Improving Sequential Recommenders through Counterfactual Augmentation of System Exposure

Ziqi Zhao
Shandong University
Qingdao, China
ziqizhao.work@gmail.com

Zhaochun Ren*
Leiden University
Leiden, The Netherlands
z.ren@liacs.leidenuniv.nl

Jiyuan Yang
Shandong University
Qingdao, China
jiyuan.yang@mail.sdu.edu.cn

Zuming Yan
Shandong University
Qingdao, China
202200450040@mail.sdu.edu.cn

Zihan Wang
University of Amsterdam
Amsterdam, The Netherlands
zhw.cypher@gmail.com

Liu Yang
Shandong University
Qingdao, China
yangliushirry@gmail.com

Pengjie Ren
Shandong University
Qingdao, China
renpengjie@sdu.edu.cn

Zhumin Chen
Shandong University
Qingdao, China
chenzhumin@sdu.edu.cn

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

Xin Xin*
Shandong University
Qingdao, China
xinxin@sdu.edu.cn

Abstract

In sequential recommendation, system exposure refers to items that are exposed to the user. Typically, the user only interactions with a few of the exposed items. Although sequential recommendation has achieved great success in predicting future user interests, existing sequential recommendation methods do not fully exploit system exposure data. Most methods only model items that have been interacted with, while the large volume of exposed but non-interacted items is overlooked. Even methods that consider system exposure typically train the recommender using only the logged historical system exposure, without exploring unseen user interests.

In this paper, we propose **c**ounterfactual **a**ugmentation over **s**ystem **e**xposure for sequential **rec**ommendation (CaseRec). To better model historical system exposure, CaseRec introduces reinforcement learning to account for different exposure rewards. CaseRec uses a decision transformer-based sequential model to take an exposure sequence as input and assigns different rewards according to the user feedback. To explore unseen user interests, CaseRec performs counterfactual augmentation, where exposed items are replaced with counterfactual items. Then, a transformer-based user simulator is used to predict the user feedback reward for the augmented items. Augmentation, together with the user simulator, gives rise to counterfactual exposure sequences to uncover new user interests. Finally, CaseRec uses the logged exposure sequences with the counterfactual exposure sequences to train a decision transformer-based sequential model for generating recommendation. Experiments on three real-world benchmarks show the effectiveness of CaseRec. Our code is available at https://github.com/ZiqiZhao1/CaseRec.

CCS Concepts

• **Information systems** → **Recommender systems**; **Retrieval models and ranking**; **Novelty in information retrieval**.

Keywords

Sequential recommendation, System exposure, Reinforcement learning, Data augmentation, Decision transformer

ACM Reference Format:
Ziqi Zhao, Zhaochun Ren, Jiyuan Yang, Zuming Yan, Zihan Wang, Liu Yang, Pengjie Ren, Zhumin Chen, Maarten de Rijke, and Xin Xin. 2025. Improving Sequential Recommenders through Counterfactual Augmentation of System Exposure. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25), July 13–18, 2025, Padua, Italy*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3726302.3730005

1 Introduction

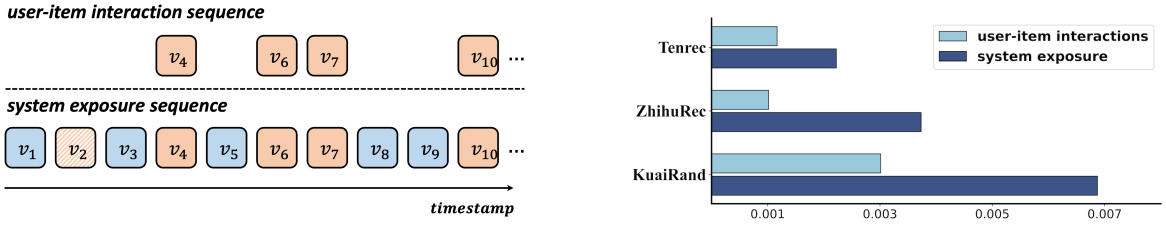
Sequential recommendation (SR) aims to predict future user interests by modeling past user-item interaction sequences [23, 25, 59, 63]. Unlike traditional collaborative filtering methods, SR models the user-item interactions as a dynamic sequence and uses (deep) sequential models to capture inherent sequential dependencies, enabling better recommendation [19, 42].

System exposure. In the interaction process between users and the recommender system, the recommender exposes a personalized list of items, from which users select interesting items to interact with. Such a system behavior sequence is referred to as *system exposure* [31, 55]. Figure 1(a) provides an example of system exposure. Figure 1(b) illustrates that users typically interact with only a few

*Corresponding authors.

SIGIR '25, Padua, Italy

ACM ISBN 979-8-4007-1592-1/2025/07
https://doi.org/10.1145/3726302.3730005



(a) Example of system exposure.

(b) Density across three datasets.

Figure 1: (a) Illustration of system exposure. Orange blocks refer to items interacted with by the user, while blue blocks refer to items exposed by the system but not interacted with by the user. The shadow item v_2 denotes that v_2 could also interest the user but is not interacted with, e.g., due to time limits. (b) System exposure is much larger than interacted data.

items from the exposed list. System exposure is generated according to recommendation policies that represent the user preference from the point of view of the recommender system. Such exposure information can contribute to improving future recommendation.

Limitations of existing methods. Although SR has achieved great success, we argue that existing SR methods fail to fully exploit system exposure data, which limits their potential and leads to sub-optimal recommendation performance.

First, most SR methods only model items that have been interacted with, while the large volume of exposed but non-interacted items is overlooked. SR methods typically assume that exposed but non-interacted items denote negative user preference [11]. But this assumption may not always hold. As an example in Figure 1(a), the shadow item v_2 denotes that item v_2 could also interest the user but is not interacted with since the user can only select one item to interact with during a limited time period. Exposed but non-interacted items may contain potential user interests. The large volume of exposure sequences helps us to better learn sequential correlations between items, and, thus, has the potential to further improve SR. Exposure sequences, including both interacted and non-interacted items, should be modeled for SR.

Second, even studies that account for system exposure typically only train the recommender using logged historical exposure and ignore unseen user interests, leading to increased exposure bias [5]. Since training SR models depends only on the logged exposure, previously deployed exposure policies may dramatically impact user-item interactions. Exposure debiasing methods introduce penalties based on inverse propensity score (IPS) [9, 10, 24, 33, 41, 50, 52, 54] or distributionally robust optimization (DRO) [55]. However, penalties on inappropriate items could negatively impact model performance. Counterfactual exposure sequences are an alternative solution to uncover new user interests and alleviate exposure bias.

Proposed methods. We propose **c**ounterfactual **a**ugmentation over **s**ystem **e**xposure for sequential **rec**ommendation (CaseRec) to address the limitations mentioned above.

To better model the whole exposure sequence, we consider different types of user feedback on exposed items. CaseRec introduces a reinforcement learning (RL)-based model to learn exposure sequences. RL has been proven to be effective in achieving reward-conditioned learning, which aligns well with modeling exposure sequences, i.e., by assigning different rewards to different types of user feedback. CaseRec uses a decision transformer (DT)-based sequential model to learn exposure sequences. The DT casts the offline RL task as a sequence modeling problem and has the ability to handle complex long sequences. To adapt the DT for modeling exposure sequences, CaseRec takes items and the corresponding user behavior as the input. It introduces a high-dimensional encoder that combines item embeddings and corresponding behavior embeddings to distinguish between different types of user feedback on the exposure sequence. Finally, CaseRec redesigns the learning objective to ensure that the model can accurately predict the next item that the user is likely to interact with.

To augment counterfactual exposure sequences and uncover new user interests, CaseRec features two augmentation strategies, *Random* and *Self-improving*, that replace part of the exposed items in a logged exposure sequence with other items. Then, a transformer-based user simulator predicts the user feedback for the augmented items. The *Random* strategy uses a simple yet effective uniform sampling strategy to replace items with randomly sampled new items, aiming at simulating user feedback under the random exposure policy and increasing the diversity of the training sequence. The *Self-improving* strategy conducts a small perturbation on a historical exposed item, and uses the current SR model to generate the following exposure sequence. The generated exposure sequences are used to train the SR model. This cycle allows the SR model to explore more of the item space near the historical exposed items. Counterfactual item replacement, together with the user simulator, constructs counterfactual exposure sequences to uncover potential user interests. Finally, CaseRec uses logged exposure sequences and counterfactual augmented sequences to train a DT-based sequential model for generating recommendations.

To demonstrate the effectiveness of CaseRec, we conduct extensive experiments on benchmark datasets. Experimental results show that CaseRec outperforms relevant state-of-the-art SR baselines and shows less exposure bias in the recommendation list.

Main contributions. Our main contributions are:

- We propose CaseRec, which uses an offline RL-based DT model to learn exposure sequences and account for various types of user feedback over exposed items.
- We propose two counterfactual augmentation strategies over system exposure to further uncover potential new user interests.
- We propose a transformer-based user simulator to imitate user feedback for the augmented counterfactual items.
- Experiments on three datasets show the effectiveness of CaseRec for generating more accurate recommendations. CaseRec also shows promising performance in reducing exposure bias.

2 Preliminaries

2.1 System exposure

The items exposed to users by a recommendation agent are referred to as system exposure. Let $\mathcal{U}$ and $\mathcal{V}$ denote the user set and the item set, respectively, where $u \in \mathcal{U}$ denotes a user and $v \in \mathcal{V}$ denotes an item. Let $\mathcal{B}$ denote the user feedback to an exposed item (e.g., clicks, views or purchases). An n-length system exposure sequence of user u can be denoted as $S^u = (v_1^u, b_1^u, v_2^u, b_2^u, \ldots, v_n^u, b_n^u)$, where v_j^u represents the j-th item exposed to user u, $b_j^u \in \mathcal{B}$ represents the feedback of user u to item v_j^u. We assume that $\mathcal{B} = \{0, 1\}$, where 1 denotes an item that has been interacted with by the user; $\mathcal{B}$ can contain additional user feedback for fine-grained user modeling.

Most SR methods only consider interacted item sequences, i.e., items with $b_j^u = 1$, while other items with $b_j^u = 0$ are overlooked. However, interacted items only account for a small portion of exposed items, and both interacted and non-interacted items should be modeled for better SR. We need to consider different types of user feedback on exposed items, which is accomplished through introducing RL-based models in this work.

2.2 Offline reinforcement learning

RL has achieved great success in conducting reward-driven decision-making. Different rewards can be assigned according to various user feedback, which naturally fits the modeling of exposure sequences.

A Markov decision process (MDP) is defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}, \mathcal{A}$ refer to a state space and action space, respectively. Given a state $s_t \in \mathcal{S}$ and action $a_t \in \mathcal{A}$ at timestep t, $P(s_{t+1}|s_t, a_t)$ and $r(s_t, a_t)$ represent the distribution of the next state s_{t+1} and the obtained immediate reward, respectively. $\gamma \in (0, 1)$ is the discount factor for future rewards. The learning objective of RL is to find an optimal policy that maximizes the expected cumulative rewards $\mathbb{E}[\sum_t \gamma^t r(s_t, a_t)]$.

Conventional RL algorithms collect the training data and learn the policy through online explorations with the environment (i.e., the user), which would affect the user experience when the policy is under-trained. To solve this problem, we use offline RL methods in this work. Offline RL learns a policy from an offline trajectory dataset pre-collected with unknown behavior policies, without online environment explorations.

The decision transformer (DT) is one of the most successful offline RL algorithms, which casts the offline RL task as a conditional sequence modeling problem. The DT possesses both the capability to handle complex long sequences and the RL characteristic of reward-conditioned learning. Unlike traditional RL approaches that estimate value functions or compute policy gradients, The DT auto-regressively generates desired future actions by modeling the trajectories of returns-to-go (RTG, i.e., expected future rewards), states s and actions a. The input trajectory of the DT is formulated as

$$\tau = (\hat{R}_1, s_1, a_1, \ldots, \hat{R}_{T-1}, s_{T-1}, a_{T-1}, \hat{R}_T, s_T), \tag{1}$$

where $\hat{R}_t = \sum_t^T \gamma^t r_t$ is the RTG for timestamp t, representing the future cumulative reward from t to T. The DT employs causally masked transformers, and the action a_T is predicted through auto-regressive supervised learning over input trajectories.

3 Methodology

We present an overview of the proposed CaseRec framework and describe its components; see Figure 2 for a helicopter view.

3.1 Model overview

We adapt the decision transformer (DT) to model exposure sequences. This is motivated by: (i) the DT can handle complex long sequences to better learn item sequential correlations from exposure sequences; and (ii) the DT can achieve reward-conditioned learning by assigning different rewards to different user feedback. Given the input trajectory $\tau_{rec} = (\hat{R}_1, s_1, a_1, \ldots, \hat{R}_T, s_T)$, the elements in the trajectory are defined as follows:

- **state** s_t represents the exposure sequence information before timestep t, which contains both the exposed the items and the user's feedback towards the items at each timestep, i.e.,

$$s_t = (v_1, b_1, v_2, b_2, \ldots, v_t, b_t). \tag{2}$$

  Besides, s_t is zero-padded or truncated to a length of $2L$, i.e., a total timestep of L before t. To map the complex exposure sequence into a hidden state, we use a gated recurrent unit (GRU)-based state encoder [8] as detailed in Section 3.2.1.
- **action** $a_t \in \mathcal{V}$ represents the item exposed to the user at timestep t, i.e., $a_t = v_{t+1}$.
- **reward** $r_t \in \mathbb{R}$ represents the reward of the action a_t. We define the reward depending on the user feedback.

$$r_t = \begin{cases} r_{uni}, \text{ if } b_{t+1} = 0 \\ r_{int}, \text{ if } b_{t+1} = 1, \end{cases} \tag{3}$$

  where r_{uni} and r_{int} are hyperparameters that represent the corresponding rewards, respectively. CaseRec can also support more flexible reward settings, e.g., assigning rewards according to the watching time for video recommendation.
- **RTG** $\hat{R}_t \in \mathbb{R}$ represents the cumulative reward from time t through to time T.

The DT-based sequential recommender takes τ_{rec} as input and predicts the action conditioned on $\hat{R}_t$ as the recommended items. Moreover, data augmentation is used to augment counterfactual items for the exposure sequence, and the user simulator is used to predict user feedback on counterfactual items. The data augmentation and the user simulator yield counterfactual exposure sequences to uncover potential new user interests. Finally, the sequential recommender is trained on the mixed dataset that contains trajectories of both logged exposure sequences and counterfactual exposure sequences.

3.2 Sequential recommender

3.2.1 Encoding exposure sequences. For a given exposure sequence, the state encoder needs to capture the sequential signals and model the user behavior signals (i.e., feedback) towards the exposed items. To this end, we introduce two independent embedding layers to encode the item and the behavior, respectively. Subsequently, item embeddings and behavior embeddings are added together to combine the item information with the corresponding behavior information. The added embeddings are fed into a GRU to capture the sequential signals and obtain the final state representation.

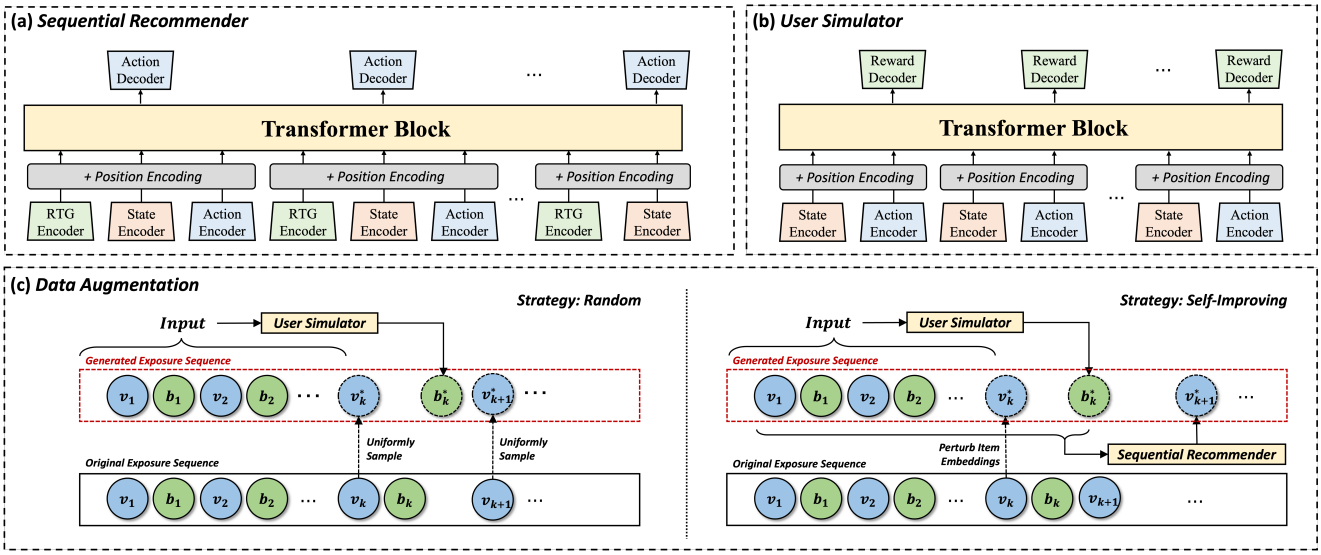


Figure 2: Overview of CaseRec. (a) The architecture of our DT-based sequential recommender, which takes system exposure as input and generates recommendation; see Section 3.2. (b) The architecture of our transformer-based user simulator, which predicts user feedback for a given item; see Section 3.3.1. (c) The counterfactual augmentation process, which illustrates two strategies: *Random* and *Self-improving*; see Section 3.3.2. The user simulator (b) is used to imitate user feedback for counterfactual items generated by the data augmentation (c), and the sequential recommender (a) is trained on both logged exposure sequences and augmented counterfactual sequences.

Given a state sequence $s_t = (v_1, b_1, v_2, b_2, \ldots, v_t, b_t)$, we first separate it into an item sequence $v_{1:t} = (v_1, v_2, \ldots, v_t)$ and a behavior sequence $b_{1:t} = (b_1, b_2, \ldots, b_t)$. The item embeddings and behavior embeddings can be defined as:

$$\mathbf{v}_i = \mathrm{E}_{item}(v_i)\ , \forall v_i \in v_{1:t} \tag{4}$$

$$\mathbf{b}_i = \mathrm{E}_{behavior}(b_i)\ , \forall b_i \in b_{1:t}, \tag{5}$$

where E_{item} and $\mathrm{E}_{behavior}$ represent the embedding functions, which could be achieved by simply looking up trainable embedding tables. $\mathbf{v}_i \in \mathbb{R}^d$ and $\mathbf{b}_i \in \mathbb{R}^d$ represent the item embedding and the behavior embedding respectively, where d denotes the embedding size. Then the added embedding can be obtained by

$$\mathbf{x}_i = \mathbf{v}_i + \mathbf{b}_i\ , \forall i \in [1, t]. \tag{6}$$

Consequently, the state embedding $\mathbf{s}_t$ for a given sequence s_t is obtained from a GRU as

$$\mathbf{s}_t = \mathrm{GRU}(\mathbf{x}_t, \mathbf{s}_{t-1}). \tag{7}$$

The GRU can also be replaced with other sequential models. For simplicity, in this paper, we use a GRU as our default encoder.

We also introduce an RTG encoder and an action encoder to embed RTG and actions. The representations for RTG $\hat{R}_t$ and action a_t are defined as:

$$\mathbf{e}_{\hat{R}_t} = \tanh(\mathrm{E}_{RTG}(\hat{R}_t)) \tag{8}$$

$$\mathbf{a}_t = \tanh(\mathrm{E}_{item}(a_t)), \tag{9}$$

where E_{RTG} and E_{item} represent the embedding functions of $\hat{R}_t$ and a_t, respectively. In our implementation, the same embedding table of item is used for both Eq. 9 of the action encoder and Eq. 4 of the state encoder, allowing the embedding table to be fully trained. Then, the representation of each tuple at timestep t can be formulated as $[\mathbf{e}_{\hat{R}_t}, \mathbf{s}_t, \mathbf{a}_t] \in \mathbb{R}^{3\times d}$. Additionally, a position embedding of timestep t is learned and added to each embedding, yielding the representation $[\mathbf{e}'_{\hat{R}_t}, \mathbf{s}'_t, \mathbf{a}'_t]$. The final representation of the input trajectory is defined as

$$\boldsymbol{\tau}_{rec} = (\mathbf{e}'_{\hat{R}_1}, \mathbf{s}'_1, \mathbf{a}'_1, \ldots, \mathbf{e}'_{\hat{R}_{T-1}}, \mathbf{s}'_{T-1}, \mathbf{a}'_{T-1}, \mathbf{e}'_{\hat{R}_T}, \mathbf{s}'_T). \tag{10}$$

3.2.2 Supervised learning. The primary model architecture is based on a unidirectional transformer layer, incorporating a multi-head self-attention mechanism [48]. The output representation of a trajectory is defined as

$$\hat{\boldsymbol{\tau}}_{rec} = \mathrm{FFN}[\mathrm{MHA}(\boldsymbol{\tau}_{rec})] = (\hat{\mathbf{e}}_{\hat{R}_1}, \hat{\mathbf{s}}_1, \hat{\mathbf{a}}_1, \ldots, \hat{\mathbf{a}}_{T-1}, \hat{\mathbf{e}}_{\hat{R}_T}, \hat{\mathbf{s}}_T). \tag{11}$$

MHA is a multi-head self-attentive layer and FFN are feed-forward neural layers with a GELU [20] activation function and skip-connection. To avoid overfitting and enable more stable learning without vanishing or exploding gradient issues, we include dropout layers and layer normalization [3]. The representation $\hat{\mathbf{s}}_t$ is fed into a fully connected layer to compute the classification logits on candidate actions for the t-th timestep as

$$\tilde{\mathbf{a}}_t = [y_1^t, y_2^t, \ldots, y_{|\mathcal{V}|}^t] = \mathbf{W}_i \hat{\mathbf{s}}_t + \mathbf{c}\ , \forall t \in [1, T], \tag{12}$$

where $\tilde{\mathbf{a}}_t$ is predicted scores for candidate actions at timestep t. $\mathbf{W}_i \in \mathbb{R}^{n\times d}$ and $\mathbf{c} \in \mathbb{R}^n$ are trainable parameters.

The learning objective of the original DT is to predict the action a_t at timestep t, while the learning objective of SR is to predict the next item that the user is likely to interact with based on the previously exposed items and user feedback. In exposure sequences, there are both interacted and non-interacted items, so directly using the auto-regressive objective of the DT would lead to a gap between training objectives. To this end, we reformulate the learning objective of CaseRec to predict only high-rewarded actions, i.e., the item that the user is likely to interact with, based on the exposed

items prior to timestep t. Therefore, the ground-truth action for $\tilde{\mathbf{a}}_t$ is defined as the next item that the user would interact with:

$$\bar{a}_t = v_k \text{ , where } k = \min\{k \mid k > t \wedge b_k = 1\}. \tag{13}$$

Then the cross-entropy loss between predicted actions and ground-truth actions is used to train the model:

$$\mathcal{L}_{rec} = \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{|\mathcal{V}|} Y_i^t \log(p_i^t), \text{ where } p_i^t = \frac{e^{y_i^t}}{\sum_{j\in\mathcal{V}} e^{y_j^t}}. \tag{14}$$

Here, Y_i^t is a binary indicator and equals 1 if the i-th item is the ground-truth $\bar{a}_t$, otherwise $Y_i^t = 0$. In the training stage, the RTG can be obtained by calculating the sum of future rewards. In the inference stage, we can obtain the RTG through auto-regressive generation. For example, in practical online deployments, we can set the expected RTG at the beginning and then decrease the RTG at each timestep according to the real user feedback. For offline inference and evaluation, we typically set the discount factor as $\gamma = 1$, and the RTG at timestep t is set as $\hat{R}_t = \sum_{k=t}^{T-1} r_k + r_{int}$, which means that the sequential recommender is expected to generate an action that can be interacted with by the user.

3.3 Counterfactual augmentation

To further explore unseen user interests, we propose two counterfactual augmentation strategies: *Random* and *Self-improving*, to generate additional exposure sequences. The main difference between the two strategies lies in the method of replacing items in the original exposure sequence. Specifically, *Random* replaces the original items with uniformly sampled items to alleviate the effect of previously deployed recommendation policies. *Self-improving* first replaces the original item by perturbing the item embedding, and then generates the following exposure sequence through the current sequential recommender in an auto-regressive manner to uncover new user interests. We ask *what's the user feedback if a counterfactual item is exposed to the user?* To this end, we introduce a user simulator to predict the user feedback for the given counterfactual items and conduct counterfactual augmentation by replacing items in the logged exposure sequences and predicting corresponding user feedback.

3.3.1 User simulator. Motivated by recent offline model-based reinforcement learning, which improves policy learning by using environment models to further explore the state-action space [26, 57, 58], a user simulator is proposed to predict the user feedback reward for counterfactual items. Considering that the exposure sequence is complex and informative, the transformer architecture, with outstanding long sequence modeling capability and generalization capability, is used to predict user feedback reward.

Specifically, given an input sequence $(s_1, a_1, s_2, a_2, \ldots, s_T, a_T)$, the goal of the user simulator is to predict the user feedback reward r_t of action a_t. The user simulator and the sequential recommender employ similar components, including the state encoder, the action encoder, and transformer blocks. The output representation at position a_t from the transformer block is fed into a fully connected layer with the binary cross-entropy loss function to optimize the simulator parameters. At the inference stage, the generated reward at the last timestep T is used to predict the user feedback reward r_T of action a_T.

Algorithm 1: Training procedure of CaseRec.

Input: sequential recommender $\mathcal{M}$, user simulator $\mathcal{G}$, augmentation ratio δ, generation length h, number of max epochs k, original dataset $\mathcal{D}$, item set $\mathcal{V}$

```
Train M with original dataset D
for i=1 to k do
    Initialized augmentation set D_aug = ∅
    while |D_aug|/|D| < δ do
        Sample s_ori = {v_i, b_i}_{i=1}^{k-1} and v_k from D
        if Self-improving then
            Get v_k^* by perturbing the embeddings of v_k
        else if Random then
            Get v_k^* by uniformly sampling from V
        Obtain b_k^* from G based on s_ori ∪ v_k^*
        s_ori = s_ori ∪ {v_k^*, b_k^*}
        for t=k+1 to k+h-1 do
            if Self-improving then
                Obtain v_t^* from M based on s_ori
            else if Random then
                Get v_t^* by uniformly sampling from V
            Obtain b_t^* from G based on s_ori ∪ v_t^*
            s_ori = s_ori ∪ {v_t^*, b_t^*}
        end
        D_aug.append(s_ori)
    end
    Train M with augment dataset D_aug
end
```

3.3.2 Data augmentation. In this section, we present the two data augmentation strategies.

The *Random* strategy. Recent work has shown that recommendation models may be influenced by prior recommendation policies, resulting in sub-optimal recommendation performance [5, 55]. To prevent the augmented sequences from being influenced by previously deployed recommendation policies, we perform random exposure simulation to generate counterfactual exposure sequences. Specifically, given a logged exposure sequence from the original training data $s_{ori} = (v_1, b_1, v_2, b_2, \ldots, v_T, b_T)$, we replace the item v_k with a uniformly sampled new item v_k^*, where $k \in [1, T]$. Then the sequence $(s_1, a_1, s_2, a_2, \ldots, s_{k-1}, v_k^*)$ can be constructed from $(v_1, b_1, v_2, b_2, \ldots, v_k^*)$ and fed into the user simulator to predict the user feedback r_{k-1}^*, i.e., b_k^* towards the new item v_k^*. This process is repeated for h times to generate a counterfactual exposure sequence as $s_{gen} = (v_1, b_1, \ldots, v_k^*, b_k^*, \ldots, v_{k+h-1}^*, b_{k+h-1}^*)$. After data augmentation, the sequential recommender is trained on trajectories generated from both logged exposure sequences and counterfactual exposure sequences through Eq. 14.

The *Self-improving* strategy. Inspired by the RL agent's improvement of its policy through continuous interaction with the environment, the *Self-improving* strategy uses the current sequential recommender to generate out-of-distribution data in an auto-regressive manner, subsequently using the data to enhance the sequential recommender further. Specifically, given a logged sequence $s_{ori} = (v_1, b_1, v_2, b_2, \ldots, v_k)$ from the original training dataset, we

first add a small amount of Gaussian noise to the item embedding of v_k to obtain new item embedding, and then replace item v_k with the v_k^* whose embedding exhibits the highest cosine similarity to the newly perturbed embedding. The corresponding sequence is then fed into the user simulator to predict the user feedback b_k^* towards the new item. Subsequently, based on the exposure sequence, we use the sequential recommender, pre-trained on the original dataset, to generate the next item and the user simulator to predict the corresponding user feedback. Similar to the *Random* strategy, this process is repeated to generate a new sequence with a length of h. After augmenting a specific number of data, these data will be used to update the parameters of the sequential recommender. The entire loop will continue until the number of max epochs is reached or early stopping is triggered. Algorithm 1 details the training procedure of CaseRec with its two augmentation strategies.

4 Experiments

We aim to answer the following research questions:

(RQ1) How does the recommendation performance of CaseRec compare to other sequential recommendation baselines?
(RQ2) How does the debiasing performance of CaseRec compare with other debiasing methods?
(RQ3) How does counterfactual augmentation influence the performance of CaseRec?
(RQ4) How does the design of CaseRec affect the recommendation performance?

4.1 Experimental settings

4.1.1 Datasets. Experiments are conducted on three datasets: ZhihuRec [17], Tenrec [60], and KuaiRand [15]. All three datasets contain user interaction data (e.g., clicks) and system exposure data (e.g., impressions). We treat the clicked items as interacted items and non-clicked items as exposed but non-interacted items.

- **ZhihuRec**[1] contains question information, answer information and user profiles. We consider the answers as the items recommended to the user.
- **Tenrec**[2] is collected from two feeds, namely articles and videos, on Tenrec's recommendation platforms. Our study focuses on the video recommendation scenario.
- **KuaiRand-Pure**[3] is a sequential recommendation dataset collected from video recommendation scenario. It contains both the standard user feedback records collected by prior recommendation policies (**KuaiRand-15policies**) and unbiased records collected by randomly exposing items (**KuaiRand-Random**).

4.1.2 Evaluation protocols. We use Recall@K and NDCG@K as metrics to evaluate the ranking performance. Results are reported with varying values of $K \in \{5, 10, 20\}$ for both metrics. We also adopt cross-validation to evaluate the performance of the models; the ratio of training, validation, and test set is 8:1:1. To ensure a fair comparison, we use user interaction sequences as the input of inference, which is consistent with the baselines. Each experiment is repeated 5 times; we report the average performance.

4.1.3 Baselines. We compare CaseRec with various state-of-the-art sequential recommendation models, including RNN-based models, transformer-based models, debiasing models, and RL-based models:

- **GRU4Rec** [21] is an RNN-based sequential recommendation model, which uses a GRU to model user–item interactions.
- **SASRec** [25] uses a left-to-right unidirectional transformer to capture user preference.
- **BERT4Rec** [46] uses a bidirectional transformer to learn sequential information.
- **CORE** [22] uses a transformer-based structure to unify representation spaces for both the encoding and decoding processes.
- **FEARec** [12] utilizes contrastive learning to capture hidden information within the frequency domain.
- **IPS** [43] eliminates popularity bias by re-weighing each interaction according to propensity score.
- **RelMF** [56] uses an effective unbiased estimator to correct the matching score between items and users.
- **DRO** [55] uses system exposure to alleviate the exposure bias by distributionally robust optimization (DRO).
- **DT** [6] is a vanilla DT model without considering the system exposure sequence.
- **SQN** [53] is an RL-based model incorporating Q-learning into a standard supervised generative sequential model.

4.1.4 Implementation details. Following [55], we preserve the last 50 clicked items as the user-interaction sequence and the last 200 exposed items as the system exposure sequence. The sequences are padded to the max length with an additional token. For CaseRec, we set the maximum interaction steps of a trajectory to $T = 20$, the maximum interaction steps of the state to $L = 10$, the augmentation length to $h = 10$, the number of transformer layers to 2, and the number of heads to 8 for the sequential recommender and user simulator. For a fair comparison, the item embedding size is set to 64 for all models. We train all models with the Adam optimizer [27]. For other baselines, the hyper-parameters are set to their optimal values as recommended by their original papers.

4.2 Overall performance (RQ1)

Table 1 compares the performance of CaseRec-R (CasRec with the *Random* strategy) and CaseRec-S (CasRec with the *Self-improving* strategy) against the baselines on three datasets. Both CaseRec-R and CaseRec-S outperform all baselines across all datasets, demonstrating the effectiveness of our approach in modeling system exposure. Compared to existing methods only using the sparse user-interaction sequence, CasRec takes full advantage of the large volume of exposure sequences to capture potential user interests. DRO [55] also uses system exposure to model the system behavior and achieve improved performance, particularly in complex datasets with a large item space like Tenrec. However, CaseRec shows a significant performance improvement over DRO, which shows that the counterfactual data generated by the data augmenter helps the sequential recommender to uncover user interests.

We also see that CaseRec-S outperforms CaseRec-R, which may be attributed to the fact that CaseRec-S explores the item space near

[1] https://github.com/THUIR/ZhihuRec-Dataset
[2] https://github.com/yuangh-x/2022-NIPS-Tenrec
[3] https://kuairand.com/

Table 1: Overall performance comparison of different methods on the three datasets. NG and Re is short for NDCG and Recall, respectively. '-R' and '-S' represent the *Random* strategy and *Self-improving* strategy, respectively. Boldface denotes the highest score. * denotes the significance p-value < 0.01 compared with the best baseline which is marked with underline.

Dataset	Metric	Method											
		GRU4Rec	SASRec	BERT4Rec	CORE	FEARec	IPS	RelMF	DRO	DT	SQN	CaseRec-R	CaseRec-S
ZhihuRec	Re@5	0.0113	0.0063	0.0138	0.0088	0.0137	0.0090	0.0101	0.0101	0.0126	0.0095	0.0528*	**0.0704***
	Re@10	0.0214	0.0201	0.0176	0.0163	0.0244	0.0189	0.0138	0.0151	0.0201	0.0232	0.0741*	**0.0942***
	Re@20	0.0251	0.0402	0.0289	0.0402	0.0445	0.0255	0.0239	0.0264	0.0437	0.0432	0.1131*	**0.1470***
	NG@5	0.0090	0.0032	0.0085	0.0050	0.0083	0.0056	0.0075	0.0076	0.0077	0.0054	0.0352*	**0.0464***
	NG@10	0.0123	0.0076	0.0097	0.0074	0.0117	0.0086	0.0087	0.0092	0.0101	0.0087	0.0421*	**0.0540***
	NG@20	0.0132	0.0125	0.0126	0.0134	0.0167	0.0102	0.0112	0.0120	0.0159	0.0143	0.0519*	**0.0672***
Tenrec	Re@5	0.0574	0.0571	0.0445	0.0463	0.0432	0.0460	0.0508	0.0627	0.0600	0.0581	0.0895*	**0.1100***
	Re@10	0.0987	0.0946	0.0848	0.0867	0.0771	0.0706	0.0687	0.1100	0.1044	0.1054	0.1419*	**0.1624***
	Re@20	0.1608	0.1526	0.1463	0.1491	0.1112	0.0971	0.0980	0.1709	0.1701	0.1687	0.2172*	**0.2459***
	NG@5	0.0362	0.0362	0.0299	0.0276	0.0258	0.0309	0.0325	0.0391	0.0353	0.0378	0.0566*	**0.0719***
	NG@10	0.0494	0.0484	0.0428	0.0405	0.0385	0.0388	0.0382	0.0542	0.0495	0.0512	0.0735*	**0.0888***
	NG@20	0.0650	0.0630	0.0583	0.0562	0.0471	0.0455	0.0456	0.0695	0.0661	0.0676	0.0925*	**0.1099***
KuaiRand -15 policies	Re@5	0.0675	0.0600	0.0518	0.0576	0.0358	0.0571	0.0327	0.0584	0.0644	0.0681	0.1333*	**0.1644***
	Re@10	0.1101	0.1039	0.0874	0.0948	0.0575	0.0973	0.0580	0.1093	0.1042	0.1079	0.2000*	**0.2311***
	Re@20	0.1706	0.1652	0.1474	0.1710	0.0962	0.1586	0.0936	0.1718	0.1634	0.1574	0.2948*	**0.3217***
	NG@5	0.0428	0.0359	0.0309	0.0333	0.0231	0.0335	0.0206	0.0350	0.0406	0.0419	0.0876*	**0.1135***
	NG@10	0.0566	0.0499	0.0423	0.0453	0.0301	0.0465	0.0288	0.0513	0.0507	0.0545	0.1088*	**0.1350***
	NG@20	0.0718	0.0651	0.0574	0.0645	0.0397	0.0619	0.0377	0.0770	0.0706	0.0720	0.1326*	**0.1579***

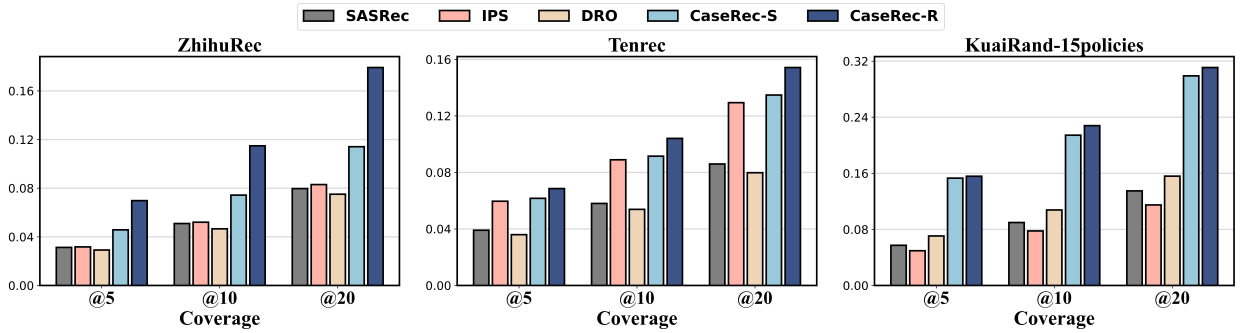


Figure 3: Evaluation on recommendation diversity.

the historical policy while CaseRec-R imitates random exploration. Finally, the improvement of CaseRec is most significant on the KuaiRand-15 policies dataset, which has a higher density than other datasets. We assume that the user simulator can more accurately capture user interests from the dense information, which leads to higher-quality generated sequence augmentation.

4.3 Debiasing performance (RQ2)

Evaluation on unbiased exposure. Exposure bias, also known as "previous model bias" [32], occurs when the recommendation model is influenced by prior recommendation policies and only exposes a subset of available items to users. Table 2 presents the debiasing performance in the unbiased KuaiRand-Random dataset, which is collected by exposing items to users randomly rather than by prior recommendation policies. To evaluate the unbiased recommendation performance learned from the biased data, we use the biased KuaiRand-15policies dataset as the training set and the unbiased KuaiRand-Random dataset as the validation and test set for each method.

From Table 2, we see that only CaseRec achieves satisfying performance in this challenging experimental setting, which indicates that counterfactual augmentation captures new user interests beyond the original dataset collected using the previous recommendation policies. Existing debiasing methods (IPS, RelMF and DRO) evaluate items and introduce penalties on those considered overexposed, achieving improved performance compared to the vanilla models. Evaluating whether an item is overexposed is a challenging task and imposing penalties on inappropriate items negatively impacts the model performance. CaseRec-R outperforms CaseRec-S in this setting, which indicates that the exposure sequence generated by random exposure simulation helps to reduce the impact of prior policies.

Evaluation on recommendation diversity. Exposure bias may cause relevant but unpopular items to be overlooked and not recommended, leading to a reduction in recommendation diversity [5, 16].

Table 2: Exposure debiasing performance on the KuaiRand-Random dataset. NG and Re is short for NDCG and Recall, respectively. '-R' and '-S' represent the *Random* strategy and *Self-improving* strategy, respectively. Boldface denotes the highest score. * denotes the significance p-value < 0.01 compared with the best baseline which is marked with underline.

Method	Metric					
	Re@5	Rec10	Re@20	NG@5	NG@10	NG@20
GRU4Rec	0.0012	0.0020	0.0050	0.0007	0.0010	0.0018
SASRec	0.0011	0.0026	0.0052	0.0006	0.0011	0.0017
SQN	0.0012	0.0028	0.0050	0.0007	0.0013	0.0016
IPS	0.0012	0.0022	0.0040	0.0007	0.0010	0.0015
RelMF	0.0010	0.0023	0.0055	0.0005	0.0010	0.0018
DRO	0.0014	0.0022	0.0046	0.0008	0.0010	0.0016
CaseRec-R	**0.0085***	**0.0139***	**0.0229***	**0.0059***	**0.0077***	**0.0099***
CaseRec-S	0.0066*	0.0133*	0.0224*	0.0042*	0.0063*	0.0086*

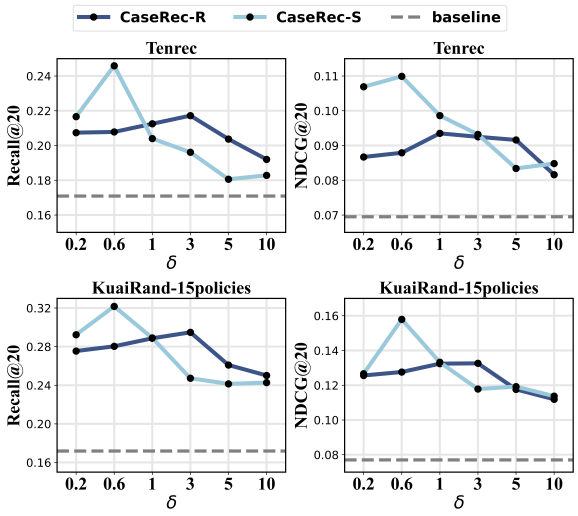


Figure 4: Impact of the augmentation ratio δ. Grey dashed lines denote the best baseline.

We conduct experiments to determine whether CaseRec can generate diversified recommendation. We use the coverage metric to measure the recommendation diversity:

$$\text{Coverage@}K = \frac{|\bigcup_{u \in \text{Test}} list_{@K}(u)|}{|\mathcal{V}|}, \tag{15}$$

where $list_{@K}(u)$ represents the the top-K recommendation list for user u. Figure 3 presents the exposure debiasing performance compared with baselines in three datasets. From Figure 3, we observe that CaseRec-R achieves the best performance across all datasets, which indicates that the baselines only depend on logged historical sequences, which only contain a small subset of all available items. In contrast, our approach explores user interests over a broader range of items, leading to more diverse lists.

To conclude, CaseRec not only achieves better recommendation performance under unbiased evaluation but also generates more diverse recommendations. This suggests that CaseRec is an alternative solution to alleviate exposure bias.

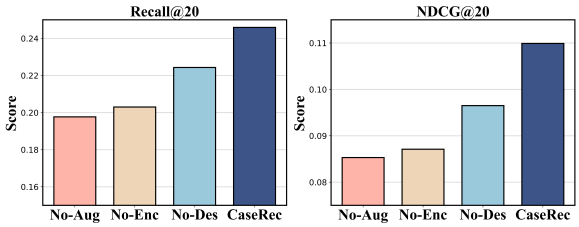


Figure 5: Ablation study.

4.4 Impact of augmentation ratio (RQ3)

We conduct experiments to demonstrate the impact of the augmentation ratio δ, which can be defined as $\delta = |\mathcal{D}_{aug}|/|\mathcal{D}_{ori}|$, where D_{aug} represents the augmented dataset and D_{ori} represents the original dataset. We vary δ from 0.2 to 10 and evaluate the performance for each augmentation ratio. The results are presented in Figure 4. We observe that the performance of both CaseRec-R and CaseRec-S in both datasets first increases and then decreases. This indicates that when the augmentation ratio increases at the beginning, the performance of CaseRec improves because the augmented exposure sequence enables the sequential recommender to further explore the user interests. However, when the augmentation ratio is too large, the performance will drop because too much noise is introduced in the dataset. In addition, compared to CaseRec-R, CaseRec-S is able to continuously improve the current recommendation strategy based on the user simulator, thus demonstrating higher learning efficiency. Regardless of the augmentation ratio, both CaseRec-S and CaseRec-R outperform the strongest baseline, highlighting the robustness of our proposed method.

4.5 Ablation study (RQ4)

We perform an ablation study to investigate the impact of each module individually. Our ablation study was conducted using the Tenrec dataset based on the *Self-improving* strategy, with the results presented in Figure 5. The variant models are:

- **No-Aug**: The recommendation model is trained using only the original dataset, without the counterfactual augmentation.
- **No-Enc**: The recommendation model uses the average of the item embeddings in a state sequence to replace the state encoder.
- **No-Des**: The recommendation model without adaption of learning objectives.

From Figure 5, we have several observations. First, CaseRec significantly outperforms 'No-Aug', demonstrating the effectiveness of counterfactual augmentation. Although 'No-Enc' shows an improvement compared to 'No-Aug', it lags behind CaseRec, which demonstrates the effectiveness of the state encoder in simultaneously capturing sequential signals and user behaviors. Finally, CaseRec outperforms 'No-Des', indicating that the redesigned learning objective effectively narrows the gap between original DT and SR, further improving performance.

5 Related work

5.1 Sequential recommendation

The purpose of sequential recommendation (SR) is to capture an item's chronological correlations. Early SR models are mainly based

on Markov chains [18, 39]. In recent years, many deep learning-based SR models have been proposed. E.g., GRU4Rec [21] uses recurrent neural networks to learn sequential information. The self-attention mechanism has shown great potential and various models have been proposed, e.g., SASRec [25], BERT4Rec [46] and S^3Rec [63]. CORE [22] uses an effective transformer-based structure to unify the representation space for the encoding and decoding processes. Recently, many contrastive learning-based sequential recommendation methods have been proposed to further improve the performance. E.g., FEARec [12] separates low-frequency information and high-frequency information and uses contrastive learning to capture hidden information.

The aforementioned SR models are based on traditional user interaction sequences and are unable to incorporate system exposure. This prevents them from fully exploiting the user interests contained in system exposure data, leading to sub-optimal recommendation performance.

5.2 Offline reinforcement learning

Offline RL learns policies from a pre-collected, static dataset of offline trajectories. Traditional offline RL methods can be classified into model-free and model-based methods. Model-free offline methods incorporate conservatism into the value function estimation [13, 14, 28–30, 45]. E.g., CQL [30] adds a regularization term into the Q-function update. Model-based offline RL methods [26, 35, 36, 38, 40, 44, 47, 57, 58, 61] learn a model of environment and generate additional data to improve the policy. Unlike the aforementioned methods, the decision transformer [6] casts the offline RL task as a conditional sequence modeling problem and achieves remarkable performance.

Offline RL has achieved successful performance in sequential recommendation scenarios [1, 7, 51, 53, 62]. For example, SQN [53] incorporates an additional output layer trained with Q-learning into a standard supervised generative sequential model and achieves improved performance. Recently, some DT-based SR models have been proposed [34, 63]. However, unlike our work, these models are user retention-oriented by treating long-term user engagement as an additional learning objective of the DT. In contrast, our work aims to generate more accurate recommendations by using the characteristics of the DT to model complex system exposure sequences.

5.3 System exposure

System exposure data, also referred to as *previous recommendations* [2] or *impressions* [37], represents the system's behavior as opposed to user interaction data, which reflects user preferences. Despite its relevance, system exposure data has received relatively little attention in the research community. Existing system exposure studies primarily focus on the negative sampling of exposed items or addressing exposure bias.

Negative sampling is a technique in which a subset of unobserved or non-preferred data points is randomly selected and treated as negative examples to improve model training efficiency. Ding et al. [11] use the exposed but not interacted items as negative examples. Wang et al. [49] propose a reinforcement negative sampler that generates exposure-like negative instances rather than directly selecting from the exposure data. Instead of simply considering the exposed but non-interacted items as negative examples, we argue that exposed but non-interacted items could also contain potential user interests.

Exposure bias happens when only subsets of the item catalog are exposed by the system, so it is unclear whether an unobserved interaction between a user and an item is attributable to the user's lack of interest or the item was not exposed to the user. Therefore, regarding non-interacted items as negative samples may lead to a misunderstanding of user's true preference and sub-optimal performance. Existing exposure debiasing methods typically introduce penalties based on IPS or DRO. E.g., Yang et al. [56] propose an IPS-based unbiased evaluator to down-weight the commonly observed interactions while up-weighting the rare ones. Yang et al. [55] use DRO to infer the debiased user preference and introduces penalties to items with high exposure probability. Our method introduces counterfactual exposure sequences and can be considered as an alternative solution to further explore user interests, leading to reduced exposure bias.

6 Conclusion

In this paper, we have presented a novel framework, i.e., CaseRec, to enhance performance on the sequential recommendation task. CaseRec leverages a decision transformer-based sequential model to take the whole exposure sequence as input to generate recommendation. In addition, CaseRec performs two counterfactual augmentation strategies, *Random* and *Self-improving*, to uncover potential user interests. Then, a transformer-based user simulator is proposed to predict the user's corresponding feedback on the augmented items. Extensive experiments on three datasets demonstrate the effectiveness of the proposed CaseRec. Finally, CaseRec uses the logged exposure sequences with the counterfactual exposure sequences to train a decision transformer-based sequential model for generating recommendation.

As to limitations, we have only considered two types of user behaviors. In future work, we plan to incorporate a broader range of user behavior. Moreover, there is a rapidly growing body of work on user simulators [4]. In future work, we aim to experiment with a broader set of user simulators.

Acknowledgments

This research was (partially) supported by the Natural Science Foundation of China (62202271, 62472261, 62372275, 62272274, T2293773), the National Key R&D Program of China with grant No. 2024YFC3307300 and No. 2022YFC3303004, the Provincial Key R&D Program of Shandong Province with grant No. 2024CXGC010108, the Natural Science Foundation of Shandong Province with grant No. ZR2024QF203, the Technology Innovation Guidance Program of Shandong Province with grant No. YDZX2024088, the Dutch Research Council (NWO), under project numbers 024.004.022, NWA.13 89.20.183, and KICH3.LTP.20.006, and the European Union's Horizon Europe program under grant agreement No. 101070212. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

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] Davide Azzalini, Fabio Azzalini, Chiara Criscuolo, Tommaso Dolci, Davide Martinenghi, and Sihem Amer-Yahia. 2022. SoCRATe: A Recommendation System with Limited-Availability Items. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 4793–4797.

[3] Jimmy Lei Ba. 2016. Layer Normalization. *arXiv preprint arXiv:1607.06450* (2016).

[4] Krisztian Balog and ChengXiang Zhai. 2024. User Simulation for Evaluating Information Access Systems. *Foundations and Trends in Information Retrieval* 18, 1-2 (2024), 1–261.

[5] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2023. Bias and Debias in Recommender System: A Survey and Future Directions. *ACM Transactions on Information Systems* 41, 3 (2023), 1–39.

[6] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision Transformer: Reinforcement Learning via Sequence Modeling. *Advances in neural information processing systems* 34 (2021), 15084–15097.

[7] Xiaocong Chen, Lina Yao, Julian McAuley, Guanglin Zhou, and Xianzhi Wang. 2023. Deep Reinforcement Learning in Recommender Systems: A Survey and New Perspectives. *Knowledge-Based Systems* 264 (2023), 110335.

[8] Kyunghyun Cho. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv preprint arXiv:1406.1078* (2014).

[9] Quanyu Dai, Haoxuan Li, Peng Wu, Zhenhua Dong, Xiao-Hua Zhou, Rui Zhang, Rui Zhang, and Jie Sun. 2022. A Generalized Doubly Robust Learning Framework for Debiasing Post-click Conversion Rate Prediction. In *KDD*. 252–262.

[10] Khalil Damak, Sami Khenissi, and Olfa Nasraoui. 2022. Debiasing the Cloze Task in Sequential Recommendation with Bidirectional Transformers. In *KDD*. 273–282.

[11] Jingtao Ding, Fuli Feng, Xiangnan He, Guanghui Yu, Yong Li, and Depeng Jin. 2018. An Improved Sampler for Bayesian Personalized Ranking by Leveraging View Data. In *Companion Proceedings of the The Web Conference 2018*. 13–14.

[12] Xinyu Du, Huanhuan Yuan, Pengpeng Zhao, Jianfeng Qu, Fuzhen Zhuang, Guanfeng Liu, Yanchi Liu, and Victor S Sheng. 2023. Frequency Enhanced Hybrid Attention Network for Sequential Recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 78–88.

[13] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing Function Approximation Error in Actor-critic Methods. In *International Conference on Machine Learning*. PMLR, 1587–1596.

[14] Scott Fujimoto, David Meger, and Doina Precup. 2019. Off-policy Deep Reinforcement Learning without Exploration. In *International Conference on Machine Learning*. PMLR, 2052–2062.

[15] Chongming Gao, Shijun Li, Yuan Zhang, Jiawei Chen, Biao Li, Wenqiang Lei, Peng Jiang, and Xiangnan He. 2022. KuaiRand: An Unbiased Sequential Recommendation Dataset with Randomly Exposed Videos. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management* (Atlanta, GA, USA) *(CIKM '22)*. 3953–3957. https://doi.org/10.1145/3511808.3557624

[16] Shantanu Gupta, Hao Wang, Zachary Lipton, and Yuyang Wang. 2021. Correcting Exposure Bias for Link Recommendation. In *International Conference on Machine Learning*. PMLR, 3953–3963.

[17] Bin Hao, Min Zhang, Weizhi Ma, Shaoyun Shi, Xinxing Yu, Houzhi Shan, Yiqun Liu, and Shaoping Ma. 2021. A Large-Scale Rich Context Query and Recommendation Dataset in Online Knowledge-Sharing. arXiv:2106.06467 [cs.IR]

[18] Ruining He, Chen Fang, Zhaowen Wang, and Julian McAuley. 2016. Vista: A Visually, Socially, and Temporally-aware Model for Artistic Recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 309–316.

[19] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.

[20] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415* (2016).

[21] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR (Poster)*.

[22] Yupeng Hou, Binbin Hu, Zhiqiang Zhang, and Wayne Xin Zhao. 2022. Core: Simple and Effective Session-based Recommendation within Consistent Representation Space. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1796–1801.

[23] Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. 2018. Reinforcement Learning to Rank in E-commerce Search Engine: Formalization, Analysis, and Application. In *KDD*. ACM, 368–377.

[24] Ray Jiang, Silvia Chiappa, Tor Lattimore, András György, and Pushmeet Kohli. 2019. Degenerate Feedback Loops in Recommender Systems. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*. 383–390.

[25] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*. IEEE, 197–206.

[26] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. 2020. Morel: Model-based Offline Reinforcement Learning. *Advances in Neural Information Processing Systems* 33 (2020), 21810–21823.

[27] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR (Poster)*.

[28] Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. 2021. Offline Reinforcement Learning with Fisher Divergence Critic Regularization. In *International Conference on Machine Learning*. PMLR, 5774–5783.

[29] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. 2019. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems* 32 (2019).

[30] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative Q-learning for Offline Reinforcement Learning. *Advances in Neural Information Processing Systems* 33 (2020), 1179–1191.

[31] Dawen Liang, Laurent Charlin, James McInerney, and David M Blei. 2016. Modeling User Exposure in Recommendation. In *WWW*. 951–961.

[32] Dugang Liu, Pengxiang Cheng, Zhenhua Dong, Xiuqiang He, Weike Pan, and Zhong Ming. 2020. A general knowledge distillation framework for counterfactual recommendation via uniform data. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 831–840.

[33] Dugang Liu, Pengxiang Cheng, Zinan Lin, Xiaolian Zhang, Zhenhua Dong, Rui Zhang, Xiuqiang He, Weike Pan, and Zhong Ming. 2023. Bounding System-Induced Biases in Recommender Systems with a Randomized Dataset. *ACM Transactions on Information Systems* 41, 4 (2023), 1–26.

[34] Ziru Liu, Shuchang Liu, Zijian Zhang, Qingpeng Cai, Xiangyu Zhao, Kesen Zhao, Lantao Hu, Peng Jiang, and Kun Gai. 2024. Sequential Recommendation for Optimizing Both Immediate Feedback and Long-term Retention. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1872–1882.

[35] Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. 2018. Plan Online, Learn Offline: Efficient Learning and Exploration via Model-based Control. *arXiv preprint arXiv:1811.01848* (2018).

[36] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. 2020. Deployment-Efficient Reinforcement Learning via Model-Based Offline Optimization. arXiv:2006.03647 [cs.LG]

[37] Fernando B. Pérez Maurera, Maurizio Ferrari Dacrema, Pablo Castells, and Paolo Cremonesi. 2023. Impression-Aware Recommender Systems. *arXiv preprint arXiv:2308.07857* (2023).

[38] Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. 2021. Offline Reinforcement Learning from Images with Latent Space Models. In *Learning for Dynamics and Control*. PMLR, 1154–1168.

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

[40] Marc Rigter, Bruno Lacerda, and Nick Hawes. 2022. RAMBO-RL: Robust Adversarial Model-based Offline Reinforcement Learning. *Advances in Neural Information Processing Systems* 35 (2022), 16082–16097.

[41] Yuta Saito, Suguru Yaginuma, Yuta Nishino, Hayato Sakata, and Kazuhide Nakata. 2020. Unbiased Recommender Learning from Missing-Not-At-Random Implicit Feedback. In *WSDM*. ACM, 501–509.

[42] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *WWW*. 285–295.

[43] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as Treatments: Debiasing Learning and Evaluation. In *International Conference on Machine Learning*. PMLR, 1670–1679.

[44] Julian Schrittwieser, Thomas Hubert, Amol Mandhane, Mohammadamin Barekatain, Ioannis Antonoglou, and David Silver. 2021. Online and Offline Reinforcement Learning by Planning with a Learned Model. *Advances in Neural Information Processing Systems* 34 (2021), 27580–27591.

[45] Noah Y. Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. 2020. Keep Doing What Worked: Behavioral Modelling Priors for Offline Reinforcement Learning. *arXiv preprint arXiv:2002.08396* (2020).

[46] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *CIKM*. 1441–1450.

[47] Phillip Swazinna, Steffen Udluft, and Thomas Runkler. 2021. Overcoming Model Bias for Robust Offline Deep Reinforcement Learning. *arXiv preprint arXiv:2008.05533* (2021).

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Advances in Neural Information Processing Systems*.

[49] Xiang Wang, Yaokun Xu, Xiangnan He, Yixin Cao, Meng Wang, and Tat-Seng Chua. 2020. Reinforced Negative Sampling over Knowledge Graph for Recommendation. In *WWW*. 99–109.

[50] Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. 2019. Doubly Robust Joint Learning for Recommendation on Data Missing Not at Random. In *International*

Conference on Machine Learning. PMLR, 6638–6647.

[51] Yuyan Wang, Mohit Sharma, Can Xu, Sriraj Badam, Qian Sun, Lee Richardson, Lisa Chung, Ed H Chi, and Minmin Chen. 2022. Surrogate for Long-term User Experience in Recommender Systems. In *KDD*. 4100–4109.

[52] Zhenlei Wang, Shiqi Shen, Zhipeng Wang, Bo Chen, Xu Chen, and Ji-Rong Wen. 2022. Unbiased Sequential Recommendation with Latent Confounders. In *WWW*. 2195–2204.

[53] Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M Jose. 2020. Self-supervised Reinforcement Learning for Recommender Systems. In *SIGIR*. 931–940.

[54] Chen Xu, Jun Xu, Xu Chen, Zhenghua Dong, and Ji-Rong Wen. 2022. Dually Enhanced Propensity Score Estimation in Sequential Recommendation. In *CIKM*. 2260–2269.

[55] Jiyuan Yang, Yue Ding, Yidan Wang, Pengjie Ren, Zhumin Chen, Fei Cai, Jun Ma, Rui Zhang, Zhaochun Ren, and Xin Xin. 2024. Debiasing Sequential Recommenders through Distributionally Robust Optimization over System Exposure. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 882–890.

[56] Longqi Yang, Yin Cui, Yuan Xuan, Chenyang Wang, Serge Belongie, and Deborah Estrin. 2018. Unbiased Offline Recommender Evaluation for Missing-not-at-random Implicit Feedback. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 279–287.

[57] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. 2021. COMBO: Conservative Offline Model-based Policy Optimization. *Advances in Neural Information Processing Systems* 34 (2021), 28954–28967.

[58] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. 2020. MOPO: Model-based Offline Policy Optimization. *Advances in Neural Information Processing Systems* 33 (2020), 14129–14142.

[59] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A Simple Convolutional Generative Network for Next Item Recommendation. In *WSDM*. ACM, 582–590.

[60] Guanghu Yuan, Fajie Yuan, Yudong Li, Beibei Kong, Shujie Li, Lei Chen, Min Yang, Chenyun Yu, Bo Hu, Zang Li, et al. 2022. Tenrec: A Large-scale Multipurpose Benchmark Dataset for Recommender Systems. *Advances in Neural Information Processing Systems* 35 (2022), 11480–11493.

[61] Xianyuan Zhan, Xiangyu Zhu, and Haoran Xu. 2022. Model-Based Offline Planning with Trajectory Pruning. *arXiv preprint arXiv:2105.07351* (2022).

[62] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018. Deep Reinforcement Learning for Page-wise Recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 95–103.

[63] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S3-Rec: Self-supervised Learning for Sequential Recommendation with Mutual Information Maximization. In *CIKM*. 1893–1902.