. The trends are similar for four datasets. This observation demonstrates the effectiveness of fusing estimated future KG information for sequential recommendation. KERL obtains the best performance when k=3 on next-item recommendation task and k=5 on next-session recommendation task. It is intuitive that the latter task requires a longer context for estimating future preference. Finally, the performance decreases if we consider longer exploration length on two recommendation tasks. A possible reason is a long exploration window is likely to incorporate noisy information and thus affect the recommendation performance.

Table 4: The impact of k-step truncated learning strategy on Hit-Ratio@10 metric on four datasets. Best performance is in bold font.

Task	Dateset	k=1	k=3	k=5	k=7	k=9
next-item	Beauty	53.9	**54.1**	53.7	53.2	52.7
	CD	72.4	**73.8**	72.9	71.9	71.3
	Books	78.9	**80.0**	79.4	78.1	77.5
	LastFM	63.3	**64.7**	63.5	62.2	61.7
next-session	Beauty	44.7	46.0	**48.2**	46.8	46.1
	CD	56.0	58.2	**60.5**	59.1	58.7
	Books	54.3	55.8	**58.2**	57.3	56.8
	LastFM	38.9	40.4	**43.1**	42.3	41.7

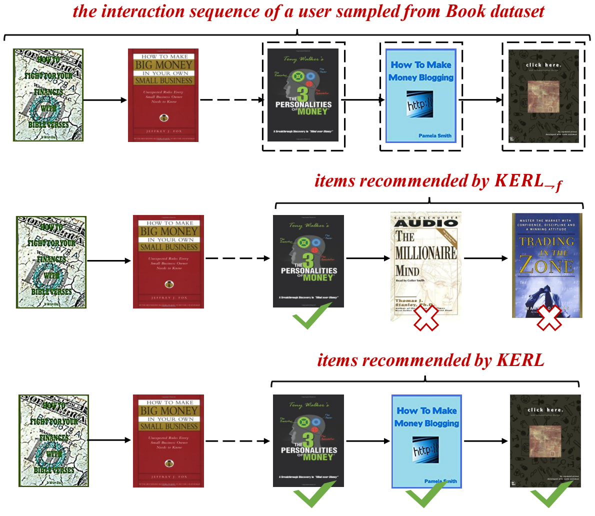
the interaction sequence of a user sampled from Book dataset
items recommended by KERL¬f
items recommended by KERL

Figure 5: Case study on next-session recommendation task. The first row is an interaction sequence (ground-truth) of a user sampled from Book dataset. The rest two rows are recommendation results of $\text{KERL}_{\neg f}$ and KERL respectively. Items in dashed boxes represent the real ones that the user has bought.

6 QUALITATIVE ANALYSIS

In our model, a major novelty is that we characterize and predict future knowledge-level preference for improving sequential recommendation. Previous experiments have shown that it is effective to improve the recommendation performance. In order to better understand why it is useful, we further construct a qualitative analysis with a case study on Book dataset in Fig. 5.

Specially, we present a snapshot of the reading sequence for a sample user. The interaction sequence is time-ordered, consisting of five books. The first three books are related to the *Finance* topic (*i.e.*, "*How to Fight for Finance*", "*Make Big Money in Business*" and "*Personalities of Money*"), while the last two books are related to the *Web design* topic (*i.e.*, "*How To Make Money Blogging*" and "*Web Design For Beginner*"). Based on this snapshot, the user is likely to be thinking of developing a website focusing on the finance topic. After inspecting into the entire reading history from the original dataset, we have found that the user is also interested in the *Technology* topic (not covered in Fig. 5 due to space limit).

For illustrating the usefulness of predicting the future preference, we prepare a new variant of our model by removing future knowledge-based preference $\mathbf{f}_{t:t+k}$ from the state representation for recommendation, denoted by $\text{KERL}_{\neg f}$. Given the first two books, we present three recommended books by both methods, respectively. It is interesting to see that $\text{KERL}_{\neg f}$ has been "trapped" in user's current preference (recommending *finance* books only), while KERL is able to successfully capture such a preference drift by making appropriate knowledge exploration (recommending both *finance* and *technology* books).

7 CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a novel knowledge-guided reinforcement learning model, called KERL, for fusing KG information into a RL framework for sequential recommendation. Specifically, we formalize the sequential recommendation task as a Markov Decision Process (MDP), and make three major technical extensions in this framework, including state representation, reward function and learning algorithm. A major novelty of our model is that KG information has been effectively utilized for both exploration and exploitation in the MDP framework. The empirical results show that our model can significantly outperform the baselines on four real-world datasets. We have also conducted detailed analysises on KERL model to illustrate the effectiveness of our extensions.

To the best of our knowledge, it is the first time that knowledge graph data has been explicitly discussed and utilized in *RL-based sequential recommenders*, especially in the exploration process. Currently, our focus lies in the utilization of knowledge information in the RL framework instead of knowledge representations. We adopt existing KG embedding methods to learn knowledge representation for items. As future work, we will consider how to adaptively learn better knowledge representation for sequential recommendation in the RL framework.

ACKNOWLEDGEMENTS

This research work was partially supported by the National Natural Science Foundation of China under Grant No. 61802029, 61872369 and 61832017, and Beijing Academy of Artificial Intelligence (BAAI) under Grant No. BAAI2020ZJ0301. This work was also supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

REFERENCES

[1] Xueying Bai, Jian Guan, and Hongning Wang. 2019. A Model-Based Reinforcement Learning with Adversarial Training for Online Recommendation. In *NeurIPS.* 10734–10745.

[2] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NIPS.* 2787–2795.

[3] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NIPS.* 2787–2795.

[4] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP.* 1724–1734.

[5] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. 2017. Sequential User-based Recurrent Neural Network Recommendations. In *RecSys.* 152–160.

[6] Yue Feng, Jun Xu, Yanyan Lan, Jiafeng Guo, Wei Zeng, and Xueqi Cheng. 2018. From Greedy Selection to Exploratory Decision-Making: Diverse Ranking with Policy-Value Networks. In *SIGIR.* 125–134.

[7] Gaole He, Junyi Li, Wayne Xin Zhao, Peiju Liu, and Ji-Rong Wen. 2020. Mining Implicit Entity Preference from User-Item Interaction Data for Knowledge Graph Completion via Adversarial Learning. In *WWW.* 740–751.

[8] Ruining He and Julian J. McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *WWW.* 507–517.

[9] Balazs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR.*

[10] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations. In *RecSys.* 241–248.

[11] Jin Huang, Zhaochun Ren, Wayne Xin Zhao, Gaole He, Ji-Rong Wen, and Daxiang Dong. 2019. Taxonomy-Aware Multi-Hop Reasoning Networks for Sequential Recommendation. In *WSDM.* 573–581.

[12] Jin Huang, Wayne Xin Zhao, Hong-Jian Dou, Ji-Rong Wen, and Edward Y. Chang. 2018. Improving Sequential Recommendation with Knowledge-Enhanced Memory Networks. In *SIGIR.* 505–514.

[13] Michael Kampffmeyer, Yinbo Chen, Xiaodan Liang, Hao Wang, Yujia Zhang, and Eric P. Xing. 2019. Rethinking Knowledge Graph Propagation for Zero-Shot Learning. In *CVPR.* 11487–11496.

[14] Wang-Cheng Kang and Julian J. McAuley. 2018. Self-Attentive Sequential Recommendation. In *ICDM.* 197–206.

[15] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer* 42, 8 (2009), 30–37.

[16] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In *CIKM.* 1419–1428.

[17] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *AAAI.* 2181–2187.

[18] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016. A Review of Relational Machine Learning for Knowledge Graphs. *Proc. IEEE* 104, 1 (2016), 11–33.

[19] Rajiv Pasricha and Julian McAuley. 2018. Translation-based factorization machines for sequential recommendation. In *RecSys.* 63–71.

[20] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. In *RecSys.* 130–137.

[21] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence Level Training with Recurrent Neural Networks. In *ICLR.*

[22] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing Personalized Markov Chains for Next-basket Recommendation. In *WWW.* 811–820.

[23] Markus Schedl. 2016. The LFM-1b Dataset for Music Retrieval and Recommendation. In *ICMR.* 103–110.

[24] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, and George van den Driessche et.al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489.

[25] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement learning - an introduction.* MIT Press, Cambridge, MA, 2018.

[26] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems. In *CIKM.* 417–426.

[27] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning Hierarchical Representation Model for NextBasket Recommendation. In *SIGIR.* 403–412.

[28] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *KDD.* 950–958.

[29] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J. Smola, and How Jing. 2017. Recurrent Recommender Networks. In *WSDM.* 495–503.

[30] Yikun Xian, Zuohui Fu, S. Muthukrishnan, Gerard de Melo, and Yongfeng Zhang. 2019. Reinforcement Knowledge Graph Reasoning for Explainable Recommendation. In *SIGIR.* 285–294.

[31] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A Dynamic Recurrent Model for Next Basket Recommendation. In *SIGIR.* 729–732.

[32] Qingheng Zhang, Zequn Sun, Wei Hu, Muhao Chen, Lingbing Guo, and Yuzhong Qu. 2019. Multi-view Knowledge Graph Embedding for Entity Alignment. In *IJCAI.* 5429–5435.

[33] Wayne Xin Zhao, Gaole He, Kunlin Yang, Hongjian Dou, Jin Huang, Siqi Ouyang, and Ji-Rong Wen. 2019. KB4Rec: A Data Set for Linking Knowledge Bases with Recommender Systems. *Data Intell.* 1, 2 (2019), 121–136.

[34] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with Negative Feedback via Pairwise Deep Reinforcement Learning. In *KDD.* 1040–1048.

[35] Lixin Zou, Long Xia, Zhuoye Ding, Jiaxing Song, Weidong Liu, and Dawei Yin. 2019. Reinforcement Learning to Optimize Long-term User Engagement in Recommender Systems. In *KDD.* 2810–2818.

[36] Lixin Zou, Long Xia, Pan Du, Zhuo Zhang, Ting Bai, Weidong Liu, Jian-Yun Nie, and Dawei Yin. 2020. Pseudo Dyna-Q: A Reinforcement Learning Framework for Interactive Recommendation. In *WSDM.* 816–824.

[37] Shihao Zou, Zhonghua Li, Mohammad Akbari, Jun Wang, and Peng Zhang. 2019. MarlRank: Multi-agent Reinforced Learning to Rank. In *CIKM.* 2073–2076.