CIRS: Bursting Filter Bubbles by Counterfactual Interactive Recommender System

CHONGMING GAO, University of Science and Technology of China, China
SHIQI WANG, Chongqing University, China
SHIJUN LI, University of Science and Technology of China, China
JIAWEI CHEN*, Zhejiang University, China
XIANGNAN HE*, University of Science and Technology of China, China
WENQIANG LEI, Sichuan University, China
BIAO LI, Kuaishou Technology Co., Ltd., China
YUAN ZHANG, Kuaishou Technology Co., Ltd., China
PENG JIANG, Kuaishou Technology Co., Ltd., China

While personalization increases the utility of recommender systems, it also brings the issue of *filter bubbles*. E.g., if the system keeps exposing and recommending the items that the user is interested in, it may also make the user feel bored and less satisfied. Existing work studies filter bubbles in static recommendation, where the effect of overexposure is hard to capture. In contrast, we believe it is more meaningful to study the issue in interactive recommendation and optimize long-term user satisfaction. Nevertheless, it is unrealistic to train the model online due to the high cost. As such, we have to leverage offline training data and disentangle the causal effect on user satisfaction.

To achieve this goal, we propose a counterfactual interactive recommender system (CIRS) that augments offline reinforcement learning (offline RL) with causal inference. The basic idea is to first learn a causal user model on historical data to capture the overexposure effect of items on user satisfaction. It then uses the learned causal user model to help the planning of the RL policy. To conduct evaluation offline, we innovatively create an authentic RL environment (KuaiEnv) based on a real-world fully observed user rating dataset. The experiments show the effectiveness of CIRS in bursting filter bubbles and achieving long-term success in interactive recommendation. The implementation of CIRS is available via https://github.com/chongminggao/CIRS-codes.

CCS Concepts: • **Information systems** → **Recommender systems**; **Personalization**; • **Theory of computation** → **Sequential decision making**.

Additional Key Words and Phrases: Filter bubble, Interactive recommendation, Causal inference, Offline reinforcement learning

ACM Reference Format:
Chongming Gao, Shiqi Wang, Shijun Li, Jiawei Chen*, Xiangnan He*, Wenqiang Lei, Biao Li, Yuan Zhang, and Peng Jiang. 2022. CIRS: Bursting Filter Bubbles by Counterfactual Interactive Recommender System. In . ACM, New York, NY, USA, 26 pages.

* Corresponding author.

Manuscript submitted to ACM

1 INTRODUCTION

Recommender systems have deeply affected our lives. They change the way of retrieving information from searching strenuously, to obtaining conveniently via the precise personalization. The system usually achieves personalization by learning on the collected behavior data of users and selecting the products that users potentially like [21, 25, 26, 38]. With time evolving and data accumulating, the recommender gradually becomes a mirror reflecting each user's interest and narrows down the recommendation lists to the items with the maximum user interest. However, this comes at a price. While enjoying the precise personalized recommendations, users have to face the fact that the variety of information shrinks. When users become isolated from the information that varies from their dominant preferences, they are stuck in the so-called "filter bubbles".

Filter bubbles are common in recommender systems. Recent studies conducted extensive experiments in large-scale recommender systems and found there are two primary causes of filter bubbles [31, 47, 58, 63, 72, 78]. From users' perspective, those who have less diverse preferences are more liable to be stuck in the bubbles. From the system's perspective, learning can lead to emphasizing the dominant interest of a user. Moreover, the system usually assumes that user satisfaction equals intrinsic interest — even though the items that user love have been overexposed, it assumes that the user satisfaction remains unchanged, which is improper. We believe that the over-exploitation behavior of the recommender algorithm is the main cause of filter bubbles, and we will use the overexposure effect as the proxy for evaluating filter bubbles.

Overexposing items has a pernicious effect on a user's satisfaction, even though the user is interested in the recommended item. For example, a user who likes dancing can be satisfied when she receives a recommendation about a dancing song. However, she may be bored after dozens of times of incessant recommendations about dancing songs and thus refuses to choose it. Therefore, it is of great significance to burst filter bubbles for a recommender system for the purpose of maximizing user satisfaction. The key is to disentangle the causal effects on user satisfaction, i.e., modeling how a user's intrinsic interest together with the overexposure effect affect the user's final satisfaction.

Existing methods address filter bubble with two strategies: (1) helping users improve their awareness of diverse social opinions [15, 22], and (2) making the model enhance diversity [1, 49, 79], serendipity [59, 93], or fairness [53] of the recommendation results. However, most of these strategies are heuristic and focus on the static recommendation setting. They cannot fundamentally address the problem since they do not directly consider the main cause of the filter bubble by modeling the overexposure effect.

In this work, we focus on the interactive recommender system (IRS) in the dynamic environment. An IRS is formulated as a sequential decision-making process, which allows the tracking and modeling of the dynamic and real-time overexposure effect. Fig. 1(a) draws an example of interactive recommendation, where a model recommends items (i.e., makes an action a) to a user based on the interaction context (i.e., state s reflecting the user's information and interaction history), then it receives user feedback (i.e., reward r representing user satisfaction). The interaction process is repeated until the user quits. The model will update its policy π_θ with the goal to maximize the cumulative satisfaction over the whole interaction process. To achieve this goal, the IRS should avoid overexposing items because users will get bored and their satisfaction will drop in filter bubbles (Fig. 1(b)).

Though appealing, this idea faces an inevitable challenge: it is difficult to learn an IRS online with real-time feedback. The reason is twofold: (1) for the model, training the policy online increases the learning time and the deploying complexity [92]; (2) for the user, interacting with a half-baked system can hurt satisfaction [19, 66]. Hence, it is necessary to train the IRS offline on historical logs before serving users online. To this end, we need to conduct causal inference

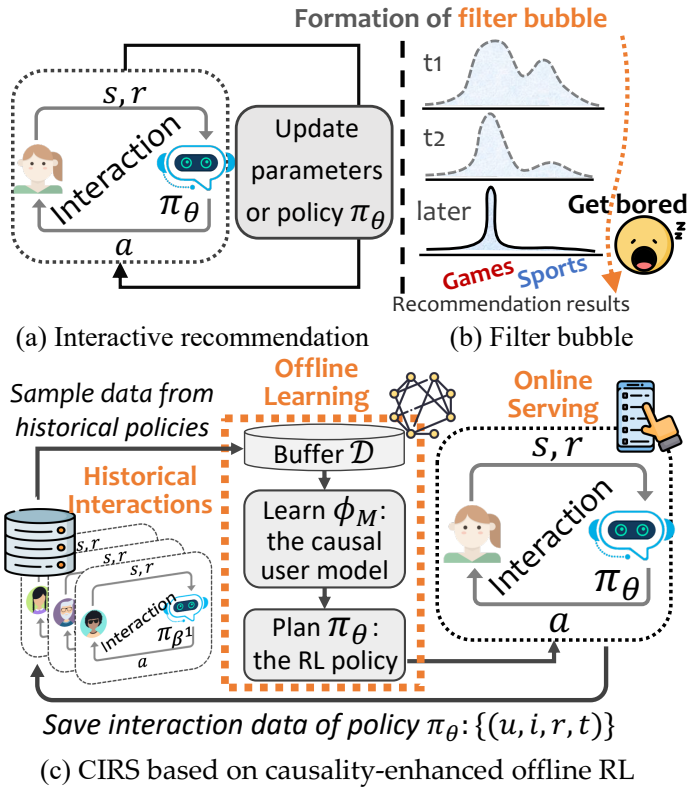


Fig. 1. Illustration of interactive recommendation and the formation of the filter bubble.

[61] on the offline data to disentangle the causal effect of user interest and overexposure. This provides an opportunity to answer a counterfactual question in the serving phase: "*Would the user still be satisfied with the interested item if it had been overexposed?*" — If the answer is no, then the filter bubble will occur once recommending the item, so we should not recommend it.

We propose a counterfactual interactive recommender system (CIRS) to achieve this goal. It enhances offline reinforcement learning (offline RL) [40] with causal inference [61]. Fig. 1(c) shows its learning framework, which contains three recurrent steps: 1) learning a causal user model to capture both user interest and item overexposure effect, 2) using the learned user model to provide *counterfactual satisfaction* (i.e., the rewards given by the causal user model instead of immediate users' feedback) for planning an RL policy, and 3) evaluating the RL policy in terms of the cumulative user satisfaction in real environments.

In addition, we propose an interactive recommendation environment, KuaiEnv, for evaluating the problem. It is created based on the KuaiRec dataset that contains a fully filled user-item matrix (density: 99.6%) collected from the Kuaishou App[1] [20]. In this matrix, IRSs can be evaluated more faithfully since there are no missing values. To effectively reflect the effect of filter bubbles, we further add a "feel bored then quit" exit mechanism in both VirtualTaobao

[1]https://www.kuaishou.com/cn

and KuaiEnv to simulate the reactions of real users. The experiments show that the proposed CIRS can capture the overexposure effect in offline learning and achieve a larger cumulative satisfaction compared with other baselines.

Our contributions are summarized as follows:

- To the best of our knowledge, this is the first work to address filter bubbles in interactive recommendation, where filter bubbles can be more naturally observed and evaluated via modeling the overexposure effect on user satisfaction.
- We solve the problem by integrating causal inference into offline RL. We are the first to combine causal inference and offline RL in interactive recommender systems.
- We conduct empirical studies on Kuaishou App to verify that overexposing items indeed hurts user experience, and we demonstrate that the proposed method can burst filter bubbles and increase cumulative satisfaction.

2 RELATED WORK

We briefly review the related works from three perspectives: filter bubbles, causality inference, and offline RL in recommendation.

2.1 Filter Bubbles in Recommendation

"Personalization filters serve up a kind of invisible autopropaganda, indoctrinating us with our own ideas, amplifying our desire for things that are familiar and leaving us oblivious to the dangers lurking in the dark territory of the unknown."

––Eli Pariser, The Filter Bubble [60]

The term filter bubble was coined by internet activist Eli Pariser [60] and used for describing the state of intellectual isolation that results from the algorithm's personalization process. Bruns [5] extensively discussed the causes and effects of filter bubbles. He claimed that the filter bubbles can be not real under certain situation, instead, it is only used as a scapegoat for avoiding addressing deep-rooted problems by blaming technology. He also discussed the filter bubble from a psychological aspect and described its political effects on society.

McKay et al. [55] investigated the filter bubble by interviewing 18 people recruited from a variety of sources about their experience of changing views on an issue that was important to them. They found an unexpected result that, instead of engaging with more closed-mind people, users are more likely to engage with disagreeable information. And these users typically passively encountered, rather than actively sought, disagreeable information. This emphasizes the significance of recommender systems that distribute the information to users. Also, Flaxman et al. [17] conduct extensive studies on online news consumption to investigate filter bubbles. They found that the majority of consumption behavior mimicked traditional reading habits, i.e., people liked visiting the homepage and receiving information passively.

In this paper, we focus on the filter bubbles in the context of recommender systems that determine the content and information to expose to users. In a recommender, when a user is stuck in a filter bubble, the algorithm will show the user limited information that conforms with their preexisting belief or interest.

In recommendation, many works systematically analyzed the causes of filter bubbles. There are in general two causes. The first cause may be users' personal intention [31, 47, 63, 72]. For example, Liu et al. [47] conducted simulation studies on news recommendations and found that users with more extreme preferences are more liable to be stuck in filter bubbles. Additionally, some researchers investigate on YouTube, the most popular video-sharing platform, how and why users fall in the misinformation filter bubble, i.e., trusting inaccurate information or extreme content with controversial topics [31, 63, 72]. Hussein et al. [31] conducted large audit experiments to investigate whether personalization (i.e.,

age, gender, geo-location, or watch history) contributes to amplifying misinformation. Their results reveal that these personalized demographics do not significantly amplify the bubble for new-coming users. However, once these users once develop a watch history of misinformation, these demographics can exert an effect. This phenomenon brings us to the second cause of the filter bubble.

The second cause is the overexploitation behavior of the recommendation algorithm, which results in the overexposure of a certain subset of items. Recent empirical studies verified that the formation of filter bubbles is mainly due to the model emphasizing certain items with dominant user interest [47, 78]. This kind of filter bubble can have a pernicious effect on user satisfaction. As demonstrated by Herlocker et al. [27], recommending already familiar items can trigger users' unsatisfactory results, due to the fact that users like novelty and serendipity.

It is necessary to ameliorate the filter bubble in recommender systems. Existing methods that aim to combat filter bubbles focus on improving users' awareness of diverse social opinions [15, 22], enhancing models' diversity [1, 49, 79], serendipity [59, 93], fairness [53] of recommendation results, correcting model behavior by watching debunking content [78]. However, these strategies focus on the static recommendation setting, where the dynamic nature of filter bubbles is hard to model. In contrast, we capture the overexposure effect, i.e., the main symptom or proxy of filter bubbles, and solve the problem in interactive recommendation.

2.2 Causality-enhanced Recommendation

Recently, causal inference (CI) has drawn a lot of attention in neural language processing (NLP) [16], computer vision (CV) [50, 82], and recommender system (RS) [4, 42, 67, 88, 95, 98]. Instead of exploiting the correlation relationships between input and output by feeding data to the black-box neural networks, CI explicitly models the causal mechanism among variables [28, 61]. In recommender systems, CI can be a powerful tool for addressing various biases in the data or the learning process, e.g., selection bias and popularity bias [10]. The most naive way to estimate the missing values is the direct method (DM) or the error-imputation-based (EIB) estimator [86], the idea is to use a hyper-parameter γ to impute all missing values. So we can learn our method for these positions using γ instead of zero. This is a very coarse-grained solution. Alternatively, many studies tried to estimate the unbiased user preference based on inverse propensity scoring (IPS) [64, 67], which is an effective CI-based method. Intuitively, when a user has a lower probability of seeing an item, we should assign increased importance to this sample and vice versa. However, IPS-based causal methods suffer from the high variance issue and it is difficult to estimate the propensity score [64]. Therefore, there are methods that combine the EIB and IPS and propose the doubly robust estimator and then show the effectiveness [86]. However, these methods have a common problem: it is hard to obtain the precise estimator, i.e., the hyper-parameter γ in EIB and the propensity score in IPS.

Recently, researchers have followed Pearl's causal inference framework [61] in RS [83, 91, 95, 98]. Generally, they organize the relationship of the variables as a triangle structure: a cause, an effect, and a confounder [8, 65, 98]. Because of the existence of the confounder, there is a spurious relationship between the cause and the effect. Therefore, we need to cut off the path and remove the effect from the confounder in the inference stage [61]. For example, Sato et al. [65] treat the features of users and items as the confounder and derive the unbiased inference by re-weighting the samples according to the features.

Another branch of work formulates the inference problem as a counterfactual learning framework, where the counterfactual variable refers to the potential outcome of the unobserved data [84, 87]. By explicitly computing the potential outcome, we can conclude whether a treatment is effective, i.e., whether taking the treatment makes a huge difference compared with doing nothing.

In this work, we generally summarize the procedure of most of the work as follows: 1) Constructing a causal graph, i.e., a representation of the structural causal model (SCM) that describes the causal relationship among the related variables. 2) In the learning stage, fitting an unbiased model (e.g., implemented as a neural network) on the training dataset based on the proposed causal graph. 3) In the inference stage, actively changing certain variables (called *intervention*) according to the certain requirement, then predicting the unbiased result of the target variable. In this work, we use this framework to explicitly model the overexposure effect of the recommender system in the filter bubble problem.

2.3 Offline RL for Recommendation

Recommender systems are designed to enhance user satisfaction or increase sales when serving online users. We usually consider recommendation as a sequential decision-making process, where the recommender policy decides to make recommendations according to previous user feedback. Common studies on static recommendation models (e.g., DeepFM [25] and LightGCN [26]) only pay attention to improving the performance in a single-round recommendation. However, this solution often assumes that the recommendation follows the I.I.D. assumption, i.e., each item or recommendation is independent and identically distributed, which is not true in many cases. For example, in slate recommendation or bundle recommendation, the conversion rate of an item does not solely depend on itself. If it is surrounded by similar but expensive items, the conversion rate can increase. This phenomenon is known as the decoy effect [76]. Besides, many items have future impacts rather than instantaneous rewards, e.g., recommending a high-priced item may not result in an instant consumption behavior now, but it can leave the user an impression that there are high-quality items in this platform and this impression can result in transactions in the future when users have the ability to consume the expensive items. The overexposure effect of filter bubbles can also have an effect on users' experience, but the effect is pernicious, i.e., it can negatively affect users' willingness to continue to use and trust this recommender system.

To directly model the long-term success in the multi-round decision-making problem, we can adopt the interactive recommender system (IRS) based on reinforcement learning (RL) [6, 7, 18, 19, 39, 46, 48, 49, 81, 94, 97, 103, 105, 107]. The object of reinforcement learning is to maximize the cumulative reward $J(\pi_\theta)$ as:

$$J(\pi) = \mathbb{E}_{\tau \sim p_\pi(\tau)} \left[\sum_{t=0}^{H} \gamma^t r\left(\mathbf{s}_t, \mathbf{a}_t\right) \right]. \tag{1}$$

where $r\left(\mathbf{s}_t, \mathbf{a}_t\right)$ is a reward given from the environment when the agent makes action $\mathbf{a}_t$ at state $\mathbf{s}_t$, $\gamma \in (0, 1]$ is a scalar discount factor. The trajectory τ is a sequence of states and actions of length H, given by $\tau = (\mathbf{s}_0, \mathbf{a}_0, \ldots, \mathbf{s}_H, \mathbf{a}_H)$. $p_\pi(\tau)$ is a trajectory distribution for a given Markov decision process (MDP) and the policy π is written as follows:

$$p_\pi(\tau) = d_0\left(\mathbf{s}_0\right) \prod_{t=0}^{H} \pi\left(\mathbf{a}_t \mid \mathbf{s}_t\right) T\left(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t\right). \tag{2}$$

where $T\left(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t\right)$ is a state transition probability that describes the dynamics of the environment system.

Though effective, learning an IRS policy with online users is impractical as it is too slow and will hurt user experience [24]. On the other hand, the recorded recommender logs and offline user feedback are easier to obtain [36]. Therefore, it is natural to think of learning a policy on the offline data, which is the core idea of offline RL [40].

Commonly used offline RL strategies in recommendation include off-policy evaluation (OPE) learning [11, 75] and model-based RL methods [13, 106]. OPE-based methods estimate the cumulative reward of the target policy π_θ using

Table 1. Six Types of Recommender Systems

	User Model	Debiasing	RL-based	Publications
Static RS	✓			[25, 26, 38, 96]
Unbiased static RS	✓	✓		[41, 44, 64, 67, 83, 98, 102, 104]
Traditional IRS			✓	[45, 46, 49, 90, 97, 103, 105, 107]
Model-based IRS	✓		✓	[2, 13, 85, 100, 101, 106]
OPE-based IRS		✓	✓	[11, 32–34, 51, 54, 75, 89]
Unbiased model-based IRS	✓	✓	✓	[12, 18, 30] **CIRS (Ours)**

the weighted data from logged policy π_β. A classical importance sampling estimator [62] is listed as follows:

$$\begin{aligned} J\left(\pi_\theta\right) &= \mathbb{E}_{\tau\sim\pi_\beta(\tau)}\left[\frac{\pi_\theta(\tau)}{\pi_\beta(\tau)}\sum_{t=0}^{H}\gamma^t r(\mathbf{s},\mathbf{a})\right] \\ &= \mathbb{E}_{\tau\sim\pi_\beta(\tau)}\left[\left(\prod_{t=0}^{H}\frac{\pi_\theta\left(\mathbf{a}_t \mid \mathbf{s}_t\right)}{\pi_\beta\left(\mathbf{a}_t \mid \mathbf{s}_t\right)}\right)\sum_{t=0}^{H}\gamma^t r(\mathbf{s},\mathbf{a})\right] \approx \sum_{i=1}^{n} w_H^i \sum_{t=0}^{H}\gamma^t r_t^i \end{aligned} \tag{3}$$

where $w_t^i = \frac{1}{n}\prod_{t'=0}^{t}\frac{\pi_\theta\left(\mathbf{a}_{t'}^i|\mathbf{s}_{t'}^i\right)}{\pi_\beta\left(\mathbf{a}_{t'}^i|\mathbf{s}_{t'}^i\right)}$ and $\left\{\left(\mathbf{s}_0^i, \mathbf{a}_0^i, r_0^i, \mathbf{s}_1^i, \ldots\right)\right\}_{i=1}^{n}$ are n trajectory samples from the historical policy π_β. However, the estimator in OPE-based methods often suffers from the high variance problem due to the divergence between the two distributions in w_t^i.

The model-based methods attempt to estimate the transition function $T(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ and the reward function r_t of the environment. From the estimated model, the RL agent can plan its trajectory accordingly on the predicted states and rewards, rather than learning rigorously on the historical trajectories. The model-based method can avoid the high-variance problem in OPE-based methods, however, it suffers from the bias issue in estimating the model. In recommender systems, various biases were specified and investigated [10]. In this work, we specify the bias as to whether the model considers the user feeling bored in the interaction.

As mentioned above, CI is a powerful tool to address the bias problem in recommender systems. Hence, we combine the model-based offline RL and the CI technique to develop an unbiased IRS that can recognize and address the filter bubble problem on the offline data. We choose to use the model-based offline RL because it has strong advantage of being sample efficient [14, 57], which is crucial in recommender system where the data is highly sparse and expensive to collect. Besides, we can directly model the bias in the extracted transition probability and reward function via a causality-enhanced model, which is also a main contribution of this work.

Here, we briefly summarized in Table 1 six types of recommenders with respect to three dimensions: (1) whether the system explicitly builds a user model trying to capture real user preference, (2) whether the system considers debiasing, and (3) whether the system has an RL-based policy. Note that there are two other works in the same category as our CIRS, but they are not designed for the filter bubble problem. In addition, the unbiasedness of them is not referred to the ability to address a certain bias effect in the recommendation problem, but to make the estimation more accurate.

3 PREREQUISITES

In this section, we introduce the problem definition and the empirical analyses of the real-world data from Kuaishou.

3.1 Problem Definition

We denote the user set as $\mathcal{U}$ and item set as $\mathcal{I}$. The set of all interaction sequences of a user $u \in \mathcal{U}$ can be denoted as $\mathcal{D}_u = \{\mathcal{S}_u^1, \mathcal{S}_u^2, \cdots, \mathcal{S}_u^{|\mathcal{D}_u|}\}$. Each $\mathcal{S}_u^k \in \mathcal{D}_u$ is the k-th interaction sequence (i.e., trajectory) recording a complete interaction process: $\mathcal{S}_u^k = \{(u, i_l, t_l)\}_{1 \le l \le |\mathcal{S}_u^k|}$, where the user u begins to interact with the system at time t_1 and quits at time $t_{|\mathcal{S}_u^k|}$, and $i_l \in \mathcal{I}$ is the recommended item at time t_l. Let $\mathbf{e}_u \in \mathbb{R}^{d_u}$ and $\mathbf{e}_i \in \mathbb{R}^{d_i}$ be the feature representation vectors of user u and item i, respectively. For the system, the task is to recommend items to users based on their preferences and interaction history. This process can be cast as a reinforcement learning problem, whose key components are summarized as follows:

- **Environment**. The agent works in the environment where the states and rewards are generated. Here, the environment is the user (either an online real user or a simulated one) who can choose to rate the recommended items from the system or quit the interaction.
- **State**. The system maintains a state $\mathbf{s}_t \in \mathbb{R}^{d_s}$ at time t is regarded as a vector representing information of all historical interactions between user u and the system prior to t. In this paper, we obtain the $\mathbf{s}_t$ by using the Transformer model [80].
- **Action**. The system makes an action a_t at time t is to recommend items to user u. Let $\mathbf{e}_{a_t} \in \mathbb{R}^{d_a}$ denote the representation vector of action a_t. In this paper, each action a_t recommends only one item i. Hence, we have $\mathbf{e}_{a_t} = \mathbf{e}_i$.
- **Reward**. The user u returns feedback as a reward score r_t reflecting its satisfaction after receiving a recommended item i. The reward can also be the *counterfactual satisfaction* predicted by the causal user model ϕ_M instead of real users' feedback.
- **State Transition**. After the agent makes an action a_t and the user gives a reward r_t, the state $\mathbf{s}_t$ will be updated to $\mathbf{s}_{t+1}$ according to a state transition probability $T(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$. In our work, this transition probability is modeled by a state tracker implemented on the Transformer model [80].
- **Policy**. The key task of the system is to optimize a target policy $\pi_\theta = \pi_\theta(a_t|\mathbf{s}_t)$ that represents the probability of making an action a_t conditioned on the state $\mathbf{s}_t$. It decides how to generate an item i to recommend and is usually implemented as a fully-connected neural network.

To make full use of the historical data, we base our method CIRS on the offline RL framework. Learning CIRS requires three recurrent steps (as illustrated by three color blocks in Fig. 3):

(1) Training a causal user model ϕ_M on historical interaction data $\{(u, i, r)\}$ to estimate not only user interest but also the effect of item overexposure.
(2) Using the learned causal user model ϕ_M (instead of real users) to train policy π_θ. In each interaction loop, ϕ_M samples a user u to interact with π_θ. When π_θ makes an action a_t (recommends i), ϕ_M provides a *counterfactual satisfaction* as the reward r. Intuitively, if π_θ has made similar recommendations before, ϕ_M shrinks the reward r.
(3) Serving the learned policy π_θ to real users and evaluating the results in the interactive environment. When the interaction ends, we save the log $\{u, i, r, t\}$ to historical data for the purpose of future learning.

The three steps can be conducted repeatedly to continuously improve ϕ_M and π_θ.

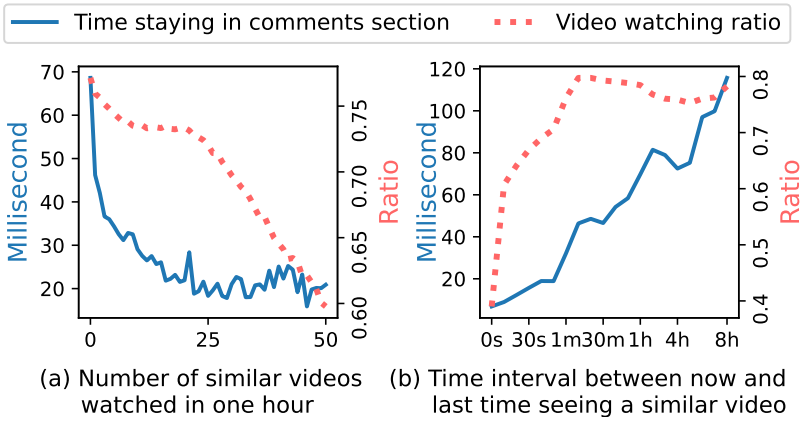


Fig. 2. Empirical studies of exposure effect on Kuaishou. Statistics of two metrics vary with the exposure effect.

3.2 Field Study of Overexposure Effect on User Satisfaction

As mentioned above, besides users' responsibility, the cause of filter bubbles is that the model incessantly recommends similar items to users. We suppose this is pernicious as users may not feel comfortable under such circumstances. Thus, we present a hypothesis as follows:

Hypothesis 1. *Users' satisfaction will drop if the recommended items (or similar items) are repeatedly exposed and recommended to them in a short time.*

To verify this hypothesis, we conduct empirical studies on real data from Kuaishou, a video-sharing mobile App. We chose 7,176 users from the video-watching user pool of the platform. We filter and collect their interaction history from August 5th, 2020 to August 11th, 2020. There are 34,215,294 views in total and 3,110,886 videos were watched. Each item is tagged with at least one and at most four-category tags, and there are 31 category tags in total.

During watching a video, users can choose to quit watching at any time by scrolling down to the next video or leaving the video-playing interface. Each video has a comments section that users can enter by clicking the "comment" button. If users are interested in a video, they will stay watching it for a longer time or enter its comments section. Therefore, we design two key metric indicators to reflect user satisfaction: One is the time duration staying in the comments section, and the other is the video watching ratio, which is the ratio of viewing time to the total video length.

To show how item exposure affects user satisfaction, we study how the two indicators change with the degree of overexposure effect. Specifically, we group all collected views with respect to (a) the number of videos with the same tag watched in one hour, or (b) the time interval between now (watching this video) and the last time viewing a video with the same tag. Then we compute the average values of the mentioned indicators in every group. The results are shown in Fig. 2. Two observations are found:

Observation 1. *User satisfaction towards a recommended item drops when the system increases the number of similar items in recent recommendations.*

Observation 2. *User satisfaction toward a recommended item drops as the time interval between two similar items is shortened.*

The result is statistically significant since even the smallest group contains enough points: 31, 277 points for the group at $x = 50$ in Fig. 2 (a) and 76, 303 points for the group at $x = (7h \sim 8h)$ in Fig. 2 (b).

In our previous work [18], an empirical analysis also demonstrated a similar phenomenon: user satisfaction decreases as the repetition rate of items or categories increases. Therefore, Hypothesis 1 is empirically proved. This suggests that the filter bubble, i.e., the overexposure effect of the recommendation algorithm, does have a pernicious impact on user satisfaction. Consequently, we can design a "feel bored then quit" exit mechanism in the constructed environments as a reflection of user behavior, which enables us to simulate the effect of filter bubbles effectively. We will illustrate it in Section 5.

4 PROPOSED METHODS

In this section, based on the observations obtained in the field studies, we propose a counterfactual interactive recommender system (CIRS) that leverages causal inference in offline RL. We first introduce CIRS's three main modules in the offline RL framework, then we describe how to leverage causal inference to disentangle the causal effects on user satisfaction.

4.1 Offline RL-based Framework

We base our CIRS model on offline RL, where we can utilize a large amount of offline data to train the interactive recommender system. The framework of CIRS is illustrated in Fig. 3. It contains three stages (shown in three color blocks): pre-learning stage, RL planning stage, and the RL evaluation stage. The functions of these three stages correspond to the three: (1) pre-learning the user model ϕ_M via supervised learning, (2) using the learned user model ϕ_M to learn RL policy π_θ by providing *counterfactual satisfaction* as the reward, and (3) evaluating policy π_θ in the real environment. Especially, the real environment can either be the real-world online environment or the simulation environment that can reflect real user behavior. Next, we separately introduce the three main components in CIRS: causal user model ϕ_M, the state tracker module, and RL-based interactive policy π_θ.

4.1.1 Causal User Model. The causal user model ϕ_M learns user interest based on the historical data and provides the counterfactual satisfaction r for the RL planning stage. It aims to explicitly disentangle the causal effect by correctly modeling the effect of item overexposure on user satisfaction. There are two sub-modules in the user model: an interest estimation module designed for computing the user's intrinsic interest y, and a counterfactual satisfaction estimation module capturing how the overexposure effect affects user satisfaction r. The details of this component will be reported in Section 4.2, where we will formally introduce the causal view of the recommender.

4.1.2 Transformer-based State Tracker. The state $\mathbf{s}_t$ in the interactive recommendation should include all critical information at time t for policy π_θ to make decisions. That includes: the feature vector $\mathbf{e}_u$ of the user u, the feature vectors of the recently recommended items, i.e., actions of the system in the interaction loop $\{\mathbf{e}_{a_1}, \cdots, \mathbf{e}_{a_t}\}$, and the user's feedback towards them $\{r_1, \cdots, r_t\}$. In order to automatically extract key information from these vectors, we use the Transformer model [80] to derive $\mathbf{s}_t$ as illustrated in Fig. 3. Transformer is a state-of-the-art sequence-to-sequence model with an attention mechanism that can capture the dependence between current input and previous input sequences [99]. In this work, we use only a two-layer encoder of Transformer. Since the input is generated sequentially, we need to add a mask to prevent future information leaking into each state of the sequence.

We further use a gate mechanism to filter information from the action $\mathbf{e}_{a_t}$ and user feedback r_t. Hence, the input for Transformer at time t is $\mathbf{e}'_{a_t} := \boldsymbol{g}_t \odot \mathbf{e}_{a_t}$. Where '$\odot$' denotes the element-wise product, and the gating vector $\boldsymbol{g}_t$ is

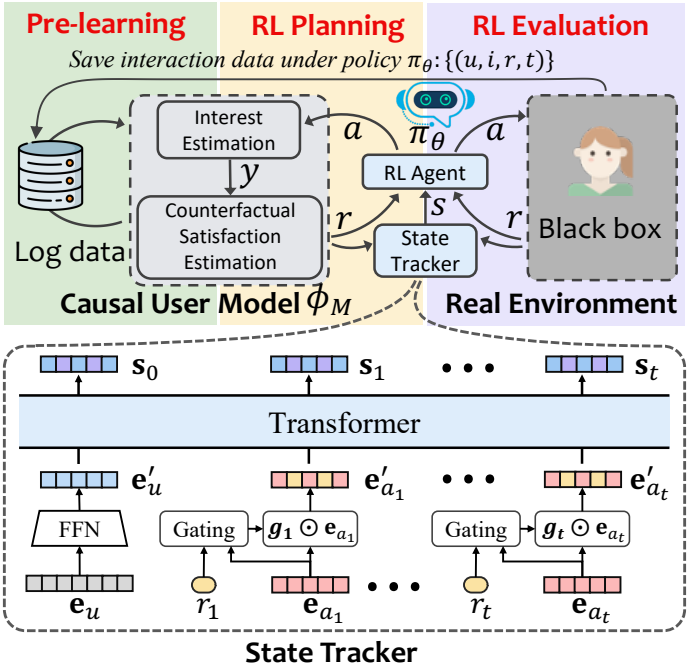


Fig. 3. Learning Framework of Counterfactual IRS.

computed as:

$$\boldsymbol{g}_t = \sigma\left(\mathbf{W} \cdot \text{Concat}\left(\mathbf{r}_t, \mathbf{e}_{a_t}\right) + \mathbf{b}\right), \tag{4}$$

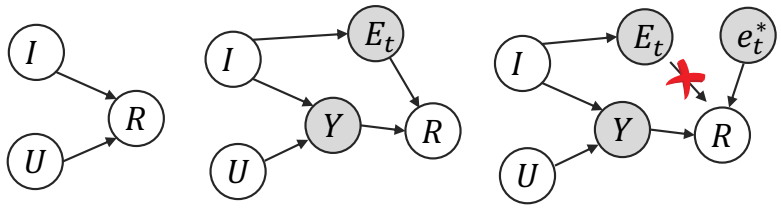
where $\mathbf{W} \in \mathbb{R}^{d_s \times (1+d_a)}$ and $\mathbf{b} \in \mathbb{R}^{d_s}$ refers to the weight matrix and bias vector, respectively. Concat$(\cdot)$ is the vector concatenation operator. Besides, we use a feed-forward network (FFN) to transform the representation vector of user u from $\mathbf{e}_u \in \mathbb{R}^{d_u}$ to $\mathbf{e}'_u \in \mathbb{R}^{d_s}$ to let it lie in the same space of $\mathbf{e}'_{a_t}$.

The feature vectors, the parameters in Transformer and the gate are initialized randomly and trained in the end-to-end manner using the gradients passed from the RL model.

It is noteworthy that the Transformer model and the gate mechanism can be replaced by other architectures, since there are other ways to extract and combine the information from the input sequence, such as a recurrent neural network (RNN)-based model [9, 23, 81]. Besides, The parameters in Transformer and the gate mechanism are fixed during the RL policy updating its parameters. Afterward, they are updated by another Adam optimizer [35], using the gradient that was back-propagated from the RL policy.

4.1.3 RL-based Interactive Recommendation Policy. We implement our interactive recommendation policy π_θ as the PPO algorithm [70]. PPO is a powerful on-policy reinforcement learning algorithm based on actor-critic framework [37] and the trust region policy optimization (TRPO) algorithm [68]. It can work in both discrete and continuous state space and action space. We aim to maximize the cumulative user satisfaction of the long-term interaction, which can be realized by maximizing the objective function of PPO:

$$\mathbb{E}_t\left[\min\left(\frac{\pi_\theta\left(a_t \mid \mathbf{s}_t\right)}{\pi_{\theta_{\text{old}}}\left(a_t \mid \mathbf{s}_t\right)}\hat{A}_t, \text{clip}\left(\frac{\pi_\theta\left(a_t \mid \mathbf{s}_t\right)}{\pi_{\theta_{\text{old}}}\left(a_t \mid \mathbf{s}_t\right)}, 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right], \tag{5}$$



(a) Causal graph in traditional RSs (b) Causal graph in our CIRS (c) Performing intervention on E_t during inference

Fig. 4. Causal graphs of traditional IRS models (a) and the CIRS model (b-c). U: a user, I: an item, R: user satisfaction (quantified by feedback such as clicks or viewing time), Y: intrinsic interest, E_t and e_t^*: the overexposure effect. Latent variables are shaded.

where ϵ is the hyperparameter that controls the maximum percentage of change that can be updated at one time. The function $\text{clip}(x, a, b)$ clips the variable x in the range of $[a, b]$. θ_{old} is the policy before updating, i.e., the interaction data are generated under policy θ_{old}. And the advantage function $\hat{A}_t$ is implemented as the generalized advantage estimator (GAE) [69] given as follows:

$$\hat{A}_t := \hat{A}_t^{\text{GAE}(\gamma,\lambda)} := \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V \tag{6}$$

where $\lambda \in [0, 1]$ is a hyperparameter making a compromise between bias and variance. δ_t^V is defined as $\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t)$, i.e., the TD residual of the approximate value function V with discount γ [74]. The value function V is defined as:

$$V(s_t) := V^{\pi_\theta,\gamma}(\mathbf{s}_t) := \mathbb{E}_{\mathbf{s}_{t+1:\infty},a_{t:\infty}}\left[\sum_{l=0}^{\infty} \gamma^l r_{t+l}\right] \tag{7}$$

It should be noted that the reward term r_t in $\hat{A}_t$ is the *counterfactual satisfaction* given by the causal user model ϕ_M instead of immediate users' feedback.

At last, to learn a good policy, we need the *counterfactual satisfaction* given by ϕ_M to be as correct and constructive as possible. Next, we will introduce how to build a causal user model ϕ_M to capture the overexposure effect thus avoiding the filter bubble.

4.2 Causal Inference-based User Satisfaction Disentanglement

We illustrate the causal graphs of the traditional recommender systems and our CIRS model in Fig. 4.

- Node U represents a certain user u, e.g., an ID or the profile feature that can represent the user.
- Node I represents an item i that is recommended to user u.
- Node R represents user u's real-time satisfaction for the recommended item i. It is the feedback such as a click or the video watching ratio.
- Node Y represents the user's intrinsic interest that is static regardless of item overexposure.
- Node E_t and e_t^* represent the overexposure effect of item i on user u. E_t is a random variable and e_t^* is the value of E_t computed in the inference stage (i.e., the RL planning stage).

Traditional recommender systems will fit user satisfaction R based on solely the feature information of user U and item I (Fig. 4 (a)). This assumes that user satisfaction equals to intrinsic interest, which is improper as stated before.

Our proposed CIRS model innovatively takes into account the overexposure effect E_t, and disentangles the causal effect on user satisfaction R. Concretely, R is generated from two causal paths:

(1) $(U, I) \rightarrow Y \rightarrow R$: This path projects user u and item i to their corresponding intrinsic interest y_{ui}. Then user satisfaction r_{ui} is proportional to y_{ui}.
(2) $I \rightarrow E_t \rightarrow R$: This path captures the real-time overexposure effect $e_t(u, i)$ of an item i on user u's satisfaction r. This effect negatively affects user satisfaction.

Intrinsic Interest Estimation. We estimate the user's intrinsic interest y_{ui} as $\hat{y}_{ui} = f_\theta(u, i)$. The estimation model $f_\theta(u, i)$ can be implemented by almost any established recommendation model, such as DeepFM [25] used in this work. We illustrated this part in the interest estimation module in Fig. 3.

Overexposure Effect Definition Considering that overexposure effect negatively affects user satisfaction, we define the overexposure effect e_t of recommending an item i to user u at time t as:

$$e_t := e_t(u, i) := \alpha_u \beta_i \sum_{(u, i_l, t_l) \in \mathcal{S}_u^k, t_l < t} \exp\left(-\frac{t - t_l}{\tau} \times \text{dist}(i, i_l)\right), \tag{8}$$

where $\text{dist}(i, i_l)$ is distance between two items i and i_l. α_u represents the *sensitivity* of user u to the overexposure effect, e.g., a user with a large α_u is more likely to feel bored when overexposed to similar content. Likewise, β_i represents the *unendurableness* of item i. For example, a classical music might be more endurable than a pop song, so β_i of the classical music is smaller. τ is a temperature hyperparameter. We will show the relationship between the learned α_u (β) and the activity of a user (the popularity of an item) in Section 5.5. Intuitively, when the recommended item i is close to the previously consumed items (e.g., i_l) of this user, i.e., $\text{dist}(i, i_l)$ is small, and its recommended time t is close to the recommended time of the previous ones (e.g., t_l of item i_l), i.e., the term $(t_l - t)$ is small, then the overexposure effect $e_t(u, i)$ will be large. This means that the recommender system is introducing a filter bubble to the user.

User Satisfaction Estimation. Generally, a similar item i_l (i.e., with smaller $\text{dist}(i, i_l)$) that was recommended recently (i.e., with smaller $t - t_l$) contributes a larger overexposure effect to item i. And e_t is the sum of the effect of all items recommended to the user u before time t. After obtaining the overexposure effect e_t via Eq. (8) and intrinsic interest $\hat{y}_{ui}$ via DeepFM, we estimate user u's satisfaction on item i as:

$$\hat{r}^t := \hat{r}^t_{ui} = \frac{\hat{y}_{ui}}{1 + e_t(u, i)}. \tag{9}$$

Therefore, a large exposure effect $e_t(u, i)$ will diminish user satisfaction $\hat{r}^t_{ui}$ even with unchanged intrinsic interest y_{ui}. In the training stage of causal user model ϕ_M, we minimize the objective function in the recommendation model. In experiments, we use the MSE loss for the VirtualTaobao and BPR loss for KuaiEnv:

$$L_{\text{MSE}} = \sum_{(u,i,t) \in \mathcal{D}} \left(\hat{r}^t_{ui} - r^t_{ui}\right)^2, \quad L_{\text{BPR}} = -\sum_{(u,i,t) \in \mathcal{D}, j \sim p_n} \log\left(\sigma\left(\hat{r}^t_{ui} - \hat{r}^t_{uj}\right)\right). \tag{10}$$

Where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the Sigmoid function. The item j is a negative instance sampled from the distribution p_n.

Counterfactual Satisfaction Estimation. In the RL planning stage, when the learned causal user model ϕ_M interacts with the policy π_θ, the overexposure effect e^*_t now is different to e_t in the pre-learning stage. Therefore, we perform the causal intervention $do(E_t = e^*_t)$ [61] by cutting off the path $I \rightarrow E_t \rightarrow R$ as shown in Fig. 4(c). Unlike traditional causal methods aiming to remove the effect of confounders [82, 83], we still need to model the correct overexposure effect e^*_t

in this stage. Note that we use the asterisk to mark out all values in this intervention stage. We compute e_t^* as:

$$e_t^*(u,i) = \gamma^* \cdot \alpha_u \beta_i \sum_{\left(u,i_l^*,t_l^*\right) \in \mathcal{S}_u^*, t_l^* < t} \exp\left(-\frac{t - t_l^*}{\tau^*} \times \text{dist}(i, i_l^*)\right), \tag{11}$$

where $\mathcal{S}_u^*$ is the new interaction trajectory produced in the RL planning stage. γ^* is a hyper parameter introduced to adjust the scale of the overexposure effect. We fix γ to be 10 throughout experiments. τ^* is the temperature hyperparmerter in the intervention stage and can have a different value with τ in Eq. (8). We estimate the *counterfactual satisfaction* as:

$$\hat{r}_{ui}^{t*} = \frac{\hat{y}_{ui}}{1 + e_t^*(u,i)}. \tag{12}$$

By now, we can use the estimated *counterfactual satisfaction* as the reward signal to update the RL policy by optimizing Eq. (5). The whole process is illustrated in the pre-learning and RL planning stage in Fig. 3. We use the Adam optimizer [35] in learning the causal user model ϕ_M, the policy π_θ, and the state tracker.

At last, by innovatively enhancing the offline RL framework with causal inference, we obtain a policy that can guarantee large user satisfaction by preventing filter bubbles, i.e., overexposing items.

5 EXPERIMENTS

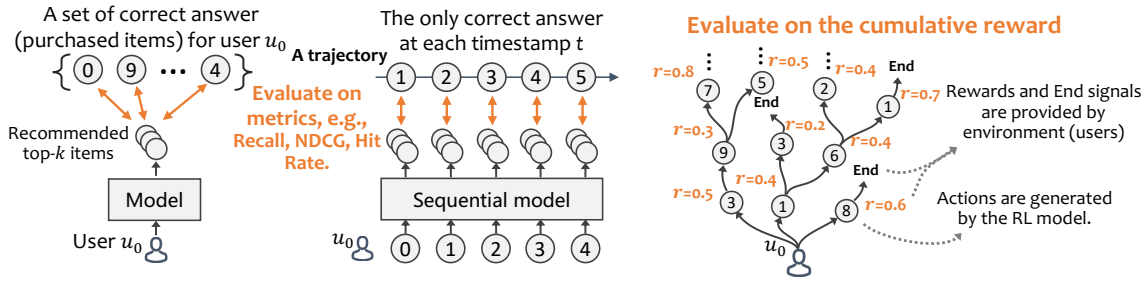
In this section, we conduct experiments to evaluate the IRS. We aim to investigate the following research questions:
(RQ1) How does CIRS perform compared with SOTA static recommendation methods and RL-based interactive recommendation policies?
(RQ2) How does CIRS perform in a limited number of interactive rounds?
(RQ3) How does CIRS perform in different environments with varying user tolerance of filter bubble?
(RQ4) What is the effect of the key parameters in CIRS?

5.1 Experimental Setup

We introduce the experimental settings with regards to the settings, environments, evaluation metrics, and state-of-the-art recommendation methods.

5.1.1 Evaluation in the Interactive Recommendation Setting. We emphasize that we evaluate all methods in the interactive setting rather than traditional static or sequential settings. Fig. 5 illustrates how static recommendation, traditional sequential recommendation, and interactive recommendation evaluate the models. Both the static and sequential recommendations use the philosophy of supervised learning, i.e., evaluating the top-k results by comparing them with a set of "correct" answers in the test set and computing metrics such as Precision, Recall, NDCG, and Hit Rate. By contrast, interactive recommendation evaluates the results by accumulating the rewards along the interaction trajectories. There is no standard answer in interactive recommendation, which is interesting yet challenging. This setting requires offline data of high quality, which hampers the related research.

Studying filter bubbles in the interactive recommendation setting is necessary. Filter bubble is the phenomenon that occurs in real-world recommendation scenarios when the recommender overexposes similar content to a user. This means user satisfaction can change dynamically, hence it is inappropriate to study the filter bubble in traditional static or sequential settings where user preference in the test set is fixed.



(a) Static Recommendation (b) Sequential Recommendation (c) Interactive Recommendation

Fig. 5. Illustration of evaluation in three different recommendation settings.

For now, interactive recommendation setting has not been widely investigated because it is hard to evaluate the model on the offline data. We overcome this problem by evaluating in the VirtualTaobao and KuaiEnv environments which are described in the following section.

5.1.2 Recommendation Environments. Traditional recommendation datasets are too sparse to evaluate the interactive recommender systems. We use two recommendation environments, VirtualTaobao[2] [71] and KuaiEnv[3]. The two environments can play the same role as the online real users. For the recommenders, an environment is like a black box as shown in the upper right corner of Fig. 3.

VirtualTaobao is a benchmark RL environment for recommendation. It is created by simulating the behaviors of real users on Taobao, one of the largest online retail platforms, via a multi-agent adversarial imitation learning (MAIL) approach. The simulated users can imitate the behavior of the real users and generate the same statistics as recorded on the Taobao platform. In the VirtualTaobao environment, a user is represented as an 88-dimensional vector $\mathbf{e}_u \in \{0,1\}^{88}$, and a recommendation is represented as a 27-dimensional vector $\mathbf{e}_i \in \mathbb{R}^{27}, \mathbf{0} \leq \mathbf{e}_i \leq \mathbf{1}$. When a model makes a recommendation $\mathbf{e}_i$, the environment will immediately return a reward signal representing user interest, i.e., a scalar $r \in \{0, 1, \cdots, 10\}$.

It provides $100,000$ logged interactions for training the offline RL policy. Since the items are represented as the continuous vectors in VirtualTaobao, we use Euclidean distance to compute the distance between two items, i.e., $\text{dist}(i, i_l)$ term in Eq. (8) and Eq. (11).

KuaiEnv is created by us on the KuaiRec dataset [20]. KuaiRec is a real-world dataset that contains a fully-observed user-item interaction matrix. The term "fully-observed" means that each user has viewed each video in the whole set and then left feedback. Therefore, unlike VirtualTaobao which simulates real users by training a model on Taobao data, i.e., the reward representing user preference is provided from a generative model, KuaiEnv uses real user historical feedback for each user-item pair, which can be more persuasive. We define the reward signal as the video watching ratio which is the ratio of viewing time to the total video length. Without loss of generality, we use this floating-point number to indicate users' intrinsic interest, i.e., we assume that the preference of a user keeps static and equals the logged ratings. We use the fully-observed matrix, i.e., *small matrix*, to evaluate the policy π_θ. For pre-learning the user model ϕ_M, we use the additional sparse user–video interactions in the *big matrix*. In the RL planning stage, we learn

[2]https://github.com/eyounx/VirtualTaobao
[3]https://kuairec.com

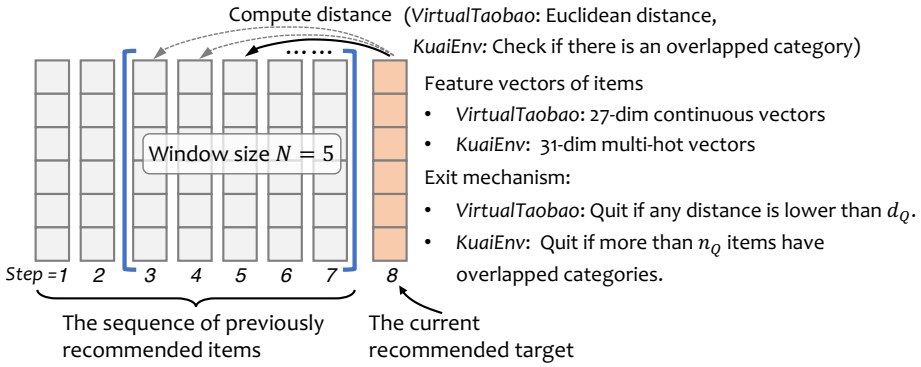


Fig. 6. Illustration of the exit mechanism.

the policy model π_θ by using the rewards provides by user model ϕ_M without levering offline data. For the details of the data, please refer to the KuaiRec dataset [20].

Each video in KuaiRec has at least one and no more than four categorical attributes, e.g., Food or Sports. Hence we use the Hamming distance for computing the term $\text{dist}(i, i_l)$ in Eq. (8) and Eq. (11).

Exit Mechanism. By now, VirtualTaobao and KuaiEnv can provide users' intrinsic interest as the reward signal. However, they cannot reflect users' responses to the overexposure effect. To this end, we introduce the "feel bored then quit" mechanism in two environments to penalize filter bubbles in evaluation. Usually, the environments will repeatedly interact with the recommender. VirtualTaobao has only a naive mechanism for ending the interaction by predicting the length of the interaction trajectory in advance. It is not in control and we alter it by considering the observations found in Section 3.2. The exit mechanism is illustrated in Fig. 6. Concretely, we compute the Euclidean distance between the recommended target and the most recent N recommended items. If any of them is lower than the threshold d_Q, the environment will quit the interaction process as the real users can feel bored and quit under such monotonous recommendation. In KuaiEnv, similarly, for the most recent N recommended items, if there are more than n_Q items in the N items have at least one attribute of the current recommended target, then the user in this environment ends the interaction process. Intuitively, a good recommender should avoid repeating highly similar items to prevent users from quitting early.

5.1.3 Evaluation Metrics. We aim to evaluate the model performance with regard to cumulative satisfaction over the whole interaction trajectory $\mathcal{S}$, i.e., $\sum_{l=0}^{|\mathcal{S}|} r_{t+l}$, where r_t is the reward signal returned by VirtualTaobao or KuaiEnv. Note that in this setting, user satisfaction is set as:

$$\text{satisfaction} = \begin{cases} \text{interest}, & \text{if no filter bubble ever occurs,} \\ 0, & \text{otherwise.} \end{cases}$$

I.e., if the recommendation does not trigger the exit mechanism, we can accumulate the rewards to represent intrinsic interest. But whenever an overexposed item triggers the exit mechanism, the interaction is interrupted and no reward can be added anymore. Thereby, the system cannot repeat to recommend the several high-quality items of the maximum confidence. Intuitively, to pursue long-term success, the recommender policy must find a trade-off between pursuing a higher single-round satisfaction and maintaining a longer interaction sequence.

We report the average cumulative satisfaction over 100 interaction sequences.

5.1.4 Baselines. We use the commonly used static recommendation models plus straightforward policies as baselines. For KuaiEnv which has rich item features, four static recommendation baselines are:

- **DeepFM** [25], which is a powerful factorization machine-based neural network containing wide and deep parts to extract knowledge from low- and high-order feature interactions. It serves as a strong backbone in the recommender framework in many companies.
- **IPS** [75] is a well-known statistical technique adjusting the target distribution by re-weighting each sample in the collected data. In recommendation, it is widely used for modeling the probability of observation in order to remove the exposure bias or selection bias in the collected data. It is easy to implement and suffers from the high variance issue [77].
- **PD** (Popularity-bias Deconfounding) [98] is a causal inference-based method that models item popularity as a confounder introducing spurious correlations between exposed items and user preference. By explicitly modeling popularity, PD can remove the popularity bias in the final recommendation stage.
- **DICE** [102] tries to disentangle popularity and user interest by separately modeling them in the so-called causal embedding. Therefore, item popularity or other unwanted factors can be removed in the recommendation stage.

It should be noticed that: (1) IPS, PD, and DICE are techniques used for debiasing, and we implement their backbone network as DeepFM. (2) All these methods are deterministic/static models, in which researchers usually take the items with top-1 or top-k highest predicted scores as the final recommendations. In our interactive setting, this manner will incur overexposure immediately. Therefore, we make the recommendation by sampling from the final logits with a Softmax layer.

We also implement basic policies in KuaiEnv:

- **Random**, which recommends random items completely.
- ϵ-**greedy**, which outputs a random result with probability ϵ and uses the results from the DeepFM model with probability $1-\epsilon$.
- **UCB** maintains an upper confidence bound for each item and follows the principle of optimism in the face of uncertainty. It means if we are uncertain about an action, we should give it a try. UCB can balance the exploration and exploitation in decision-making process.

For VirtualTaobao, since the users and items are given by feature vectors, we can only implement a multilayer perceptron (**MLP**) with sampling on the Softmax results and the ϵ-greedy strategy. Besides, we implement the powerful RL baseline, PPO [70], on the same offline RL framework, i.e., it is learned by interacting with the user model but without the causal inference module. For comparison, we denoted this baseline as **CIRS w/o CI**.

Note that PPO is one component in the model-based offline RL and it can be replaced by other RL models (e.g., DQN [56], Actor-Critic [37], or DDPG [43]). **CIRS w/o CI** can be deemed as a framework that summarizes model-based offline RL methods [29, 30]. For example, the difference with Huang et al. [30] is that they used MF as the user model while we use DeepFM; the difference with Huang et al. [29] is that they investigated a lot of sequential models as state trackers while we use a Transformer-based model.

5.2 Overall Performance Comparison

We evaluate the proposed CIRS and baselines in two environments. We use grid search to tune the optimal parameters for all methods. For example, in VirtualTaobao, the key parameter τ^* is searched in $\{0.001, 0.005, 0.01, 0.1, 0.5, 1.0, 5.0\}$

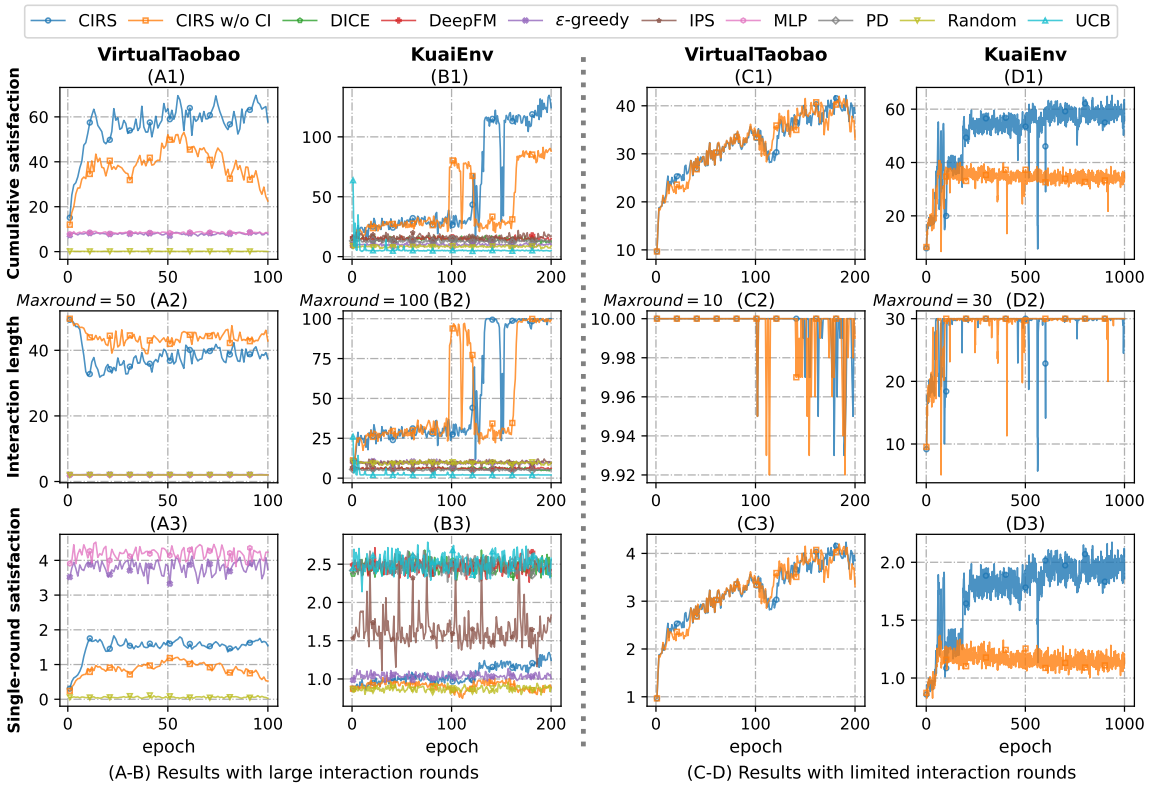


Fig. 7. Results of all methods with large interaction rounds (Section 5.2) and limited interaction rounds (Section 5.3).

and τ is searched in $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0\}$. The results are illustrated in Fig. 10. For more implementation details, please visit the instruction via this Github link[4].

For general comparison, we do not limit the length of the interaction and set the max round to be large enough (but feasible to implement). We set the max round to be 50 and 100 for VirtualTaobao and KuaiEnv, respectively. For the parameters in the environment setting, we set the window size, i.e., the number of the most recent recommendations, to be $N = 5$ and the exit threshold to be $d_Q = 3.0$ for VirtualTaobao and $N = 1$, $n_Q = 1$ for KuaiEnv.

The results are shown in Fig. 7 (A-B). The first row shows the cumulative satisfaction, which is the global metric to evaluate the recommender systems. The second row and third rows show the details of the user satisfaction, i.e., the length of the interaction trajectory and the single-round satisfaction, respectively. From (A1) and (B1), we can see the proposed CIRS achieves the maximal average cumulative satisfaction after several epochs in both VirtualTaobao and KuaiEnv.

In the first few epochs in VirtualTaobao, the performances of both CIRS and CIRS w/o CI improve because the RL policy gradually finds the correct user preference so the satisfaction in each round increases (A3). Interestingly, the increase in the single-round satisfaction compromises the length of trajectory at the beginning (A2). Later, the length gradually becomes stable, and the policy of CIRS eventually finds a balance point between length and single-round

[4]https://github.com/chongminggao/CIRS-codes/tree/main/reproduce_results_of_our_paper

satisfaction, thus achieving the maximal cumulative satisfaction. However, without the causal inference module, i.e., as shown by CIRS w/o CI, the policy becomes unstable and the performance degenerates with the epoch increases. This phenomenon demonstrates the effectiveness of causal inference in capturing the overexposure effect and thus avoiding repeatedly recommending items.

In KuaiEnv, CIRS also achieves the largest cumulative satisfaction (B1) after enough epochs. Unlike in VirtualTaobao, the performance increases mainly because of the increase in the interaction length. As shown in (B2), the interaction lengths of both CIRS and CIRS w.o. CI increases to the maximum length of 100 after about 160 epochs. In addition, the cumulative satisfaction of CIRS further increases at about 180 epochs in (B1), which is due to the possibility of further improving the single-round performance (B3). For both VirtualTaobao and KuaiEnv, CIRS beats the counterpart method CIRS w/o CI, which demonstrates the effectiveness of the causal module in CIRS.

For other baselines, we can see that all other methods except Random can achieve better single-round performance (A3 and B3). However, their recommendation results are too limited and narrow even with the randomness introduced by the basic policies (i.e., Random sampling, Softmax-based sampling, and ϵ-greedy). Note that in VirtualTaobao, even the random sampling cannot bring a longer interaction sequence because of the curse of dimensionality: The action space has 88 dimensions, therefore, the Euclidean distance of any two random points becomes statistically indiscriminate. The result of IPS fluctuates intensely in terms of the single-round performance (B3), which is due to the widely discussed high variance issue [75]. The interaction lengths of ϵ-greedy and IPS are longer than other methods (i.e., DICE, PD, and DeepFM) in (B2). This is because the two methods have the ability to explore the item space during the whole interaction process. Compared with these two naive methods, UCB is a policy that can automatically balance exploration and exploitation, it has the best performance at the beginning. However, after several epochs of exploration, the policy enhances its belief in certain items and thus leads to getting stuck in the filter bubble. Therefore, UCB ends up with the lowest interaction length as shown in (B2) but with the maximum single-round satisfaction in (B3).

To conclude, except for the deep RL policy-based methods (i.e., CIRS and CIRS w/o CI), static recommendation models with heuristic policies (i.e., Softmax-based sampling, and ϵ-greedy, and UCB) cannot overcome the overexposure effect, thus lead to the filter bubbles and result in low user satisfaction. Furthermore, by comparing CIRS with CIRS w/o CI, we show the effectiveness of causal inference in the offline RL framework.

5.3 Results with Limited Interaction Rounds

In real-world recommendation scenarios, users have limited energy and will not spend too many rounds interacting with the recommender. Therefore, we limit the max round to 10 and 30 in VirtualTaobao and KuaiEnv, respectively. We aim to investigate whether the policy can exploit to improve the single-round satisfaction under such a situation. We alter the exit threshold $d_Q = 1.0$ in VirtualTaobao for better demonstration.

From the results in Fig. 7 (C-D), we can see that CIRS outperforms CIRS w/o CI in KuaiEnv (D1 and D3), and produces a similar performance in VirtualTaobao (C1 and C3). In VirtualTaobao, both the two policies achieve the same level of single-round performance (greater than 4.0) after approximately 150 epochs. And the performance is even similar to the static methods (C3 and A3). In KuaiEnv, both the two policies achieve higher single-round performances (D3) compared with that in the former setting (B3). Especially, CIRS has a great improvement in single-round performance (D3 and B3), which means that it is suitable for real-world interactive recommendation scenarios with limited interaction rounds. Actually, in (B1 and B3), we can also see that the performance of CIRS continues to increase at $epoch = 200$. This means CIRS has potential even under huge rounds, given enough training time. The results in (C3 and D3) show that the knowledge, i.e., the correct user preference can be distilled from the causal user model to the RL policy. This further

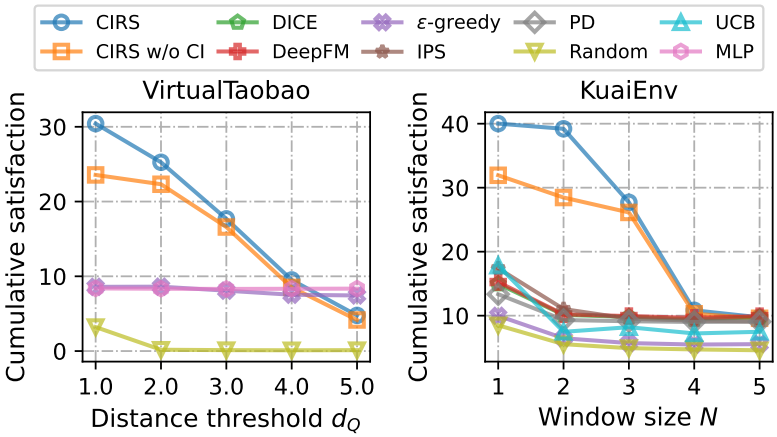


Fig. 8. Results under different user sensitivity

demonstrates the effectiveness of our model-based RL framework. Again, in this setting, we conclude that by integrating with causal inference, we let CIRS outperform its counterpart.

5.4 Results with Different User Sensitivity

To validate the generality of CRS, we vary the parameters: the distance threshold d_Q and window size N to illustrate their effects in VirtualTaobao and KuaiEnv, respectively. A large d_Q or N means that users get more sensitive to filter bubbles and become easier to quit the interaction. The results in Fig. 8 show that CIRS outperforms all baseline methods when users are less sensitive, i.e., small d_Q in VirtualTaobao and small N in KuaiEnv. CIRS obtains the best cumulative satisfaction because it can avoid repeating recommending highly similar items and thus can maintain a long interaction length.

However, the performance of CIRS inevitably decreases when users become more sensitive, though it can still beat its counterpart CIRS w/o CI. When $d_Q \geq 4$ or $N \geq 4$, CIRS and CIRS w/o CI can only achieve the same or even worse performance compared with other baselines. This means that facing extremely picky users, even the model enhanced by causal inference cannot alleviate the dissatisfaction caused by overexposing any item. When $d_Q = 5$ or $N = 5$, the two RL-based models even cannot beat DeepFM or MLP which are served as the teacher models in the two model-based RL methods. This is because the RL-based methods do not have the opportunity to conduct their explore-exploit philosophy when the user is too picky.

Meanwhile, other baselines have similar performance under different user sensitivity — the recommendations make users feel bored and quit even though the user is more tolerant to filter bubbles (i.e., less sensitive to item overexposure). This also demonstrates they are not suitable for addressing the filter bubble issue.

5.5 Effect of Key Parameters

We investigate the effect of the key parameters of CIRS in KuaiEnv. We compare the learned α_u and β_i in the user model with two statistics of data, i.e., the activity of the users and the popularity of items, which are derived by summing the rows and columns of the *big matrix*. We show the results in Fig. 9. The results are intuitive to understand: user sensitivity,

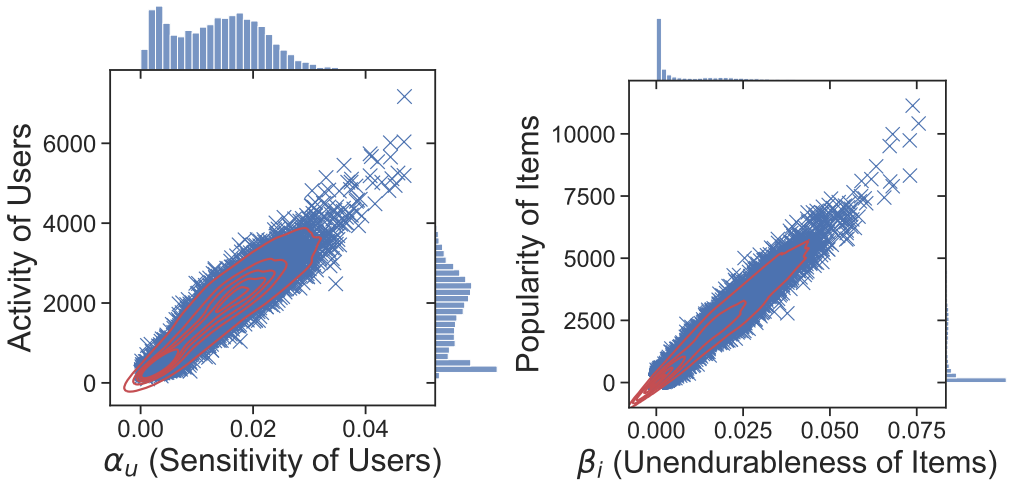
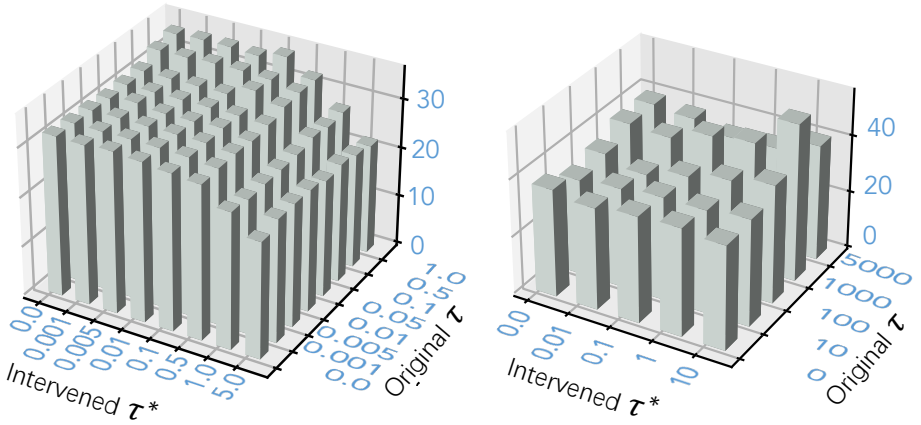


Fig. 9. Relationship between the learned α_u, β_i and the data statistics (user activity and item popularity).



(a) VirtualTaobao (b) KuaiEnv

Fig. 10. Cumulative satisfaction with different τ-τ^* pairs.

i.e., α_u, is proportional to the user activity, i.e., an active user is easier to get bored when viewing overexposed videos because he/she may have seen many similar content before. Similarly, item unendurableness, i.e., β_i, is proportional to the item popularity, i.e., popular videos are less endurable when they are overexposed. This also explains why popular items become outdated quickly.

Furthermore, we investigate the effect of different combinations of τ (in Eq. (8)) and τ^* (in Eq. (11)) on the cumulative satisfaction. CIRS with $\tau = 0, \tau^* = 0$ degenerates to CIRS w/o CI since both $e_t(u,i)$ and $e_t^*(u,i)$ become 0, i.e., the modeling of user satisfaction will not take into account the overexposure effect. The results in Fig. 10 demonstrate that suitable τ-τ^* pairs indeed improve the performance compared to CIRS w/o CI. Note that the orders of magnitude of the

τ-axis and τ^*-axis differ greatly in KuaiEnv, because the unit of time in Eq. (8) and Eq. (11) are different. The former uses the second(s) in log data and the latter uses the step(s) in the RL planning and evaluation stages.

6 CONCLUSION AND DISCUSSION

This work studies filter bubbles in the interactive recommendation setting. Different from the static and sequential recommendation settings which use the philosophy of supervised learning, the interactive setting evaluates the RL-based policies by accumulating the rewards along the interaction trajectories.

The interactive recommendation setting provides a practical way to track and estimate the filter bubble, which is the phenomenon that occurs in real-world recommendation scenarios when the recommender overexposes similar content to a user. We conduct field studies on music and video recommendation datasets and show that user satisfaction will drop with the increasing of similar content, which spurs us to remove filter bubbles in recommender systems.

We propose a counterfactual interactive recommender system (CIRS), leveraging causal inference in offline RL to deduce users' varying satisfaction. CIRS utilizes a causal user model that can disentangle the intrinsic user interest from the overexposure effect of items. The causal user model provides unbiased *counterfactual rewards* for learning the RL policy. To conduct evaluations, we innovatively create a faithful RL environment, KuaiEnv, based on a real-world fully-observed user rating dataset. Extensive experiments demonstrate that the proposed method can burst filter bubbles and increase users' cumulative satisfaction. The experiments show that CIRS can obtain optimal cumulative satisfaction by finding the trade-off between pursuing a high single-round satisfaction and maintaining a long-lasting interaction.

Our work has several noteworthy contributions. The most important one is that we demonstrate the right way to evaluate RL-based methods in the interactive recommendation setting, i.e., evaluating the decision-makers (i.e., recommendation policies) by the cumulative reward. In reality, real users do not have any standard answers in their minds when they use recommender systems; and the companies care whether the model can make users satisfied in the long term. Therefore, the interactive recommendation setting can well describe real-world recommendation scenarios.

However, many previous works still evaluate the RL-based methods via the static or sequential settings, i.e., evaluating the top-k results by comparing them with a set of "correct" answers in the test set and computing metrics such as Precision, Recall, NDCG, and Hit Rate [73, 90]. We understand why they chose to evaluate that way: the evaluation of RL is notoriously hard on offline data. To overcome this problem, we create the KuaiEnv environment in which each user's preference towards all items is known. With this environment, researchers can conduct faithful evaluation without having to synthesize user preference in simulated user-item matrices [11, 29, 30].

By modeling and alleviating filter bubble issues in the interactive recommendation setting, we demonstrate the potential research directions and possible solutions in the recommendation community. In the future, we believe that the interactive recommendation will draw a lot of research attention. It is interesting to explore other types of biases in this setting. Combining causal inference and reinforcement learning is promising since causal inference can provide an explicit guide to optimize models and thus introduce explainability in RL [3, 52].

ACKNOWLEDGEMENTS

This work is supported by the National Key Research and Development Program of China (2021ZD0111802), the National Natural Science Foundation of China (61972372, U19A2079, 62121002), and the CCCD Key Lab of Ministry of Culture and Tourism.

REFERENCES

[1] Guy Aridor, Duarte Goncalves, and Shan Sikdar. 2020. Deconstructing the Filter Bubble: User Decision-Making and Recommender Systems. In *RecSys '20*. 82–91.

[2] Xueying Bai, Jian Guan, and Hongning Wang. 2019. A Model-Based Reinforcement Learning with Adversarial Training for Online Recommendation. In *NeurIPS '19*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.).

[3] Elias Bareinboim. 2020. Causal Reinforcement Learning. In *ICML 2020 Tutorial*.

[4] Stephen Bonner and Flavian Vasile. 2018. Causal Embeddings for Recommendation. In *RecSys '18*. 104–112.

[5] Axel Bruns. 2019. *Are Filter Bubbles Real?* John Wiley & Sons.

[6] Qingpeng Cai, Shuchang Liu, Xueliang Wang, Tianyou Zuo, Wentao Xie, Bin Yang, Dong Zheng, Peng Jiang, and Kun Gai. 2023. Reinforcing User Retention in a Billion Scale Short Video Recommender System. *arXiv preprint arXiv:2302.01724* (2023).

[7] Qingpeng Cai, Zhenghai Xue, Chi Zhang, Wanqi Xue, Shuchang Liu, Ruohan Zhan, Xueliang Wang, Tianyou Zuo, Wentao Xie, Dong Zheng, et al. 2023. Two-Stage Constrained Actor-Critic for Short Video Recommendation. *arXiv preprint arXiv:2302.01680* (2023).

[8] Allison J. B. Chaney, Brandon M. Stewart, and Barbara E. Engelhardt. 2018. How Algorithmic Confounding in Recommendation Systems Increases Homogeneity and Decreases Utility. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys '18)*. 224–232.

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

[10] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He†. 2022. Bias and Debias in Recommender System: A Survey and Future Directions. *ACM Transactions on Information Systems* (oct 2022). https://doi.org/10.1145/3564284 Just Accepted.

[11] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H. Chi. 2019. Top-K Off-Policy Correction for a REINFORCE Recommender System. In *WSDM '19*. 456–464.

[12] Minmin Chen, Bo Chang, Can Xu, and Ed H. Chi. 2021. User Response Models to Improve a REINFORCE Recommender System. In *WSDM '21*. 121–129.

[13] Xinshi Chen, Shuang Li, Hui Li, Shaohua Jiang, Yuan Qi, and Le Song. 2019. Generative adversarial user model for reinforcement learning based recommendation system. In *ICML '19*. 1052–1061.

[14] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. 2013. Gaussian Processes for Data-efficient Learning in Robotics and Control. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37, 2 (2013), 408–423.

[15] Tim Donkers and Jürgen Ziegler. 2021. The Dual Echo Chamber: Modeling Social Media Polarization for Interventional Recommending. In *RecSys '21*. 12–22.

[16] Amir Feder, Katherine A Keith, Emaad Manzoor, Reid Pryzant, Dhanya Sridhar, Zach Wood-Doughty, Jacob Eisenstein, Justin Grimmer, Roi Reichart, Margaret E Roberts, et al. 2021. Causal Inference in Natural Language Processing: Estimation, Prediction, Interpretation and Beyond. *arXiv preprint arXiv:2109.00725* (2021).

[17] Seth Flaxman, Sharad Goel, and Justin M Rao. 2016. Filter Bubbles, Echo Chambers, and Online News Consumption. *Public Opinion Quarterly* 80, S1 (2016), 298–320.

[18] Chongming Gao, Kexin Huang, Jiawei Chen, Yuan Zhang, Biao Li, Peng Jiang, Shiqi Wang, Zhong Zhang, and Xiangnan He. 2023. Alleviating Matthew Effect of Offline Reinforcement Learning in Interactive Recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Taipei, Taiwan) *(SIGIR '23)*. 11 pages. https://doi.org/10.1145/3539618.3591636

[19] Chongming Gao, Wenqiang Lei, Xiangnan He, Maarten de Rijke, and Tat-Seng Chua. 2021. Advances and Challenges in Conversational Recommender Systems: A Survey. *AI Open* 2 (2021), 100–126.

[20] Chongming Gao, Shijun Li, Wenqiang Lei, Jiawei Chen, Biao Li, Peng Jiang, Xiangnan He, Jiaxin Mao, and Tat-Seng Chua. 2022. KuaiRec: A Fully-Observed Dataset and Insights for Evaluating Recommender Systems. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management* (Atlanta, GA, USA) *(CIKM '22)*. 540–550. https://doi.org/10.1145/3511808.3557220

[21] Chongming Gao, Shuai Yuan, Zhong Zhang, Hongzhi Yin, and Junming Shao. 2019. BLOMA: Explain Collaborative Filtering via Boosted Local Rank-One Matrix Approximation. In *DASFAA '19*. Springer, 487–490.

[22] Mingkun Gao, Hyo Jin Do, and Wai-Tat Fu. 2018. Burst Your Bubble! An Intelligent System for Improving Awareness of Diverse Social Opinions. In *IUI '18*. 371–383.

[23] Yingqiang Ge, Shuchang Liu, Ruoyuan Gao, Yikun Xian, Yunqi Li, Xiangyu Zhao, Changhua Pei, Fei Sun, Junfeng Ge, Wenwu Ou, and Yongfeng Zhang. 2021. Towards Long-Term Fairness in Recommendation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining (WSDM '21)*. 445–453.

[24] Alexandre Gilotte, Clément Calauzènes, Thomas Nedelec, Alexandre Abraham, and Simon Dollé. 2018. Offline A/B Testing for Recommender Systems. In *WSDM '18*. 198–206.

[25] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine Based Neural Network for CTR Prediction. In *IJCAI'17*. 1725–1731.

[26] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR '20*. 639–648.

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

[28] MA Hernán and JM Robins. 2020. *Causal Inference: What If*. Boca Raton: Chapman & Hall/CRC.

[29] Jin Huang, Harrie Oosterhuis, Bunyamin Cetinkaya, Thijs Rood, and Maarten de Rijke. 2022. State Encoders in Reinforcement Learning for Recommendation: A Reproducibility Study. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 2738–2748. https://doi.org/10.1145/3477495.3531716

[30] Jin Huang, Harrie Oosterhuis, Maarten de Rijke, and Herke van Hoof. 2020. Keeping Dataset Biases out of the Simulation: A Debiased Simulator for Reinforcement Learning Based Recommender Systems. In *RecSys '20*. 190–199.

[31] Eslam Hussein, Prerna Juneja, and Tanushree Mitra. 2020. Measuring Misinformation in Video Search Platforms: An Audit Study on YouTube. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW1, Article 048 (May 2020).

[32] Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Morgane Lustman, Vince Gatto, Paul Covington, et al. 2019. Reinforcement learning for slate-based recommender systems: A tractable decomposition and practical methodology. *arXiv preprint arXiv:1905.12767* (2019).

[33] Rolf Jagerman, Ilya Markov, and Maarten de Rijke. 2019. When People Change Their Mind: Off-Policy Evaluation in Non-Stationary Recommendation Environments. In *WSDM '19*. 447–455.

[34] Olivier Jeunen and Bart Goethals. 2021. Pessimistic Reward Models for Off-Policy Learning in Recommendation. In *RecSys '21*. 63–74.

[35] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* (2014).

[36] Haruka Kiyohara, Kosuke Kawakami, and Yuta Saito. 2021. Accelerating Offline Reinforcement Learning Application in Real-Time Bidding and Recommendation: Potential Use of Simulation. *arXiv preprint arXiv:2109.08331* (2021).

[37] Vijay Konda and John Tsitsiklis. 1999. Actor-critic algorithms. *Advances in neural information processing systems* 12 (1999).

[38] Yehuda Koren. 2008. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In *KDD '08*. 426–434.

[39] Wenqiang Lei, Chongming Gao, and Maarten de Rijke. 2021. RecSys 2021 Tutorial on Conversational Recommendation: Formulation, Methods, and Evaluation. In *Proceedings of the 15th ACM Conference on Recommender Systems* (Amsterdam, Netherlands) *(RecSys '21)*. Association for Computing Machinery, New York, NY, USA, 842–844. https://doi.org/10.1145/3460231.3473325

[40] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *arXiv preprint arXiv:2005.01643* (2020).

[41] Qian Li, Xiangmeng Wang, and Guandong Xu. 2021. Be Causal: De-biasing Social Network Confounding in Recommendation. *arXiv preprint arXiv:2105.07775* (2021).

[42] Dawen Liang, Laurent Charlin, James McInerney, and David M. Blei. 2016. Modeling User Exposure in Recommendation. In *WWW '16*. 951–961.

[43] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[44] 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 *SIGIR '20*. 831–840.

[45] Feng Liu, Huifeng Guo, Xutao Li, Ruiming Tang, Yunming Ye, and Xiuqiang He. 2020. End-to-End Deep Reinforcement Learning Based Recommendation with Supervised Embedding. In *WSDM '20*. 384–392.

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

[47] Ping Liu, Karthik Shivaram, Aron Culotta, Matthew A. Shapiro, and Mustafa Bilgic. 2021. The Interaction between Political Typology and Filter Bubbles in News Recommendation Algorithms. In *WWW '21*. 3791–3801.

[48] Shuchang Liu, Qingpeng Cai, Bowen Sun, Yuhao Wang, Ji Jiang, Dong Zheng, Kun Gai, Peng Jiang, Xiangyu Zhao, and Yongfeng Zhang. 2023. Exploration and Regularization of the Latent Action Space in Recommendation. *arXiv preprint arXiv:2302.03431* (2023).

[49] Yong Liu, Yingtai Xiao, Qiong Wu, Chunyan Miao, Juyong Zhang, Binqiang Zhao, and Haihong Tang. 2020. Diversified Interactive Recommendation with Implicit Feedback. In *AAAI '20*. 4932–4939.

[50] David Lopez-Paz, Robert Nishihara, Soumith Chintala, Bernhard Scholkopf, and Léon Bottou. 2017. Discovering causal signals in images. In *CVPR '17*. 6979–6987.

[51] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Ji Yang, Minmin Chen, Jiaxi Tang, Lichan Hong, and Ed H. Chi. 2020. Off-Policy Learning in Two-Stage Recommender Systems. In *WWW '20*. 463–473.

[52] Prashan Madumal, Tim Miller, Liz Sonenberg, and Frank Vetere. 2020. Explainable reinforcement learning through a causal lens. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 2493–2500.

[53] Farzan Masrour, Tyler Wilson, Heng Yan, Pang-Ning Tan, and Abdol Esfahanian. 2020. Bursting the Filter Bubble: Fairness-aware Network Link Prediction. In *AAAI '20*. 841–848.

[54] James McInerney, Brian Brost, Praveen Chandar, Rishabh Mehrotra, and Benjamin Carterette. 2020. Counterfactual Evaluation of Slate Recommendations with Sequential Reward Interactions. In *KDD '20*. 1779–1788.

[55] Dana McKay, Kaipin Owyong, Stephann Makri, and Marisela Gutierrez Lopez. 2022. Turn and Face the Strange: Investigating Filter Bubble Bursting Information Interactions *(CHIIR '22)*. 233–242.

[56] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[57] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. 2018. Neural Network Dynamics for Model-based Deep Reinforcement Learning with Model-free Fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 7559–7566.

[58] Tien T. Nguyen, Pik-Mai Hui, F. Maxwell Harper, Loren Terveen, and Joseph A. Konstan. 2014. Exploring the Filter Bubble: The Effect of Using Recommender Systems on Content Diversity. In *WWW '14*. 677–686.

[59] Zachary A. Pardos and Weijie Jiang. 2020. Designing for Serendipity in a University Course Recommendation System. In *LAK '20*. 350–359.

[60] Eli Pariser. 2011. *The filter bubble: How the new personalized web is changing what we read and how we think*. Penguin.

[61] Judea Pearl. 2009. *Causality*. Cambridge University Press.

[62] Doina Precup. 2000. Eligibility Traces for Off-policy Policy Evaluation. *Computer Science Department Faculty Publication Series* (2000), 80.

[63] Manoel Horta Ribeiro, Raphael Ottoni, Robert West, Virgílio A. F. Almeida, and Wagner Meira. 2020. Auditing Radicalization Pathways on YouTube. In *FAT* '20*. 131–141.

[64] Yuta Saito, Suguru Yaginuma, Yuta Nishino, Hayato Sakata, and Kazuhide Nakata. 2020. Unbiased Recommender Learning from Missing-Not-At-Random Implicit Feedback. In *WSDM '20*. 501–509.

[65] Masahiro Sato, Sho Takemori, Janmajay Singh, and Tomoko Ohkuma. 2020. Unbiased Learning for the Causal Effect of Recommendation. In *Fourteenth ACM Conference on Recommender Systems*. 378–387.

[66] Tobias Schnabel, Paul N. Bennett, Susan T. Dumais, and Thorsten Joachims. 2018. Short-Term Satisfaction and Long-Term Coverage: Understanding How Users Tolerate Algorithmic Exploration. In *WSDM '18*. 513–521.

[67] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as Treatments: Debiasing Learning and Evaluation. In *ICML '16*. 1670–1679.

[68] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *ICML '15*. 1889–1897.

[69] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional Continuous Control Using Generalized Advantage Estimation. *arXiv preprint arXiv:1506.02438* (2015).

[70] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[71] Jing-Cheng Shi, Yang Yu, Qing Da, Shi-Yong Chen, and An-Xiang Zeng. 2019. Virtual-Taobao: Virtualizing Real-world Online Retail Environment for Reinforcement Learning. In *AAAI '19*. 4902–4909.

[72] Larissa Spinelli and Mark Crovella. 2020. How YouTube Leads Privacy-Seeking Users Away from Reliable Information. In *UMAP '20 Adjunct*. 244–251.

[73] Dusan Stamenkovic, Alexandros Karatzoglou, Ioannis Arapakis, Xin Xin, and Kleomenis Katevas. 2022. Choosing the Best of Both Worlds: Diverse and Novel Recommendations through Multi-Objective Reinforcement Learning. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining* (Virtual Event, AZ, USA) *(WSDM '22)*. Association for Computing Machinery, New York, NY, USA, 957–965. https://doi.org/10.1145/3488560.3498471

[74] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement Learning: An Introduction*. MIT press.

[75] Adith Swaminathan and Thorsten Joachims. 2015. Counterfactual Risk Minimization: Learning from Logged Bandit Feedback. In *ICML '15*. 814–823.

[76] Erich Christian Teppan and Alexander Felfernig. 2012. Minimization of Decoy Effects in Recommender Result Sets. *Web Intelligence and Agent Systems* 10, 4 (2012), 385–395.

[77] Philip S. Thomas and Emma Brunskill. 2016. Data-Efficient off-Policy Policy Evaluation for Reinforcement Learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48* (New York, NY, USA) *(ICML '16)*. JMLR.org, 2139–2148.

[78] Matus Tomlein, Branislav Pecher, Jakub Simko, Ivan Srba, Robert Moro, Elena Stefancova, Michal Kompan, Andrea Hrckova, Juraj Podrouzek, and Maria Bielikova. 2021. An Audit of Misinformation Filter Bubbles on YouTube: Bubble Bursting and Recent Behavior Changes. In *RecSys '21*. 1–11.

[79] Antonela Tommasel, Juan Manuel Rodriguez, and Daniela Godoy. 2021. I Want to Break Free! Recommending Friends from Outside the Echo Chamber. In *RecSys '21*. 23–33.

[80] 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 *NeurIPS '17*.

[81] Shiqi Wang, Chongming Gao, Min Gao, Junliang Yu, Zongwei Wang, and Hongzhi Yin. 2022. Who Are the Best Adopters? User Selection Model for Free Trial Item Promotion. *IEEE Transactions on Big Data* (2022).

[82] Tan Wang, Jianqiang Huang, Hanwang Zhang, and Qianru Sun. 2020. Visual commonsense r-cnn. In *CVPR '20*. 10760–10770.

[83] Wenjie Wang, Fuli Feng, Xiangnan He, Xiang Wang, and Tat-Seng Chua. 2021. Deconfounded Recommendation for Alleviating Bias Amplification. In *KDD '21*. 1717–1725.

[84] Wenjie Wang, Fuli Feng, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. 2021. Clicks Can Be Cheating: Counterfactual Recommendation for Mitigating Clickbait Issue. In *SIGIR '21*. 1288–1297.

[85] Wenlin Wang, Hongteng Xu, Ruiyi Zhang, Wenqi Wang, Piyush Rai, and Lawrence Carin. 2021. Learning to recommend from sparse data via generative user feedback. In *AAAI '21*.

[86] 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.

[87] Zifeng Wang, Xi Chen, Rui Wen, Shao-Lun Huang, Ercan Kuruoglu, and Yefeng Zheng. 2020. Information Theoretic Counterfactual Learning from Missing-not-at-random Feedback. *Advances in Neural Information Processing Systems* 33 (2020), 1854–1864.

[88] Zhenlei Wang, Jingsen Zhang, Hongteng Xu, Xu Chen, Yongfeng Zhang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Counterfactual Data-Augmented Sequential Recommendation. In *SIGIR '21*. 347–356.

[89] Teng Xiao and Donglin Wang. 2021. A General Offline Reinforcement Learning Framework for Interactive Recommendation. In *AAAI '21*.

[90] Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M. Jose. 2020. Self-Supervised Reinforcement Learning for Recommender Systems. In *SIGIR '20*. 931–940.

[91] Shuyuan Xu, Yingqiang Ge, Yunqi Li, Zuohui Fu, Xu Chen, and Yongfeng Zhang. 2021. Causal Collaborative Filtering. *arXiv preprint arXiv:2102.01868* (2021).

[92] Ya Xu, Nanyu Chen, Addrian Fernandez, Omar Sinno, and Anmol Bhasin. 2015. From Infrastructure to Culture: A/B Testing Challenges in Large Scale Social Networks. In *KDD '15*. 2227–2236.

[93] Yuanbo Xu, Yongjian Yang, En Wang, Jiayu Han, Fuzhen Zhuang, Zhiwen Yu, and Hui Xiong. 2020. Neural Serendipity Recommendation: Exploring the Balance between Accuracy and Novelty with Sparse Explicit Feedback. *ACM Trans. Knowl. Discov. Data* 14, 4, Article 50 (June 2020).

[94] Wanqi Xue, Qingpeng Cai, Ruohan Zhan, Dong Zheng, Peng Jiang, and Bo An. 2023. ResAct: Reinforcing Long-term Engagement in Sequential Recommendation with Residual Actor *(ICLR '23)*.

[95] Mengyue Yang, Quanyu Dai, Zhenhua Dong, Xu Chen, Xiuqiang He, and Jun Wang. 2021. Top-N Recommendation with Counterfactual User Preference Simulation. In *CIKM '21*.

[96] Junliang Yu, Min Gao, Hongzhi Yin, Jundong Li, Chongming Gao, and Qinyong Wang. 2019. Generating Reliable Friends via Adversarial Training to Improve Social Recommendation. In *ICDM '19*. IEEE, 768–777.

[97] Ruiyi Zhang, Tong Yu, Yilin Shen, Hongxia Jin, Changyou Chen, and Lawrence Carin. 2019. Reward Constrained Interactive Recommendation with Natural Language Feedback. In *NeurIPS '19*.

[98] Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. 2021. Causal Intervention for Leveraging Popularity Bias in Recommendation. In *SIGIR '21*. 11–20.

[99] Zhong Zhang, Nian Shao, Chongming Gao, Rui Miao, Qinli Yang, and Junming Shao. 2022. Mixhead: Breaking the Low-rank Bottleneck in Multi-head Attention Language Models. *Knowledge-Based Systems* (2022), 108075.

[100] Xiangyu Zhao, Long Xia, Lixin Zou, Hui Liu, Dawei Yin, and Jiliang Tang. 2020. Whole-Chain Recommendations. In *CIKM '20*. 1883–1891.

[101] Xiangyu Zhao, Long Xia, Lixin Zou, Hui Liu, Dawei Yin, and Jiliang Tang. 2021. UserSim: User Simulation via Supervised Generative Adversarial Network. In *WWW '21*. 3582–3589.

[102] Yu Zheng, Chen Gao, Xiang Li, Xiangnan He, Yong Li, and Depeng Jin. 2021. Disentangling User Interest and Conformity for Recommendation with Causal Embedding. In *WWW '21*. 2980–2991.

[103] Sijin Zhou, Xinyi Dai, Haokun Chen, Weinan Zhang, Kan Ren, Ruiming Tang, Xiuqiang He, and Yong Yu. 2020. Interactive Recommender System via Knowledge Graph-Enhanced Reinforcement Learning. In *SIGIR '20*. 179–188.

[104] Ziwei Zhu, Yun He, Xing Zhao, and James Caverlee. 2021. Popularity Bias in Dynamic Recommendation. In *KDD '21*. 2439–2449.

[105] Lixin Zou, Long Xia, Zhuoye Ding, Jiaxing Song, Weidong Liu, and Dawei Yin. 2019. Reinforcement Learning to Optimize Long-Term User Engagement in Recommender Systems. In *KDD '19*. 2810–2818.

[106] Lixin Zou, Long Xia, Pan Du, Zhuo Zhang, Ting Bai, Weidong Liu, Jian-Yun Nie, and Dawei Yin. 2020. Pseudo Dyna-Q: A Reinforcement Learning Framework for Interactive Recommendation. In *WSDM '20*. 816–824.

[107] Lixin Zou, Long Xia, Yulong Gu, Xiangyu Zhao, Weidong Liu, Jimmy Xiangji Huang, and Dawei Yin. 2020. Neural Interactive Collaborative Filtering. In *SIGIR '20*. 749–758.