Repetition and Exploration in Offline Reinforcement Learning-based Recommendations

Ming Li
University of Amsterdam
Amsterdam, The Netherlands
m.li@uva.nl

Jin Huang
University of Amsterdam
Amsterdam, The Netherlands
j.huang2@uva.nl

Maarten de Rijke
University of Amsterdam
Amsterdam, The Netherlands
m.derijke@uva.nl

ABSTRACT

Methods for reinforcement learning for recommendation (RL4Rec) have been gaining a substantial amount of attention, as they can optimize long-term user engagement. To avoid expensive online interactions with actual users, offline RL4Rec has been proposed to optimize methods based on logged user interactions. The evaluation of offline RL4Rec methods solely depends on the overall performance of the resulting recommendations, and thus may inaccurately reflect true performance. We study the evaluation of offline RL4Rec methods from a repetition-and-exploration perspective, where we separately evaluate and compare the performance of recommending relevant repeat items (*i.e.,* items that a user has previously interacted with) and exploratory items (*i.e.,* items that the user has not interacted with so far). Our experimental results reveal a significant disparity between repetition performance and exploration performance of RL4Rec methods. Furthermore, we find that the consideration of future gains sensitively affects the optimization of RL4Rec methods. Our findings regarding repetition performance and exploration performance provide valuable insights for the future evaluation and optimization of RL4Rec methods.

CCS CONCEPTS

• **Information systems → Recommender systems**; *Retrieval models and ranking*.

KEYWORDS

Recommender Systems; Reinforcement Learning for Recommendations; Evaluation

ACM Reference Format:
Ming Li, Jin Huang, and Maarten de Rijke. 2023. Repetition and Exploration in Offline Reinforcement Learning-based Recommendations. In *Proceedings of Workshop on Deep Reinforcement Learning for Information Retrieval at CIKM'23 (DRL4IR@CIKM).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
DRL4IR@CIKM, October 22, 2023, Birmingham, UK
© 2023 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

RL4Rec is increasingly attracting attention, in both academia and industry, due to its capacity to optimize long-term user engagement [1, 24, 45]. The goal of RL4Rec is to optimize a policy that can receive maximum cumulative reward over multiple sequential recommendations. Due to the disadvantages of deploying RL4Rec online, *e.g.,* it is time-consuming and hurts the user experience [14, 17, 22], *offline* RL4Rec, which includes offline *learning* and offline *evaluation*, is a more practical alternative. Offline RL4Rec methods base their learning and evaluation only on logged user interactions and do not require interactions with users [25, 37, 45, 46]. As a result, offline RL4Rec exhibits much higher stability and computational efficiency than online RL4Rec, despite the fact that offline RL4Rec has the potential for sub-optimality [3, 10].

Evaluating RL4Rec is crucial for ensuring that recommendations are accurate, valuable, and adaptable to user needs [25, 29, 37, 40, 45, 46]. The commonly used evaluation strategy is to evaluate the average performance of all recommendations generated by a RL4Rec method [25, 37, 45, 46]. This overall evaluation approach ignores the distinction between different user interaction patterns and may result in poor performance in recommending certain items, *e.g.,* those that the user never interacted with. Ekstrand et al. [9] argue that the evaluation of recommendation methods should consider the distribution of utility between and within different user groups to more deeply understand recommendation performance. We expect that the current evaluation of RL4Rec methods that solely relies on overall performance is misleading and may lead to a decline in performance when confronting actual users. Hence, there is a real need to evaluate RL4Recs from different perspectives, not just in terms of overall performance.

Previous work has found important differences between user interactions that concern so-called *repeat items*, *i.e.,* items that a user has previously interacted with, and those that concern so-called *exploratory items*, *i.e.,* items that a user has not interacted with so far [11, 19–21]. Li et al. [21] have found that there are significant disparities between repetition performance (*i.e.,* the performance of recommending repeat items) and exploration performance in (*i.e.,* the performance of recommending exploratory items) when evaluating sequential recommendation methods in real-world datasets. Recommending repeat items proves to be considerably easier and yields markedly higher accuracy than recommending exploratory items. So far, a systematic comparison of repetition performance and exploration performance has not been performed for RL4Rec. We believe it is important to perform such a comparison as it could shed light on how RL4Rec handles the challenges associated with repetition and exploration, thereby contributing to a more comprehensive understanding of its capabilities and limitations in real-world scenarios.

In this paper, we address this knowledge gap and provide a comprehensive experimental performance analysis of RL4Rec methods from the perspective of repetition and exploration. Our experimental results reveal that: (i) there is a huge imbalance between the repetition performance and exploration performance of RL4Rec methods; and (ii) the task of recommending items to users who will next purchase a repeat item is much easier than the task of recommending items to users who will next purchase an exploratory item. This indicates that relying solely on overall accuracy has its limitations in terms of accurately reflecting true performance.

Besides the evaluation of RL4Rec methods, we also expect the optimization of RL4Rec methods to be affected by different user interactions that concern repeat items and exploratory items, as they are learned from such interactions. Specifically, we investigate how a RL4Rec method is affected by the degree to which future rewards (*i.e.,* long-term reward) are considered, during its optimization with these interactions. Our experimental results suggest that placing greater emphasis on long-term reward results in an increase in the novelty in both repetition and exploratory recommendation tasks, albeit with a decrease in accuracy.

2 RELATED WORK

2.1 RL4Rec and its evaluation

Various reinforcement learning (RL) methods, including Deep Q-Network (DQN), REINFORCE, and actor-critic, are being used in RL4Rec methods to improve recommender systems (RSs), *e.g.,* optimizing long-term user engagement. Among those, DQN is the most widely used RL method, and the resulting DQN-based recommendation (DQN4Rec) methods have been widely applied in a variety of recommendation scenarios, *e.g.,* e-commerce product recommendation [41], tip recommendation [6], news recommendation [44], and recommendation mixed with advertisements [43]. Different from DQN, which is a value-based method and estimates Q-values to find the best actions, REINFORCE is a policy gradient method that directly optimizes the recommendation policy for improved performance [31]. It is commonly used in conversational RSs [30] and explainable RSs [35]. Actor-critic is a hybrid RL approach that combines the elements of both value-based and policy gradient methods, allowing it to benefit from both components [23]. Actor-critic-based recommendation methods are able to handle large action spaces in RSs [8] and have been used for diverse recommendation tasks [40, 42].

The evaluation of RL4Rec methods primarily focuses solely on the overall performance of the resulting recommendations generated by these methods, which may inaccurately reflect their true performance.

2.2 Evaluation of RS

Over the years, a lot of effort has been put into the evaluation of RS using different metrics, e.g., accuracy, diversity [36], novelty [12, 15], fairness [2, 33, 34], allocation of item exposure [21]. Among them, accuracy is one of the most important evaluation metrics in recommender systems, as it represents the ability to find relevant items that meet user preferences, which is the default focus of RS. Several empirical studies investigated the accuracy evaluation from different aspects, e.g., reproducibility [13], versions of implementations [28], dataset splitting methods [38]. However, these studies mainly focus on the overall average accuracy, which fails to assess how a model performs at a more fine-grained level. Recently, Li et al. [20] proposed a set of evaluation metrics to evaluate the next-basket recommendation (NBR) performance from the perspective of repetition and exploration. They uncover an imbalance between repetition performance and exploration performance. They argue that using average accuracy is not sufficient to evaluate the recommendation performance, as it may suffer from several issues, e.g., long-tail contribution of performance, unfairness across different users, etc. Following this study, the perspective of repetition and exploration in evaluation has gotten increasing attention, and the distinction between repetition and exploration has been found in more recommendation scenarios, e.g., sequential item recommendation [21], reverse next period recommendation [19], and conversational recommendation [11]. Recently, Ekstrand et al. [9] emphasized that understanding the fine-grained metric distributions is important to provide an appropriate evaluation of recommender systems.

However, the evaluation from the repetition and exploration perspective has not been investigated for RL4Rec methods, which is the gap this paper addresses.

3 PRELIMINARIES

3.1 Offline DQN4Rec

We focus on a recommendation task in which items from the item set $\mathcal{I}$ are recommended to users from the user set $\mathcal{U}$ [29]. We follow the common RL4Rec setting where the recommendation task is modelled as a Markov decision process (MDP) [4, 5, 13, 14]:

State space $\mathcal{S}$: A state I_u stores historical item sequence of user $u \in \mathcal{U}$, denoted as $I_u = [i_1, i_2, \dots, i_t]$ with item $i_t \in \mathcal{I}$ that the user interacted at timestamp t.

Action space $\mathcal{A}$: An action is to recommend an item i_{t+1} to user u by the RS method based on state I_u.

Reward $\mathcal{R}$: The immediate reward $r(I_u, i_{t+1})$ is the user's feedback $r_{u,i_{t+1}} \in [0, 1]$. If the user clicks the recommended item i_{t+1}, $r_{u,i_{t+1}} = 1$, otherwise, $r_{u,i_{t+1}} = 0$.

Transition probability $\mathcal{P}$: After the user provides feedback $r_{u,i_{t+1}}$, the state transitions deterministically to the next state $I'_u = [i_1, i_2, \dots, i_{t+1}]$.

Discount factor γ: $\gamma \in [0, 1]$ governs the significance that the RS method attaches to future rewards: if $\gamma = 0$, only the immediate reward is considered; if $\gamma = 1$, all future rewards are equally considered.

Offline DQN4Rec methods are only learned from logged user data $\mathcal{D}$, without further interactions with users. Following Huang et al. [13], we implement a DQN4Rec method that consists of two components: a state encoder to predict state-action function $\widehat{Q}(I_u, i; \theta)$ and a DQN method to optimize the expected discounted cumulative reward.

The state encoder component. Neural networks $\mathcal{M}$, *e.g.,* transformers [32] or gated recurrent units (GRUs) [7], are widely used in state encoders to encode a state into a dense representation

$\boldsymbol{p}_u$ [13, 26]. The state-action value function $\widehat{Q}(I_u, i)$ is computed as the dot-product of the state representation $\boldsymbol{p}_u$ and the item representation $\boldsymbol{q}_i$:

$$\widehat{Q}(I_u, i) = \boldsymbol{p}_u^\top \cdot \boldsymbol{q}_i; \quad \boldsymbol{p}_u = \mathcal{M}(I_u; \theta), \tag{1}$$

where θ denotes the parameters of state representation model $\mathcal{M}$. The generated $\widehat{Q}(I_u, i)$ is applied in the following DQN component.

The DQN component. DQN implements a behavior network that is separated from the target network to ensure the stability of the training process. These networks have the same structure but are updated in different ways. In the DQN4Rec method that we implement, two networks are two state encoders that have the same structure and use the same item embeddings, but use different parameters of model $\mathcal{M}$.

Given a transition $(I_u, i_{t+1}, r_{u,i_{t+1}}, I'_u) \in \mathcal{D}$, we use the behavior network to estimate a state-action value function $\widehat{Q}(I_u, i_{t+1}; \theta)$ on state-action pairs (I_u, i_{t+1}), where θ denotes the parameters of the behavior network; and we use the target network to estimate a state-action value function $\widehat{Q}'(I'_u, i; \theta')$ for any item $i \in \mathcal{I}$ given state I'_u, where θ' denotes the parameters of the behavior network. The parameter θ in the behavior network is updated by minimizing the following loss using the Adam optimizer [16]:

$$\begin{aligned} \mathcal{L} &= \frac{1}{\mathcal{D}} \sum\nolimits_{(I_u, i_{t+1}, r_{u,i_{t+1}}, I'_u) \in \mathcal{D}} \delta^{\mathrm{TD}}, \\ \delta^{\mathrm{TD}} &= \left(r_{u,i_{t+1}} + \gamma \max_i \widehat{Q}'(I'_u, i; \theta') - \widehat{Q}(I_u, i_{t+1}; \theta) \right)^2 . \end{aligned} \tag{2}$$

Following Lillicrap et al. [23], the parameter θ' in the target network is updated by:

$$\theta' \longleftarrow \tau\theta + (1 - \tau)\theta', \tag{3}$$

where τ controls the rate at which the target network is updated with the parameter of the behavior network and is set to 0.01 [40].

3.2 Evaluation

3.2.1 Repetition and exploration. Following Li et al. [21], for each user u, we divide items into *repeat items* $i \in I_u$ (i.e., items that user u has interacted with before) and *explore items* $i \in \bar{I}_u = \mathcal{I} \setminus I_u$ (i.e., items that user u has not interacted with before). Then, given the next item that users will purchase, we can divide users into the following two types:

(i) **Repeat-next users** (RNU): Users who will purchase a repeat item in their subsequent step; and
(ii) **Explore-next users** (ENU): Users who will purchase an explore (new) item in their subsequent step.

3.2.2 Metrics. Following Li et al. [21], we adopt two widely used accuracy metrics for the sequential recommendation task, *i.e.,* Recall@K, and NDCG@K, which focus on measuring how accurate the recommendations are. Recall@K measures the proportion of relevant items recommended within the top-K items. NDCG additionally considers the position of relevant items in the recommendation list.

Apart from these accuracy-related metrics, we also measure the novelty of the recommendation, which is defined as follows:

$$\text{Novelty}_u@K = \frac{\sum r = 1^K h(u, r) \cdot \log_2(r + 1)}{\sum_{r=1}^K \log_2(r + 1)} \tag{4}$$

Table 1: Statistics of the processed datasets. RNU denotes repeat-next users; ENU denotes explore-next users. The ENU proportion can be computed as $1 -$ RNU proportion**.**

Dataset	#Items	#Users	#RNU	#ENU	RNU proportion
Diginetica	35,042	75,739	22,610	53,129	38.1%
Yoochoose	30,833	1,878,967	715,518	1,163,449	29.8%

where $h(u, r) = 1$ if the r-th item in the recommended list to user u is a new item, otherwise $h(u, r) = 0$. Explore-next users expect the recommendation with a high novelty, as they are inclined to buy new items in their subsequent purchase, while repeat-next users exhibit a contrary tendency. Note that the positions of the items in the recommendation list are also considered to measure the novelty.

In our analysis below, Metric@K denotes the overall average performance on the given metric; and Metric^{rep}@K and Metric^{expl}@K denote the repetition performance on repeat-next users and exploration performance on explore-next users, respectively.

4 EXPERIMENTS

4.1 Research questions

In this study, we address the following research questions:

(RQ1) How do RL4Rec methods perform from the repetition and exploration perspective?
(RQ2) How are RL4Rec methods affected by the degree to which long-term reward is considered during its optimization?

4.2 Experimental set up

4.2.1 Dataset. Following [21], we use two widely used datasets, *i.e.,* Diginetica and Yoochoose, with different proportions of repeat-next users to support our analysis.

Diginetica is a CIKM2016 dataset, which contains e-commerce search sessions with unique session ids. We regard each session as the historical interaction of a user.[1]

Yoochoose is a RecSys2015 dataset, which is a collection of users' sequential click events from a retailer.[2]

We adhere to the preprocessing approach employed in prior research, denoted as the "5-core" method. Items with fewer than 5 occurrences and users with interaction sequences shorter than 5 items are excluded. The user sequence length cap is set to 50, with sequences exceeding this length being truncated. To split the dataset into training, validation, and test partitions, we adopt a leave-one-out method: for every sequence, the ultimate interaction is the test label, the second-to-last interaction is the validation label, and the third-to-last interaction is the training label. A summary of the preprocessed datasets can be found in Table 1.

4.2.2 State encoder for DQN4Rec. As we have explained in Section 3.1, the DQN4Rec methods that we use in our experiments can be integrated with different neural networks. Specifically, we apply a transformer [32] and a GRU [7] to construct the state encoder for DQN4Rec, resulting in two DQN4Rec methods, *i.e.,* DQN-Transformer and DQN-GRU, respectively.

[1] https://competitions.codalab.org/competitions/11161
[2] https://www.kaggle.com/datasets/chadgostopp/recsys-challenge-2015

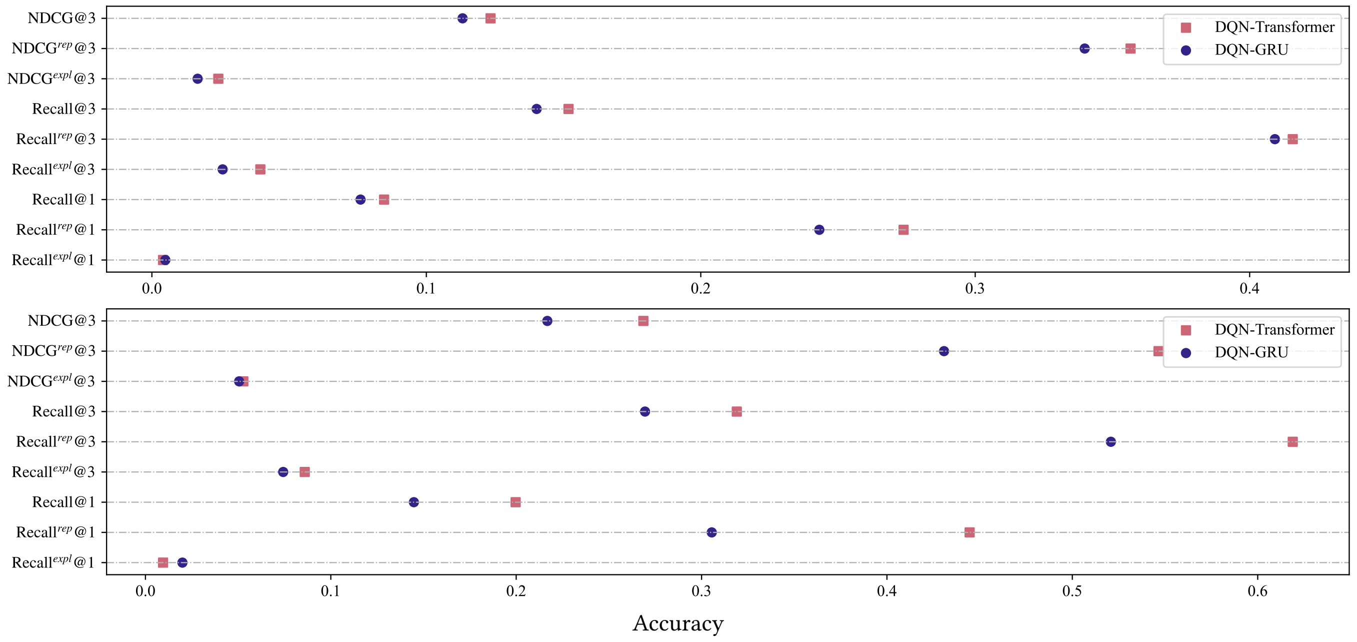

Figure 1: Recommendation accuracy for all users, repeat-next users and explore-next users on the Diginetica (top) and Yoochoose (bottom) datasets.

4.2.3 *Configuration.* We implement DQN-Transformer and DQN-GRU based on RecBole [39] using Pytorch [27]. We tune the hyper-parameters per DQN4Rec method through a grid search in the following range: the number of negative samples for each positive sample is searched from $\{1, 5, 10, 20, 50\}$, the embedding size is searched from $\{32, 64, 128\}$, and the γ is searched from $\{0, 0.1, 0.2, \ldots, 1.0\}$. τ is set to 0.01 according to [40]. The dropout ratio is set to 0.1 and the Adam optimizer [32] with a learning rate of 0.001 is employed to optimize the parameters. For the state encoder of DQN-Transformer, we use 2 transformer layers with 8 heads. For the state encoder of DQN-GRU, we set the number of GRU layers as 1. We train each model using a 12G TITAN X GPU and each experiment is repeated 5 times to make results more reliable. We will share our source code upon publication of this paper.

4.3 Experimental results

4.3.1 *Evaluation: repetition vs. exploration.* To answer RQ1, we analyze the overall performance on all users. We also analyze the fine-grained performance for repeat-next users and explore-next users separately.

Accuracy. The accuracy results are displayed in Figure 1. For all the accuracy-related metrics, the higher the value the better the performance. Several observations can be made based on Figure 1:

(1) DQN4Rec methods with different state encoders have different performance across various metrics. DQN-Transformer outperforms DQN-GRU w.r.t. overall accuracy on both datasets.
(2) Both DQN-Transformer and DQN-GRU suffer from the huge accuracy imbalance between repetition and exploration on both datasets. Both DQN4Rec methods achieve noticeably better recommendation accuracy for repeat-next users in comparison to explore-next users across various accuracy metrics. This suggests that RL4Rec methods have unfair performance w.r.t. different types of users.
(3) Compared to DQN-GRU, DQN-Transformer achieves better overall Recall@1, but worse Recall^{expl}@1, which accounts for over 70% of the user population on the Yoochoose dataset. This reveals that higher overall average performance does not necessarily translate into better performance across users.

The above observations suggest that repeat-next users contribute a significant portion of the overall average performance, although they constitute a relatively small proportion of the user population compared to explore-next users. The overall average accuracy is not sufficient to represent the performance of the RL4Rec methods, for example, optimizing methods based only on repetition could be a "shortcut" to get a good overall performance [11, 18–21].

Novelty. In general, lower novelty signifies better outcomes for repeat-next users, as it suggests that more repeat items are recommended. Conversely, higher novelty is preferable for explore-next users, as it means that more explore items are recommended. The

Table 2: The novelty of the recommendation for repeat-next users and explore-next users. The results are averages of 5 independent runs.

Dataset	Method	Novelty@1		Novelty@3	
		RNU	ENU	RNU	ENU
Diginetica	DQN-Transformer	0.189	0.222	0.548	0.582
	DQN-GRU	0.277	0.326	0.526	0.546
Yoochoose	DQN-Transformer	0.190	0.224	0.541	0.582
	DQN-GRU	0.357	0.492	0.556	0.626

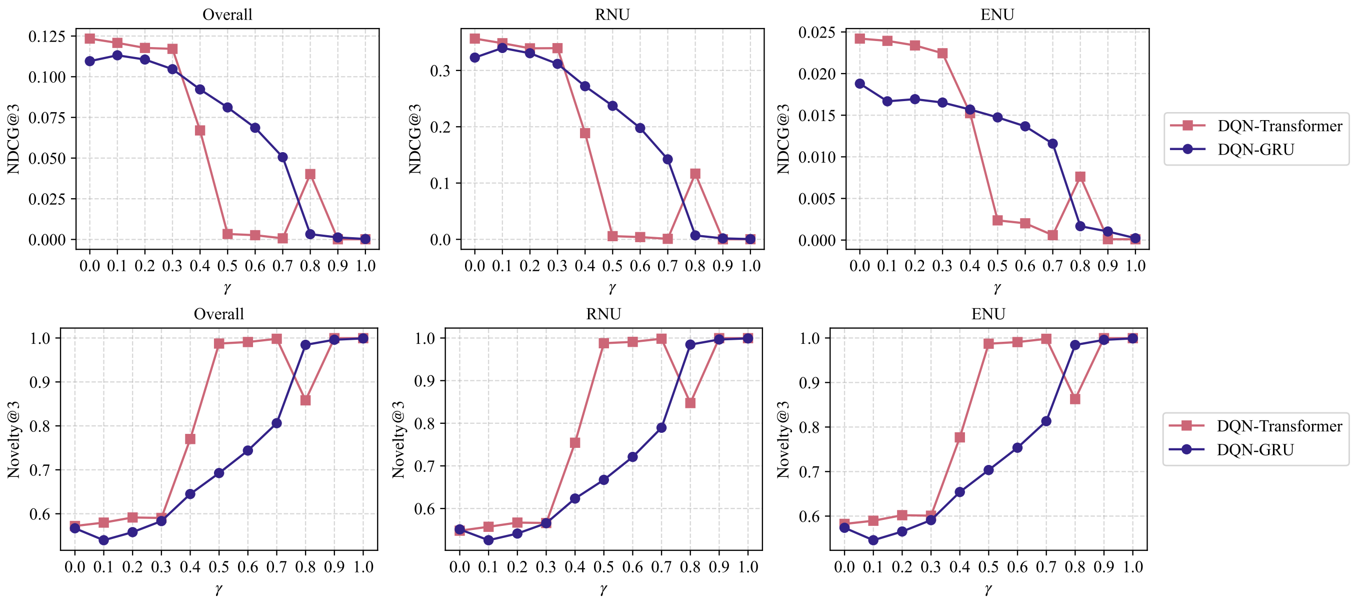

Figure 2: The accuracy (top) and novelty performance (bottom) for all users (Overall), repeat-next users (RNU) and explore-next users (ENU) with γ varying from 0 to 1 on the Diginetica dataset.

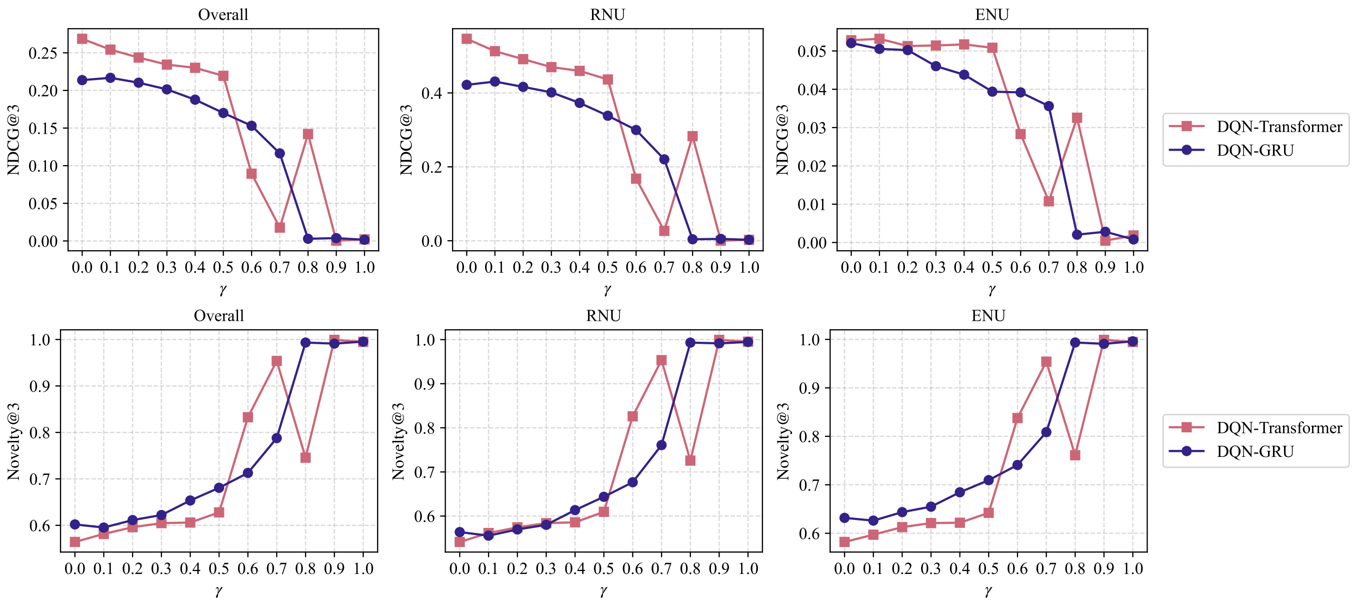

Figure 3: Accuracy (top) and novelty performance (bottom) for all users (Overall), repeat-next users (RNU) and explore-next users (ENU) with γ varying from 0 to 1 on the Yoochoose dataset.

novelty results for all users and different user sub-groups are shown in Table 2. We have the following observations:

(1) DQN4Rec with different state encoders have varying levels of novelty of the recommendation. For both repeat-next users and explore-next users, the novelty of recommendations generated by DQN-GRU is higher than those from DQN-Transformer when $K = 1$. This indicates that compared to DQN-Transformer, DQN-GRU favors the explore-next users w.r.t. the types of recommended items.

(2) On both datasets, the novelty of DQN-Transformer and DQN-GRU recommendations for explore-next users is consistently higher than for repeat-next users, however, their difference is relatively smaller w.r.t. Novelty@3, which indicates that both methods may have difficulty in classifying the repetition and exploration preferences of users.

The aforementioned findings indicate that the DQN4Rec methods with different state encoders favor different user groups (*i.e.,* repeat-next users and explore-next users). The overall average novelty is insufficient to represent the novelty performance of RL4Rec methods, as different types of users expect different novelty of the recommendation.

On both evaluation metrics, the RL4Rec methods perform significantly differently on repeat-next users and explore-next users. This highlights the importance of evaluating RL4Rec methods from the perspective of repetition and exploration.

4.3.2 Optimization: long-term vs. short-term. To answer RQ2, we investigate the influence of the optimization process on the repetition and exploration performance. Specifically, we conduct experiments with a varying discount factor γ, which controls the balance between immediate and future rewards in the model's optimization process. A higher value of γ places more importance on long-term rewards, encouraging the model to consider potential future benefits. Conversely, a lower value of γ emphasizes short-term rewards, making the model focus more on immediate gains. Similarly, we analyze this impact on the conventional overall performance, as well as the performances from the repetition and exploration perspective. Due to space limitations, we only showcase the NDCG@3 and Novelty@3 as the accuracy metric and novelty metric, respectively. The experimental results for Diginetica and Yoochoose are shown in Figure 2 and Figure 3, respectively. We have several observations, as follows:

- On both datasets for both DQN-Transformer and DQN-GRU, as the value of γ increases (i.e., as we place more importance on long-term rewards during the optimization process), the novelty of recommendation increases. One possible reason for these results is that, when the optimization focuses more on the short-term reward, the DQN model is more biased to learn the "repeat shortcut" [11, 18] to minimize the loss, which leads to recommended more repeat items (i.e., low novelty).
- As γ and novelty increase, the overall accuracy and repetition accuracy decrease. This observation is not surprising, as the model recommends more new items, (i.e., being biased to the relatively difficult task), which will result in low overall accuracy and repetition accuracy. Interestingly, we find that, as the recommendation novelty increases for explore-next users, the exploration performance does not increase, which indicates that emphasizing long-term reward does not necessarily improve the model's accuracy performance on explore-next users.
- Compared to DQN-GRU, DQN-Transformer has better performance w.r.t. the accuracy in most cases when the optimization focuses more on the short-term reward, i.e., $\gamma < 0.3$ on Diginetica dataset and $\gamma \leq 0.5$ on Yoochoose dataset. When the value of γ is large, both methods exhibit low accuracy, and in some cases, they may even encounter a collapse in the optimization. Additionally, in comparison to DQN-GRU, the DQN-Transformer approach displays higher instability and a greater susceptibility to collapsing. We suspect that using a transformer as state encoder exhibits a better ability to capture the item relations in a dynamic setting, however, due to its complexity, it is susceptible to collapse when optimization places excessive emphasis on long-term rewards.

The above findings indicate that the optimization of DQN4Rec methods in a scenario with a mix of interactions concerning repeat items and exploratory items is very sensitive to the discount factor that controls the trade-off between the long-term and short-term rewards.

5 CONCLUSION

In this paper, we have investigated the evaluation of offline RL4Rec methods from the perspective of repetition and exploration. Our experimental results on two real-world datasets exhibit a huge imbalance between repetition performance and exploration performance of the DQN4Rec methods that we considered. Specifically, these methods achieve much higher accuracy on repeat-next users than on explore-next users. This suggests that an evaluation that solely relies on the overall performance is insufficient to represent the performance of methods. Instead, a fine-grained evaluation that considers repetition and exploration could enhance the evaluation.

Furthermore, we investigated how the discount factor that controls the long-term and short-term reward trade-offs affects the performance of DQN4Rec methods from the repetition and exploration perspective. We find that emphasizing the long-term reward leads to an increase in recommendation novelty, but might lead to a decrease in accuracy in the scenario with both repetition and exploration interactions.

Our research expands on the optimization and evaluation of existing RL4Rec methods from the perspective of repetition and exploration, thus providing valuable insights into the directions of improving RL4Rec methods. Future work could improve RL4Rec methods by considering the perspective of repetition and exploratorion, *e.g.,* designing a reward function for RL4Rec that is able to distinguish between repetition and exploration.

REFERENCES

[1] M. Mehdi Afsar, Trafford Crump, and Behrouz Far. 2022. Reinforcement Learning Based Recommender Systems: A Survey. *Comput. Surveys* 55, 7 (2022), 1–38.

[2] Jesús Bobadilla, Raúl Lara-Cabrera, Ángel González-Prieto, and Fernando Ortega. 2020. Deepfair: Deep Learning for Improving Fairness in Recommender Systems. *arXiv preprint arXiv:2006.05255* (2020).

[3] Emanuele Cavenaghi, Gabriele Sottocornola, Fabio Stella, and Markus Zanker. 2023. A Systematic Study on Reproducibility of Reinforcement Learning in Recommendation Systems. *ACM Transactions on Recommender Systems* (2023).

[4] Haokun Chen, Xinyi Dai, Han Cai, Weinan Zhang, Xuejian Wang, Ruiming Tang, Yuzhou Zhang, and Yong Yu. 2019. Large-scale Interactive Recommendation with Tree-structured Policy Gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3312–3320.

[5] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. 2019. Top-k Off-policy Correction for a REINFORCE Recommender System. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. ACM, 456–464.

[6] Shi-Yong Chen, Yang Yu, Qing Da, Jun Tan, Hai-Kuan Huang, and Hai-Hong Tang. 2018. Stabilizing Reinforcement Learning in Dynamic Environment with Application to Online Recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1187–1196.

[7] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder-decoder Approaches. *arXiv preprint arXiv:1409.1259* (2014).

[8] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. 2015. Deep Reinforcement Learning in Large Discrete Action Spaces. *arXiv preprint arXiv:1512.07679* (2015).

[9] Michael D. Ekstrand, Ben Carterette, and Fernando Diaz. 2023. Distributionally-Informed Recommender System Evaluation. *ACM Transactions on Recommender Systems* (2023).

[10] Naieme Hazrati and Francesco Ricci. 2022. Recommender Systems Effect on the Evolution of Users' Choices Distribution. *Information Processing & Management* 59, 1 (2022), 102766.

[11] Zhankui He, Zhouhang Xie, Rahul Jha, Harald Steck, Dawen Liang, Yesu Feng, Bodhisattwa Prasad Majumder, Nathan Kallus, and Julian McAuley. 2023. Large

Language Models as Zero-Shot Conversational Recommenders. *arXiv preprint arXiv:2308.10053* (2023).

[12] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. 2004. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems* 22, 1 (2004), 5–53.

[13] Jin Huang, Harrie Oosterhuis, Bunyamin Cetinkaya, Thijs Rood, and Maarten de Rijke. 2022. State Encoders in Reinforcement Learning for Recommendation: A Reproducibility Study. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2738–2748.

[14] Jin Huang, Harrie Oosterhuis, Maarten de Rijke, and Herke van Hoof. 2020. Keeping Dataset Biases Out of the Simulation: A Debiased Simulator for Reinforcement Learning Based Recommender Systems. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 190–199.

[15] Marius Kaminskas and Derek Bridge. 2016. Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-accuracy Objectives in Recommender Systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 7, 1 (2016), 1–42.

[16] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations*.

[17] Lihong Li, Jin Young Kim, and Imed Zitouni. 2015. Toward Predicting the Outcome of an A/B Experiment for Search Relevance. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. 37–46.

[18] Ming Li, Mozhdeh Ariannezhad, Andrew Yates, and Maarten de Rijke. 2023. Masked and Swapped sequence modeling for Next Novel Basket Recommendation in Grocery Shopping. In *RecSys 2023: 17th ACM Conference on Recommender Systems*. ACM.

[19] Ming Li, Mozhdeh Ariannezhad, Andrew Yates, and Maarten de Rijke. 2023. Who Will Purchase this Item Next? Reverse Next Period Recommendation in Grocery Shopping. *ACM Transactions on Recommender Systems* 1, 2 (2023), Article 10.

[20] Ming Li, Sami Jullien, Mozhdeh Ariannezhad, and Maarten de Rijke. 2023. A Next Basket Recommendation Reality Check. *ACM Transactions on Information Systems* 41, 4 (2023), Article 116.

[21] Ming Li, Ali Vardasbi, Andrew Yates, and Maarten de Rijke. 2023. Repetition and Exploration in Sequential Recommendation. In *SIGIR 2023: 46th international ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2532–2541.

[22] Elad Liebman, Maytal Saar-Tsechansky, and Peter Stone. 2014. DJ-MC: A Reinforcement-learning Agent for Music Playlist Recommendation. *arXiv preprint arXiv:1401.1880* (2014).

[23] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous Control with Deep Reinforcement Learning. *arXiv preprint arXiv:1509.02971* (2015).

[24] Yuanguo Lin, Yong Liu, Fan Lin, Lixin Zou, Pengcheng Wu, Wenhua Zeng, Huanhuan Chen, and Chunyan Miao. 2023. A Survey on Reinforcement Learning for Recommender Systems. *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[25] Feng Liu, Ruiming Tang, Xutao Li, Weinan Zhang, Yunming Ye, Haokun Chen, Huifeng Guo, and Yuzhou Zhang. 2018. Deep Reinforcement Learning Based Recommendation with Explicit User-item Interactions Modeling. *arXiv preprint arXiv:1810.12027* (2018).

[26] Feng Liu, Ruiming Tang, Xutao Li, Weinan Zhang, Yunming Ye, Haokun Chen, Huifeng Guo, Yuzhou Zhang, and Xiuqiang He. 2020. State Representation Modeling for Deep Reinforcement Learning based Recommendation. *Knowledge-Based Systems* 205 (2020), 106170.

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An Imperative Style, High-performance Deep Learning Library. *Advances in Neural Information Processing Systems* 32 (2019).

[28] Aleksandr Petrov and Craig Macdonald. 2022. A Systematic Review and Replicability Study of BERT4Rec for Sequential Recommendation. In *Proceedings of the 16th ACM Conference on Recommender Systems*. 436–447.

[29] Harald Steck. 2013. Evaluation of Recommendations: Rating-prediction and Ranking. In *Proceedings of the 7th ACM conference on Recommender systems*. 213–220.

[30] Yueming Sun and Yi Zhang. 2018. Conversational Recommender System. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 235–244.

[31] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2018.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. *arXiv preprint arXiv:1706.03762* (2017).

[33] Haolun Wu, Bhaskar Mitra, Chen Ma, Fernando Diaz, and Xue Liu. 2022. Joint Multisided Exposure Fairness for Recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 703–714.

[34] Yao Wu, Jian Cao, Guandong Xu, and Yudong Tan. 2021. TFROM: A Two-Sided Fairness-Aware Recommendation Model for Both Customers and Providers. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1013–1022.

[35] Yikun Xian, Zuohui Fu, Shan Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. 2019. Reinforcement Knowledge Graph Reasoning for Explainable Recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 285–294.

[36] Mi Zhang and Neil Hurley. 2008. Avoiding Monotony: Improving the Diversity of Recommendation Lists. In *Proceedings of the 2008 ACM conference on Recommender systems*. 123–130.

[37] Weinan Zhang, Xiangyu Zhao, Li Zhao, Dawei Yin, and Grace Hui Yang. 2021. *DRL4IR: 2nd Workshop on Deep Reinforcement Learning for Information Retrieval*. Association for Computing Machinery, New York, NY, USA, 2681–2684.

[38] Wayne Xin Zhao, Junhua Chen, Pengfei Wang, Qi Gu, and Ji-Rong Wen. 2020. Revisiting Alternative Experimental Settings for Evaluating Top-n Item Recommendation Algorithms. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2329–2332.

[39] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, Yingqian Min, Zhichao Feng, Xinyan Fan, Xu Chen, Pengfei Wang, Wendi Ji, Yaliang Li, Xiaoling Wang, and Ji-Ron Wen. 2021. RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 4653–4664.

[40] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018. Deep Reinforcement Learning for Page-wise Recommendations. In *RecSys*. ACM, 95–103.

[41] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with Negative Feedback via Pairwise Deep Reinforcement Learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1040–1048.

[42] Xiangyu Zhao, Liang Zhang, Long Xia, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2017. Deep Reinforcement Learning for List-Wise Recommendations. *arXiv preprint arXiv:1801.00209* (2017).

[43] Xiangyu Zhao, Xudong Zheng, Xiwang Yang, Xiaobing Liu, and Jiliang Tang. 2020. Jointly Learning to Recommend and Advertise. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3319–3327.

[44] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. DRN: A Deep Reinforcement Learning Framework for News Recommendation. In *Proceedings of the 2018 World Wide Web Conference*. 167–176.

[45] 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 *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2810–2818.

[46] Lixin Zou, Long Xia, Yulong Gu, Xiangyu Zhao, Weidong Liu, Jimmy Xiangji Huang, and Dawei Yin. 2020. Neural Interactive Collaborative Filtering. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 749–758.