Knowledge and Information Systems (2022) 64:2239–2265
https://doi.org/10.1007/s10115-022-01711-7

REGULAR PAPER

CDARL: a contrastive discriminator-augmented reinforcement learning framework for sequential recommendations

Zhuang Liu[1] · Yunpu Ma[2] · Marcel Hildebrandt[3] · Yuanxin Ouyang[1] · Zhang Xiong[4]

Received: 29 August 2021 / Revised: 22 June 2022 / Accepted: 26 June 2022 /
Published online: 15 July 2022

Abstract
Sequential recommendations play a crucial role in many real-world applications. Due to the sequential nature, reinforcement learning has been employed to iteratively produce recommendations based on an observed stream of user behavior. In this setting, a recommendation agent interacts with the environments (users) by sequentially recommending items (actions) to maximize users' overall long-term cumulative rewards. However, most reinforcement learning-based recommendation models only focus on extrinsic rewards based on user feedback, leading to sub-optimal policies if user-item interactions are sparse and fail to obtain the dynamic rewards based on the users' preferences. As a remedy, we propose a dynamic intrinsic reward signal integrated with a contrastive discriminator-augmented reinforcement learning framework. Concretely, our framework contains two modules: (1) a contrastive learning module is employed to learn the representation of item sequences; (2) an intrinsic reward learning function to imitate the user's internal dynamics. Furthermore, we combine static extrinsic reward and dynamic intrinsic reward to train a sequential recommender system

✉ Zhuang Liu
liuzhuang@buaa.edu.cn

Yunpu Ma
cognitive.yunpu@gmail.com

Marcel Hildebrandt
marcel.hildebrandt@siemens.com

Yuanxin Ouyang
oyyx@buaa.edu.cn

Zhang Xiong
xiongz@buaa.edu.cn

1 State Key Laboratory of Software Development Environment, Beihang University, Beijing, China

2 Ludwig-Maximilians-Universität München Lehrstuhl für Datenbanksysteme und Data Mining, München, Germany

3 University of Munich and Siemens AG, München, Germany

4 Engineering Research Center of Advanced Computer Application Technology, Ministry of Education, Beihang University, Beijing, China

based on double Q-learning. We integrate our framework with five representative sequential recommendation models. Specifically, our framework augments these recommendation models with two output layers: the supervised layer that applies cross-entropy loss to perform ranking and the other for reinforcement learning. Experimental results on two real-world datasets demonstrate that the proposed framework outperforms several sequential recommendation baselines and exploration with intrinsic reward baselines.

Keywords Sequential recommendation · Contrastive learning · Reinforcement learning · Intrinsic reward

1 Introduction

Recommender systems have been widely deployed in many web services such as e-commerce platforms, social networks, and media streaming sites. In these scenarios, users' interests hidden in their behaviors are intrinsically and evolving over time, which is difficult to make appropriate recommendations [35]. To address this problem, various methods have been proposed to make sequential recommendations (SRs) by capturing users' dynamics interests from their historical interactions (see [14, 15, 35]). Unlike conventional recommendation methods, SRs model the users' temporal preferences in a sequence and generate the next-item recommendation accordingly.

Considerable research efforts have been dedicated to SRs. One line of work represents the stream of historical user-item interactions as unidirectional sequences and encodes them via recurrent neural networks (RNNs) to predict the users' future behavior (see [13, 14, 20]). Recently, another line of research achieved state-of-the-art performance based on applying graph neural networks (GNNs) to compute SRs (see [5, 31–33]). The underlying idea of these works is to model observed user-item interactions as directed subgraphs by the timestamps in ascending order. The resulting topological patterns can in turn be encoded via the GNNs to produce accurate and expressive embeddings of the items.

Besides, SRs can also be phrased as a Markov decision process (MDP) so that reinforcement learning (RL) techniques can be employed to improve the recommendation performance. However, it is often infeasible to train a RL-based recommender system in an online fashion because one would need to expose real users to unrefined recommendations during the training process [36]. As a result, learning the policy from logged implicit feedback is of vital importance. Xin et al. [36] propose a self-supervised RL framework with reward-driven properties. Graph convolutional Q-network (GCQN) [19] is a graph-based RL recommendation model, where an RL agent aims to obtain more refined representations for states and actions based on exploiting graph structures. Generally speaking, in RL-based recommender systems, the recommendation agent interacts with the environments (users) by sequentially recommending items (actions) to maximize the users' overall long-term cumulative rewards. In most scenarios, these rewards are artificially given based on user feedbacks. For example, if a user u interacts with a recommended item, then the agent receives a positive reward [19]. In [36], to distinguish the different behaviors of users (click or purchase), the authors define the click reward r_c and the purchase reward r_p, respectively. Here, we take these artificially designed rewards as extrinsic rewards.

However, these extrinsic rewards suffer from the following three problems: Firstly, in real-world recommendation scenarios, extrinsic rewards based on observed interactions are sparse. Hence, even though a set of actions may correspond to helpful recommendations, it is difficult to identify them and assign positive rewards because of a lag of observed user feedback. Second, due to the static nature of extrinsic rewards, the environment is not guaranteed to have internal dynamics that reward the agent for recommendations corresponding with the users' preference. Third, extrinsic rewards do not incorporate the need for diverse recommendations and fail to enforce a policy that explores the state space during training. Therefore, we argue that only considering extrinsic rewards in a RL setting may be insufficient and lead to sub-optimal results.

To tackle the aforementioned problems, we develop a contrastive discriminator-augmented reinforcement learning (CDARL) framework based on a dynamic reward function that incorporates the principles of contrastive learning. The proposed framework can be easily integrated into existing recommendation models. Specifically, given a user's historical interaction sequence, an encoder is deployed to compute a low-dimensional vector representation of the sequence. We consider this to be the state representation $\mathbf{s}_t$ at time t and augment the recommendation model with two final output layers. One of them is the supervised layer trained with a cross-entropy loss to compute an optimal ranking of the items. The second layer is trained with RL, which acts as a regularizer to improve the recommendation performance.

To fit the parameters of the RL module, several item sequence augmentation methods are applied to each training sequence to generate a set of perturbation sequences (also known as views). Here, we randomly perturb the sub-sequence twice, leading to two perturbed versions of the original sequence. Subsequently, we also apply the same recommendation model to produce state representations $\mathbf{s}_t'$ and $\mathbf{s}_t''$. Moreover, we leverage a contrastive loss to maximize their similarity. In this way, it can infer accurate user representations and select appealing items for each user. In order to obtain the intrinsic reward, we need to measure the distribution of the environmental states. Here, we apply a transition encoder to transfer the current state to the next state. Next, we train a discriminator to distinguish between the predicted next state and the real next state. In this way, we can obtain the current user state. In order to encourage the recommendation agent to explore novel states, we consider the contrastive loss and the discriminator's classification loss as intrinsic rewards. That means frequently visited states are accompanied by low intrinsic rewards, imposing more attention on exploring novel states. We verify the effectiveness of CDARL by integrating it into four state-of-the-art sequential recommendation models.

To summarize, our main contributions are as follows:

- We propose a contrastive discriminator-augmented reinforcement learning framework, which can be integrated into several state-of-the-art models for sequential recommendations.
- We design a novel, intrinsic reward learning function that aims to imitate the user's internal dynamics. We show that this module leads to a performance boost of about 5%.
- We conduct an extensive, empirical study on two public datasets. The results demonstrate that our framework outperforms the existing state-of-the-art recommendation methods and exploration methods by a significant margin.

The remaining of this paper is organized as follows. We briefly review related work in Sect. 2. The preliminary concepts are introduced in Sect. 3. In Sect. 4, we present the proposed CDARL model in detail. The results of the experiments are analyzed in Sect. 5. Finally, we conclude the paper and list some future works in Sect. 6.

2 Related work

In this section, we briefly review the related works in the area of reinforcement learning for recommender systems, contrastive learning, and exploration with intrinsic rewards.

Reinforcement learning for recommender systems. In recent years, reinforcement learning has been widely examined in the context of recommender systems. In particular, Q-learning techniques were successfully integrated into plenty of recommender systems [19, 36, 40]. Most of the approaches use function approximators (e.g., a neural network) that compute approximations of the state-action value function. In turn, a policy is derived by selecting the actions (i.e., recommendations) that maximize the value function. Besides, policy-based optimization is another strategy for RL-based recommendation, which directly fits a policy function that maximizes the expected rewards. Chen et al. [2] propose a policy-gradient-based recommendation algorithm to address data biases in learning from logged feedback collected from multiple behavior policies. The authors later extend this work to two-stage recommender systems with a candidate generator in the first stage and a more powerful ranking tool in the second stage (see [23]). Another attempt [34] is a graph-based explainable recommendation framework based on RL, in which a policy-guided path reasoning unit is trained to provide recommended items along with the corresponding paths in the graph.

Contrastive learning. Contrastive learning aims to learn high-quality representation in a self-supervised manner. It has been widely adopted in computer vision, natural language processing, and recommender systems. In the computer vision community, self-supervised visual representation learning frameworks have been proposed, including SimCLR [3, 4] and MoCo [6, 11], which use image augmentation or momentum updated memory bank to learn the embeddings. When it comes to the area of natural language processing, many works aim to acquire universal word representation via self-supervised techniques [7, 24]. In the recommender system field, contrastive learning is used to improve the performance of recommendations [41]. CP4Rec [35] is a contrastive pre-training framework for sequential recommendation, which is equipped with the contrastive learning framework to extract self-supervised signals just from raw data and utilizes them to pre-train the recommendation model. In addition, contrastive methods have also been developed for graph representation [10, 21, 26] and sequence learning [12, 28].

Exploration in RL. Exploration is a crucial aspect of finding optimal RL policies. To face the challenges of exploration, several works based on extrinsic rewards and intrinsic rewards have been proposed. Extrinsic rewards are artificially given based on user feedback (see [19, 36]). They define different user behaviors (click or purchase) as different rewards and then maximize users' overall cumulative rewards. While intrinsic rewards constitute a promising solution to provide qualitative guidance for state exploration. For example, the intrinsic curiosity module (ICM) is an effective exploration algorithm for RL [1, 25]. Thereby, curiosity is defined as the error in an agent's ability to predict the consequence of its actions. Besides, EMI [16] aims to extract predictive signals that can be used to guide exploration based on mutual information. In [17], empowerment is regarded as an intrinsic reward signal to enable the agent to explore by maximizing its influence over the near future. Although intrinsic rewards have been studied in several RL literature, there is still little work on intrinsic rewards applied to RL-based recommender systems.

3 Preliminaries

In this section, we introduce a set of preliminary concepts that will be used in this paper, including the formulation of the SR problem and the corresponding RL setup.

3.1 Sequential recommendation

Let $\mathcal{I}$ denote the set of items. Hence, $|\mathcal{I}|$ indicates the number of items. A (chronologically ordered) user-item interaction sequence can be represented as $x_{1:t} = \{x_1, x_2, \ldots, x_{t-1}, x_t\}$, where $x_i \in \mathcal{I}(0 < i \leq t)$ is the i-th interacted item, its embedding is defined as $\mathbf{e}_{x_i}$. The task of making sequential recommendations consists of predicting the next item x_{t+1} that a user is likely to interact with at the $t+1$-th step given the sequence of previous interactions $x_{1:t}$.

3.2 RL for recommendation

In terms of RL, we model the sequential recommendation problem as a MDP. Thereby, the recommender agent interacts with the environment $\mathcal{E}$ (users) by sequentially recommending items (actions) with the goal to maximize the cumulative reward. More formally, this leads to a MDP consisting of the quintuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ defined as follows:

- **State space** $\mathcal{S}$: A state $s_t \in \mathcal{S}$ is defined as the historical interactions of a user before time t. For our purpose, the state of a user can be represented by the internal state of a sequential model G (e.g., $\mathbf{s}_t = G(x_{1:t})$).
- **Action space** $\mathcal{A}$: An action $a_t \in \mathcal{A}$ consists of recommending a set of items to a user at time t. In this work, we focus on offline data so that we can get the action at time t from the user-item interaction, e.g., $a_t = x_{t+1}$. Thus, we assume that the recommender agent only recommends one item to the user at each time step.
- **Transition probability** $\mathcal{P}$: The transition probability $\mathcal{P} : (\mathbf{s}_{t+1}, \mathbf{s}_t, a_t) \mapsto p(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$ indicates the probability of transitioning from state $\mathbf{s}_t$ to $\mathbf{s}_{t+1}$ when the recommender agent performs the action a_t.
- **Reward** $\mathcal{R}$: After the recommender agent performed an action a_t given the state $\mathbf{s}_t$, it receives an immediate reward given by $\mathcal{R} : (\mathbf{s}_t, a_t) \mapsto r(\mathbf{s}_t, a_t)$ according to the user's feedback (e.g., via a click or purchase).
- **Discount factor** γ: The parameter $\gamma \in [0, 1]$ denotes the discount factor for future rewards. It is noteworthy that when $\gamma = 0$, the recommender agent only considers the immediate reward. $\gamma = 1$ implies that all future rewards are weighted equally.

With the notations and definitions above, the reinforcement learning objective corresponds to the maximization of the cumulative expected rewards

$$\max_{\pi_\theta} \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{|\tau|} \gamma^t r(\mathbf{s}_t, a_t) \right], \tag{1}$$

where π_θ is the target policy which maps a state $\mathbf{s}_t \in \mathcal{S}$ to an action distribution, θ denotes all trainable parameters of the policy, and $\tau = (\mathbf{s}_0, a_0, \mathbf{s}_1, \ldots)$ is the trajectory of states and actions sampled from the policy π_θ and the transition probability $\mathcal{P}$, respectively.

4 Methodology

In this section, we first give an overview of the proposed framework CDARL. Then, we elaborate each module component in detail and finally discuss how to train the model parameters.

4.1 An overview of the proposed framework

The architecture of the proposed CDARL framework is shown in Fig. 1. Given a user's historical interaction sequence, a sequential model G produces a state representation $\mathbf{s}_t$. The representation is passed to the decoder model, which produces recommendation scores for each item. A sequence augmentation module is applied to generate two perturbed sequences, leading to perturbed state encodings produced by G. We leverage a contrastive loss to distinguish whether the two representations are derived from the same user historical sequence. To achieve this target, we maximize the similarity between the encodings of the modified views derived from the same sequence. Subsequently, we combine the two perturbed states with actions to produce two predicted next states for the perturbed sequences. In order to obtain a dynamic intrinsic reward signal, we utilize a discriminator and employ a classification loss to discriminate between the real next state and the predicted next state. After that, a linear combination of the contrastive loss and the discriminator's classification loss is formed, serving as intrinsic rewards to imitate the users' internal dynamics, encouraging the recommendation agent to explore novel states. Finally, we combine extrinsic and intrinsic rewards and employ temporal difference (TD) learning to update the policy. As we show later in the experimental section, the proposed framework can be easily integrated into existing sequential recommendation models.

4.2 Contrastive representation learning for sequences

Let $\mathbf{M}_I \in \mathbb{R}^{|\mathcal{I}|\times d}$ denote the item embedding matrix, where d is the embedding dimension. It maps the high-dimensional one-hot representation of an item to a low-dimensional dense representation. Given an item sequence $x_{1:t}$, we apply a look-up operation from $\mathbf{M}_I$ to form the input embedding matrix $\mathbf{E}_I \in \mathbb{R}^{t\times d}$. Here, G denotes a sequential recommendation model that encodes the input sequence into a latent representation $\mathbf{s}_t$. Stacking a decoder f on top of G allows to obtain a vector of recommendation scores $f \circ G(x_{1:t}) = (y_1, \ldots, y_{|\mathcal{I}|})^{\top}$, where $\circ$ denotes the composition of two functions. Then, the cross-entropy loss is deployed

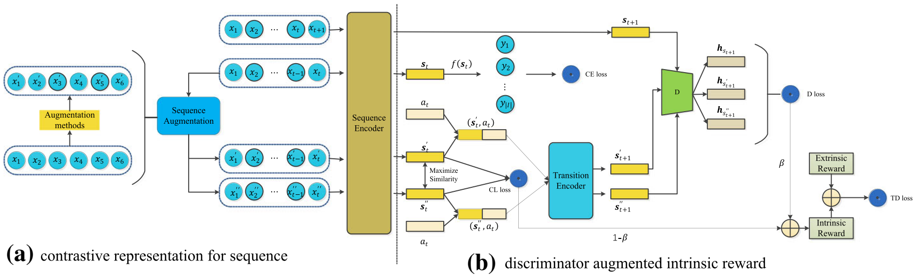

Fig. 1 The architecture of the proposed CDARL framework. Overall, our framework contains two modules: **a** contrastive learning is employed to learn the representation of sequences. **b** A discriminator-augmented intrinsic reward learning function is designed for RL-based sequential recommendations

to learn the ranking task

$$L_{CE} = -\sum_{i=1}^{|\mathcal{I}|} Y_i \log(p_i), \text{ where } \quad p_i = \frac{e^{y_i}}{\sum_{j=1}^{|\mathcal{I}|} e^{y_j}}. \tag{2}$$

Y_i is an indicator function with $Y_i = 1$ if the user interacted with the i-th item in the next time step and $Y_i = 0$, otherwise.

To obtain a better representation for each item sequence, we consider applying the contrastive learning method illustrated in Fig. 1(a). In this paper, we mainly focus on the following augmentation approaches, which can construct different views of the same item sequence.

Item embedding mask. For each item x_i in the sequence $X = [x_1, x_2, \ldots, x_t]$, we randomly mask some dimensions of its embedding $\mathbf{e}_{x_i}$. Here, we change some dimensions of $\mathbf{e}_{x_i}$ to 0 with probability p_{mask}.

Item sequence crop. For each item sequence X, we randomly select a continuous sub-sequence with the sequence length $l_{crop} = \lfloor p_{crop} * |X| \rfloor$. More formally, we first randomly select a position c for each sequence, then intercept the continuous sub-sequence from c to the end of this sequence with a proportion p_{crop}. Therefore, this augmentation method can be formulated as:

$$X^{crop} = f_{crop}(X) = [x_c, x_{c+1}, \ldots, x_{c+l_{crop}-1}]. \tag{3}$$

Item sequence delete. For each item sequence X, we randomly delete some items in the sequence. Here, the deletion probability is set as p_{delete}, and the length of the sequence after deletion is $l_{delete} = \lfloor p_{delete} * |X| \rfloor$. Specifically, we randomly select l_{delete} locations of the sequence and delete the items at the corresponding locations. Therefore, this augmentation method can be formulated as:

$$X^{delete} = f_{delete}(X) = [x_i, x_{i+1}, \ldots, x_{l_{delete}}]. \tag{4}$$

Item sequence insert. Similar to the delete operator, we randomly insert some items (randomly sample from $\mathcal{I}$) to the item sequence X with a probability p_{insert}, and the length of the sequence after insertion is $l_{insert} = |X| + \lfloor p_{insert} * |X| \rfloor$. This augmentation method can be formulated as:

$$X^{insert} = f_{insert}(X) = [x_i, x_{i+1}, \ldots, x_{l_{insert}}]. \tag{5}$$

Item sequence reorder. According to [27, 30], the precise ordering of the interactions is often irrelevant. Therefore, we adopt item permutation for the sequence augmentation. More concretely, for each item sequence X, we first randomly choose a position c. Then we randomly shuffle the continuous sub-sequence from position c to the end with a proportion of $p_{reorder}$. We define the length of the reordered sub-sequence as $l_{reorder} = \lfloor p_{reorder} * |X| \rfloor$. Then, the continuous sub-sequence $[x_c, x_{c+1}, \ldots, x_{c+l_{reorder}-1}]$ is reordered so that we obtain $[x'_c, x'_{c+1}, \ldots, x'_{c+l_{reorder}-1}]$ via random shuffling. Thus, the sequence augmentation method can be formulated as

$$X^{reorder} = f_{reorder}(X) = [x_1, x_2, \ldots, x'_c, \ldots, x'_{c+l-1} \ldots, x_t]. \tag{6}$$

Based on these augmentation methods mentioned above, at each iteration, we randomly perturb each item sequence twice with the same augmentation method and generate two modified sequences X' and X''. The sequential model G is employed to encode the two perturbed item sequences leading to $\mathbf{s}'_t = G(X')$ and $\mathbf{s}''_t = G(X'')$. It is worth noting that we also employ different augmentation methods to generate these two perturbed sequences.

Namely, we first randomly sample one of the above augmentation methods to generate a perturbed sequence and then randomly sample another augmentation method to generate the second perturbed sequence. We define this method as **random combination**.

Finally, a contrastive loss function is applied to distinguish whether the two perturbed sequence representations are derived from the same user historical sequence. Therefore, we maximize the similarity between the representations of the modified views derived from the same user historical sequence and minimize the similarity derived from different item sequences. That lead to

$$L_{CL} = -\frac{1}{|\mathcal{D}|} \sum_{\left(\mathbf{s}_t', \mathbf{s}_t''\right) \in \mathcal{D}} \log \frac{e^{\left(\phi\left(\mathbf{s}_t', \mathbf{s}_t''\right)/\tau\right)}}{e^{\left(\phi\left(\mathbf{s}_t', \mathbf{s}_t''\right)/\tau\right)} + \sum\limits_{\mathbf{s}^- \in \mathbf{S}^-} e^{\left(\phi\left(\mathbf{s}_t', \mathbf{s}^-\right)/\tau\right)}}, \tag{7}$$

where $\mathcal{D} = \left\{\left(\mathbf{s}_t', \mathbf{s}_t''\right)\right\}$ denotes the set of perturbed states. ϕ is a similarity function. In our case, we apply the cosine similarity to measure the difference between the encodings $\phi(\mathbf{u}, \mathbf{v}) = \mathbf{u}^\top \mathbf{v}/\|\mathbf{u}\|\|\mathbf{v}\|$. $\mathbf{S}^-$ denotes the set of negative samples. Here, we treat the other $2(B-1)$ augmented states within the same minibatch as negative examples, where B is the batch size. $\tau > 0$ is the temperature parameter.

4.3 Discriminator-augmented intrinsic reward

In this subsection, we design an intrinsic reward learning function for RL-based sequential recommendations. It aims to encourage the recommendation agent to explore novel states and imitate the user's internal dynamics based on its behavior.

To ease the notation, we treat $\mathbf{s}_t'$ and $\mathbf{s}_t''$ as the states for the RL-based sequential recommendation. Recall that $a_t = x_{t+1}$ denotes the action at time t. In order to obtain an intrinsic reward learning function defined on the set of all state-action pairs, our proposed CDARL framework is based on a contrastive learning principle and a discriminator.

As illustrated in Fig. 1(b), we introduce a transition encoder $f_\theta\left(\mathbf{s}_{t+1} | \mathbf{s}_t, a_t\right)$ that transfers state $\mathbf{s}_t$ to $\mathbf{s}_{t+1}$ based on the selected action. In order to obtain the predicted next state, we first concatenate the perturbed states and actions to form two state-action pairs $\left(\mathbf{s}_t', a_t\right)$ and $(\mathbf{s}_t'', a_t)$. Then two fully connected layers are employed leading to

$$\mathbf{s}_{t+1}' = \sigma\left(\mathbf{W}^{(2)} \sigma\left(\mathbf{W}^{(1)}\left(\mathbf{s}_t' \| a_t\right)\right)\right) \tag{8}$$

$$\mathbf{s}_{t+1}'' = \sigma\left(\mathbf{W}^{(2)} \sigma\left(\mathbf{W}^{(1)}(\mathbf{s}_t'' \| a_t)\right)\right), \tag{9}$$

where σ denotes the activation function (e.g., ReLU), $\|$ is the concatenation operator, and $\mathbf{W}^{(1)}, \mathbf{W}^{(2)}$ are the trainable parameters.

Subsequently, we train a discriminator to distinguish between the real next state in the recommendation environment and the predicted next state obtained from the transition encoder. Here, the real next state $\mathbf{s}_{t+1}$ can be obtained from the recommendation model G based on the item sequence $x_{1:t+1}$, namely, $\mathbf{s}_{t+1} = G\left(x_{1:t+1}\right)$. More formally, we still apply two fully connected layers to obtain the probabilities

$$p_{true} = \sigma\left(\mathbf{W}^{(4)} \phi\left(\mathbf{W}^{(3)} \mathbf{s}_{t+1}\right)\right) \tag{10}$$

$$p'_{fake} = \sigma\left(\mathbf{W}^{(4)} \phi\left(\mathbf{W}^{(3)} \mathbf{s}_{t+1}'\right)\right) \tag{11}$$

$$p''_{fake} = \sigma\left(\mathbf{W}^{(4)}\phi\left(\mathbf{W}^{(3)}\mathbf{s}''_{t+1}\right)\right), \tag{12}$$

where σ (e.g., sigmoid) and ϕ (e.g., ReLU) are activation functions. $\mathbf{W}^{(3)}$ and $\mathbf{W}^{(4)}$ are the trainable parameters. Then, the loss function of the discriminator is given by

$$L_D = (-\log p_{true} - \log(1 - p'_{fake})) + (-\log p_{true} - \log(1 - p''_{fake})). \tag{13}$$

In order to encourage the recommendation agent to explore novel states, we treat the combination of the contrastive learning loss and the discriminator's classification loss as intrinsic rewards

$$r_i = (1 - \beta)L_{CL} + \beta L_D, \tag{14}$$

where $\beta \in [0, 1]$ determines the weight assigned to the two parts of the intrinsic reward. Thus, the total reward for the recommendation agent can be defined as

$$r = r_e + r_i, \tag{15}$$

where r_e denotes extrinsic reward (e.g., 0.2 for click reward and 1.0 for purchase reward) and r_i denotes the intrinsic reward defined above.

For the training of the RL component and consistency with [36], we adopt double Q-learning [29] which is more stable and robust in the off-policy setting. We apply a fully connected layer on the top of recommendation model G to calculate the Q-values

$$Q(\mathbf{s}_t, a_t) = \delta\left(\mathbf{s}_t\mathbf{h}_t^T + b\right), \tag{16}$$

where δ denotes the activation function, $\mathbf{h}_t$ and b are the trainable parameters. Finally, we utilize the one-step TD loss to train the double Q-learning model

$$L_{TD} = \left(r + \gamma \max_{a'} Q'\left(\mathbf{s}_{t+1}, a'\right) - Q(\mathbf{s}_t, a_t)\right)^2, \tag{17}$$

where Q' is another Q function given by

$$Q'(\mathbf{s}_t, a_t) = \delta\left(\mathbf{s}_t\mathbf{h}_t'^T + b'\right), \tag{18}$$

where $\mathbf{h}'_t$ and b' are the trainable parameters. Each Q function is updated with a value from the other Q function for the next state.

4.4 Model training

To effectively learn the parameters of our framework, we apply the combined loss function

$$\mathcal{L} = \lambda \cdot L_{CE} + (1 - \lambda) \cdot (L_{CL} + L_D + L_{TD}), \tag{19}$$

where L_{CE} aims to perform the ranking task, L_{CL} and L_D in Eq. (14) are employed to explore the novel state while in Eq. (19) aims to obtain refined representations for each item sequence. L_{TD} is used to train the RL part for the sequential recommendation. λ is a coefficient that balances different losses. In this paper, the ranking task can be regarded as the main task. The remaining tasks are auxiliary tasks. We employ a hyper-parameter λ to adjust the weight between the main task and auxiliary tasks. During testing, we directly utilize the trained parameters to predict the next item for each item sequence.

5 Experiments

In this section, we conduct extensive experiments on two real-world sequential recommendation datasets to evaluate the effectiveness of the CDARL framework. We aim to answer the following research questions:

- **RQ1**: Can the CDARL framework augment existing sequential recommendation algorithms, such as GRU4Rec and SASRec, etc.
- **RQ2**: How does CDARL's intrinsic reward module perform compared with other intrinsic reward methods, such as ICM and GIRIL, etc.
- **RQ3**: What is the contribution of the different designs of CDARL (e.g., different augmentation methods, the contrastive learning module, or the intrinsic reward module) to the overall performance? How does the reward change during models' learning?
- **RQ4**: How do different hyperparameter settings (e.g., discount factor γ, perturbation rate p and intrinsic reward weight β) impact the model performance?

5.1 Experimental settings

In this subsection, we mainly introduce some experimental settings, including datasets, evaluation metrics, baselines, parameter settings, etc.

5.1.1 Datasets

We conduct experiments on two datasets collected from real-world e-commerce platforms. The statistics of these datasets after preprocessing are summarized in Table 1.

RC15 is a session-based dataset collected from the RecSys Challenge 2015 (see https://recsys.acm.org/recsys15/challenge/). Each session contains a sequence of clicks and purchases. Following [36], we remove the sessions that have a length smaller than three and sample 200k sessions. We sort the interaction records in one session by the timestamps in ascending order.

RetailRocket is collected from a real-world e-commerce website (see https://www.kaggle.com/retailrocket/ecommerce-dataset). It contains sequential events corresponding to a user viewing an item's detail page or adding it to the cart. Similar to [36], we treat views as clicks and adding to the cart as purchases. We only keep the 3-core datasets and filter unpopular items that participate in less than three interactions and the sequences with length smaller than three.

For each dataset, we randomly assign 80% of the sequences to the training set and 10% to the validation set. The remaining 10% serve as test set. Following [36], we evaluate each sequence in the validation and test set one-by-one and compute the rank of the item in the

Table 1 Statistics of the datasets

Dataset	RC15	RetailRocket
#Sequences	2000000	195523
#Items	26702	70852
#Clicks	1110965	1176680
#Purchases	43946	57269

next step. The ranking is computed on the basis of the whole item set. Each experiment is repeated 5 times with a different random split, and the average performance is reported.

5.1.2 Evaluation metrics

To evaluate the recommendation performance of the CDARL framework, we employ two widely used ranking-based metrics: top-k hit ratio (HR@k) and top-k normalized discounted cumulative gain (NDCG@k). Following [36], we calculate HR and NDCG for clicks and purchases, respectively, and report HR@k and NDCG@k with $k = 5, 10, 20$. For all these metrics, the higher the value, the better the performance.

5.1.3 Baselines

To verify the effectiveness of CDARL, we focus on two aspects: First, we compare our framework with several representative sequential recommendation models. In particular, we compare the performance of these models with and without incorporating the CDARL framework, enabling a systematic performance evaluation in a controlled setting. Second, we compare our framework with several other exploration methods based on intrinsic rewards.

Thereby, we employ the following recommendation baselines:

- **GRU4Rec** [14] applies a gated recurrent unit (GRU) with a ranking based loss to session-based recommendations.
- **Caser** [27] is a convolutional neural network-based recommendation method, which captures high-order Markov chains by applying horizontal and vertical convolutional operations.
- **NItNet** [38] is a simple convolutional generative model for session-based recommendations. It enlarges the receptive field and residual connections to increase the network depth.
- **SASRec** [15] is a self-attention-based sequential model, which can capture long-term semantics.
- **GRU-SQN**, **Caser-SQN**, **NItNet-SQN**, **SASRec-SQN** [36] integrate self-supervised Q-learning (SQN) with GRU, Caser, NItNet, and SASRec, respectively, to further improve the recommendation performance.
- **CP4Rec** [35] is a contrastive pre-training framework for sequential recommendation, which is the state-of-the-art model.
- **CP4Rec-SQN** is based on CP4Rec. We integrate SQN with it to augment the recommendation performance.

Moreover, we consider the following exploration baselines:

- **ICM** [25] is an exploration algorithm for reward learning based on curiosity.
- **GIRIL** [37] is a reward learning model which generates intrinsic reward signals via a generative model. It enables the agent to do sampling-based self-supervised exploration.
- **EMI** [16] is an exploration method that extracts the predictive signal that can be used for exploration within a compact representation space.
- **DAM** [9] is a discriminator-based reward learning method. It trains a discriminative model to access the predicted next state and true next state and takes the classification loss as intrinsic rewards.

5.1.4 Parameter settings

Following [36], we treat the last 10 items before the target timestamp as the input sequences. We optimize all models using Adam [8]. Its main advantage is that the learning rate can be self-adaptive during the training phase, which eases the pain of choosing a proper learning rate. It is worth noting that we also try to use other optimizers, such as AdaGrad [22], RMSProp [18] and AdaDelta [39], but Adam performs best. The item embedding size is fixed to 64 and the batch size to 256 for all methods. The learning rate is set to 0.01 for RC15 and 0.005 for RetailRocket. We fix the extrinsic reward following [36], e.g., the click reward is set to 0.2, while the purchase reward is set to 1.0. Following the setting in [35], we apply the cosine similarity with temperature τ as the similarity function for the two datasets and τ is tuned in $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. The perturbation rate p and the weight for the intrinsic reward β is tuned in the sets $\{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$, and $\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$, respectively. The loss weight λ and discount factor γ is searched in $\{0, 0.1, \ldots, 0.9, 1\}$ and $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$, respectively. All other hyperparameters are assigned according to the settings specified in the original publications. We train all models on a single NVIDIA GeForce GTX 1080 Ti GPU.

5.2 Recommendation comparison (RQ1)

Tables 2 and 3 summarize the best results of all considered recommendation baselines on RetailRocket and RC15, respectively. Bold scores indicate the best performance. Note that we report the best performance among all the augmentation methods mentioned in Sect. 4.2 for each CDARL-based baseline and that the improvements of CDARL over other baselines are all statistically significant (i.e., the p-value ≤ 0.05 by a t test).

On the RetailRocket dataset, the proposed CDARL framework achieves consistently better performance than the corresponding recommendation baselines when predicting both clicks and purchases in terms of all evaluation metrics. Specifically, CDARL exhibits more significant performance gains when combined with GRU4Rec, Caser, and NItNet compared to SASRec and CP4Rec. For example, compared with GRU4Rec, GRU-CDARL exhibits a performance boost of around 10-12 percentage points when predicting purchases and a boost of around 6-7 percentage points when predicting clicks. GRU-CDARL also has an increase of 4-8 percentage points when compared to GRU-SQN. It is noteworthy that CP4Rec is the current state-of-the-art model, while CP4Rec-CDARL still outperforms CP4Rec and CP4Rec-SQN in terms of all evaluation metrics. Here, SQN is another reinforcement learning method for the recommendation task, which integrates a Q-learning module into a recommendation algorithm that aims to model the long-term cumulative extrinsic reward. It is apparent that SQN consistently outperforms the base model for both click and purchase recommendations. However, SQN considers only extrinsic rewards and ignores intrinsic rewards. The CDARL framework integrates extrinsic and intrinsic rewards to improve the recommendation performance further when predicting clicks and purchases. This indicates that it is not sufficient to consider only extrinsic rewards since they are artificially designed and cannot be changed with the environment. In contrast, intrinsic rewards can dynamically change based on the environment, which can imitate the user's internal dynamics and obtain better recommendation performance.

When it comes to the RC15 dataset, CDARL achieves considerable improvements when combined with the GRU4Rec and NItNet models compared to SQN. However, applied to the Caser model, CDARL is slightly worse than SQN with respect to NDCG when predicting

Table 2 Top-k recommendation performance comparison among all considered recommendation baselines on RetailRocket

Methods	Purchase						Click					
	HR			NDCG			HR			NDCG		
	@5	@10	@20	@5	@10	@20	@5	@10	@20	@5	@10	@20
GRU4Rec	0.4608	0.5107	0.5564	0.3834	0.3995	0.4111	0.2233	0.2673	0.3082	0.1735	0.1878	0.1981
GRU-SQN	0.5069	0.5589	0.5946	0.4130	0.4289	0.4392	0.2487	0.2967	0.3406	0.1939	0.2094	0.2205
GRU-CDARL	**0.5882**	**0.6371**	**0.6730**	**0.5047**	**0.5205**	**0.5296**	**0.2921**	**0.3482**	**0.4004**	**0.2277**	**0.2459**	**0.2591**
Caser	0.3491	0.3857	0.4198	0.2935	0.3053	0.3141	0.1966	0.2302	0.2628	0.1566	0.1675	0.1758
Caser-SQN	0.3674	0.4050	0.4409	0.3089	0.3210	0.3301	0.2089	0.2454	0.2803	0.1661	0.1778	0.1867
Caser-CDARL	**0.4073**	**0.4521**	**0.4822**	**0.3438**	**0.3563**	**0.3655**	**0.2153**	**0.2532**	**0.2879**	**0.1716**	**0.1838**	**0.1924**
NItNet	0.5630	0.6127	0.6477	0.4630	0.4792	0.4881	0.2495	0.2990	0.3419	0.1906	0.2067	0.2175
NItNet-SQN	0.5895	0.6403	0.6766	0.4860	0.5026	0.5118	0.2610	0.3129	0.3586	0.1982	0.2150	0.2266
NItNet-CDARL	**0.6305**	**0.6721**	**0.7020**	**0.5336**	**0.5472**	**0.5551**	**0.2892**	**0.3437**	**0.3922**	**0.2254**	**0.2432**	**0.2553**
SASRec	0.5267	0.5916	0.6341	0.4298	0.4510	0.4618	0.2541	0.3085	0.3570	0.1931	0.2107	0.2230
SASRec-SQN	0.5681	0.6203	0.6619	0.4617	0.4806	0.4914	0.2761	0.3302	0.3803	0.2104	0.2279	0.2406
SASRec-CDARL	**0.5780**	**0.6315**	**0.6785**	**0.4799**	**0.4981**	**0.5088**	**0.2806**	**0.3386**	**0.3913**	**0.2145**	**0.2327**	**0.2458**
CP4Rec	0.5356	0.6005	0.6492	0.4362	0.4561	0.4669	0.2610	0.3159	0.3677	0.1986	0.2165	0.2296
CP4Rec-SQN	0.5560	0.6211	0.6655	0.4550	0.4762	0.4875	0.2750	0.3346	0.3875	0.2090	0.2283	0.2417
CP4Rec-CDARL	**0.5819**	**0.6394**	**0.6846**	**0.4790**	**0.4981**	**0.5083**	**0.2797**	**0.3375**	**0.3910**	**0.2128**	**0.2314**	**0.2448**

Table 3 Top-*k* recommendation performance comparison among all considered recommendation baselines on RC15

Methods	Purchase						Click					
	HR			NDCG			HR			NDCG		
	@5	@10	@20	@5	@10	@20	@5	@10	@20	@5	@10	@20
GRU4Rec	0.3994	0.5183	0.6067	0.2824	0.3204	0.3429	0.2876	0.3793	0.4581	0.1982	0.2279	0.2478
GRU-SQN	0.4228	0.5333	0.6233	0.3016	0.3376	0.3605	0.3020	0.3946	0.4741	0.2093	0.2394	0.2587
GRU-CDARL	**0.4726**	**0.5906**	**0.6853**	**0.3395**	**0.3779**	**0.4018**	**0.3228**	**0.4198**	**0.5022**	**0.2244**	**0.2552**	**0.2752**
Caser	0.4475	0.5559	0.6393	0.3211	0.3565	0.3775	0.2728	0.3593	0.4371	0.1896	0.2177	0.2372
Caser-SQN	0.4553	0.5637	0.6417	**0.3302**	**0.3653**	**0.3862**	0.2742	0.3613	0.4381	0.1909	0.2192	0.2386
Caser-CDARL	**0.4571**	**0.5667**	**0.6442**	0.3295	0.3648	0.3847	**0.3164**	**0.4105**	**0.4878**	**0.2199**	**0.2506**	**0.2702**
NItNet	0.3632	0.4716	0.5558	0.2547	0.2900	0.3114	0.2950	0.3885	0.4684	0.2030	0.2332	0.2535
NItNet-SQN	0.3845	0.4945	0.5766	0.2736	0.3094	0.3302	0.3091	0.4037	0.4835	0.2137	0.2442	0.2645
NItNet-CDARL	**0.4271**	**0.5316**	**0.6168**	**0.3065**	**0.3381**	**0.3593**	**0.3366**	**0.4328**	**0.5117**	**0.2345**	**0.2657**	**0.2858**
SASRec	0.4228	0.5418	0.6329	0.2938	0.3326	0.3558	0.3187	0.4164	0.4974	0.2200	0.2515	0.2720
SASRec-SQN	0.4336	0.5505	0.6442	0.3067	0.3435	0.3674	0.3272	0.4255	0.5066	**0.2263**	0.2580	0.2786
SASRec-CDARL	**0.4521**	**0.5667**	**0.6558**	**0.3172**	**0.3555**	**0.3783**	**0.3282**	**0.4274**	**0.5097**	0.2260	**0.2584**	**0.2792**
CP4Rec	0.4235	0.5463	0.6415	0.2935	0.3336	0.3579	0.3148	0.4132	0.4954	0.2169	0.2489	0.2694
CP4Rec-SQN	0.4380	0.5618	**0.6549**	0.3077	0.3480	0.3717	0.3242	0.4240	0.5067	0.2236	0.2560	0.2770
CP4Rec-CDARL	**0.4447**	**0.5642**	0.6523	**0.3114**	**0.3503**	**0.3725**	**0.3277**	**0.4279**	**0.5090**	**0.2262**	**0.2589**	**0.2795**

purchases. Concerning the SASRec model, CDARL is slightly outperformed by SQN with respect to NDCG@5 when predicting clicks. For the CP4Rec model, CDARL is slightly worse than SQN concerning HR@20 when predicting purchases. On all other tasks, CDARL achieves a better recommendation performance. This further highlights the positive effect of intrinsic rewards on the recommendation tasks.

For a more fine-grained analysis, CDARL achieves larger performance increases when predicting purchases compared to clicks. On both datasets, the number of purchases is much smaller than the number of clicks. In turn, the intrinsic rewards are better suited for enforcing exploration in the case of sparse data. In summary, the proposed CDARL framework consistently improves the base model's recommendation performance and outperforms SQN in most cases.

5.3 Exploration comparison (RQ2)

In this subsection, we analyze different exploration methods with intrinsic rewards. Table 4 summarizes the results of all exploration baselines on RC15 and RetailRocket. We choose GRU as the base model. Bold scores indicate the best performance in each column. Underlined scores mark the second best results. Note that we report the best performance among all the augmentation methods mentioned in Sect. 4.2 for CDARL. The last row for each dataset shows the relative improvements of our method compared to the strongest results, which is significant at p-value ≤ 0.05. Also, note that NON means we only consider extrinsic reward and ignore intrinsic reward, equivalent to GRU-SQN.

We observe that CDARL achieves consistently better performances than the corresponding exploration baselines in terms of all evaluation metrics. Moreover, the performance boost on RetailRocket is more significant than the one on RC15. NON achieves poor performance on the two datasets. This indicates that extrinsic rewards are insufficient to explore the novel state. ICM, GIRIL, EMI, and DAM are the most commonly used exploration methods for reinforcement learning. They have been deployed in several environments, including Atari games, 3D navigation in Unity, and real-world robotic manipulation task using Sawyer's arm.

Although these exploration methods have achieved better performances in these fields, they are not suitable for sequential recommendations. For example, on the RC15 dataset, GIRIL does not improve the recommendation performance when predicting purchases. Indeed, it even leads to lower performance. On the RetailRocket dataset, ICM, GIRIL, and EMI also lead to a performance drop in the purchase recommendations. Although these exploration methods slightly outperform GRU-SQN in some cases, these small gains are insufficient to verify their overall effectiveness. These results further show that it is necessary to study exploration methods suitable for sequential recommendation tasks carefully.

The proposed CDARL framework considers the contrastive loss and the discriminator's classification loss as intrinsic rewards to encourage exploring the novel state. Both the self-supervised learning method and the state discriminator consider the characteristics of item sequences making CDARL more suitable as an exploration method for sequential recommendation tasks. Especially on the RetailRocket dataset, CDARL leads to a performance increase of around 6–8% when predicting purchases and 3–6% when predicting clicks. Moreover, it leads to a performance increase of 16.04% in HR@5, 19.77% in NDCG@5 compared to the strongest baselines when predicting purchases, and 16.10% in HR@5, 16.71% in NDCG@5 for click prediction. The p-value of significance tests is less than 0.05, which demonstrates the improvement is significant. These results further underline the effectiveness of the proposed exploration method.

Table 4 A comparison of the different exploration baselines on RC15 and RetailRocket

Datasets	Methods	Purchase						Click					
		HR			NDCG			HR			NDCG		
		@5	@10	@20	@5	@10	@20	@5	@10	@20	@5	@10	@20
RC15	NON	0.4228	0.5333	0.6233	0.3016	0.3376	0.3605	0.3020	0.3946	0.4741	0.2093	0.2394	0.2587
	ICM	0.4329	0.5473	0.6351	0.3115	0.3478	0.3705	0.3037	0.3978	0.4778	0.2105	0.2410	0.2613
	GIRIL	0.4167	0.5283	0.6246	0.2920	0.3290	0.3534	0.3046	0.3994	0.4786	0.2109	0.2417	0.2618
	EMI	0.4320	0.5491	0.6337	0.3069	0.3449	0.3665	0.3025	0.3966	0.4764	0.2101	0.2406	0.2608
	DAM	0.4290	0.5383	0.6238	0.3046	0.3403	0.3630	0.3026	0.3946	0.4740	0.2098	0.2396	0.2596
	CDARL	**0.4726**	**0.5906**	**0.6853**	**0.3395**	**0.3779**	**0.4018**	**0.3228**	**0.4198**	**0.5022**	**0.2244**	**0.2552**	**0.2752**
	Improv.	9.17%	7.56%	7.90%	8.99%	8.65%	8.45%	5.98%	5.11%	4.93%	6.40%	5.59%	5.12%
RetailRocket	NON	0.5069	0.5589	0.5946	0.4130	0.4289	0.4392	0.2487	0.2967	0.3406	0.1939	0.2094	0.2205
	ICM	0.5010	0.5532	0.5980	0.4200	0.4369	0.4483	0.2516	0.3005	0.3470	0.1951	0.2107	0.2223
	GIRIL	0.4997	0.5521	0.5955	0.4161	0.4323	0.4436	0.2500	0.2985	0.3442	0.1934	0.2091	0.2204
	EMI	0.4978	0.5549	0.5938	0.4147	0.4309	0.4426	0.2508	0.2988	0.3448	0.1948	0.2103	0.2220
	DAM	0.5031	0.5589	0.6010	0.4214	0.4395	0.4502	0.2509	0.3009	0.3463	0.1940	0.2103	0.2218
	CDARL	**0.5882**	**0.6371**	**0.6730**	**0.5047**	**0.5205**	**0.5296**	**0.2921**	**0.3482**	**0.4004**	**0.2277**	**0.2459**	**0.2591**
	Improv.	16.04%	13.99%	11.98%	19.77%	18.43%	17.64%	16.10%	15.72%	15.39%	16.71%	16.71%	16.55%

5.4 Study of CDARL (RQ3)

To answer **RQ3**, in this subsection, we mainly focus on studying different designs of CDARL. We start by exploring different augmentation methods. And then analyze each component in our framework via an ablation study. Finally, we discuss how the reward changes during models' learning.

5.4.1 Comparison of different augmentation methods

In this subsection, we analyze how different augmentation methods impact the performance and choose GRU as the base model. Table 5 shows the performance of each augmentation method on RC15 and RetailRocket when predicting clicks and purchases, respectively. Bold scores indicate the best performance. We observe that the item embedding mask achieves poor performance on the two datasets. In contrast, the performance produced by the item sequence perturbation is consistently better than that of item embedding perturbation. This shows that item sequence perturbation can obtain better representations for item sequence and result in better recommendation performance. For a fine-grained analysis, on the RC15 dataset, random combination obtains the best performance when predicting purchases; item sequence crop and reorder results best performance when predicting clicks. This is due to the sequences of click are denser than purchase, item sequence crop and reorder can reduce the noise in the sequence to a certain extent to obtain better performance. On the RetailRocket dataset, item sequence crop leads to best performance when predicting clicks, and the random combination results in best performance when predicting purchases. In general, different augmentation methods can bring different recommendation performances. We need to choose an appropriate augmentation method for different datasets.

5.4.2 Ablation study

Since there are several distinct components in our framework, we analyze their impact on the overall performance via an ablation study. More concretely, we choose GRU as the base model and item sequence reordering as augmentation method and consider the following variants:

- **GRU** utilizes a GRU to model the input sequences and considers neither extrinsic reward nor intrinsic reward.
- **SQN**: Based on GRU, **SQN** introduces a Q-learning head that aims to model the cumulative extrinsic rewards.
- **GRU-CL** utilizes a contrastive loss to train the GRU model.
- **CDARL-C** only treats the discriminator's classification loss as intrinsic rewards neglecting the contrastive learning loss.
- **CDARL-D** considers the contrastive learning loss as intrinsic rewards but neglects the discriminator.
- **CDARL** considers all features of our proposed framework, including both extrinsic rewards and intrinsic rewards (e.g., the contrastive loss and discriminator's classification loss).

Figure 2 shows the performance of each variant on RC15 and RetailRocket when predicting clicks and purchases, respectively. It is apparent that the experimental results of these five

Table 5 A comparison of different data augmentation methods on RC15 and RetailRocket

Datasets	Methods	Purchase						Click					
		HR			NDCG			HR			NDCG		
		@5	@10	@20	@5	@10	@20	@5	@10	@20	@5	@10	@20
RC15	mask	0.4225	0.5302	0.6208	0.3055	0.3405	0.3619	0.3187	0.4111	0.4901	0.2219	0.2519	0.2720
	crop	0.4246	0.5459	0.6402	0.3067	0.3461	0.3701	**0.3228**	**0.4198**	**0.5022**	0.2228	0.2543	**0.2752**
	delete	0.4502	0.5634	0.6521	0.3170	0.3537	0.3763	0.3190	0.4155	0.4965	0.2208	0.2520	0.2725
	insert	0.4507	0.5666	0.6593	0.3219	0.3598	0.3833	0.3162	0.4110	0.4907	0.2198	0.2506	0.2708
	reorder	0.4398	0.5530	0.6436	0.3132	0.3499	0.3729	0.3226	0.4180	0.4966	**0.2244**	**0.2552**	**0.2752**
	combined	**0.4726**	**0.5906**	**0.6853**	**0.3395**	**0.3779**	**0.4018**	0.3110	0.4076	0.4923	0.2147	0.2460	0.2675
RetailRocket	mask	0.5003	0.5438	0.5812	0.4261	0.4403	0.4498	0.2568	0.3074	0.3543	0.2013	0.2177	0.2295
	crop	**0.5882**	**0.6371**	**0.6730**	**0.5047**	**0.5205**	**0.5296**	0.2877	0.3447	0.3955	0.2231	0.2416	0.2545
	delete	0.5541	0.6069	0.6579	0.4709	0.4881	0.5010	0.2867	0.3423	0.3952	0.2225	0.2405	0.2539
	insert	0.5428	0.5917	0.6288	0.4697	0.4839	0.4929	0.2818	0.3340	0.3833	0.2198	0.2366	0.2491
	reorder	0.5593	0.6059	0.6447	0.4730	0.4870	0.4975	0.2825	0.3368	0.3869	0.2179	0.2355	0.2482
	combined	0.5455	0.5978	0.6451	0.4599	0.4768	0.4889	**0.2921**	**0.3482**	**0.4004**	**0.2277**	**0.2459**	**0.2591**

mask represents embedding augmentation method; *crop*, *delete*, *insert*, and *reorder* represent item sequence augmentation methods; *combined* represents their random combination

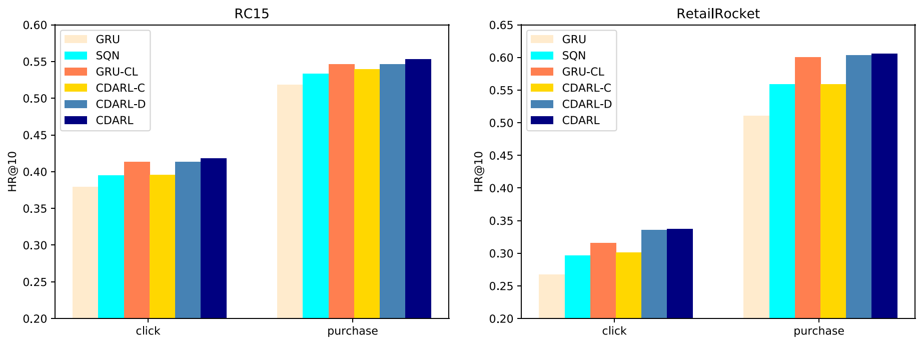

Fig. 2 An ablation study with HR@10 on RC15 and RetailRocket

variants on the two datasets show the same trend. Specifically, GRU exhibits the worst performance indicating that it does not suffice to capture item sequences based on a sequential model alone. SQN outperforms GRU, which indicates that extrinsic rewards can improve the recommendation performance to a certain extent. GRU-CL outperforms SQN further; the results show that HR@10 decreases about 7.4% than SQN on RetailRocket when predicting purchases since it is equipped with the contrastive learning framework to extract self-supervised signals from the raw data and improves the recommendation performance. In addition, the results of the latter three variants are better than SQN and GRU, which shows that intrinsic rewards further improve the recommendation performance. More concretely, CDARL-D outperforms CDARL-C, especially on the RetailRocket dataset. CDARL-D decreases about 11.6% for HR@10 when predicting clicks and 8% when predicting purchases. This indicates that contrastive learning as the intrinsic reward is better suited to explore novel states. CDARL achieves the best performance since the two intrinsic rewards parts enforce the environment's exploration when predicting both clicks and purchases.

5.4.3 In-depth analysis of reward changes w.r.t epoch

In this subsection, we make an in-depth discussion as to how the reward changes during models' learning. To enable a fair comparison, we still choose GRU as the base model and item sequence reordering as the augmentation method. Figure 3 shows how the cumulative reward and HR changes during the training process on RC15 and RetailRocket when predicting clicks and purchases, respectively. For the sake of fairness, we show the top 10 recommendation results, and the cumulative reward here refers specifically to the extrinsic reward. We analyze whether intrinsic rewards can promote the accumulation of extrinsic rewards to improve the recommendation performance.

On the RC15 dataset, whether SQN or CDARL, cumulative_reward@10 increases with training epochs and finally stabilizes. However, under the same epoch, CDARL gets more cumulative reward than SQN, which indicates that when adding intrinsic reward, the recommendation agent explores more novel states and then obtains more extrinsic reward. Similarly, HR@10 also increases gradually and stabilizes when predicting clicks and purchases, but CDARL consistently outperforms SQN. This shows that the introduction of intrinsic rewards further enhances the exploration ability of the recommendation agent and obtains better recommendation performance. Regarding the RetailRocket dataset, in the first 4 epochs, SQN achieves more cumulative rewards and recommendation performance than CDARL. However, with the increase of training epochs, CDARL gradually outperformed SQN and finally

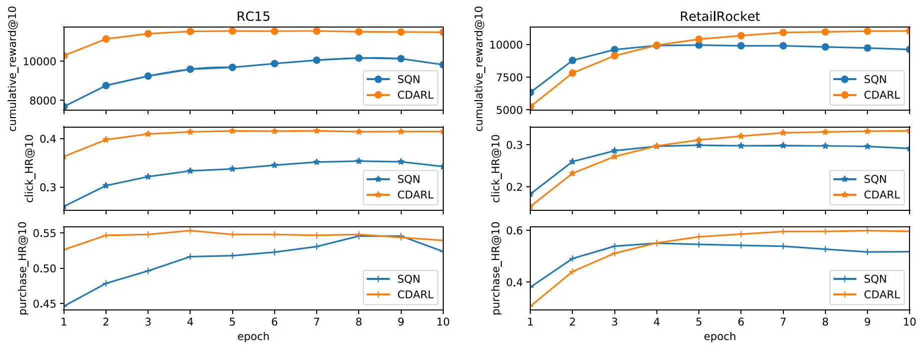

Fig. 3 In-depth analysis of reward changes with HR@10 on RC15 and RetailRocket

stabilized. One possible reason is that the click and purchase sequences of the RetailRocket dataset are denser than that of RC15. In the early training process, the extrinsic reward is enough for the recommendation agent to explore the users' state. The addition of the intrinsic reward has a negative effect. However, as training epochs increase, it is harder to explore novel states with extrinsic reward alone. At this point, the intrinsic rewards come into play, driving the recommendation model to achieve better performance. In general, the introduction of intrinsic rewards has different performances on different datasets but will ultimately play a positive role in improving the recommendation performance.

5.5 Parameter sensitivity analysis (RQ4)

In this subsection, we investigate how the different hyperparameters (e.g., the discount factor γ, the item sequence perturbation rate p, and the intrinsic reward weight β) affect the performance of CDARL. We analyze one hyper-parameter at a time by fixing the remaining hyper-parameters at their optimal value. As above, we select GRU as the base model and item sequence reordering as the augmentation method.

5.5.1 Impact of the discount factor γ

Figure 4 illustrates the HR@10 and NDCG@10 of GRU-CDARL with different discount factors on RC15 and RetailRocket datasets. We can see that the recommendation performance of GRU-CDARL improves when the discount factor γ increases from 0 to 0.6. If we continue increasing γ from 0.6 to 1.0, the recommendation performance will decrease sharply. When $\gamma = 0$, the recommender agent only considers the immediate reward but ignores the long-term reward, which results in a slightly worse performance. When $\gamma = 1$, the performance decreases suddenly. This is due to the average sequence length of the two datasets being only 6. If we consider the long-term reward excessively, it will introduce some noise, resulting in the degradation of the recommendation performance.

5.5.2 Impact of the perturbation rate p

Figure 5 shows the HR@10 and NDCG@10 for GRU-CDARL while the perturbation rate p varies from 0.5 to 1.0. We can see that on the RC15 dataset, when predicting clicks, as p

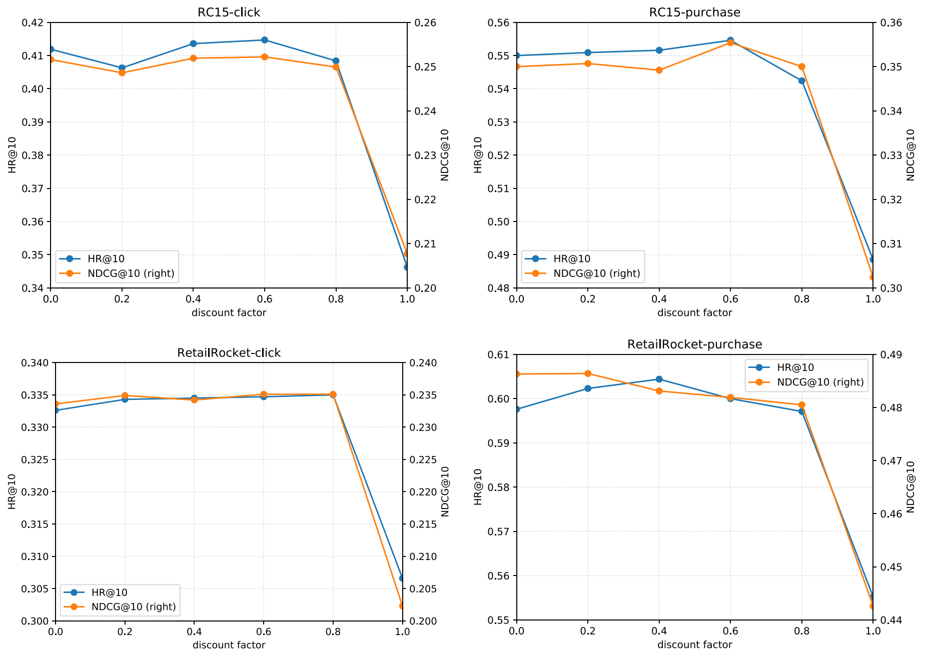

Fig. 4 The performance (with respect to HR@10 and NDCG@10) of GRU-CDARL obtained with different discount factors γ on RC15 and RetailRocket when predicting clicks and purchases

increases, HR@10 and NDCG@10 fluctuate in a small range before dropping sharply. This sudden drop is due to the effect that overly excessive perturbations will destroy the temporal structure of the original item sequence. When $p = 0.6$, HR@10 and NDCG@10 reach the optimal value. When predicting purchases, as p increases, HR@10 and NDCG@10 begin to rise after a slight decrease. When $p = 1$, GRU-CDARL achieves the best performance since the purchase sequences are relatively sparse in the user-item interactions. Thus, even significant perturbations have little impact on them and can improve the recommendation performance. On the RetailRocket dataset, with increasing values of p, HR@10 and NDCG@10 both decrease when predicting clicks and purchases since the RetailRocket dataset is denser than RC15 and the sequence orders have a more significant impact on the recommendation performance. When we set p to 0.5, CDARL achieves the best performance.

5.5.3 Impact of the intrinsic reward weight β

To study the impact of the intrinsic reward weight β, we tune the parameter β in the interval [0, 1]. Figure 6 shows the performance with respect to HR@10 and NDCG@10 for GRU-CDARL on the two datasets when predicting clicks and purchases. Recall, $\beta = 0$ implies that we only treat the contrastive learning loss as intrinsic rewards to explore the novel states. For $\beta = 1$, we only treat the discriminator's classification loss as intrinsic rewards.

It can be seen that HR@10 and NDCG@10 have the same trend at different intrinsic reward weights. On the RC15 dataset, HR@10 and NDCG@10 fluctuate in a small range when predicting clicks. For $\beta = 1.0$, both HR@10 and NDCG@10 reach the optimal value showing that the discriminator's classification loss has a greater impact on exploring the novel states for the click sequences. When predicting purchases, HR@10 and NDCG@10 exhibit

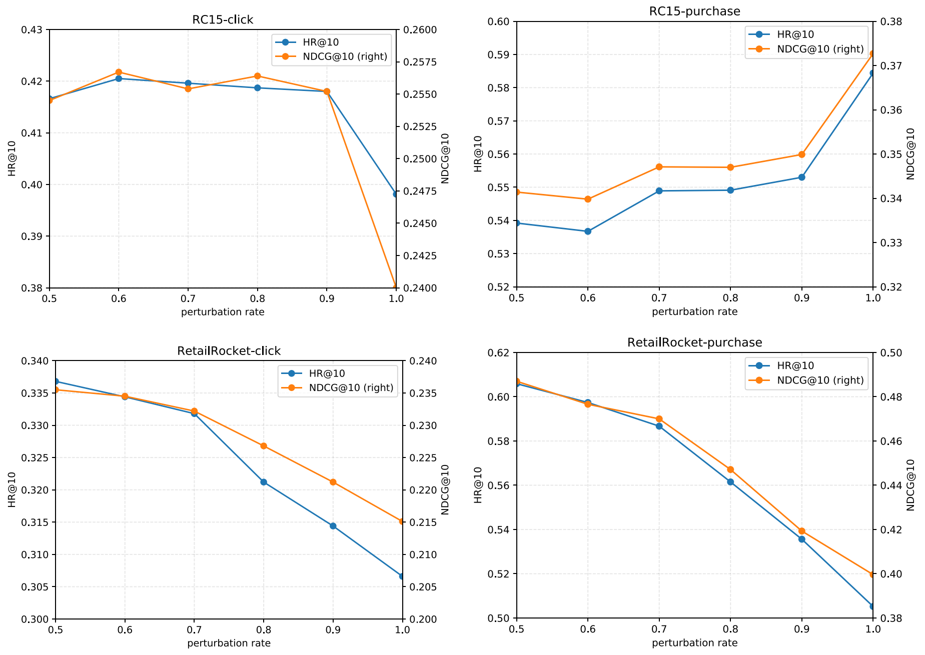

Fig. 5 The performance (with respect to HR@10 and NDCG@10) of GRU-CDARL obtained with different perturbation rates p on RC15 and RetailRocket when predicting clicks and purchases

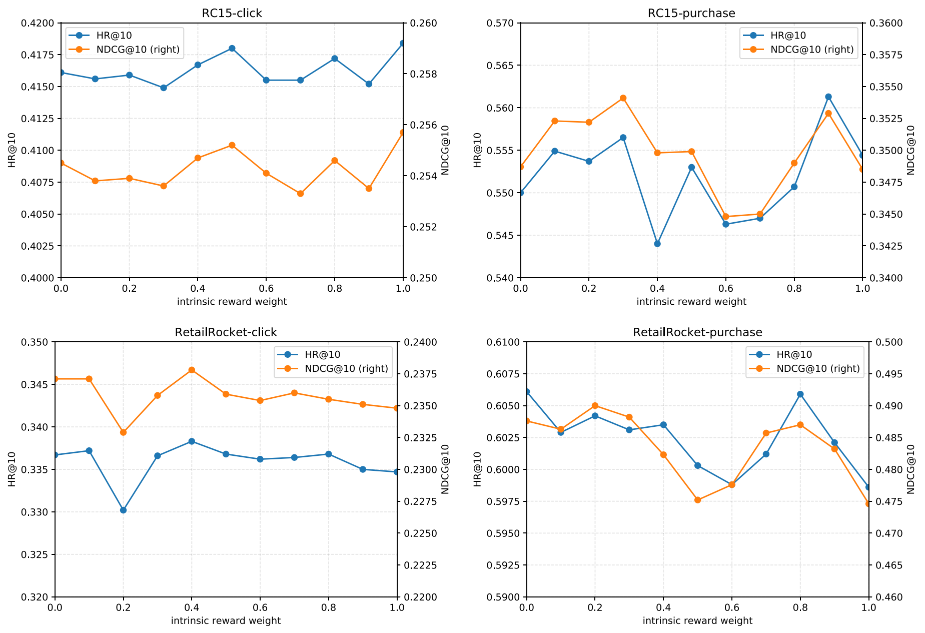

Fig. 6 The performance (with respect to HR@10 and NDCG@10) of GRU-CDARL obtained with different intrinsic reward weight β on RC15 and RetailRocket when predicting clicks and purchases

relatively large changes. When $\beta = 0.9$, HR@10 achieves the optimal value of 0.5613. For $\beta = 0.3$, NDCG@10 achieves the optimal value of 0.3541. This indicates that the exploration of the purchase sequences needs to combine different intrinsic rewards to achieve the best recommendation performance. For the RetailRocket dataset, there are similar findings as for the RC15 dataset. When predicting clicks, we have that $\beta = 0.4$ leads to optimal values with respect to HR@10 and NDCG@10. However, when predicting purchases, we set $\beta = 0$ (i.e., we only consider the contrastive learning loss as intrinsic rewards) to achieve the best HR@10. For $\beta = 0.8$, NDCG@10 reaches the optimal value. This demonstrates that distinct components of intrinsic rewards have a different impact on recommendation performance.

6 Conclusions and future work

We proposed CDARL, a contrastive discriminator-augmented reinforcement learning framework that addresses the sequential recommendation task. The underlying rationale is to construct a dynamic, intrinsic reward function that leverages contrastive learning principles. In addition, we combined extrinsic and intrinsic rewards with training a double Q-learning model to improve the recommendation performance. In order to verify the effectiveness of our framework, we integrated the CDARL framework with five representative sequential recommendation models. Thereby, we compared our framework with four exploration methods with intrinsic rewards. Experimental results on two real-world datasets demonstrate that the proposed framework leads to significant performance gains. Especially on the RetailRocket dataset, CDARL achieves consistently better performance than both recommendation and exploration baselines. This indicated that the designed intrinsic rewards were critical to sequential recommendation tasks, which can dynamically change based on the environment and imitate the user's internal dynamics. This was also the main reason for obtaining better recommendation performance. For future work, we plan to investigate whether the CDARL can be integrated into other recommendation frameworks such as graph-based or context-aware recommendation systems.

Acknowledgements This work was partially supported by the National Natural Science Foundation of China (No.61977002), the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A and the State Key Laboratory of Software Development Environment of China (No. SKLSDE-2022ZX-14). The authors of this work take full responsibilities for its content. We thank the anonymous reviewers for their insightful comments and suggestions on this paper.

Funding This research is partially funded by the National Natural Science Foundation of China (No. 61977002), the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A and the State Key Laboratory of Software Development Environment of China (No. SKLSDE-2022ZX-14).

Availability of data and materials The datasets can be found in https://recsys.acm.org/recsys15/challenge/ and https://www.kaggle.com/retailrocket/ecommerce-dataset.

Declarations

Conflict of interest The authors declare that there is no conflict of interests regarding the publication of this article.

Code availability We train all models on a single NVIDIA GeForce GTX 1080 Ti GPU.

References

1. Burda Y, Edwards H, Pathak D, Storkey AJ, Darrell T, Efros AA (2019) Large-scale study of curiosity-driven learning. ICLR (poster). OpenReview.net
2. Chen M, Beutel A, Covington P, Jain S, Belletti F, Chi EH (2019) Top-k off-policy correction for a REINFORCE recommender system. In: WSDM. ACM, pp 456–464
3. Chen T, Kornblith S, Norouzi M, Hinton GE (2020) A simple framework for contrastive learning of visual representations. In: ICML, vol 119. PMLR, pp 1597–1607
4. Chen T, Kornblith S, Swersky K, Norouzi M, Hinton GE (2020) Big self-supervised models are strong semi-supervised learners. In: Neurips
5. Chen T, Wong RC (2020) Handling information loss of graph neural networks for session-based recommendation. In: KDD. ACM, pp 1172–1180
6. Chen X, Fan H, Girshick RB, He K (2020) Improved baselines with momentum contrastive learning. CoRR, arXiv:2003.04297
7. Devlin J, Chang M, Lee K, Toutanova K (2019) BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT (1). Association for Computational Linguistics, pp 4171–4186
8. Glorot X, Bengio Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In: AISTATS, vol 9. JMLR.org, pp 249–256
9. Haghgoo B, Zhou A, Sharma A, Finn C (2021) Discriminator augmented model-based reinforcement learning. CoRR, arXiv:2103.12999
10. Hassani K, Ahmadi AHK (2020) Contrastive multi-view representation learning on graphs. In: ICML, vol 119. PMLR, pp 4116–4126
11. He K, Fan H, Wu Y, Xie S, Girshick RB (2020) Momentum contrast for unsupervised visual representation learning. In: CVPR. IEEE, pp 9726–9735
12. Hénaff OJ (2020) Data-efficient image recognition with contrastive predictive coding. In: ICML, vol 119. PMLR, pp 4182–4192
13. Hidasi B, Karatzoglou A (2018) Recurrent neural networks with top-k gains for session-based recommendations. In: CIKM. ACM, pp 843–852
14. Hidasi B, Karatzoglou A, Baltrunas L, Tikk D (2016) Session-based recommendations with recurrent neural networks. In: ICLR (poster)
15. Kang W, McAuley JJ (2018) Self-attentive sequential recommendation. In: ICDM. IEEE Computer Society, pp 197–206
16. Kim H, Kim J, Jeong Y, Levine S, Song HO (2019) EMI: exploration with mutual information. In: ICML, vol 97. PMLR, pp 3360–3369
17. Kumar NM (2018) Empowerment-driven exploration using mutual information estimation. CoRR, arXiv:1810.05533
18. Kurbiel T, Khaleghian S (2017) Training of deep neural networks based on distance measures using rmsprop. arXiv preprint arXiv:1708.01911
19. Lei Y, Pei H, Yan H, Li W (2020) Reinforcement learning based recommendation with graph convolutional q-network. In: SIGIR. ACM, pp 1757–1760
20. Li J, Ren P, Chen Z, Ren Z, Lian T, Ma J (2017) Neural attentive session-based recommendation. In: CIKM. ACM, pp 1419–1428
21. Li Y, Gu C, Dullien T, Vinyals O, Kohli P (2019) Graph matching networks for learning the similarity of graph structured objects. In: ICML, vol 97. PMLR, pp 3835–3845
22. Lydia A, Francis S (2019) Adagrad—an optimizer for stochastic gradient descent. Int J Inf Comput Sci 6(5):566–568
23. Ma J, Zhao Z, Yi X, Yang J, Chen M, Tang J, Chi EH (2020) Off-policy learning in two-stage recommender systems. In: WWW. ACM/IW3C2, pp 463–473
24. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: NIPS, pp 3111–3119
25. Pathak D, Agrawal P, Efros AA, Darrell T (2017) Curiositydriven exploration by self-supervised prediction. In: ICML, vol 70. PMLR, pp 2778–2787
26. Sun F, Hoffmann J, Verma V, Tang J (2020) Infograph: unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In: ICLR. OpenReview.net
27. Tang J, Wang K (2018) Personalized top-n sequential recommendation via convolutional sequence embedding. In: WSDM. ACM, pp 565–573
28. van den Oord A, Li Y, Vinyals O (2018) Representation learning with contrastive predictive coding. CoRR, arXiv:1807.03748
29. van Hasselt H (2010) Double q-learning. In: NIPS. Curran Associates, Inc., pp 2613–2621

30. Wang S, Hu L, Wang Y, Cao L, Sheng QZ, Orgun MA (2019) Sequential recommender systems: challenges, progress and prospects. In: IJCAI. ijcai.org, pp 6332–6338
31. Wang W, Zhang W, Liu S, Liu Q, Zhang B, Lin L, Zha H (2020) Beyond clicks: modeling multi-relational item graph for session-based target behavior prediction. In: WWW. ACM/IW3C2, pp 3056–3062
32. Wang Z, Wei W, Cong G, Li X, Mao X, Qiu M (2020) Global context enhanced graph neural networks for session-based recommendation. In: SIGIR. ACM, pp 169–178
33. Wu S, Tang Y, Zhu Y, Wang L, Xie X, Tan T (2019) Session-based recommendation with graph neural networks. In: AAAI. AAAI Press, pp 346–353
34. Xian Y, Fu Z, Muthukrishnan S, de Melo G, Zhang Y (2019) Reinforcement knowledge graph reasoning for explainable recommendation. In: SIGIR. ACM, pp 285–294
35. Xie X, Sun F, Liu Z, Gao J, Ding B, Cui B (2020) Contrastive pre-training for sequential recommendation. CoRR, arXiv:2010.14395
36. Xin X, Karatzoglou A, Arapakis I, Jose JM (2020) Self-supervised reinforcement learning for recommender systems. In: SIGIR. ACM, pp 931–940
37. Yu X, Lyu Y, Tsang IW (2020) Intrinsic reward driven imitation learning via generative model. In: ICML, vol 119. PMLR, pp 10925–10935
38. Yuan F, Karatzoglou A, Arapakis I, Jose JM, He X (2019) A simple convolutional generative network for next item recommendation. In: WSDM. ACM, pp 582–590
39. Zeiler MD (2012) Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701
40. Zhao X, Zhang L, Ding Z, Xia L, Tang J, Yin D (2018) Recommendations with negative feedback via pairwise deep reinforcement learning. In: KDD. ACM, pp 1040–1048
41. Zhou C, Ma J, Zhang J, Zhou J, Yang H (2021) Contrastive learning for debiased candidate generation in large-scale recommender systems

Zhuang Liu received a master's degree from Beihang University in 2020. He is currently studying for a Ph.D. degree in the Laboratory of Engineering, Research Center of the Advanced Computer Application Technology, School of Computer Science, Beihang University. His main research interests are recommender systems and network representation learning.

Yunpu Ma received a master's degree in high-energy physics from the Ludwig-Maximilians-University of München in 2016. In the same year, he joined Siemens as a Ph.D. student of computer science. His current research is primarily centered on different topics of artificial intelligence and quantum machine learning. One topic is the cognitive perspective of sematic and episodic knowledge graphs. He is also working on quantum algorithms, causal inference, NLP, and graph-related algorithms.

Marcel Hildebrandt received a Ph.D. degree in computer science from the Ludwig-Maximilians-University of München in 2021. He is a research scientist with Siemens CT currently. His research is primarily centered on deep learning on graphs, recommender systems, and (visual) question answering.

Yuanxin Ouyang is Professor at Beihang University, China. She received Ph.D. and B.Sc. degrees from Beihang University in 2005 and 1997, respectively. Her area of research covers recommender system, data mining, social networks and service computing.

Zhang Xiong is Professor in the School of Computer Science of Engineering, Beihang University and Director of the Advanced Computer Application Research Engineering, Center of National Educational Ministry of China. He has published over 100 referred papers in international journals and conference proceedings and won a National Science and Technology Progress Award. His research interests and publications span from smart cities, knowledge management, information systems, intelligent transportation systems, etc.