On the Unexpected Effectiveness of Reinforcement Learning for Sequential Recommendation

Álvaro Labarca [1 2] **Denis Parra** [1 3 2 4] **Rodrigo Toro Icarte** [1 2]

Abstract

In recent years, Reinforcement Learning (RL) has shown great promise in session-based recommendation. Sequential models that use RL have reached state-of-the-art performance for the Next-item Prediction (NIP) task. This result is intriguing, as the NIP task only evaluates how well the system can correctly recommend the next item to the user, while the goal of RL is to find a policy that optimizes rewards in the long term – sometimes at the expense of suboptimal short-term performance. Then, how can RL improve the system's performance on short-term metrics? This article investigates this question by exploring proxy learning objectives, which we identify as goals RL models might be following, and thus could explain the performance boost. We found that RL – when used as an auxiliary loss – promotes the learning of embeddings that capture information about the user's previously interacted items. Subsequently, we replaced the RL objective with a straightforward auxiliary loss designed to predict the number of items the user interacted with. This substitution results in performance gains comparable to RL. These findings pave the way to improve performance and understanding of RL methods for recommender systems.

1. Introduction

Recommender Systems (RecSys) help us select the most appropriate items or actions in the complexities of an information-saturated world (Jannach et al., 2010; Burke et al., 2011; Parra & Sahebi, 2013). RecSys have achieved their purpose by learning user preferences and thus suggesting items in personalized and contextual ways (Karatzoglou et al., 2010; Rendle et al., 2011; Villa et al., 2020; Adomavicius et al., 2021). These approaches seek to learn users' static preferences, assuming each item in the users' historical interactions is equally important. This, however, does not always hold true. In some scenarios, the user's preferences are dynamic (Fang et al., 2020) or the recommendations are based only on the user interactions during a session, without any previous history (Xin et al., 2020). Session-based recommendation is applied in several areas (e-commerce, video, news, advertisement) and it is traditionally addressed with sequence-aware recommender systems (Quadrana et al., 2018), being Reinforcement Learning (RL) one of the techniques gaining more traction in recent years (Afsar et al., 2022).

For many years, recommendation systems have utilized reinforcement learning through the application of multi-armed bandits (MABs) (Li et al., 2010; Nguyen & Kofod-Petersen, 2014; Li et al., 2016). These methods address the interactive nature of recommendation systems and the cold-start problem (Elena et al., 2021), but they tend to focus on short-term preferences. More recently, deep reinforcement learning (DRL) approaches, such as value-based methods like Q-learning (Lei et al., 2019) and policy-based methods like REINFORCE (Chen et al., 2021a) have been introduced to model long-term preferences, which can translate into better long-term user engagement (Zou et al., 2019).

However, RL-based methods are usually evaluated under short-term metrics, such as next-item prediction (NIP) (Deffayet et al., 2023). Under this NIP evaluation protocol, RL techniques have achieved promising results – often outperforming traditional self-supervised methods (Lee et al., 2022; Zhao et al., 2018b; Stamenkovic et al., 2022; Wang, 2020). This result, however, is counter-intuitive. Even though RL has many characteristics that make it a good alternative for sequential recommendation, as Deffayet et al. (2023) pointed out, NIP cannot capture the potential benefits introduced by the use of RL algorithms. NIP is a one-shot accuracy-oriented protocol. The goal of RL is to maximize

[1]Department of Computer Science, Pontificia Universidad Católica de Chile, Santiago, Chile [2]National Center for Artificial Intelligence (CENIA), Santiago, Chile [3]Instituto Milenio en Ingeniería e Inteligencia Artificial para la Salud (iHealth), Santiago, Chile [4]Instituto Milenio Fundamentos de los Datos (IMFD), Santiago, Chile. Correspondence to: Álvaro Labarca <aalabarca@uc.cl>, Denis Parra <dparras@uc.cl>, Rodrigo Toro Icarte <rntoro@uc.cl>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

the long-term reward. As a result, RL will tend to make poor short-term recommendations in order to make better recommendations in the future. Then, why is RL improving the NIP performance across so many different works? How does RL still manage to outperform self-supervised sequential methods?

In this paper, we investigate those questions. We first formally prove that the NIP performance of an optimal RL policy can be arbitrarily worse than the performance of a self-supervised model. This further shows that, in theory, there are no reasons to expect that RL would improve the performance of recommender systems under NIP metrics.

Then, we analyze the case of the *Self-Supervised Q-learning (SQN)* framework proposed by Xin et al. (2020). SQN shows that adding RL as an auxiliary loss to a self-supervised model improves the model's generalization performance with respect to NIP metrics. We analyze this result and show that learning an optimal policy with RL might not be the reason why methods such as SQN improve the NIP performance. Instead, RL seems to be implicitly learning to encode information about the user's past interactions. We then replaced the RL objective with a simple auxiliary loss that predicted the number of past interactions of the user, and we observed similar performance gains to those seen when using RL as an auxiliary loss.

Thus, we contribute to the research on RL recommendation systems by (i) formally analyzing the discrepancy between expected and obtained results with RL modeling for sequence-aware RecSys, (ii) studying the mechanism that allows RL to improve the NIP performance of self-supervised methods, and (iii) showing that replacing RL with more straightforward objectives can still achieve similar performance gains.

2. Sequential Recommendation

Sequential recommendation refers to recommender systems where the sequential nature of events (e.g., interactions, sessions) is considered to model user preferences and make recommendations (Hidasi & Czapp, 2023). This allows models to capture sequential patterns in users' short-term and dynamic behaviors (Tang & Wang, 2018).

Sequential recommendation problems are typically defined as a task of next item recommendation (Ludewig & Jannach, 2018). Let $\mathcal{I}$ denote the item set and $x_{1:t} = \{x_1, x_2, \ldots, x_{t-1}, x_t\}$ denote a user-item interaction sequence up to time t, where $x_i \in \mathcal{I} (1 \leq i \leq t)$ represents the interacted item at time step i and $x_{1:0} = \emptyset$. Then the goal is to recommend the most relevant next item x_{t+1} given the user's previous interactions $x_{1:t}$.

Next-item Prediction (NIP) is an offline evaluation protocol adopted in many sequential recommendation studies (Tang & Wang, 2018; Zhao et al., 2018b; Xin et al., 2022a; Hidasi et al., 2016). Deffayet et al. (2023) defined the NIP task as ensuring that the next interacted item in the history log is among the top items ranked by the model, given the sequence of past interactions. For example, given a logged interaction sequence $\{x_1, x_2, x_3\}$, the goal of NIP is to predict x_1 given $\emptyset$, x_2 given $\{x_1\}$, and x_3 given $\{x_1, x_2\}$. Performance is then measured according to ranking metrics concerning the next target item (e.g., hit rate, nDCG, etc).

The most common approach to tackle this problem is to train a *self-supervised model* $f_\theta(\cdot|x_{1:t})$ that, given the user's past interactions $x_{1:t}$, predicts the next item x_{t+1} that the user is going to interact with (Kang & McAuley, 2018). Here, f_θ is a neural network with parameters θ. Its output is a probability distribution over the possible items that might be recommended to the user. This network is usually trained using a cross-entropy loss:

$$L_s = -\sum_{x \in \mathcal{D}} \sum_{t=0}^{|x|-1} \log \left(f_\theta \left(x_{t+1} | x_{1:t} \right) \right), \tag{1}$$

where $\mathcal{D}$ is a set of training user-item interaction sequences.

3. Reinforcement Learning (RL)

RL agents learn optimal behavior by interacting with an environment (Sutton & Barto, 2018). The environment is usually modeled as a *Markov Decision Problem (MDP)*. An MDP is a tuple $(\mathcal{S}, \mathcal{A}, p, r, \rho_0, \gamma)$, where $\mathcal{S}$ is a finite set of *states*, $\mathcal{A}$ is a finite set of *actions*, $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the *state transition probability*, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the *reward function*, ρ_0 is the *initial state distribution*, and γ is the *discount factor*.

A *policy* $\pi(a|s)$ is a probability distribution over actions given a state. In *episodic tasks*, each episode begins by sampling an initial state $s_0 \sim \rho_0$ from the environment. Then, at time step t, the agent observes the current state s_t and selects an action $a_t \sim \pi(\cdot|s_t)$ according to its current policy. As a result, the next state $s_{t+1} \sim p(\cdot|s_t, a_t)$ and immediate reward $r_{t+1} = r(s_t, a_t)$ are obtained from the environment. The process repeats from $t+1$ until reaching a *terminal* state – which ends the episode.

The goal is to find an *optimal policy* π_*. That is, a policy that maximizes the expected *discounted return* that the agent gets while interacting with the environment:

$$\pi_* = \arg\max_\pi \mathbb{E}_\pi \left[\sum_{t=1}^{H} \gamma^k R_t \right], \tag{2}$$

where $\mathbb{E}_\pi[\cdot]$ represents the expected value of following policy π, H is the episode length, and R_t is a random variable representing the immediate reward at time step t.

We can formulate the sequential recommendation problem as a MDP (Shani et al., 2005; Xin et al., 2020), where the agent looks to maximize the cumulative reward interacting with the environment (users) by choosing actions (recommendations) and receiving rewards. The reward signal is typically defined in terms of the *type of interaction* with the item. For instance, if the possible interactions are *click* and *buy*, the reward signal could be high when the user buys the recommended item, low when the user only clicks on the item, and zero when the user does not interact with the item.

One way to learn optimal policies is by learning *Q-functions*. The Q-function $q_\pi(s, a)$ represents the expected discounted return of taking action a in state s and following policy π thereafter. We can use *Temporal-Difference (TD)* learning to estimate Q-functions from experiences. An experience is a tuple (s, a, r, s', a') where s is a state where the agent performed action a and, as a result, reached state s' and received an immediate reward r. An experience might also contain the next action a' that the agent performed from s'.

Given a set of training experiences $\mathcal{D}$, we can train the parameters θ of a neural network Q_θ to approximate q_π for a given policy π by minimizing the following loss function:

$$L_{\text{eval}} = \sum_{(s,a,r,s',a') \in \mathcal{D}} \left(r + \gamma Q_{\theta'}(s', a') - Q_\theta(s, a)\right)^2, \tag{3}$$

where θ' are the parameters of a *target network* (Van Hasselt et al., 2016). This problem is known as *policy evaluation* since it evaluates how good a policy π is.

We could also estimate the *optimal Q-function* q_*. The optimal Q-function is the Q-function of an optimal policy. To do so, we just have to change the loss function by the following loss (Mnih et al., 2015):

$$L_{\text{DQN}} = \sum_{(s,a,r,s') \in \mathcal{D}} \left(r + \gamma \max_{a'} Q_{\theta'}(s', a') - Q_\theta(s, a)\right)^2 \tag{4}$$

This is known as the *control problem* since if we learn $Q_\theta \approx q_*$ then an optimal policy might be obtained by always selecting the action with the highest $Q_\theta(s, a)$ for all $s \in S$.

4. An Intriguing Result About RL in RecSys

Different works have shown that RL improves the NIP performance of recommender systems (e.g., Xin et al., 2020; Stamenkovic et al., 2022; Antaris & Rafailidis, 2021; Xiao & Wang, 2021). However, from a theoretical point of view, there is no reason to expect that. RL's learning objective is to find a policy that collects maximal cumulative reward from the environment. Since this objective is defined in terms of *cumulative reward*, an optimal policy might first recommend niche items in order to discover what the user likes and then exploit such knowledge to make excellent long-term recommendations. Unfortunately, NIP metrics do not evaluate performance in the long term. They purely evaluate the next item prediction over a fixed set of sequences – penalizing the recommendation of niche items. And the long-term advantages of RL do not compensate for that because the testing sequences do not depend on previous recommendations made by the agent (Deffayet et al., 2023).

Because of this misalignment, an optimal policy might perform arbitrarily worse than the self-supervised model from Equation 1 according to NIP metrics. To prove this, we first define a *sequential recommender* as any function that ranks items given the user's previous interactions.

Definition 4.1 (Sequential Recommender)**.** Given a finite set of items $\mathcal{I}$, a *sequential recommender* $\mathcal{R} : \mathcal{I}^* \times \mathcal{I} \to \{1, \cdots, |\mathcal{I}|\}$ is any function that utilizes the user's past interactions $u \in \mathcal{I}^*$ to rank the items $\mathcal{I}$. That is, $\mathcal{R}(u, x)$ is an integer between 1 and $|\mathcal{I}|$, where $\mathcal{R}(u, x)$ is 1 when item x is at the top of the ranking. In addition, every item must be assigned a different position in the ranking: $\mathcal{R}(u, x) \neq \mathcal{R}(u, x')$ if $x \neq x'$ for all $u \in \mathcal{I}^*$ and $x, x' \in \mathcal{I}$.

We now define a *NIP metric* as a function that, given a set of interaction sequences, assigns a score between zero and one to sequential recommenders. This score is equal to the average score of each next-item prediction. The NIP metric is *consistent* if the score provided to each next-item prediction is 1 when the next item is at the top of the ranking, 0 when the next item is at the bottom of the ranking, and the scores are non-increasing in between.

Definition 4.2 (Consistency)**.** Given a finite set of items $\mathcal{I}$, a sequential recommender $\mathcal{R} : \mathcal{I}^* \times \mathcal{I} \to \{1, \cdots, |\mathcal{I}|\}$, a scoring function $s : \{1, \cdots, |\mathcal{I}|\} \to [0, 1]$, and a set of item sequences $\mathcal{D}$, a *NIP metric* is defined as the average score of the position of the next items in $\mathcal{D}$ according to the scoring function s and ranking defined by $\mathcal{R}$:

$$\mathcal{N}(\mathcal{D}, \mathcal{R}) = \frac{1}{N} \sum_{x \in \mathcal{D}} \sum_{t=0}^{|x|-1} s(\mathcal{R}(x_{1:t}, x_{t+1}))$$

where $N = \sum_{x \in \mathcal{D}} |x|$ is the total number of next-step recommendations. Moreover, we say that the NIP metric is *consistent* if $s(1) = 1$, $s(|\mathcal{I}|) = 0$, and $s(i) \geq s(i + 1)$ for all $i \in \{1, \cdots, |\mathcal{I}| - 1\}$.

Note that classic NIP metrics such as Hit Ratio (HR) (Zhang et al., 2018) and normalized discounted cumulative gain (nDCG) (Järvelin & Kekäläinen, 2002) are consistent. Finally, we show that the relative NIP performance of an optimal policy might be arbitrarily worse than the performance of an optimal cross-entropy solution.

Theorem 4.3. *For any consistent NIP metric $\mathcal{N}$ and discount factor $\gamma > 0$, the relative NIP performance of an optimal policy π_* can be arbitrarily worse than the performance of an optimal solution f_* according to L_s (Equation 1).*

Proof. In Appendix A, we show that for any natural number $n > 1$, we can build a dataset $\mathcal{D}$ such that:

$$\frac{\mathcal{N}(\mathcal{D}, f_*)}{\mathcal{N}(\mathcal{D}, \pi_*)} > n \ .$$

Then, as we increase the value of n, the ratio between the performance of f_* and π_* can grow arbitrarily large. □

Moreover, the NIP performance of an optimal policy is bounded, up to some degree, by the performance of an optimal cross-entropy solution f_*, as shown below:

Theorem 4.4. *Let's consider any consistent NIP metric $\mathcal{N}$ and set of interaction sequences $\mathcal{D}$. Let f_* be an optimal solution to the cross-entropy loss L_s from equation 1 assuming f_* has sufficient capacity. Let π be any optimal policy with respect to $\mathcal{D}$. Then, $\mathcal{N}(\mathcal{D}, f_*) \geq \mathcal{N}(\mathcal{D}, \pi_*)$.*

Proof. See Appendix B. □

Then, why is RL improving the NIP performance?

5. The SQN Learning Framework

To study why RL improves the performance in RecSys, we analyze the Self-Supervised Q-learning (SQN) framework (Xin et al., 2020). SQN was one of the first approaches to demonstrate that, under identical conditions, adding RL improves the generalization performance of sequential recommenders under NIP metrics. This result held across different models and datasets. It also inspired a family of RL+RecSys methods that follow the same learning principle (e.g., Xin et al., 2022a; Stamenkovic et al., 2022; Gao et al., 2022; Antaris & Rafailidis, 2021).

SQN augments standard sequential self-supervised models with a second output layer for the RL implementation, as shown in Figure 1(a). This model has two outputs. The *action logits* output is a soft-max layer that decides which item to recommend next. It is trained using the standard cross-entropy loss from Equation 1. The second output is the RL layer. It has one output per item and it estimates the values of an optimal Q-function. This layer is trained using the DQN loss from Equation 4. As such, the model's combined loss function is as follows:

$$L_{SQN} = L_s + L_{\text{DQN}} \tag{5}$$

Note that RL acts as a regularizer in SQN. In fact, the recommendations are selected according to the *action logits*.

Finally, it is worth discussing the reward function and discount factor used in the experimental evaluation of SQN. We believe that both – the selected rewards and discount – played a crucial role in SQN's good performance. In their experimental setting, there were two types of interactions: *buy* and *click*. When the user bought an item, the reward was 5. When the user clicked on an item, the reward was 1. The discount factor was set to 0.5.

6. Finding Proxy Learning Signals for NIP

Our hypothesis is that RL, by itself, is not responsible for the performance improvements. We believe that a clever combination of reward signals and discount factors entails useful auxiliary losses for self-supervised models. But RL, as the process of learning an optimal policy from data, might not be needed to do so. If this were the case, we should be able to replace RL with a simple auxiliary loss without losing performance.

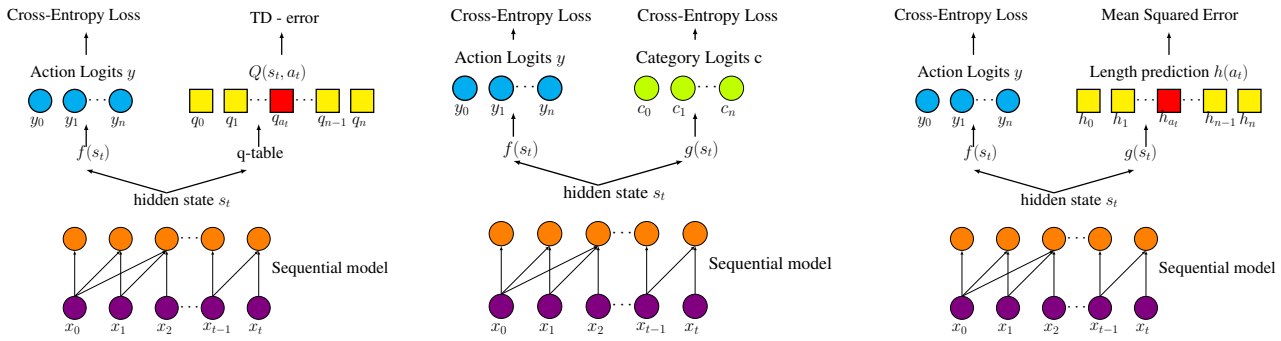
In this section, we explore this idea from two perspectives. The top-down approach tries to understand how the L_{DQN} loss (Equation 4) behaves in sequential RecSys, given the reward function and discount factor that are typically used. The bottom-up approach follows an empirical methodology to understand what features can better explain the embeddings learned when using RL as an auxiliary loss.

6.1. Top-Down Approach: Predict Types of Interactions

Analyzing the behavior of L_{DQN} (Equation 4) in SQN is challenging because of the maximization term. The term $\max_{a'} Q_{\theta'}(s', a')$ asks the network to predict how much expected return would be received if item a' is recommended from state s'. This could be an item that has never been recommended to the user. As such, we don't really know its value. The user might buy the item or click on the item. Depending on that, the reward would be 1 or 5.

As an alternative, we propose a variation of SQN where L_{DQN} is replaced by L_{eval} (from Equation 3). In contrast to L_{DQN}, L_{eval} doesn't attempt to learn an optimal policy. It simply evaluates the policy that was used to create the dataset. As a result, the maximization term is replaced by $Q_{\theta'}(s', a')$. That is, how much expected return the agent will receive if it recommends the next item in the sequence. The key advantage is that, since we always recommend the next item in the sequence, we know its real reward. This makes analyzing L_{eval} much simpler than analyzing L_{DQN}.

We also note that the empirical performance of SQN does not drop significantly when replacing L_{DQN} by L_{eval}. Table 1 compares the NIP performance of GRU, a standard self-supervised model (Hidasi et al., 2016), with other methods in RetailRocket. GRU-SQN is SQN when training the RL head using L_{DQN}. GRU-EVAL is our variation of SQN that trains the RL head using L_{eval}. As the table shows, both versions of SQN outperform the self-supervised model, and there is not much difference between GRU-SQN and GRU-EVAL. Hence, we focus our analysis on GRU-EVAL.



(a) SQN architecture (Xin et al., 2020) (b) Interaction-based architecture (c) Length-based architecture

Figure 1. The SQN and proposed architectures

Recall that L_{eval} finds a set of parameters θ such that the model Q_θ approximates the Q-function q_π of a policy π. In this case, π is the policy used to create the training set. That is, a policy that always recommends the next item that appears in the training sequence. In the empirical evaluation of SQN, Xin et al. (2020) used a discount factor of 0.5. This is a fairly aggressive discount. Considering that the reward was 1 for clicking and 5 for buying, the value of q_π will have a lower bound of 1 and an upper bound of 10. Moreover, only the first two or three immediate rewards account for most of the value that we are trying to predict:

$$q_\pi(s_t, a_t) = r_{t+1} + \frac{1}{2} r_{t+2} + \frac{1}{4} r_{t+3} + \sum_{k=3}^{\infty} \frac{1}{2^k} r_{t+k+1}$$
$$\leq r_{t+1} + \frac{1}{2} r_{t+2} + \frac{1}{4} r_{t+3} + 1.25$$

Therefore, to correctly predict $q_\pi(s_t, a_t)$, $Q_\theta(s_t, a_t)$ must implicitly learn to predict the type of interactions that the user will have with the next two or three items. For instance, predicting $Q_\theta(s_t, a_t) = 1$ means the user will click on the next item and then the sequence ends. On the other hand, predicting $Q_\theta(s_t, a_t) = 7.5$ means that the user will buy two items, and then the session ends.

As a result, we believe that adding L_{eval} as an auxiliary loss encourages the self-supervised model to learn embeddings that can predict the type of interactions the user will have with the next items. Doing so could improve the model's generalization because knowing whether the user will click or buy the next item constrains the items that make sense to recommend. For instance, if we know that the user will buy two items, the model could increase the probability of recommending an item that tends to be purchased together with other items (e.g., complementary items).

To test our hypothesis, we replaced the RL head with a simple categorical classification problem, as shown in Figure 1(b). We call this model **cat**. The cat model adds an auxiliary loss to the self-supervised model. The auxiliary loss is a cross-entropy that predicts the types of interactions with the following two items. In total, there are six classes in this problem: *Click-Done*, *Click-Click*, *Click-Buy*, *Buy-Done*, *Buy-Click*, and *Buy-Buy*. Since we know the future interaction types during training, it is trivial to compute the supervision for this auxiliary loss. We also experimented with a **cat3 model**, which is identical to the cat model, but it predicts the interaction types with the following three items.

Table 1. Top-k recommendation performance comparison for **buy** interactions on the RetailRocket dataset. Boldface denotes the highest performance. * denotes that the model outperforms the self-supervised baseline with a significance p-value < 0.05.

Model	HR@5	NG@5	HR@20	NG@20
GRU	0.1601	0.1248	0.2306	0.1456
GRU-SQN	*0.1921	0.1519	*0.2698	0.1743
GRU-EVAL	*0.1962	*0.1545	*0.2718	*0.1762
GRU-CAT	0.1644	0.1282	*0.2384	0.1495
GRU-CAT3	*0.1696	*0.1325	*0.2435	*0.1536
GRU-HIST	***0.1980**	***0.1549**	***0.2747**	***0.1770**
GRU-FUT	0.0390	0.0312	0.0569	0.0375

Table 1 shows the performance in RetailRocket of GRU using cat and cat3 as auxiliary losses. We note that cat and cat3 are not RL objectives. Still, they significantly improve the performance of GRU. On the other hand, there is a clear gap between the performance of the cat models and SQN. Somehow, the cat model doesn't fully explain the performance gains of using RL in RecSys. This motivated our second approach to explore why RL works in RecSys.

6.2. Bottom-Up Approach: Feature Importance

Given the lackluster results obtained using the cat method, we take a different approach. It might be that L_{DQN} is

Table 2. List of features used for linear regression analysis. SHAP-XG shows the feature importance scores derived using SHAP values from the XGBoost model on the **RetailRocket** dataset using the **Caser** self-supervised model.

Feature	Description	SHAP-XG
hist-length	Number of past user interactions in the sequence.	18.4964
fut-length	Number of future user interactions in the sequence.	5.8296
hist-buys	Number of items the user bought in past interactions.	2.8757
fut-buys	Number of items the user will buy in future interactions.	2.7714
Q-Value	The expected return following the sequence in the history log.	1.7878

not really learning a good approximation of the optimal Q-function. After all, we are using RL over a fixed set of sequences. This setting is known as offline RL (Levine et al., 2020), and it is known to be quite challenging (Dulac-Arnold et al., 2021). Thus, we decided to empirically look for features that correlate well with the Q-values that are learned using L_{DQN}.

Specifically, we trained a standard SQN model and obtained all the Q-value $Q_\theta(x_{1:t}, \cdot)$ predictions on the test set $\mathcal{D}_{\text{test}}$. Note that $Q_\theta(x_{1:t}, \cdot)$ is a vector of size $|\mathcal{I}|$ (i.e., there is one value per item). Then, for each sequence $x \in \mathcal{D}_{\text{test}}$, we compute twelve features that, we believe, could explain the Q-value predictions. We then performed a correlation test between the features and calculated their *variance inflation factor (VIF)* to avoid significant multicollinearity (Daoud, 2017). After this preprocessing, only five features remained – shown in Table 2. Finally, we performed feature importance analyses to discover the features that explain Q_θ better.

We conducted three feature importance analyses on the RetailRocket dataset: (1) linear regression, (2) SHAP (Lundberg & Lee, 2017) after training predictive models with XGBoost (Chen & Guestrin, 2016), and (3) SHAP after training predictive models with linear regression – obtaining similar feature rankings (see Table 11 in Appendix). This ranking is shown in Table 2, including the SHAP values from the trained XGBoost model using the Caser model as a reference. Surprisingly, the most relevant feature is the number of items the user has interacted with in the session prior to the target timestamp (hist-length), followed by the number of future interactions in the session (fut-length).

Our feature analysis suggests that RL might improve the NIP performance because it learns to remember the number of items the user has interacted with. To test this hypothesis, we propose the **hist model** (shown in Figure 1(c)). The hist model replaces the RL head from SQN with a prediction of the number of items that the user has interacted with. This head is trained using a *mean square error* loss between the prediction and the target number. As Table 1 shows, the hist model performs remarkably well. It is competitive with GRU-SQN and it does not use RL. This result sheds some light on why RL improves the NIP performance in RecSys.

Finally, we also investigated the performance of a **fut model**, which is similar to the hist model but predicts the number of future interactions. We explored this model because the *fut-length* feature was the second most important feature according to our feature analysis. Unfortunately, this model performs poorly – as shown in Table 1.

7. Experimental Settings

In this section, we describe the details of our experiments. We followed the recent guidelines proposed by Hidasi & Czapp (2023) to prevent a flawed evaluation.

Datasets. Hidasi & Czapp (2023) pointed out that many public rating-based datasets commonly used for sequential recommendation evaluation, such as MovieLens and Amazon (Beauty), are not appropriate for this task, as the time of rating is disjoint from the time of user-item interaction.

On the other hand, they validated the sequential nature of session-based datasets like RetailRocket, denoting it as an appropriate dataset for sequential recommendation evaluation. The **RetailRocket** dataset contains sequential data collected from a real-world e-commerce website. We followed the setting by Xin et al. (2020) by treating views as clicks and adding to cart actions as purchases. The dataset contains information about $195,525$ unique sessions with $1,233,949$ click or purchase events across $70,852$ items.

We also tested our models on the **RC15** dataset. This is another session-based dataset constructed from retailer session data. It was proposed as part of the RecSys Challenge 2015. This dataset contains $200,000$ sessions, $26,702$ unique items, and $1,154,911$ interaction events.

Evaluation protocol. In the original SQN implementation, Xin et al. (2020) followed a random split for training, testing, and validation. This protocol might be flawed (Hidasi & Czapp, 2023), since user preferences and behavior have been noted to drift over time (Tsymbal, 2004), causing information to leak when using train and test sessions with time overlaps. To prevent this issue, we adopt a time-based split (Hidasi & Czapp, 2023). We implement an 8:1:1 split ratio for train, validation, and test by using the first 80% of sessions for the train set and the last 10% for the test set.

We use two widely adopted metrics to evaluate the recommendation performance, following the evaluation procedure of previous work (Xin et al., 2020): Hit Ratio (HR) (Zhang et al., 2018), and normalized discounted cumulative gain (NDCG) (Järvelin & Kekäläinen, 2002). HR@K is a recall-

Table 3. Performance for **buy** interactions on the **RetailRocket** dataset. NG stands for NDCG. Boldface denotes the highest performance and underlines the runner-up. * denotes a significant difference when tested against the self-supervised baselines (GRU, NIN, CASER, and SAS).

MODEL	HR@5	NG@5	HR@20	NG@20
GRU	0.1601	0.1248	0.2306	0.1450
GRU-SQN	*0.1921	0.1519	*0.2698	0.1743
GRU-EVAL	*0.1962	*0.1545	*0.2718	*0.1762
GRU-CAT	0.1644	0.1282	*0.2384	0.1495
GRU-HIST	***0.1980**	***0.1549**	***0.2747**	***0.1770**
NIN	0.2282	0.1785	0.3215	0.2053
NIN-SQN	*0.3307	*0.2577	*0.4418	*0.2901
NIN-EVAL	***0.3308**	***0.2582**	***0.4421**	***0.2905**
NIN-CAT	*0.2505	0.1959	*0.3483	0.2242
NIN-HIST	*0.3001	*0.2348	*0.4110	*0.2667
CASER	0.1682	0.1361	0.2217	0.1514
CASER-SQN	*0.2020	*0.1601	*0.2715	*0.1801
CASER-EVAL	0.2000	0.1610	*0.2638	0.1794
CASER-CAT	0.1765	0.1434	0.2307	0.1591
CASER-HIST	***0.2399**	***0.1893**	***0.3296**	***0.2152**
SAS	0.2458	0.1872	0.3509	0.2176
SAS-SQN	***0.3012**	***0.2280**	***0.4227**	***0.2634**
SAS-EVAL	0.2963	*0.2242	*0.4195	*0.2600
SAS-CAT	*0.2636	*0.1999	*0.3731	*0.2315
SAS-HIST	*0.2960	*0.2231	0.3847	*0.2573

Table 4. Performance for **buy** interactions on the **RC15** dataset. NG stands for NDCG. Boldface denotes the highest performance and underlines the runner-up. * denotes a significant difference when tested against the self-supervised baselines (GRU, NIN, CASER, and SAS).

MODEL	HR@5	NG@5	HR@20	NG@20
GRU	0.3599	0.2511	0.5612	0.3098
GRU-SQN	0.3736	0.2626	0.5761	0.3218
GRU-EVAL	0.3704	0.2620	0.5776	0.3225
GRU-CAT	0.3579	0.2491	0.5551	0.3068
GRU-HIST	***0.3841**	***0.2714**	***0.5901**	***0.3316**
NIN	0.3208	0.2243	0.5054	0.2783
NIN-SQN	***0.3610**	***0.2548**	***0.5606**	***0.3132**
NIN-EVAL	*0.3483	*0.2463	0.5342	*0.3021
NIN-CAT	*0.3381	*0.2384	*0.5253	0.2923
NIN-HIST	*0.3558	*0.2520	*0.5521	*0.3092
CASER	0.4095	0.2898	0.6192	0.3511
CASER-SQN	*0.4263	*0.3014	*0.6342	*0.3621
CASER-EVAL	*0.4340	**0.3255**	*0.6448	*0.3714
CASER-CAT	*0.4351	*0.3106	*0.6386	*0.3699
CASER-HIST	***0.4528**	*0.3197	***0.6686**	***0.3830**
SAS	0.3706	0.2596	0.5598	0.3150
SAS-SQN	0.3831	0.2689	0.5805	*0.3266
SAS-EVAL	***0.4115**	***0.2872**	***0.6216**	***0.3483**
SAS-CAT	*0.4049	*0.2804	*0.6159	*0.3423
SAS-HIST	0.3741	0.2634	0.5634	0.3189

based metric that measures the ratio of events for which the correct item was among the top-k recommended items against the total number of events. NDCG@k is a ranking-based metric that assigns a higher value if the correct item is ranked better within the top-k recommended items.

Baselines. For the sequential-recommendation models used in the proposed model's architecture, we use four state-of-the-art models: GRU (Hidasi et al., 2016), Caser (Tang & Wang, 2018), NextItNet (NIN) (Yuan et al., 2019), and SASRec (SAS) (Kang & McAuley, 2018).

For SQN, we used the same hyperparameters as the original implementation (Xin et al., 2020) and trained the models for 50 epochs. Each experiment was run 5 times, with the average performance reported. Table 5 in the Appendix shows the implementation details for each model.

8. Results

Table 3 summarizes the results of a NIP evaluation protocol on purchase interactions in the RetailRocket dataset. We can see that our proposed models that replace the RL loss with a supervised version consistently outperform the respective self-supervised baseline, most of the time by a significant difference. We can see on Table 4 that these findings are consistent on the RC15 dataset. This result highlights the value of enhancing traditional recommendation models with proxy objective functions to increase performance.

We see that the proposed EVAL update did not significantly impact the model's performance compared to the SQN performance. The EVAL model narrowly outperforms the SQN model when paired with the GRU and NextItNet models on the RetailRocket dataset, while it does so when paired with the Caser and SASRec models on the RC15 dataset. As discussed earlier, this allows for a better mathematical interpretation of the learning process during training while not significantly impacting performance.

Regarding the **cat model**, it consistently and often significantly outperforms the self-supervised baseline on both datasets, showing the advantage of learning to predict interaction types of upcoming events on the next-item prediction task. However, there is a considerable gap between the performance of the cat model and the SQN variants.

The **hist model** achieves results comparable to the SQN model, outperforming SQN when paired with the GRU and Caser models on both datasets. On the RetailRocket dataset (Table 3), the best results overall are achieved by the NextItNet-EVAL model, while on the RC15 dataset (Table 4) they were achieved by Caser-hist. However, the goal of this paper is not to propose a novel model that outperforms the current state-of-the-art but rather to explore why RL shows an intriguing performance increase in a short-tem

evaluation setting. The fact that learning the length of the past sequence manages to achieve comparable performance to SQN brings us one step closer to understanding RL performance in RecSys.

Finally, the performance differences between using different self-supervised networks can be explained by the diverse mechanisms these networks have, impacting the information available in the hidden state given to the auxiliary head. Between the RNN-based GRU network, the CNN-based NextItNet and Caser networks, and the transformer-based SASRec network, different features from the original input sequence might be lost or accentuated, impacting the performance of the auxiliary task. The performances in click prediction yielded similar conclusions to the ones of buy prediction, and the full results can be found in Table 6.

9. Related Work

Recent works have investigated potential flaws in sequential recommendation evaluation. Hidasi & Czapp (2023) showed how improper dataset pre-processing and evaluation design can lead to flawed results. In contrast, Deffayet et al. (2023) highlighted problems in the offline evaluation for RL, specifically in the Next-item prediction (NIP) protocol. They discussed that NIP only accounts for short-term rewards and, as such, the reward maximization objective of RL is likely to deteriorate the results of this evaluation. In this paper, we further explored this observation. We formally showed that the NIP performance of an optimal RL policy might be arbitrarily worse than the performance of a self-supervised model. We then explored why RL methods can still improve the NIP performance and found that RL, as the process of finding optimal policies, is not exactly what produces the performance gains. Our experiments showed that we can replace RL with simple auxiliary functions and observe similar gains. However, the gap between RL and our auxiliary functions is not entirely closed. Exploring this remaining gap is an exciting direction for future work.

Xin et al. (2020) proposed SQN, a self-supervised framework that uses the TD-update rule from RL as a regularizer in a sequential, self-supervised prediction algorithm. Based on the SQN framework, Xin et al. (2022a) implemented a negative sampling technique, while Stamenkovic et al. (2022) changed the RL head with a multi-objective agent looking to enhance the diversity and novelty in the recommendations. Other methods, like SAR (Antaris & Rafailidis, 2021) and VPQ (Gao et al., 2022), also use RL as a regularizing agent in a cross-entropy-based self-supervised task. Each of these studies outperforms traditional self-supervised sequential networks on the NIP task. In this paper, we look to address the misalignment between the network objective and the NIP evaluation protocol discussed in Section 4. In this paper, we use SQN as the basis for our studies, but Theorems 4.3 and 4.4 apply equally to all the studies discussed here. Analyzing what these different RL methods are learning and why they improve the NIP performance is an important direction for future work.

SDAC (Xiao & Wang, 2021) proposes a framework based on a stochastic Actor-Critic method based on a probability formulation while implementing regularization approaches to reduce the extrapolation error. Unlike the previously discussed models, SDAC does not use the RL component as a regularizer for a supervised model. Since it's an RL-based model evaluated with the NIP protocol, Theorems 4.3 and 4.4 still apply to this framework. As such, a similar analysis to the one proposed in this paper could be applied to it.

Other studies have approached the sequential recommendation task using RL; however, instead of directly using the NIP protocol to evaluate, they take the items the target user interacted with in the test set and look to re-rank them, looking to place higher-value interactions in the top positions (e.g., Zhao et al., 2018a; 2020b; 2018b). While this evaluation method differs from NIP, it's still a one-shot evaluation protocol for an RL method. Thus, the discussion in Section 4 still applies to them. We believe that exploring the reason behind the misalignment between the model's objective and results is an interesting direction for future work.

While this paper aims to explore the aforementioned discrepancy, it does not contradict the potential long-term benefits that using RL might bring to RecSys. However, a simulator would be needed to evaluate those benefits properly. While there is still no standardized simulation platform or benchmark specific to RL-based RecSys (Chen et al., 2021b), and simulators are still not mature enough to be widely used as standardized baselines (Hidasi & Czapp, 2023), simulator research has been gaining more attention in RecSys (Loepp, 2022). Some simulators have been proposed to simulate sequential recommendation scenarios (e.g., McInerney et al., 2021; Hazrati & Ricci, 2022; Xie et al., 2018). At the same time, different works have developed their own ways to simulate unseen user feedback (e.g., Chen et al., 2019a;b; Zhang et al., 2020; Zou et al., 2020), managing to outperform the supervised baselines. By developing a framework capable of simulating dynamic user behavior under different conditions, the long-term benefits from RL could be assessed in an offline setting.

Finally, it is worth mentioning that the scope of this investigation applies only to RL-based sequential recommendation models that use the NIP protocol for evaluation. Research outside sequential recommendation includes conversational recommendation (e.g., Sun & Zhang, 2018; Ren et al., 2020; Lei et al., 2020), explainable recommendation (e.g., Zhao et al., 2020a; Liu et al., 2021; Park et al., 2022), and medical treatment recommendations (e.g., Nemati et al., 2016; Raghu et al., 2017). Other sequential approaches like

Prompt-Learning (Xin et al., 2022b) also fall outside this paper's scope. Similarly, the Multi-Armed Bandit approaches fall outside of this investigation since our theorems only apply to models with a discount factor $\gamma > 0$.

10. Conclusion and Future Work

This paper addressed an overlooked discrepancy in RL-based recommender system research. Somehow, RL-based methods achieve better performance under short-term metrics, even though they primarily aim to maximize long-term rewards.

We studied this discrepancy and found that RL allows self-supervised models to learn embeddings that encode information about the user's past interactions. In particular, when adding RL as an auxiliary loss, the network learns Q-value estimates that correlate with the length of the interaction sequence. Then, we replaced the RL objective with a simple prediction of the current sequence length and obtained similar performance gains to those obtained using RL.

This is an interesting result. It suggests that RL, as the process of finding an optimal policy, might not be the reason behind the performance improvement seen in sequential recommendation. But, at the same time, studying its behavior might allow us to find useful auxiliary losses for sequential recommendation, such as hist, cat, or fut.

Our work opens different avenues for future work. In particular, while the hist model achieved comparable performance to SQN overall, the results vary depending on which self-supervised method is used. And we are not entirely sure why that is the case. We believe that SQN outperforms hist in NIN and (sometimes) in SAS because those architectures have specialized mechanisms that work well with large sequences. For instance, NIN uses dilated convolutions in order to process longer sequences than traditional convolutional models (such as Caser). As such, it might be easier for NIN to consider the sequence length in their predictions. This could explain why using hist as an auxiliary loss did not significantly impact its performance. That said, further experimentation is required to verify this hypothesis.

Another direction for future work is to study how to exploit fut auxiliary loss better. Fut was the second most relevant feature according to our analysis in Section 6.2 but its empirical performance was terrible (see Tables 6 & 7 in the Appendix). A fundamental difficulty of using fut as an auxiliary loss is its high variance. For instance, consider a user who has not interacted with any item yet. The fut auxiliary loss asks the network to predict how many items this new user will interact with. The best possible guess is the average length of the sequences across the training data. But that guess is way off most of the time. Since fut is a mean square error loss, incorrect predictions result in large values (with high variance) – making it hard to optimize using gradient descent. Addressing this issue might allow us to take advantage of fut as an auxiliary loss in future work.

Finally, since the SQN learning framework is used as the basis for other works in RL (Xin et al., 2022a; Stamenkovic et al., 2022), we expect that the conclusions of our work should extend to other RL-based research in RecSys. We also hope this work serves as the first step towards better understanding the unexpected effectiveness of RL for sequential recommendation.

Acknowledgements

We thank Alexandros Karatzoglou for an insightful discussion about sequential recommendation and reinforcement learning at an early stage of this research project. His deep understanding of how RL is being used in recommender systems helped us better define the scope of our work.

We gratefully acknowledge funding from the National Center for Artificial Intelligence CENIA FB210017 (Basal ANID), the Millenium Initiative research centers iHealth ICN2021_004 and IMFD ICN17_002, and Fondecyt grants 11230762 and 1231724.

Impact Statement

The research carried out in this article might affect recommendation systems in terms of biases against certain groups of users, which were not investigated in this work. This might result, for instance, in certain unrepresented groups having smaller performance gains than others. This is left for future research, but we also warn about using this method in a production environment to check for potential biases.

References

Adomavicius, G., Bauman, K., Tuzhilin, A., and Unger, M. Context-aware recommender systems: From foundations to recent developments context-aware recommender systems. In *Recommender Systems Handbook*, pp. 211–250. Springer, 2021.

Afsar, M. M., Crump, T., and Far, B. Reinforcement learning based recommender systems: A survey. *ACM Computing Surveys*, 55(7):1–38, 2022.

Antaris, S. and Rafailidis, D. Sequence adaptation via reinforcement learning in recommender systems. In *Proceedings of the 15th ACM Conference on Recommender Systems (RecSys)*, pp. 714–718, 2021.

Burke, R., Felfernig, A., and Göker, M. H. Recommender systems: An overview. *Ai Magazine*, 32(3):13–18, 2011.

Chen, H., Dai, X., Cai, H., Zhang, W., Wang, X., Tang,

R., Zhang, Y., and Yu, Y. Large-scale interactive recommendation with tree-structured policy gradient. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*, pp. 3312–3320, 2019a.

Chen, M., Chang, B., Xu, C., and Chi, E. H. User response models to improve a reinforce recommender system. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 121–129, 2021a.

Chen, T. and Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 785–794, 2016.

Chen, X., Li, S., Li, H., Jiang, S., Qi, Y., and Song, L. Generative adversarial user model for reinforcement learning based recommendation system. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pp. 1052–1061, 2019b.

Chen, X., Yao, L., McAuley, J., Zhou, G., and Wang, X. A survey of deep reinforcement learning in recommender systems: A systematic review and future directions. *arXiv preprint arXiv:2109.03540*, 2021b.

Daoud, J. I. Multicollinearity and regression analysis. In *Journal of Physics: Conference Series (JPCS)*, volume 949, pp. 012009, 2017.

Deffayet, R., Thonet, T., Renders, J.-M., and de Rijke, M. Offline evaluation for reinforcement learning-based recommendation: a critical issue and some alternatives. *ACM SIGIR Forum*, 56(2):1–14, 2023.

Dulac-Arnold, G., Levine, N., Mankowitz, D. J., Li, J., Paduraru, C., Gowal, S., and Hester, T. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.

Elena, G., Milos, K., and Eugene, I. Survey of multiarmed bandit algorithms applied to recommendation systems. *International Journal of Open Information Technologies*, 9(4):12–27, 2021.

Fang, H., Zhang, D., Shu, Y., and Guo, G. Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations. *ACM Transactions on Information Systems (TOIS)*, 39(1):1–42, 2020.

Gao, C., Xu, K., Zhou, K., Li, L., Wang, X., Yuan, B., and Zhao, P. Value penalized q-learning for recommender systems. In *Proceedings of the 45th International ACM Special Interest Group on Information Retrieval SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 2008–2012, 2022.

Hazrati, N. and Ricci, F. Simulating users' interactions with recommender systems. In *Adjunct Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization (UMAP)*, pp. 95–98, 2022.

Hidasi, B. and Czapp, Á. T. Widespread flaws in offline evaluation of recommender systems. In *Proceedings of the 17th ACM Conference on Recommender Systems (RecSys)*, pp. 848–855, 2023.

Hidasi, B., Karatzoglou, A., Baltrunas, L., and Tikk, D. Session-based recommendations with recurrent neural networks. 2016.

Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. *Recommender systems: an introduction*. Cambridge University Press, 2010.

Järvelin, K. and Kekäläinen, J. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

Kang, W.-C. and McAuley, J. Self-attentive sequential recommendation. In *Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM)*, pp. 197–206. IEEE, 2018.

Karatzoglou, A., Amatriain, X., Baltrunas, L., and Oliver, N. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys)*, pp. 79–86, 2010.

Lee, H., Hwang, D., Min, K., and Choo, J. Towards validating long-term user feedbacks in interactive recommendation systems. In *Proceedings of the 45th International ACM Special Interest Group on Information Retrieval SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 2607–2611, 2022.

Lei, W., He, X., Miao, Y., Wu, Q., Hong, R., Kan, M.-Y., and Chua, T.-S. Estimation-action-reflection: Towards deep interaction between conversational and recommender systems. In *Proceedings of the 13th ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 304–312, 2020.

Lei, Y., Wang, Z., Li, W., and Pei, H. Social attentive deep q-network for recommendation. In *Proceedings of the 42nd International ACM Special Interest Group on Information Retrieval SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 1189–1192, 2019.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Li, L., Chu, W., Langford, J., and Schapire, R. E. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web (WWW)*, pp. 661–670, 2010.

Li, S., Karatzoglou, A., and Gentile, C. Collaborative filtering bandits. In *Proceedings of the 39th International ACM Special Interest Group on Information Retrieval SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 539–548, 2016.

Liu, D., Lian, J., Liu, Z., Wang, X., Sun, G., and Xie, X. Reinforced anchor knowledge graph generation for news recommendation reasoning. In *Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1055–1065, 2021.

Loepp, B. Recommender systems alone are not everything: Towards a broader perspective in the evaluation of recommender systems. In *Proceedings of the 2nd Workshop on the Perspectives on the Evaluation of Recommender Systems (PERSPECTIVES)*, 2022.

Ludewig, M. and Jannach, D. Evaluation of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction*, 28:331–390, 2018.

Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

McInerney, J., Elahi, E., Basilico, J., Raimond, Y., and Jebara, T. Accordion: a trainable simulator for long-term interactive systems. In *Proceedings of the 15th ACM Conference on Recommender Systems (RecSys)*, pp. 102–113, 2021.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Nemati, S., Ghassemi, M. M., and Clifford, G. D. Optimal medication dosing from suboptimal clinical examples: A deep reinforcement learning approach. In *Proceedings of the 38th International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 2978–2981, 2016.

Nguyen, H. T. and Kofod-Petersen, A. Using multi-armed bandit to solve cold-start problems in recommender systems at telco. In *Proceedings of the 2nd International Conference on Mining Intelligence and Knowledge Exploration (MIKE)*, pp. 21–30, 2014.

Park, S.-J., Chae, D.-K., Bae, H.-K., Park, S., and Kim, S.-W. Reinforcement learning over sentiment-augmented knowledge graphs towards accurate and explainable recommendation. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 784–793, 2022.

Parra, D. and Sahebi, S. Recommender systems: Sources of knowledge and evaluation metrics. In *Advanced Techniques in Web Intelligence-2: Web User Browsing Behaviour and Preference Analysis*, pp. 149–175. Springer, 2013.

Quadrana, M., Cremonesi, P., and Jannach, D. Sequence-aware recommender systems. *ACM computing surveys (CSUR)*, 51(4):1–36, 2018.

Raghu, A., Komorowski, M., Ahmed, I., Celi, L., Szolovits, P., and Ghassemi, M. Deep reinforcement learning for sepsis treatment. *arXiv preprint arXiv:1711.09602*, 2017.

Ren, X., Yin, H., Chen, T., Wang, H., Hung, N. Q. V., Huang, Z., and Zhang, X. Crsal: Conversational recommender systems with adversarial learning. *ACM Transactions on Information Systems (TOIS)*, 38(4):1–40, 2020.

Rendle, S., Gantner, Z., Freudenthaler, C., and Schmidt-Thieme, L. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th International ACM Special Interest Group on Information Retrieval SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 635–644, 2011.

Shani, G., Heckerman, D., Brafman, R. I., and Boutilier, C. An mdp-based recommender system. *Journal of Machine Learning Research*, 6(9), 2005.

Stamenkovic, D., Karatzoglou, A., Arapakis, I., Xin, X., and Katevas, K. Choosing the best of both worlds: Diverse and novel recommendations through multi-objective reinforcement learning. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 957–965, 2022.

Sun, Y. and Zhang, Y. Conversational recommender system. In *Proceedings of the 41st International ACM Special Interest Group on Information Retrieval SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 235–244, 2018.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Tang, J. and Wang, K. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 565–573, 2018.

Tsymbal, A. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 106(2):58, 2004.

Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*, 2016.

Villa, A., Araujo, V., Cattan, F., and Parra, D. Interpretable contextual team-aware item recommendation: application in multiplayer online battle arena games. In *Proceedings of the 14th ACM Conference on Recommender Systems (RecSys)*, pp. 503–508, 2020.

Wang, Y. A hybrid recommendation for music based on reinforcement learning. In *Proceedings of the 24th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD)*, pp. 91–103, 2020.

Xiao, T. and Wang, D. A general offline reinforcement learning framework for interactive recommendation. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 4512–4520, 2021.

Xie, H., Wang, D. D., Rao, Y., Wong, T.-L., Raymond, L. Y., Chen, L., and Wang, F. L. Incorporating user experience into critiquing-based recommender systems: a collaborative approach based on compound critiquing. *International Journal of Machine Learning and Cybernetics*, 9: 837–852, 2018.

Xin, X., Karatzoglou, A., Arapakis, I., and Jose, J. M. Self-supervised reinforcement learning for recommender systems. In *Proceedings of the 43rd International ACM Special Interest Group on Information Retrieval SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 931–940, 2020.

Xin, X., Karatzoglou, A., Arapakis, I., and Jose, J. M. Supervised advantage actor-critic for recommender systems. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 1186–1196, 2022a.

Xin, X., Pimentel, T., Karatzoglou, A., Ren, P., Christakopoulou, K., and Ren, Z. Rethinking reinforcement learning for recommendation: A prompt perspective. In *Proceedings of the 45th International ACM Special Interest Group on Information Retrieval SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 1347–1357, 2022b.

Yuan, F., Karatzoglou, A., Arapakis, I., Jose, J. M., and He, X. A simple convolutional generative network for next item recommendation. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 582–590, 2019.

Zhang, Q., Cao, L., Zhu, C., Li, Z., and Sun, J. Coupledcf: Learning explicit and implicit user-item couplings in recommendation for deep collaborative filtering. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

Zhang, R., Yu, T., Shen, Y., Jin, H., Chen, C., and Carin, L. Reward constrained interactive recommendation with natural language feedback. In *Proceedings of the 33rd Conference on Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Zhao, K., Wang, X., Zhang, Y., Zhao, L., Liu, Z., Xing, C., and Xie, X. Leveraging demonstrations for reinforcement recommendation reasoning over knowledge graphs. In *Proceedings of the 43rd International ACM Special Interest Group on Information Retrieval SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 239–248, 2020a.

Zhao, X., Xia, L., Zhang, L., Ding, Z., Yin, D., and Tang, J. Deep reinforcement learning for page-wise recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys)*, pp. 95–103, 2018a.

Zhao, X., Zhang, L., Ding, Z., Xia, L., Tang, J., and Yin, D. Recommendations with negative feedback via pairwise deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1040–1048, 2018b.

Zhao, X., Xia, L., Zou, L., Liu, H., Yin, D., and Tang, J. Whole-chain recommendations. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 1883–1891, 2020b.

Zou, L., Xia, L., Ding, Z., Song, J., Liu, W., and Yin, D. Reinforcement learning to optimize long-term user engagement in recommender systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 2810–2818, 2019.

Zou, L., Xia, L., Du, P., Zhang, Z., Bai, T., Liu, W., Nie, J.-Y., and Yin, D. Pseudo dyna-q: A reinforcement learning framework for interactive recommendation. In *Proceedings of the 13th ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 816–824, 2020.

A. Proof Theorem 4.3

Let's consider a sequential recommendation problem with two items $\mathcal{I} = \{x_1, x_2\}$. In this problem, there are two possible interaction sequences: $\{x_1, x_2\}$ and $\{x_2\}$. For simplicity, let's consider that the training and testing sets are identical. This set $\mathcal{D}$ contains one trace $\{x_1, x_2\}$ and $n > 1$ copies of the trace $\{x_2\}$.

Given $\mathcal{D}$, an optimal solution f_* to the cross-entropy loss from Equation 1 – assuming that f_* has enough capacity to encode such a solution – is the following: $f_*(x_1|\emptyset) = \frac{1}{n+1}$, $f_*(x_2|\emptyset) = \frac{n}{n+1}$, and $f_*(x_2|\{x_1\}) = 1$. That is, f_* first recommends x_2 because most users in $\mathcal{D}$ interacted first with x_2. But if we are in a situation where the user interacts first with x_1, f_* recommends x_2 after that. As a result, the NIP performance of f_* over $\mathcal{D}$ is the following:

$$\mathcal{N}(\mathcal{D}, f_*) = \frac{1}{n+2}\left(n \cdot s(1) + s(2) + s(1)\right) = \frac{n+1}{n+2},$$

where the term $n \cdot s(1)$ refers to the n sequences $\{x_2\} \in \mathcal{D}$ and $s(2) + s(1)$ refers to the one sequence $\{x_1, x_2\} \in \mathcal{D}$ where the first x_1 is recommended last by f_* but then x_2 is correctly recommended first from x_1. Since $\mathcal{N}$ is consistent and this problem has only two items, $s(1) = 1$ and $s(2) = 0$.

On the other hand, an optimal policy π_* for $\mathcal{D}$ first recommends x_1 because the expected discounted return of recommending x_1 is $q_*(\emptyset, x_1) = r(1 + \gamma)$ whereas the Q-function of recommending x_2 is $q_*(\emptyset, x_2) = r$. Thus, as long as we define a positive reward r for interacting with an item and we use a discount of $\gamma > 0$, then $\pi_*(x_1|\emptyset) = 1$. And once the user interacts with x_1, the next recommendation will be x_2: $\pi_*(x_2|x_1) = 1$. Therefore, according to the NIP performance, π_* will fail at recommending x_1 instead of x_2 in n sequences in $\mathcal{D}$ (although, it will correctly recommend x_1 and then x_2 in the single example where that situation happens in $\mathcal{D}$):

$$\mathcal{N}(\mathcal{D}, \pi_*) = \frac{1}{n+2}\left(n \cdot s(2) + s(1) + s(1)\right) = \frac{2}{n+2},$$

Hence, the ratio between the scores of f_* and π_* is the following:

$$\frac{\mathcal{N}(\mathcal{D}, f_*)}{\mathcal{N}(\mathcal{D}, \pi_*)} = n + \frac{1}{2} > n.$$

Then, as we increase the value of n the optimal policy π_* can perform arbitrarily worse than f_* according to any consistent NIP metric.

B. Proof Theorem 4.4

Let $\mathcal{R}_\pi$ be a sequential recommender that ranks items according to a policy π and $\mathcal{R}_f$ be a sequential recommender that ranks items according to f_*. Let $C_\mathcal{D} : \mathcal{I}^* \to \mathbb{N}$ be a function that returns the number of times a subsequence appears in $\mathcal{D}$. In particular, $C_\mathcal{D}(x_{1:t})$ is equal to the number of sequences in $\mathcal{D}$ that begins with $x_{1:t}$. Then,

$$f_*(x_{1:t}, y) = \frac{C(x_{1:t} \circ y \mid \mathcal{D})}{C(x_{1:t} \mid \mathcal{D})} \quad \text{for all } y \in \mathcal{I}, \tag{6}$$

where $x \circ y$ represents the concatenation of x and y. That is, the probability of predicting y from $x_{1:t}$ is equal to the number of times that y is the next item after sequence $x_{1:t}$ in $\mathcal{D}$. Then, we can represent $\mathcal{N}$ in terms of the function C, where all the subsequences $x_s \subset \mathcal{D}$ are grouped together.

$$\mathcal{N}(\mathcal{D}, \mathcal{R}) = \frac{1}{N} \sum_{x_s \subset \mathcal{D}} \sum_{y \in \mathcal{I}} C(x_s \circ y \mid \mathcal{D}) \cdot s(\mathcal{R}(x_s, y))$$

Let $\mathcal{N}(\mathcal{D}, \mathcal{R}, x_s)$ be the following:

$$\mathcal{N}(\mathcal{D}, \mathcal{R}, x_s) = \sum_{y \in \mathcal{I}} C(x_s \circ y \mid \mathcal{D}) \cdot s(\mathcal{R}(x_s, y))$$

Then, we will prove that, for all subsequence $x_s \subset \mathcal{D}$:

$$\mathcal{N}(\mathcal{D}, \mathcal{R}_f, x_s) = \sum_{y \in \mathcal{I}} C(x_s \circ y \mid \mathcal{D}) \cdot s(\mathcal{R}_f(x_s, y)) \geq \sum_{y \in \mathcal{I}} C(x_s \circ y \mid \mathcal{D}) \cdot s(\mathcal{R}_\pi(x_s, y)) = \mathcal{N}(\mathcal{D}, \mathcal{R}_\pi, x_s) \tag{7}$$

As stated in Equation 6, $\mathcal{R}_f(x_s, y)$ ranks the items according to $C(x_s \circ y \mid \mathcal{D})$. The item with higher counter $C(x_s \circ y \mid \mathcal{D})$ is ranked first. Then, the item with the second higher count and so for. In addition, the scoring function s is non-increasing because $\mathcal{N}$ is consistent.

We will prove that Equation 7 holds by showing that no other ranking could achieve a higher $\mathcal{N}(\mathcal{D}, \mathcal{R}_f, x_s)$ value than $\mathcal{R}_f$, for any x_s. We will do so by contradiction. Let's assume that there exists a policy π such that its ranking $\mathcal{R}_\pi(x_s, \cdot)$ for the next item given the interaction sequence x_s has higher $\mathcal{N}(\mathcal{D}, \mathcal{R}_\pi, x_s)$ value. For now, let's consider that the only difference between the rankings $\mathcal{R}_f(x_s, \cdot)$ and $\mathcal{R}_\pi(x_s, \cdot)$ is in the location of two items y_1 and y_2 that are swapped. That is, $\mathcal{R}_f(x_s, y_1) = \mathcal{R}_\pi(x_s, y_2)$ and $\mathcal{R}_f(x_s, y_2) = \mathcal{R}_\pi(x_s, y_1)$. Let's say that f_* ranks y_1 higher than y_2. Let p_1^f be the position of y_1 according to f_* and p_2^f be the position of y_2. Conversely, let p_1^π be the position of y_1 according to π and p_2^π be the position of y_2.

Since f_* ranks y_1 higher than y_2, then $C_1 = C(x_s \circ y_1 \mid \mathcal{D}) \geq C(x_s \circ y_2 \mid \mathcal{D}) = C_2$. Therefore,

$$C_1 = C_2 + \epsilon \, ,$$

for some $\epsilon \geq 0$. In addition, we know that the scoring function s is non-increasing. That means that:

$$s(p_1^f) = s(p_2^f) + \beta \, ,$$

where $\beta \geq 0$. We now prove that the swap cannot increase the value of $\mathcal{N}(\mathcal{D}, \mathcal{R}_f, x_s)$.

$$\begin{aligned}
C_1 \cdot s(p_1^f) + C_2 \cdot s(p_2^f) &= (C_2 + \epsilon) \cdot (s(p_2^f) + \beta) + C_2 \cdot s(p_2^f) && (8)\\
&= C_2 \cdot s(p_2^f) + C_2 \cdot \beta + \epsilon \cdot s(p_2^f) + C_2 \cdot s(p_2^f) + \epsilon \cdot \beta && (9)\\
&\geq C_2 \cdot s(p_2^f) + C_2 \cdot \beta + \epsilon \cdot s(p_2^f) + C_2 \cdot s(p_2^f) && (10)\\
&= C_2 \cdot s(p_2^f) + C_2 \cdot \beta + \epsilon \cdot s(p_2^f) + C_2 \cdot s(p_2^f) && (11)\\
&= C_2 \cdot (s(p_2^f) + \beta) + s(p_2^f) \cdot (C_2 + \epsilon) && (12)\\
&= C_2 \cdot s(p_1^f) + s(p_2^f) \cdot C_1 && (13)\\
&= C_2 \cdot s(p_2^\pi) + s(p_1^\pi) \cdot C_1 && (14)\\
& && (15)
\end{aligned}$$

Thus, swapping the order of y_1 and y_2 cannot increase the value of $\mathcal{N}(\mathcal{D}, \mathcal{R}_f, x_s)$. And, for the same reason, making more than one swaps cannot increase the value of $\mathcal{N}(\mathcal{D}, \mathcal{R}_f, x_s)$. This means that, regardless the policy π, $\mathcal{N}(\mathcal{D}, \mathcal{R}_f, x_s) \geq \mathcal{N}(\mathcal{D}, \mathcal{R}_\pi, x_s)$. And since this relation holds for all $x_s \subset \mathcal{D}$, $\mathcal{N}(\mathcal{D}, f_*) \geq \mathcal{N}(\mathcal{D}, \pi_*)$ – proving the theorem.

C. Parameter Setting

Table 5. Model parameters used in training. **Batch**: Batch size used. **lr**: learning rate. **h_factor**: Hidden factor or item embedding size. **filter#** Number of horizontal filters used in Caser. **f_sizes**: The size of the horizontal filters in Caser. **Head#**: Number of heads in self-attention in SASRec. **dropout**: Dropout Rate. **CR**: Click Reward. **BR**: Buy Reward

Model	Optimizer	Epochs	Batch	lr	γ	h_factor	filter#	f_sizes	Head#	dropout	CR	BR
GRU	Adam	50	256	0.005	0.5	64	-	-	-	0	1	5
NIN	Adam	50	256	0.005	0.5	64	-	-	-	0	1	5
Caser	Adam	50	256	0.005	0.5	64	16	[2,3,4]	-	0.1	1	5
SAS	Adam	50	256	0.005	0.5	64	-		1	0.1	1	5

D. Complete Results

Table 6. Top-k recommendation performance comparison for **click** interactions on different models on the **RetailRocket** dataset. NG is short for NDCG. Boldface denotes the highest performance and underlines denote the runner-up. * denotes that the model outperforms the self-supervised baseline with a significance p-value < 0.05.

MODEL	HR@5	NG@5	HR@10	NG@10	HR@15	NG@15	HR@20	NG@20
GRU	0.1205	0.0938	0.1472	0.1024	0.1633	0.1067	0.1751	0.1094
GRU-SQN	*0.1416	*0.1105	*0.1705	*0.1199	*0.1874	*0.1244	*0.1999	*0.1273
GRU-SAC	***0.1475**	***0.1148**	***0.1774**	***0.1245**	**0.1947**	***0.1291**	***0.2069**	***0.1320**
GRU-EVAL	*0.1426	*0.1109	*0.1714	*0.1203	*0.1888	*0.1249	*0.2009	*0.1277
GRU-CAT	*0.1228	*0.0955	*0.1497	*0.1042	*0.1665	*0.1086	*0.1784	*0.1114
GRU-CAT3	*0.1249	*0.0969	*0.1514	*0.1054	*0.1677	*0.1097	*0.1797	*0.1126
GRU-HIST	*0.1434	*0.1118	*0.1717	*0.1210	*0.1880	*0.1253	*0.1997	*0.1281
GRU-FUT	0.0390	0.0312	0.0474	0.0339	0.0528	0.0353	0.0569	0.0375
NIN	0.1345	0.1059	0.1612	0.1145	0.1774	0.1188	0.1892	0.1216
NIN-SQN	***0.1673**	***0.1310**	*0.1996	***0.1414**	*0.2178	***0.1463**	*0.2305	***0.1493**
NIN-SAC	*0.1671	*0.1301	***0.1999**	*0.1407	***0.2186**	*0.1457	***0.2317**	*0.1488
NIN-EVAL	*0.1668	*0.1306	*0.1993	*0.1411	*0.2176	*0.1459	*0.2302	*0.1489
NIN-CAT	*0.1431	*0.1121	*0.1721	*0.1215	*0.1890	*0.1259	*0.2014	*0.1289
NIN-CAT3	*0.1436	*0.1129	*0.1732	*0.1225	0.1903	*0.1271	*0.2027	*0.1300
NIN-HIST	*0.1638	*0.1284	*0.1951	*0.1386	*0.2130	*0.1433	*0.2256	0.1463
NIN-FUT	0.0939	0.0750	0.1100	0.0802	0.1194	0.0827	0.1263	0.0830
CASER	0.1400	0.1107	0.1640	0.1185	0.1781	0.1222	0.1877	0.1245
CASER-SQN	*0.1560	0.1218	*0.1849	*0.1312	*0.2018	*0.1357	*0.2136	*0.1385
CASER-SAC	*0.1539	0.1190	*0.1836	*0.1286	*0.2012	*0.1333	*0.2132	*0.1361
CASER-EVAL	0.1577	0.1246	0.1849	0.1334	0.2005	0.1375	0.2116	0.1401
CASER-CAT	0.1415	0.1135	0.1652	0.1212	0.1790	0.1249	0.1888	0.1344
CASER-CAT3	0.1565	*0.1241	*0.1830	*0.1327	*0.1982	*0.1367	0.2090	*0.1393
CASER-HIST	**0.1669**	***0.1300**	***0.1979**	**0.1401**	***0.2160**	***0.1449**	***0.2283**	***0.1478**
CASER-FUT	0.0248	0.0189	0.0316	0.0210	0.0358	0.0222	0.0393	0.0230
SAS	0.1635	0.1249	0.1982	0.1361	0.2176	0.1413	0.2313	0.1445
SAS-SQN	0.1835	*0.1397	*0.2228	*0.1524	*0.2445	*0.1582	*0.2597	*0.1618
SAS-SAC	***0.1852**	***0.1398**	***0.2266**	***0.1532**	***0.2496**	***0.1593**	***0.2649**	***0.1629**
SAS-EVAL	0.1815	*0.1382	*0.2212	*0.1511	*0.2433	*0.1569	*0.2584	0.1605
SAS-CAT	*0.1689	*0.1285	*0.2057	*0.1404	*0.2259	*0.1458	*0.2399	*0.1491
SAS-CAT3	0.1669	0.1271	*0.2034	0.1390	*0.2239	0.1444	*0.2380	0.1477
SAS-HIST	*0.1812	*0.1376	*0.2204	*0.1503	*0.2424	*0.1561	*0.2576	*0.1597
SAS-FUT	0.0099	0.0071	0.0138	0.0083	0.0168	0.0091	0.0193	0.0097

Table 7. Top-k recommendation performance comparison for **buy** interactions on different models on the **RetailRocket** dataset. NG is short for NDCG. Boldface denotes the highest performance and underlines denote the runner-up. * denotes that the model outperforms the self-supervised baseline with a significance p-value < 0.05.

MODEL	HR@5	NG@5	HR@10	NG@10	HR@15	NG@15	HR@20	NG@20
GRU	0.1601	0.1248	0.1932	0.1355	0.2151	0.1413	0.2306	0.1456
GRU-SQN	*0.1921	0.1519	*0.2304	0.1643	*0.2531	0.1703	*0.2698	0.1743
GRU-SAC	0.1973	*0.1546	*0.2338	*0.1664	*0.2575	*0.1727	*0.2734	*0.1764
GRU-EVAL	*0.1962	*0.1545	*0.2333	*0.1664	*0.2555	*0.1723	*0.2718	*0.1762
GRU-CAT	0.1644	0.1282	*0.2004	*0.1400	*0.2225	0.1457	*0.2384	0.1495
GRU-CAT3	*0.1696	*0.1325	*0.2044	*0.1438	*0.2272	*0.1498	*0.2435	*0.1536
GRU-HIST	***0.1980**	***0.1549**	***0.2352**	***0.1670**	***0.2588**	***0.1732**	***0.2747**	***0.1770**
GRU-FUT	0.0390	0.0312	0.0474	0.0339	0.0528	0.0353	0.0569	0.0375
NIN	0.2282	0.1785	0.2746	0.1935	0.3018	0.2007	0.3215	0.2053
NIN-SQN	*0.3307	*0.2577	***0.3890**	*0.2767	*0.4200	*0.2849	*0.4418	*0.2901
NIN-SAC	*0.3303	***0.2594**	*0.3880	***0.2781**	***0.4201**	***0.2867**	***0.4422**	***0.2919**
NIN-EVAL	***0.3308**	*0.2582	*0.3885	*0.2770	*0.4199	*0.2853	*0.4421	*0.2905
NIN-CAT	*0.2505	0.1959	*0.2998	0.2119	*0.3280	0.2194	*0.3483	0.2242
NIN-CAT3	*0.2543	*0.1973	*0.3046	*0.2136	*0.3346	*0.2215	*0.3553	*0.2264
NIN-HIST	*0.3001	*0.2348	*0.3566	*0.2529	*0.3896	*0.2616	*0.4110	*0.2667
NIN-FUT	0.1980	0.1581	0.2301	0.1685	0.2490	0.1735	0.2623	0.1767
CASER	0.1682	0.1361	0.1947	0.1446	0.2105	0.1488	0.2217	0.1514
CASER-SQN	*0.2020	*0.1601	*0.2367	*0.1713	*0.2568	*0.1766	*0.2715	*0.1801
CASER-SAC	*0.1979	*0.1559	*0.2237	*0.1684	*0.2581	*0.1741	*0.2737	*0.1778
CASER-EVAL	0.2000	0.1610	*0.2321	0.1714	*0.2505	0.1762	*0.2638	0.1794
CASER-CAT	0.1765	0.1434	0.2032	0.1521	0.2193	0.1564	0.2307	0.1591
CASER-CAT3	0.1936	*0.1565	*0.2254	*0.1668	*0.2420	*0.1712	*0.2541	*0.1740
CASER-HIST	***0.2399**	***0.1893**	***0.2857**	***0.2041**	***0.3112**	***0.2108**	***0.3296**	***0.2152**
CASER-FUT	0.0299	0.0222	0.0378	0.0247	0.0435	0.0262	0.0481	0.0273
SAS	0.2458	0.1872	0.2995	0.2046	0.3283	0.2123	0.3509	0.2176
SAS-SQN	*0.3012	*0.2280	*0.3657	*0.2490	*0.3989	*0.2577	*0.4227	*0.2634
SAS-SAC	***0.3143**	***0.2379**	***0.3810**	***0.2596**	***0.4153**	***0.2687**	***0.4382**	***0.2741**
SAS-EVAL	0.2963	*0.2242	*0.3619	*0.2454	*0.3962	*0.2545	*0.4195	*0.2600
SAS-CAT	*0.2636	*0.1999	*0.3190	*0.2178	*0.3504	*0.2261	*0.3731	*0.2315
SAS-CAT3	0.2572	0.1948	*0.3130	0.2129	*0.3437	0.2211	*0.3651	0.2261
SAS-HIST	*0.2960	*0.2231	0.3315	*0.2428	0.3630	*0.2519	0.3847	*0.2573
SAS-FUT	0.0207	0.0144	0.0285	0.0170	0.0349	0.0187	0.0400	0.0199

Table 8. Top-k recommendation performance comparison for **click** interactions on different models on the **RC15** dataset. NG is short for NDCG. Boldface denotes the highest performance and underlines denote the runner-up. * denotes that the model outperforms the self-supervised baseline with a significance p-value < 0.05.

MODEL	HR@5	NG@5	HR@10	NG@10	HR@15	NG@15	HR@20	NG@20
GRU	0.2676	0.1840	0.3527	0.2116	0.3976	0.2235	0.4274	0.2306
GRU-SQN	**0.2923**	0.2030	**0.3818**	**0.2320**	0.4268	0.2439	**0.4569**	***0.2510**
GRU-EVAL	0.2918	**0.2065**	0.3812	0.2319	**0.4272**	**0.2441**	0.4567	0.2509
GRU-CAT	0.2822	0.1953	0.3707	0.2240	0.4159	0.2360	0.4450	0.2429
GRU-HIST	0.2723	0.1882	0.3598	0.2166	0.4052	0.2285	0.4355	0.2358
NIN	0.2718	0.1883	0.3567	0.2158	0.4003	0.2274	0.4294	0.2343
NIN-SQN	***0.2985**	*0.2067	***0.3900**	***0.2363**	***0.4365**	***0.2487**	***0.4667**	***0.2558**
NIN-EVAL	0.2981	***0.2068**	*0.3876	*0.2358	*0.4330	*0.2479	*0.4625	*0.2548
NIN-CAT	0.2868	*0.1992	*0.3740	*0.2274	0.4186	*0.2392	*0.4479	*0.2461
NIN-HIST	*0.2953	*0.2048	*0.3856	0.2330	*0.4323	*0.2465	*0.4623	*0.2536
CASER	0.2734	0.1898	0.3580	0.2172	0.4021	0.2289	0.4315	0.2358
CASER-SQN	*0.2767	*0.1916	*0.3629	*0.2196	*0.4082	*0.2316	*0.4387	*0.2388
CASER-EVAL	*0.2761	*0.1915	*0.3622	0.2194	*0.4074	*0.2314	*0.4372	*0.2384
CASER-CAT	*0.2771	*0.1924	*0.3628	*0.2202	*0.4080	*0.2321	*0.4375	*0.2391
CASER-HIST	***0.2834**	***0.1959**	***0.3742**	***0.2253**	***0.4219**	***0.2379**	***0.4535**	***0.2454**
SAS	0.3239	0.2246	0.4201	0.2559	0.4668	0.2683	0.4962	0.2752
SAS-SQN	*0.3304	0.2290	*0.4272	0.2605	*0.4751	*0.2732	*0.5052	*0.2803
SAS-EVAL	0.3272	0.2268	*0.4243	0.2583	*0.4714	0.2708	*0.5018	0.2780
SAS-CAT	0.3218	0.2225	0.4177	0.2537	0.4652	0.2663	0.4956	0.2734
SAS-HIST	***0.3333**	***0.2318**	***0.4297**	***0.2631**	***0.4769**	***0.2756**	***0.5067**	***0.2827**

Table 9. Top-k recommendation performance comparison for **buy** interactions on different models on the **RC15** dataset. NG is short for NDCG. Boldface denotes the highest performance and underlines denote the runner-up. * denotes that the model outperforms the self-supervised baseline with a significance p-value < 0.05.

MODEL	HR@5	NG@5	HR@10	NG@10	HR@15	NG@15	HR@20	NG@20
GRU	0.3599	0.2511	0.4685	0.2864	0.5240	0.3010	0.5612	0.3098
GRU-SQN	0.3736	0.2626	0.4832	0.2983	0.5394	0.3132	0.5761	0.3218
GRU-EVAL	0.3704	0.2620	0.4819	0.2983	0.5403	0.3137	0.5776	0.3225
GRU-CAT	0.3579	0.2491	0.4653	0.2840	0.5190	0.2982	0.5551	0.3068
GRU-HIST	***0.3841**	***0.2714**	***0.4972**	***0.3080**	***0.5527**	***0.3228**	***0.5901**	***0.3316**
NIN	0.3208	0.2243	0.4217	0.2571	0.4724	0.2707	0.5054	0.2783
NIN-SQN	***0.3610**	***0.2548**	***0.4703**	***0.2903**	***0.5234**	***0.3044**	***0.5606**	***0.3132**
NIN-EVAL	*0.3483	*0.2463	*0.4463	0.2782	*0.4994	*0.2927	0.5342	*0.3021
NIN-CAT	*0.3381	*0.2384	*0.4368	*0.2699	*0.4912	*0.2842	*0.5253	0.2923
NIN-HIST	*0.3558	*0.2520	*0.4609	*0.2868	*0.5152	*0.3004	*0.5521	*0.3092
CASER	0.4095	0.2898	0.5251	0.3272	0.5816	0.3422	0.6192	0.3511
CASER-SQN	*0.4263	*0.3014	*0.5394	*0.3389	*0.5990	*0.3538	*0.6342	*0.3621
CASER-EVAL	*0.4340	**0.3255**	*0.5512	*0.3477	*0.6071	*0.3632	*0.6448	*0.3714
CASER-CAT	*0.4351	*0.3106	*0.5462	*0.3468	*0.6016	*0.3611	*0.6386	*0.3699
CASER-HIST	***0.4528**	*0.3197	***0.5724**	***0.3586**	***0.6317**	***0.3743**	***0.6686**	***0.3830**
SAS	0.3706	0.2596	0.4741	0.2933	0.5262	0.3070	0.5598	0.3150
SAS-SQN	0.3831	0.2689	0.4920	*0.3062	0.5462	*0.3185	0.5805	*0.3266
SAS-EVAL	***0.4115**	***0.2872**	***0.5275**	***0.3245**	***0.5828**	***0.3391**	***0.6216**	***0.3483**
SAS-CAT	*0.4049	*0.2804	*0.5192	*0.3178	*0.5764	*0.3330	*0.6159	*0.3423
SAS-HIST	0.3741	0.2634	0.4795	0.2979	0.5289	0.3108	0.5634	0.3189

E. Feature Importance Analysis

Table 10. Initial list of features before filtering

Feature	Description
Interaction	Categorical feature denoting the interaction type (click, buy) at the target timestamp
Interaction_2	Categorical feature denoting the interaction type (click, buy, done) at timestamp $t+1$
Interaction_3	Categorical feature denoting the interaction type (click, buy, done) at timestamp $t+2$
Is_done	Binary feature denoting whether the sequence finishes at timestamp t
hist-length	Number of past user interactions in the sequence.
fut-length	Number of future user interactions in the sequence.
total-length	Number of interactions in the complete sequence.
Q-Value (eval)	The expected return following the sequence in the history log.
hist-buys	Number of items the user bought in past interactions.
fut-buys	Number of items the user will buy in future interactions.
Steps2Buy	Number of steps until the next buy interaction in the sequence.

Table 11. Feature importance analysis for each SQN model on the RetailRocket dataset. β represents the feature coefficient on the linear regression model, SHAP-LR is the SHAP score using a linear regression model, and SHAP-XG is the SHAP score using XGBoost. Boldface denotes the highest importance score under each metric

GRU-SQN				NextItNet-SQN			
Feature	β	SHAP-LR	SHAP-XG	Feature	β	SHAP-LR	SHAP-XG
hist-length	**0.0711**	**0.1446**	**0.0998**	hist-length	**0.0339**	**0.0672**	0.0916
fut-length	0.0342	0.0721	0.0445	fut-length	0.0327	0.0653	**0.1085**
Q-Value	0.0091	0.0043	0.0284	Q-Value	0.0292	0.0146	0.0291
hist-buys	0.0352	0.0458	0.0570	hist-buys	0.0139	0.0182	0.0817
fut-buys	0.0097	0.0114	0.0370	fut-buys	0.0246	0.0314	0.0598
Caser-SQN				SASRec-SQN			
Feature	β	SHAP-LR	SHAP-XG	Feature	β	SHAP-LR	SHAP-XG
hist-length	**7.8921**	**15.5814**	**18.4964**	hist-length	**0.0638**	**0.1297**	**0.0696**
fut-length	1.8460	3.5634	5.8296	fut-length	0.0261	0.0546	0.0533
Q-Value	0.5914	0.2555	1.7878	Q-Value	0.0159	0.0072	0.0252
hist-buys	5.7520	7.2642	2.8757	hist-buys	0.0282	0.0371	0.0391
fut-buys	2.1184	2.6344	2.7714	fut-buys	0.0110	0.0141	0.0321

F. RL-Based Loss

Table 12. Top-k recommendation performance comparison on the RetailRocket dataset between the baseline, self-supervised models, their SQN counterpart and the model using only the RL-based loss for recommendations.

clicks								
Model	HR@5	NG@5	HR@10	NG@10	HR@15	NG@15	HR@20	NG@20
GRU	0.1205	0.0938	0.1472	0.1024	0.1633	0.1067	0.1751	0.1094
GRU-SQN	0.1416	0.1105	0.1705	0.1199	0.1874	0.1244	0.1999	0.1273
GRU-RL	0.0001	0.0001	0.0001	0.0001	0.0003	0.0001	0.0004	0.0001
NIN	0.1345	0.1059	0.1612	0.1145	0.1774	0.1188	0.1892	0.1216
NIN-SQN	0.1673	0.1310	0.1996	0.1414	0.2178	0.1463	0.2305	0.1493
NIN-RL	0.0014	0.0009	0.0020	0.0011	0.0026	0.0012	0.0033	0.0014
Caser	0.1400	0.1107	0.1640	0.1185	0.1781	0.1222	0.1877	0.1245
Caser-SQN	0.1560	0.1218	0.1849	0.1312	0.2018	0.1357	0.2136	0.1385
Caser-RL	0.0017	0.0014	0.0020	0.0015	0.0023	0.0015	0.0031	0.0017
SAS	0.1635	0.1249	0.1982	0.1361	0.2176	0.1413	0.2313	0.1445
SAS-SQN	0.1835	0.1397	0.2228	0.1524	0.2445	0.1582	0.2597	0.1618
SAS-RL	0.0000	0.0000	0.0001	0.0000	0.0001	0.0000	0.0002	0.0001
buys								
GRU	0.1205	0.0938	0.1472	0.1024	0.1633	0.1067	0.1751	0.1094
GRU-SQN	0.1416	0.1105	0.1705	0.1199	0.1874	0.1244	0.1999	0.1273
GRU-RL	0.0000	0.0000	0.0000	0.0000	0.0001	0.0000	0.0003	0.0001
NIN	0.2282	0.1785	0.2746	0.1935	0.3018	0.2007	0.3215	0.2053
NIN-SQN	0.3307	0.2577	0.3890	0.2767	0.4200	0.2849	0.4418	0.2901
NIN-RL	0.0009	0.0005	0.0014	0.0006	0.0016	0.0007	0.0020	0.0008
Caser	0.1400	0.1107	0.1640	0.1185	0.1781	0.1222	0.1877	0.1245
Caser-SQN	0.1560	0.1218	0.1849	0.1312	0.2018	0.1357	0.2136	0.1385
Caser-RL	0.0025	0.0021	0.0027	0.0022	0.0033	0.0024	0.0042	0.0026
SAS	0.2458	0.1872	0.2995	0.2046	0.3283	0.2123	0.3509	0.2176
SAS-SQN	0.3012	0.2280	0.3657	0.2490	0.3989	0.2577	0.4227	0.2634
SAS-RL	0.0000	0.0000	0.0001	0.0000	0.0001	0.0000	0.0001	0.0000