Causal Collaborative Filtering

Shuyuan Xu
Rutgers University
New Brunswick, NJ, US
shuyuan.xu@rutgers.edu

Yingqiang Ge
Rutgers University
New Brunswick, NJ, US
yingqiang.ge@rutgers.edu

Yunqi Li
Rutgers University
New Brunswick, NJ, US
yunqi.li@rutgers.edu

Zuohui Fu
Rutgers University
New Brunswick, NJ, US
zuohui.fu@rutgers.edu

Xu Chen
Renmin University of China
Beijing, China
xu.chen@ruc.edu.cn

Yongfeng Zhang
Rutgers University
New Brunswick, NJ, US
yongfeng.zhang@rutgers.edu

ABSTRACT

Many of the traditional recommendation algorithms are designed based on the fundamental idea of mining or learning correlative patterns from data to estimate the user-item correlative preference. However, pure correlative learning may lead to Simpson's paradox in predictions, and thus results in sacrificed recommendation performance. Simpson's paradox is a well-known statistical phenomenon, which causes confusions in statistical conclusions and ignoring the paradox may result in inaccurate decisions. Fortunately, causal and counterfactual modeling can help us to think outside of the observational data for user modeling and personalization so as to tackle such issues. In this paper, we propose **C**ausal **C**ollaborative **F**iltering (CCF) — a general framework for modeling causality in collaborative filtering and recommendation. We provide a unified causal view of CF and mathematically show that many of the traditional CF algorithms are actually special cases of CCF under simplified causal graphs. We then propose a conditional intervention approach for *do*-operations so that we can estimate the user-item causal preference based on the observational data. Finally, we further propose a general counterfactual constrained learning framework for estimating the user-item preferences. Experiments are conducted on two types of real-world datasets—traditional and randomized trial data—and results show that our framework can improve the recommendation performance and reduce the Simpson's paradox problem of many CF algorithms.

CCS CONCEPTS

• **Information systems → Recommender systems**; • **Computing methodologies → Machine learning**.

KEYWORDS

Collaborative Filtering; Causal Analysis; Counterfactual Reasoning; Simpson's Paradox; Recommender Systems

ICTIR '23, July 23, 2023, Taipei, Taiwan

ACM ISBN 979-8-4007-0073-6/23/07…$15.00
https://doi.org/10.1145/3578337.3605122

ACM Reference Format:
Shuyuan Xu, Yingqiang Ge, Yunqi Li, Zuohui Fu, Xu Chen, and Yongfeng Zhang. 2023. Causal Collaborative Filtering. In *Proceedings of the 2023 ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '23), July 23, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3578337.3605122

1 INTRODUCTION

Recommender systems are important and valuable tools to provide sophisticated services for many Web-based services such as e-commerce, social networks and online media systems. Collaborative Filtering (CF) [11, 43] algorithms, among others, are fundamental algorithms that support the underlying mechanism of recommender systems.

Most of the existing CF models are developed based on associative user-item preference learning. However, associative learning may be vulnerable to the Simpson's paradox [17, 35] and thus leads to sacrificed recommendation performance. Simpson's paradox refers to the phenomenon that the statistical conclusion from the total observational data disagrees with that from the sub-groups of data [17]. Take Table 1 as a toy example, for two candidate items v_1 and v_2, we have the feedback (e.g., like or dislike) of the users who interacted with v_1 or v_2, assuming that there are 100 users for each item. Suppose the 100 users can be divided into two groups G_1 and G_2, e.g., based on gender, age or income. For each group, we have the percentage of users in the group who like the corresponding item, and we can also calculate the overall percentage of users who like each item. We can see that it is possible that v_2 is more likely to be recommended than v_1 according to the data of each group, but v_1 is more likely to be recommended than v_2 according to the overall data, leading to the Simpson's paradox.

Such Simpson's paradox also exists for real-world data. Following the paradox detection method in [17], we show observations on the MovieLens-100K data. For each user, we rank the user's interacted items according to ratings and each user's top-K items are considered recommended by the user. Figure 1(a) shows the percentage of item pairs that have paradox among all possible item pairs in the observational data, with K ranging from 10 to 200 and users grouped by gender or age (age threshold is 35). We can see that a large percentage of item pairs results in Simpson's paradox.

Additionally, such paradox could be learned into associative CF models which leads to paradoxes in the final recommendation list. Take Matrix Factorization (MF) [37] on MovieLens-100K as an example. Based on the full user-item ranking score matrix completed

	Item v_1	Item v_2
User Group G_1	62.5% (50/80)	**66.6%** (20/30)
User Group G_2	50.0% (10/20)	**54.3%** (38/70)
Overall	**60.0%** (60/100)	58.0% (58/100)

Table 1: A toy example of Simpson's paradox in recommendation, where two candidate items v_1 and v_2 are considered. (x/y) represents that there are x users like the item within y users who have interacted with the item.

by the well-trained MF model, each user recommends his or her top-K ranked items and for each item we randomly sample 100 users. Figure 1(b) shows the percentage of item pairs that have paradox with K ranging from 10 to 200, which indicates the existence of Simpson's paradox in the final recommendation lists. As a result, Simpson's paradox exists in both observational data and the predicted data, which may mislead the recommendation results. Thus mitigating Simpson's paradox will help improve the recommendation performance (we will show that in Section 5).

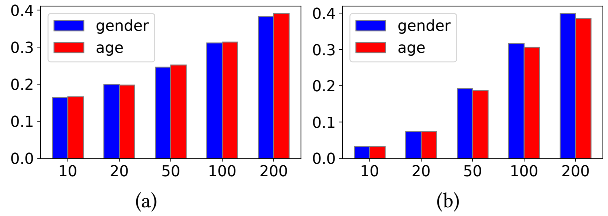
One important approach to mitigating Simpson's paradox is causal inference and *do*-operations [35]. In this paper, we propose a causal collaborative filtering (CCF) model as a simple and principled framework that seamlessly integrates causal inference and recommendation for reduced paradox and better recommendations.

Intuitively, estimating the user-item causal preference can be interpreted as answering a *what if* question: what would be the user's preference on an item if we intervene to recommend the item to the user [55]. Using standard mathematical language of causal inference [35], the above *what if* question can be represented as $P(y|u, do(v))$, where u, v is a user-item pair and y is the preference score to be estimated for the pair, e.g., $y = 1$ for likes and 0 for dislikes. In the CCF framework, *do*-operation is used to represent the causal preference if we intervene to recommend item v instead of passively observing item v in training data. More interestingly, we show that traditional CF models are actually special cases of CCF under simplified causal graphs (Figure 2), and CCF is a general framework for casual learning in recommendation which can be applied over various causal graphs.

Except for the above conceptual contribution, this work also provides technical contributions. More specifically, a great challenge is how to estimate $P(y|u, do(v))$. In this work, we propose a conditional intervention approach to estimating $P(y|u, do(v))$ based on observational data. Specifically, we adopt the causal graph in Figure 2(d) for conditional intervention, which considers the user interaction history X for mediator analysis. Moreover, solving the conditional intervention requires counterfactual reasoning, and we propose a counterfactual constrained learning framework for counterfactual reasoning in both discrete and continuous space to estimate the causal preference $P(y|u, do(v))$. We conduct extensive experiments on real-world datasets. Experimental results show that CCF reduces Simpson's paradox and significantly improves the recommendation performance.

2 RELATED WORK

Existing literature usually categorizes the recommendation algorithms into three major types: collaborative filtering, content-based recommendation and hybrid method [1, 18, 70]. Due to the wide scope of literature of recommender systems (RS), it is hardly possible to cover all of the RS algorithms, so we review some representative methods based on collaborative filtering in this section, and a more comprehensive review can be seen in [1, 11, 18, 70].



(a) (b)

Figure 1: (a) The percentage of item pairs that have paradox in the observational data of MovieLens-100k. (b) The percentage of item pairs that have paradox on the full user-item ranking score matrix completed by MF on MovieLens-100K.

Collaborative Filtering (CF) [11] is based on a key idea that similar users may share similar interests and similar items may be liked by similar users. Early memory-based CF models—such as user-based CF [23, 38] and item-based CF [27, 40]—calculate the similarity between users or items for recommendation based on pre-defined similarity functions such as cosine similarity. To extract latent semantic meanings from the matrix, researchers later explored learned user and item vector representations to calculate the matching score of each user-item pair for recommendation, including Latent Factor Models (LFM) such as matrix factorization [24], tensor factorization [22] and factorization machines [36]. The development of deep learning and neural networks has further extended CF. The relevant methods can be broadly classified into two categories: similarity learning approach and representation learning approach. The similarity learning approach adopts simple user and item representations (such as one-hot) and learns a complex matching function (such as a prediction network) to calculate user-item matching scores [10, 15, 63], while the representation learning approach learns rich user and item representations and adopts a simple matching function (e.g., inner product) for efficient matching score calculation [2, 29, 68, 71, 73]. Another important direction is learning to rank for recommendation, which learns the relative ordering of items instead of the absolute scores, such as Bayesian Personalized Ranking (BPR) [37].

Most existing methods learn correlative patterns from data for matching and recommendation based on either simple or complex matching functions. However, advancing from correlative learning to causal learning is an important problem [35]. The community has explored causal modeling on several different perspectives. For example, researchers adopted causal models to generate explanations for recommendation [12, 48, 50], considered fairness under counterfactual settings [26, 30], corrected data bias for rankings in search [3, 16, 21, 32, 33, 54], recommendation [7, 28, 42, 44, 53, 55, 57, 58, 72, 74], advertising [67] and evaluating the ranking models [64], estimated the uplift effect of recommendations [41, 42, 59], explored data augmentation [56, 65, 69] as well as multimodal information such as text [60] based on causal methods. Some related works are proposed to estimate $P(y|u, do(v))$ for recommendation as well, for example, Zhang et al. [72] leverage popularity bias for recommendation, Xu et al. [62] design a causal model for mitigating echo chambers while maintaining comparable performance, etc. Unlike the existing causal recommendation works, our work focuses on mitigating Simpson's paradox.

Simpson's paradox is a common statistical phenomenon and it appears in many artificial intelligence applications and real-life scenarios [31, 51]. This phenomenon was originally observed in 1951 [47] and was later named as the "Simpson's Paradox"[6]. Simpson's paradox has attracted the attention of many computer scientists in recent years. In general machine learning, existing literature mainly focuses on detecting Simpson' paradox automatically [4, 5, 45, 45, 46, 61]. In recommender systems, Jadidinejad et al. [17] propose a method to address the Simpson's paradox in offline evaluation. To the best of our knowledge, none of the existing works aims at proposing models to mitigate Simpson's paradox in model prediction for improved recommendation performance.

3 A UNIFIED CAUSAL VIEW OF CF

We provide a unified causal view of collaborative filtering (CF) in this section. Specifically, we show that the fundamental goal of many CF algorithms is to estimate the causal effect $P(y|u, do(v))$. The key difference between various CF models is that they assume different causal graphs to calculate $P(y|u, do(v))$. When the causal graph is too simple or even unrealistic, the causal effect will naturally degenerate to association relations that are considered in traditional CF models. We now show how different CF models fit into the unified causal view under $P(y|u, do(v))$.

3.1 Non-Personalized Model

Non-personalized recommendation models, such as most popular recommendation [19], assume a simple causal graph without the user node, as shown in Figure 2(a). Since user is excluded from consideration and since item is a root node in the graph, we have $P(y|u, do(v)) = P(y|do(v)) = P(y|v)$, and $P(y|v)$ naturally represents the popularity of item v in the data.

3.2 Associative Matching Models

Most CF algorithms fall into the user-item associative matching category. These models assume a causal graph shown in Figure 2(b), where user node U and item node V constitute a collider to influence preference node Y. Basically, these models assume that the appearance of users and items are independent from each other in observational data (though this may be an unrealistic assumption), and since both U and V are root nodes, we have $P(y|u, do(v)) = P(y|u, v)$, which can thus be estimated from observational data.

The main difference of various models is how to design the matching function to estimate $P(y|u, v)$, e.g., user-based CF assumes $P(Y = 1|u, v) \propto \frac{1}{|N(u)|} \sum_{u' \in N(u)} y_{u',v}$, where $N(u)$ are the neighbours of user u. Matrix factorization (MF) models, such as [24], assume $P(Y = 1|u, v) \propto \mathbf{u}^\top \mathbf{v}$ or $\propto \mathbf{u}^\top \mathbf{v} + b_u + b_v + b$. Some neural network-based models such as [10, 13, 63] assume $P(Y = 1|u, v) \propto \mathrm{NN}(\mathbf{u}, \mathbf{v})$, where NN is a neural network for similarity matching. More complex deep representation learning models such as sequential models [9, 14, 25, 49] and graph-based models [2, 52, 66, 68] can be represented as $P(Y = 1|u, v) \propto \mathrm{NN}(\mathbf{NN}(u), \mathbf{NN}(v))$, where a neural similarity network NN is applied on top of the neural representation learning network **NN**.

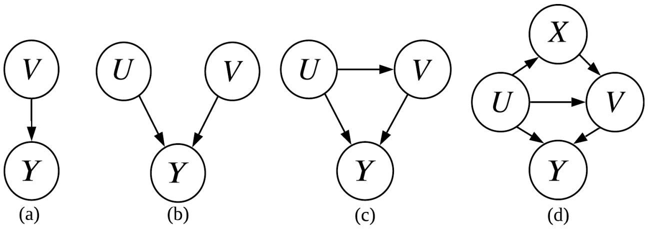


Figure 2: Many traditional CF models are special cases of CCF under simplified causal graphs. In the graphs, U is user, V is item, X is user interaction history, Y is preference score. (a) Causal graph for non-personalized models. (b) Causal graph for similarity matching-based CF models. (c) Causal graph that considers the causality from user to item [7]. (d) Causal graph used in our framework to demonstrate the idea of CCF, using user interaction history X as a mediator.

3.3 Causal Reasoning Model

Some causal models [7, 20, 44] are aware of the dependencies between user and item, thus assume the causal graph in Figure 2(c), which extends Figure 2(b) by removing the independence assumption between user and item. In this case, the u-specific causal effect $P(y|u, do(v))$ by definition requires interventional reasoning. Depending on if or not we have complete control of the recommendation platform, we have the following two approaches to estimate $P(y|u, do(v))$.

3.3.1 ***Direct Intervention Models.*** If we have complete control of the recommendation platform or have access to a randomized treatment dataset where user is randomly exposed to items, then the straightforward way of estimating $P(y|u, do(v))$ is through direct intervention [7, 55, 72]. We refine Figure 2(c) as Figure 3(a) to show the structural equations $V = g(U)$ and $Y = f(U, V)$, which represent the two steps of the recommendation pipeline. $V = g(U)$ represents the de facto recommendation model in the system that decides what items are exposed to the user, and $Y = f(U, V)$ represents the user's preference on the exposed item. To estimate $P(y|u, do(v))$, we resort to the most original definition of intervention to get the manipulated causal graph as shown in Figure 3(b) [35, p.54]. We thus have $P(y|u, do(v)) = P_m(y|u, v)$, where P_m is the probability distribution according to the manipulated causal graph. To estimate $P_m(y|u, v)$, we can apply a randomized exposure policy by either showing random items to users or manipulating the observational data to simulate a random policy and thus to implement the independence between U and V. This treatment will help us to collect an unbiased dataset to estimate $P_m(y|u, v)$. More details can be seen in [7, 55, 72].

3.3.2 ***Inverse Propensity Scoring (IPS) Models.*** In many cases, we do not have complete control of the recommendation platform or access to the randomized treatment data. The basic idea of inverse propensity scoring (IPS) methods is to turn the outcomes of an observational study into pseudo-randomized trials by re-weighting the samples [7], so that $P(y|u, do(v))$ can be estimated from the observational data [20, 44]. More formally, according to the recommendation pipeline shown in Figure 3(a), the observed user preference r_{uv} is considered as $r_{uv} \propto P(y|u, do(v))P(v|u)$, which is the multiplication between the user's real preference $P(y|u, do(v))$ and the probability that user u had a chance to see the item $P(v|u)$.

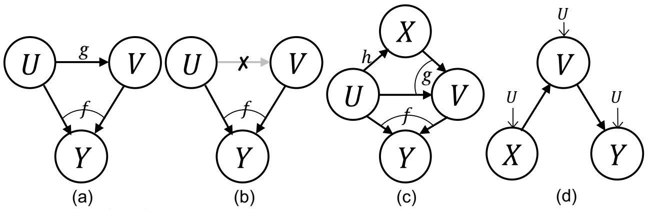


Figure 3: (a-b) Causal graphs before and after manipulation. (c-d) Reorganize causal graph using U as exogenous variable.

As a result, we have $P(y|u, do(v)) \propto \frac{r_{uv}}{P(v|u)}$, which means that each example in the observational data boosts its probability by a factor equal to $1/P(v|u)$, which corrects the observational data by removing the exposure bias.

4 THE PROPOSED FRAMEWORK

In this section, we will start from the causal graph and then introduce the techniques for estimating $P(y|u, do(v))$, including conditional intervention, counterfactual reasoning, and finally a flexible counterfactual constrained learning framework that can be applied on any existing CF model for recommendation.

4.1 The Causal Graph

As mentioned above, in many cases of recommender systems we want to answer counterfactual questions such as *what if* an item had (or had not) been recommended, or *what if* the user had a different interaction history. Such imaginary cases constitute the *counterfactual world*, in contrast to what happened in the *real world*.

To enable counterfactual reasoning, we extend the causal graph from Figure 3(a) to Figure 3(c) to consider user's interaction histories X for mediator analysis. Specifically, the casual model includes three structural equations: (1) $X = h(U)$, which returns a user's history X. In the most simple case, it can be a database retrieval operation that returns a user's interaction history; (2) $V = g(U, X)$, which is the already deployed recommendation algorithm of the system that returns the recommended item V based on the user and the user's interaction history; (3) $Y = f(U, V)$, which is the user preference function that we do not know but we want to estimate.

We should acknowledge that the causal graph in Figure 3(c) is not a once-and-for-all solution for recommender systems, because practical systems are very complicated that involve many other factors. However, we consider this causal graph in the work because the structural equation $V = g(U, X)$ is general enough to include a wide scope of recommendation algorithms, including both sequential and non-sequential methods. With the help of the causal graph, our framework aims to estimate $P(y|u, do(v))$ for reduced paradox and enhanced performance, which we will show in the following.

4.2 Conditional Intervention

To estimate $P(y|u, do(v))$, we first identify that $\{U, X\}$ is a set of variables that satisfy the backdoor criterion [35, p.61] for the casual effect $V \longrightarrow Y$. Since we already conditioned on U for personalization, the only variable that leads to variations in V is user interaction history X, as a result, we adopt conditional intervention [35, p.70][34, p.113] to estimate $P(y|u, do(v))$.

More specifically, the recommendation policy $V = g(U, X)$ provides recommendation V based on the user U and history X, written as $do(V = g(U, X))$. To find out the distribution of the outcome Y that results from this policy, we seek to estimate $P(Y = y|U = u, do(V = g(U, X)))$. We will show that identifying the effect of such policies is equivalent to identifying the expression for the (u, x)-specific effect $P(Y = y|U = u, X = x, do(V = v))$ [35, p.71].

$$
\begin{aligned}
&P(y|u, do(v)) \doteq P(Y = y|U = u, do(V = g(U, X)))\\
&\overset{1}{=} \sum_{x} P(Y = y|U = u, do(V = g(U, X)), X = x) \times\\
&\qquad P(X = x|U = u, do(V = g(U, X)))\\
&\overset{2}{=} \sum_{x} P(Y = y|U = u, X = x, do(V = g(u, x)))P(X = x|U = u)\\
&\overset{3}{=} \sum_{x} P(Y = y|U = u, X = x, do(V = v))|_{v=g(u,x)}P(X = x|U = u)\\
&\overset{4}{=} \sum_{x} P(y|u, x, v)|_{v=g(u,x)}P(x|u) = E_{x|u}[P(y|u, x, v)|_{v=g(u,x)}]
\end{aligned}
\tag{1}
$$

From the last step in Eq.(1) we can see that the key difference between the causal model $P(y|u, do(v))$ and traditional associative models $P(y|u, v)$ is the existence of the conditional probability term $P(x|u)$ in the final step. In step 4, $P(y|u, x, v)|_{v=g(u,x)}$ stands for the preference estimation of the deployed recommendation model $V = g(U, X)$. Traditional models only consider the real world but not the counterfactual world, as a result, the conditional probability $P(x|u) = 1$ for observed user history x, while for unobserved history x', $P(x'|u) = 0$. In this case, we see that the summation in step 4 will only include observed history x and thus $P(y|u, do(v))$ naturally degenerates to the original recommendation model $V = g(U, X)$.

However, the observed history x does not mean that the user is destined to interact with the items in x—the user just happened to interact with x, i.e., if the user had a chance to be recommended with different items x' in the counterfactual world, the user may also interact with those items, and thus the probability $P(x'|u)$ is not 0. As a result, the calculation of Eq.(1) requires counterfactual reasoning where the user history had been $X = x'$, which is beyond the observational data $X = x$.

4.3 Counterfactual Reasoning

Counterfactual reasoning enables more refined intervention at individual level [35, p.78,93]. In this work, the *individual level* refers to each user $U = u$ for personalization purpose. To better understand this, the causal graph in Figure 3(c) is equivalently transformed into Figure 3(d), where U serves as the exogenous variable. As a result, counterfactual reasoning is individualized on each user.

To enable counterfactual reasoning to calculate Eq.(1), let's consider a record (u, x, v, y) in the observational data, meaning that user u's real history is x, and then the system logged user's preference on item v which is y, e.g., we can consider binary preference values using $y = 1$ for likes and $y = 0$ for dislikes, but the framework can also be applied over multiple preference values. According to Eq.(1), the user preference estimation $y = f(u, v)$ is expressed as

$$
\begin{aligned}
&y = f(u, v) \propto P(y|u, do(v))\\
&\quad = \sum_{\tilde{x}} P(y|u, \tilde{x}, v)|_{v=g(u,\tilde{x})}P(\tilde{x}|u) = E_{\tilde{x}|u}[P(y|u, \tilde{x}, v)|_{v=g(u,\tilde{x})}]
\end{aligned}
\tag{2}
$$

To distinguish from the single real-world history x, we use $\tilde{x}$ to represent any possible user history, including both the real history x and possible counterfactual histories x'. Eq.(2) means that the

Table 2: Different heuristic rules to create counterfactual examples, the corresponding counterfactual question, and some intuitive toy examples. In the toy examples, the user's real interaction history x includes items $a\ b\ c$, and items at the right side of the arrow is the counterfactual history x'. Multiple counterfactual histories can be constructed from the real history x.

Heuristic Rule	Counterfactual Question	Toy Example
Keep One (K1)	What if the user only interacted with one history item?	$a\ b\ c \rightarrow a;\ a\ b\ c \rightarrow b;\ a\ b\ c \rightarrow c$
Delete One (D1)	What if the user did not interact with one of the history items?	$a\ b\ c \rightarrow b\ c;\ a\ b\ c \rightarrow a\ c;\ a\ b\ c \rightarrow a\ b$
Replace One (R1)	What if one of the history items were different?	$a\ b\ c \rightarrow a'b\ c;\ a\ b\ c \rightarrow a\ b'c;\ a\ b\ c \rightarrow a\ b\ c'$

estimation of $P(y|u,do(v))$ can be achieved by correcting the original recommendation algorithm's estimation $P(y|u,x,v)|_{v=g(u,x)}$ using counterfactual histories x'. More specifically, the estimation for $P(y|u,do(v))$ is the *expected* estimation of $P(y|u,x,v)|_{v=g(u,x)}$, where the expectation is taken over all possible histories (including real and counterfactual histories) when item v is recommended.

4.3.1 *Generate Counterfactual Examples*.

Counterfactual reasoning requires generating counterfactual examples based on minimal changes [35, p.92]. We start with a heuristic-based approach for counterfactual example generation and we will generalize to a learning-based approach in the next section.

We adopt three heuristic rules to generate counterfactual histories x' by applying modifications to the real history x (Table 2). The Keep One (K1) rule only keeps one item of the user's real history, the Delete One (D1) rule removes one item from the user's real history, and the Replace One (R1) rule replaces one item of the user's real history with another item. For the R1 rule, depending on how the item is replaced, we have two variants: R1-random (R1r)—the item is replaced with a random item, and R1-nearest (R1n)—the item is replaced with its nearest neighbour based on embedding similarity. We will introduce more details in the experiments.

4.3.2 *Select Counterfactual Examples*.

Consider the training example (u,x,v,y) where the user's real history is x, and we are able to generate m counterfactual histories $\{x'_1, x'_2 \cdots x'_m\}$ using one of the heuristic rules. Conditional intervention (Section 4.2) requires $v = g(u,\tilde{x})$, i.e., the same item v should be recommended (i.e., within the top-k recommendation list) by the recommendation algorithm $g(\cdot,\cdot)$ under counterfactual histories (since we are considering $do(v)$ instead of just v in the condition). However, not all of the counterfactual histories $\{x'_1, x'_2 \cdots x'_m\}$ guarantee that item v is recommended under the algorithm. As a result, we execute the recommendation algorithm $g(\cdot,\cdot)$ over each counterfactual history x'_i and obtain the top-k recommendation list $\mathcal{V}'_i = g(u,x'_i)$, where k is a hyper-parameter to be tuned (will be introduced in the experiments). If the target item $v \in \mathcal{V}'_i$, then we keep the counterfactual example (u,x'_i,v,y). Suppose n of the m counterfactual histories are eventually selected, we will have a set of counterfactual examples $\{(u,x'_i,v,y)\}_{i=1}^n$.

4.3.3 *Calculate the Expectation*.

We then calculate $P(y|u,do(v))$ based on the real observation (u,x,v,y) and the counterfactual examples $\{(u,x'_i,v,y)\}_{i=1}^n$ according to Eq.(2). For simplicity, we consider $P(\tilde{x}|u)$ as a piecewise uniform distribution over the real and counterfactual histories, i.e.,

$$P(\tilde{x}|u) = \begin{cases} \alpha, \text{ when } \tilde{x} = x \\ \beta, \text{ when } \tilde{x} = x'_i,\ i \in \{1, 2 \cdots n\} \end{cases}, \alpha + n\beta = 1 \tag{3}$$

where α is the probability of the real example x, and β is the probability of each counterfactual example x'_i. Since x is already observed, we apply a higher probability to x than x'_i, i.e., $\alpha > \beta > 0$. Generalizing to even more complex distributions such as Gaussian or Gamma distribution will be considered in the future. Then we have:

$$\begin{aligned} P(y|u,do(v)) &= \sum_{\tilde{x}} P(y|u,\tilde{x},v)|_{v=g(u,\tilde{x})} P(\tilde{x}|u) \\ &= \alpha\ P_g(y|u,x,v) + \beta \sum_{i=1}^{n} P_g(y|u,x'_i,v) \end{aligned} \tag{4}$$

where P_g is the probability estimation of the base recommendation algorithm $v = g(u,x)$.

4.4 Counterfactual Constrained Learning

In practical recommender systems, the ranking probability score $P_g(y|u,x,v)$ is usually learned by optimizing a loss function $L(g)$ such as the rating prediction loss [24] or the pair-wise ranking loss [37]. As noted before, however, the estimated probability $P_g(y|u,x,v)$ could be unreliable due to unrealistic model assumptions or data bias. As a result, what we really want is the probability score of $P(y|u,do(v))$ for item ranking. To learn the values of $P(y|u,do(v))$, we propose a counterfactual constrained learning approach, which requires the base recommender's probability estimation $P_g(y|u,x,v)$ to be equal to $P(y|u,do(v))$, and thus we can safely use the learned $P_g(y|u,x,v)$ scores for item ranking and recommendation:

$$\begin{aligned} &\text{minimize } L(g) \\ &\text{s.t. } P_g(y|u,x,v) = P(y|u,do(v)) \quad \forall u \in \mathcal{U},\ \forall v \in \mathcal{V} \end{aligned} \tag{5}$$

where $L(g)$ is the loss function of a base recommendation algorithm $g(u,x)$, $\mathcal{U}$ is the set of users, and $\mathcal{V}$ is the set of items.

Actually, the constraint $P_g(y|u,x,v) = P(y|u,do(v))$ is naturally supported by the causal graph (Figure 3(c)). The reason is that in the graph, both U alone and $\{U,X\}$ as a set satisfy the backdoor criterion for $V \rightarrow Y$, as a result, we have $P_g(y|u,x,v) = P(y|u,do(v))$. Careful readers may ask if $P_g(y|u,x,v) = P(y|u,do(v))$, then why can't we just directly use the estimation $P_g(y|u,x,v)$ of the base recommender for recommendation? The reason is that we only have the accurate $P_g(y|u,x,v)$ scores for the observed (u,v) pairs in the dataset, which are the already observed user preference (ratings, clicks, etc.) on the item. These pairs do not need any estimation and according to the causal graph they can be used as $P(y|u,do(v))$ because the system already exposed the items to the user and collected the user's preference. However, recommender system needs the $P_g(y|u,x,v)$ scores for the unobserved (u,v) pairs to make recommendations, and these scores need to be estimated using a model. As discussed before, most traditional CF models assume simplified causal graphs to estimate $P_g(y|u,x,v)$ based on associative learning, which may lead to unreliable or even biased estimations. As a result, we need to explicitly add the constraint to the learning procedure to make sure $P_g(y|u, x, v) = P(y|u, do(v))$ is guaranteed for both observed and unobserved (u, v) pairs. In the following, we further derive Eq.(5) to make it learnable.

4.4.1 ***Counterfactual Learning in Discrete Space.*** We first propose a discrete version of the counterfactual constrained learning algorithm for any base recommender, which conducts counterfactual reasoning in a discrete item space. We already know the expression for $P(y|u, do(v))$ (i.e., Eq.(4)). Using $P(y|u, x, v)$ for $P_g(y|u, x, v)$ for notation simplicity, the constraint can be written as:

$$
\begin{aligned}
&P(y|u{,}x, v) = P(y|u, do(v)) = \alpha\ P(y|u, x, v) + \beta \sum_{i=1}^{n} P(y|u, x'_i, v)\\
&\Leftrightarrow (1-\alpha)\ P(y|u, x, v) = \beta \sum_{i=1}^{n} P(y|u, x'_i, v)\\
&\Leftrightarrow \sum_{i=1}^{n} P(y|u, x'_i, v) = \frac{1-\alpha}{\beta} \cdot P(y|u, x, v) = n \cdot P(y|u, x, v)\\
&\Leftrightarrow \sum_{i=1}^{n} P(y|u, x'_i, v) - n \cdot P(y|u, x, v) = 0
\end{aligned}
\tag{6}
$$

To make the constraint optimizable, we relax the equality constraint to an inequality constraint, i.e.,

$$
\begin{aligned}
&\text{minimize } L(g)\\
&\text{s.t.} \Big| \sum_{x' \in C(u,v)} P(y|u, x', v) - |C(u, v)| \cdot P(y|u, x, v) \Big| \le \epsilon\\
&\forall u \in \mathcal{U},\ \forall v \in \mathcal{I}(u) \cup \mathcal{S}(u)
\end{aligned}
\tag{7}
$$

where $C(u, v)$ is the set of counterfactual histories of user u under the target item v (section 4.3.1 and 4.3.2), $|C(u, v)|$ represents the size of the set (i.e. n in Eq.(6)), $\mathcal{I}(u)$ is the set of interacted items of user u, and ϵ is a parameter controlling how rigorous is the constraint. In Eq.(7), since the item space is very huge, it is impractical to apply the constraint on all items in practice. As a result, we sample a set of items for each user, i.e., $\mathcal{S}(u)$, where $|\mathcal{S}(u)| = |\mathcal{I}(u)|$. For easy implementation, we apply absolute value inequality to constrain the upper bound of the above inequality:

$$
\begin{aligned}
&\Big| \sum_{x' \in C(u,v)} P(y|u, x', v) - |C(u, v)| \cdot P(y|u, x, v) \Big|\\
&\le \sum_{x' \in C(u,v)} \Big| P(y|u, x', v) - P(y|u, x, v) \Big| \le \epsilon
\end{aligned}
\tag{8}
$$

Therefore, we define the final counterfactual learning in discrete space as following:

$$
\begin{aligned}
&\text{minimize } L(g)\\
&\text{s.t.} \sum_{x' \in C(u,v)} \Big| P(y|u, x', v) - P(y|u, x, v) \Big| \le \epsilon\\
&\forall u \in \mathcal{U},\ \forall v \in \mathcal{I}(u) \cup \mathcal{S}(u)
\end{aligned}
\tag{9}
$$

According to Eq.(8), we can see that satisfying the constraint in Eq.(9) naturally leads to satisfying the constraint in Eq.(7).

4.4.2 ***Counterfactual Learning in Continuous Space.*** Many recommendation models represent users, items and histories as embedding vectors in a latent space. If a user's history x is represented as an embedding vector $\mathbf{x}$, then we can directly create latent counterfactual histories $\mathbf{x}'$ by slightly perturbing vector $\mathbf{x}$ in the latent space. Similarly, let $L(g)$ be the loss function of a base recommendation algorithm $g(u, x)$, CCF in continuous space aims to learn $L(g)$ under a continuous counterfactual constraint:

$$
\begin{aligned}
&\text{minimize } L(g)\\
&\text{s.t.} \int_{\mathbf{x}'} \Big| P(y|u, x', v) - P(y|u, x, v) \Big| \le \epsilon_1,\ \|\mathbf{x}' - \mathbf{x}\|_2 \le \epsilon_2\\
&\forall u \in \mathcal{U},\ \forall v \in \mathcal{I}(u) \cup \mathcal{S}(u)
\end{aligned}
\tag{10}
$$

where $\mathbf{x}$ is the embedding of user u's real history x, $\mathbf{x}'$ is a latent vector selected from the small ϵ_2-neighbourhood of vector $\mathbf{x}$, and the integration can be calculated based on Monte Carlo sampling. All other parameters have the same meaning as Eq.(9).

4.5 Model Learning and Optimization

To solve the above constrained optimization problem, we formulate the problem as a tractable optimization problem by the Lagrange Multiplier Method. For the discrete space version, we convert the objective in Eq.(9) to the following Lagrange optimization form:

$$
\begin{aligned}
&\text{minimize } L(g) + \omega L_c\\
&L_c = \sum_{u \in \mathcal{U}} \sum_{v \in \mathcal{I}(u) \cup \mathcal{S}(u)} \max\Big\{0, \sum_{x' \in C(u,v)} \Big| P(y|u, x', v) - P(y|u, x, v) \Big| - \epsilon \Big\}
\end{aligned}
\tag{11}
$$

where ω is a parameter controlling the weight of the constraint.

While for continuous space, we process the constraint similarly. The difference is that the parameter ϵ_2 in Eq.(10) is used to restrict the distance between counterfactual histories and the real history.

$$
\begin{aligned}
&\text{minimize } L(g) + \omega L_c\\
&\text{s.t. } \|\mathbf{x}' - \mathbf{x}\|_2 \le \epsilon_2\\
&L_c = \sum_{u \in \mathcal{U}} \sum_{v \in \mathcal{I}(u) \cup \mathcal{S}(u)} \max\Big\{0, \int_{\mathbf{x}'} \Big| P(y|u, x', v) - P(y|u, x, v) \Big| - \epsilon_1 \Big\}
\end{aligned}
\tag{12}
$$

We still apply Monte Carlo sampling for integration calculation. The counterfactual constrained learning framework is flexible and can be applied on many base recommender algorithms g, which we will show in the experiments.

5 EXPERIMENTS

We conduct experiments to explore CCF from different perspectives. In particular, we aim to answer the following research questions: **RQ1**: What is the overall performance of the CCF framework, can CCF improve the recommendation performance? **RQ2**: Can CCF reduce Simpson's paradox? **RQ3**: How different heuristic rules influence the performance? **RQ4**: Is it necessary to select the counterfactual examples after they are generated? **RQ5**: What is the impact of the counterfactual constraint in the learning objective? We will first describe the datasets, baselines and then provide our answers to the above questions.

5.1 Data Description

Our experiments are conducted on two types of datasets. The first type is frequently used benchmark dataset **MovieLen-100k**[1]. For the second type, to better show that our framework can help to

[1] https://grouplens.org/datasets/movielens/

Table 3: Performance of all three recommendation models on MovieLens-100k and Coat Shopping. The relative improvement is calculated against the original performance. For recommendation results, the higher the better. For Simpson's paradox results, the lower the better. Positive improvements are bold and the highest is underlined.

Models		ML100k								Coat Shopping							
		Recommendation				Simpson's Paradox				Recommendation				Simpson's Paradox			
		nDCG@10		Hit@1		Gender		Age		nDCG@10		Hit@1		Gender		Age	
		value	imp	value	imp	value	imp	value	imp	value	imp	value	imp	value	imp	value	imp
MF	Original	0.3647	-	0.1490	-	0.1918	-	0.1864	-	0.2042	-	0.1561	-	0.4415	-	0.3236	-
	CausE	0.3784	**3.8%**	0.1651	**10.8%**	0.1205	**37.2%**	0.1123	**39.8%**	0.2100	**2.8%**	0.1899	**21.7%**	0.4462	-1.1%	0.3124	**3.5%**
	IPS	0.3696	**1.3%**	0.1618	**8.6%**	0.1966	-2.5%	0.1954	-4.8%	0.2261	**10.7%**	0.2068	**32.5%**	0.4392	**0.5%**	0.3217	**0.6%**
	DCF	0.3686	**1.1%**	0.1543	**3.6%**	0.2190	-14.2%	0.2091	-12.2%	0.2209	**8.2%**	0.1983	**27.0%**	0.4615	-4.5%	0.3622	-11.9%
	DICE	0.3692	**1.2%**	0.1543	**3.6%**	0.2344	-22.2%	0.2273	-21.9%	0.2303	**12.8%**	0.2068	**32.4%**	0.5610	-27.1%	0.4433	-37.0%
	MACR	0.3671	**0.7%**	0.1586	**6.4%**	0.1516	**21.0%**	0.1235	**33.7%**	0.2197	**7.6%**	0.2236	**43.2%**	0.3941	**10.7%**	0.2936	**9.3%**
	CCF$_{\text{K1}}$	0.3661	**0.4%**	0.1554	**4.3%**	0.1834	**4.4%**	0.1822	**2.3%**	0.2324	**13.8%**	0.2363	**51.4%**	0.3547	**19.7%**	0.2621	**19.0%**
	CCF$_{\text{D1}}$	0.3781	**3.7%**	0.1683	**13.0%**	0.1125	**41.3%**	0.1087	**41.7%**	0.2061	**0.9%**	0.2152	**37.9%**	0.4217	**4.5%**	0.3078	**4.9%**
	CCF$_{\text{R1r}}$	0.3673	**0.7%**	0.1554	**4.3%**	0.1342	**30.0%**	0.1195	**35.9%**	0.2165	**6.0%**	0.2194	**40.6%**	0.3642	**17.5%**	0.2810	**13.2%**
	CCF$_{\text{R1n}}$	0.3734	**2.4%**	0.1533	**2.9%**	0.1217	**36.5%**	0.1203	**35.5%**	0.2089	**2.3%**	0.2110	**35.2%**	0.4012	**9.1%**	0.2991	**7.6%**
	CCF$_{\text{C}}$	0.3729	**2.2%**	0.1597	**7.2%**	0.1142	**40.5%**	0.1062	**43.0%**	0.2150	**5.3%**	0.2068	**32.5%**	0.4206	**4.7%**	0.3109	**3.9%**
GRU4Rec	Original	0.4087	-	0.1865	-	0.1460	-	0.1397	-	0.1147	-	0.0759	-	0.1252	-	0.1260	-
	CausE	0.4111	**0.6%**	0.1908	**2.3%**	0.1366	**6.4%**	0.1313	**6.0%**	0.1157	**0.9%**	0.0802	**5.7%**	0.1170	**6.5%**	0.1006	**20.2%**
	IPS	0.4136	**1.2%**	0.1876	**0.6%**	0.1480	-1.4%	0.1292	**7.5%**	0.1160	**1.1%**	0.0802	**5.7%**	0.1239	**1.0%**	0.1204	**4.4%**
	DCF	0.4158	**1.7%**	0.1951	**4.6%**	0.1392	**4.7%**	0.1330	**4.8%**	0.1174	**2.4%**	0.0717	-5.5%	0.1317	-5.2%	0.1282	-1.7%
	DICE	0.4158	**1.7%**	0.1929	**3.4%**	0.1325	**9.2%**	0.1128	**19.3%**	0.1273	**11.0%**	0.0886	**16.7%**	0.1362	-8.8%	0.1101	**12.6%**
	MACR	0.4211	**3.0%**	0.1875	**0.5%**	0.1304	**10.7%**	0.1167	**16.5%**	0.1289	**12.8%**	0.0928	**22.3%**	0.1437	-14.8%	0.1521	-20.7%
	CCF$_{\text{K1}}$	0.4225	**3.4%**	0.2015	**8.0%**	0.1152	**21.1%**	0.1088	**22.1%**	0.1170	**2.0%**	0.0675	-11.1%	0.1241	**0.9%**	0.1225	**2.8%**
	CCF$_{\text{D1}}$	0.4281	**4.7%**	0.1972	**5.7%**	0.1235	**15.4%**	0.1145	**18.0%**	0.1226	**6.9%**	0.0886	**16.7%**	0.1185	**5.4%**	0.1013	**19.6%**
	CCF$_{\text{R1r}}$	0.4241	**3.8%**	0.1972	**5.7%**	0.1257	**13.9%**	0.1174	**16.0%**	0.1299	**13.3%**	0.0802	**5.7%**	0.1139	**9.0%**	0.1093	**13.3%**
	CCF$_{\text{R1n}}$	0.4235	**3.6%**	0.2015	**8.0%**	0.1143	**21.7%**	0.1113	**20.3%**	0.1264	**10.2%**	0.0802	**5.7%**	0.1082	**13.6%**	0.1154	**8.4%**
	CCF$_{\text{C}}$	0.4238	**3.7%**	0.2015	**8.0%**	0.1245	**14.7%**	0.1095	**21.6%**	0.1352	**17.9%**	0.0970	**27.8%**	0.1047	**16.4%**	0.0895	**29.0%**
NCR	Original	0.4227	-	0.1972	-	0.1054	-	0.0828	-	0.2608	-	0.0506	-	0.1320	-	0.1128	-
	CausE	0.4234	**0.2%**	0.2090	**6.0%**	0.0829	**21.3%**	0.0798	**3.6%**	0.2813	**7.9%**	0.0886	**75.1%**	0.1236	**6.4%**	0.1057	**6.3%**
	IPS	0.4237	**0.2%**	0.2036	**3.2%**	0.0874	**17.1%**	0.0826	**0.2%**	0.2916	**11.8%**	0.1350	**166.8%**	0.1285	**2.7%**	0.1094	**3.0%**
	DCF	0.4201	-0.6%	0.1940	-1.6%	0.0859	**18.5%**	0.0789	**4.7%**	0.2689	**3.1%**	0.0802	**58.5%**	0.1347	-2.0%	0.1135	-0.6%
	DICE	0.4199	-0.7%	0.1994	**1.1%**	0.1211	-14.9%	0.1149	-38.8%	0.2892	**10.9%**	0.0928	**83.4%**	0.1421	-7.7%	0.1207	-7.0%
	MACR	0.4231	**0.1%**	0.2101	**6.5%**	0.0892	**15.4%**	0.0818	**1.2%**	0.3011	**15.5%**	0.1392	**175.1%**	0.1175	**11.0%**	0.0987	**12.5%**
	CCF$_{\text{K1}}$	0.4094	-3.1%	0.1940	-1.6%	0.1047	**0.7%**	0.0835	-0.8%	0.2896	**11.0%**	0.1139	**125.1%**	0.1213	**8.1%**	0.1083	**4.0%**
	CCF$_{\text{D1}}$	0.4144	-2.0%	0.1897	-3.8%	0.1103	-4.6%	0.0844	-1.9%	0.3098	**18.8%**	0.1308	**158.5%**	0.1139	**13.7%**	0.1021	**9.5%**
	CCF$_{\text{R1r}}$	0.4271	**1.0%**	0.2004	**1.6%**	0.0854	**19.0%**	0.0713	**13.9%**	0.2874	**10.2%**	0.1013	**100.2%**	0.1229	**6.9%**	0.1102	**2.3%**
	CCF$_{\text{R1n}}$	0.4195	-0.8%	0.2111	**7.0%**	0.0881	**16.4%**	0.0803	**3.0%**	0.2816	**8.0%**	0.0970	**91.7%**	0.1243	**5.8%**	0.1079	**4.3%**
	CCF$_{\text{C}}$	0.4274	**1.1%**	0.2058	**4.4%**	0.0871	**17.4%**	0.0733	**11.5%**	0.3095	**18.7%**	0.1266	**150.2%**	0.1188	**10.0%**	0.0968	**14.2%**

capture users' preference, we apply our framework on the **Coat Shopping**[2] dataset. A special property of this dataset is that the testing data are collected from randomized trials, i.e., users give feedback on random items.

5.2 Baseline Models

We employ five causal frameworks for comparison. **CausE** [7] is a direct intervention model, which creates randomized treatment data for causal learning. **IPS** [39] is an Inverse Propensity Scoring-based model, which uses a user-independent propensity estimator to re-weight the training samples. **DCF** [55] is a deconfounded recommender, which uses an exposure model to construct a substitute confounder. **DICE** [74] is a framework for disentangling user interest and conformity for recommendation with causal embedding. **MACR** [57] is a model-agnostic framework for alleviating popularity bias issue in recommender systems.

Meanwhile, we test five versions of our framework. **CCF**$_{\text{K1}}$, **CCF**$_{\text{D1}}$, **CCF**$_{\text{R1r}}$, **CCF**$_{\text{R1n}}$ are CCF in discrete space under different heuristic rules. **CCF**$_{\text{C}}$ is CCF in continuous space.

We apply all above frameworks on three base recommendation models, including a matching model (MF), a sequential model (GRU4Rec) and a reasoning model (NCR). **MF** [37] uses Matrix Factorization [24] as the prediction function under Bayesian personalized ranking. **GRU4Rec** [14] uses Gated Recurrent Units (GRU) to capture sequential patterns. **NCR** [8] organizes the logic expressions as neural networks for reasoning and recommendation.

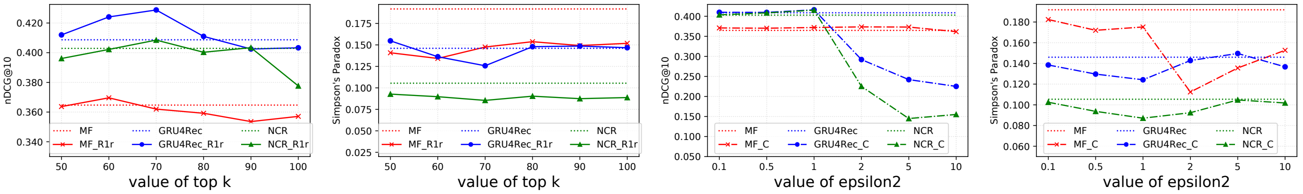
5.3 Overall Performance

We answer **RQ1** and **RQ2** in this section by showing the recommendation performance and Simpson's paradox performance of applying causal frameworks (CausE, IPS, DCF, DICE, MACR, CCF$_*$) on the three recommendation models in Table 3.

For recommendation performance, we can see that in most cases causal frameworks can bring positive improvement to the recommendation models. Comparing all frameworks, we see that for all of the recommendation models, the largest average improvement over four datasets is mostly brought by our CCF framework.

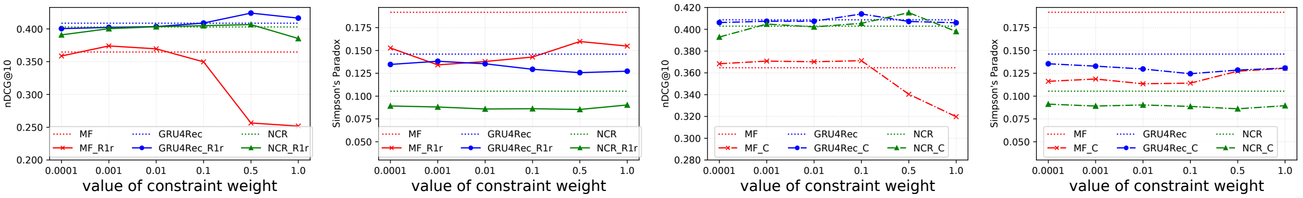
For Simpson's paradox evaluation, we follow the paradox detection method in [17]. More specifically, we split users into two groups according to gender (a binary feature in the dataset) or age (the split threshold is 35). Similar to the example in Figure 1(b), each user only recommends his or her top-K items of highest predicted scores, where K is set to 50. We randomly sample 100 users for each item and calculate the percentage of item pairs that

[2] https://www.cs.cornell.edu/~schnabts/mnar/



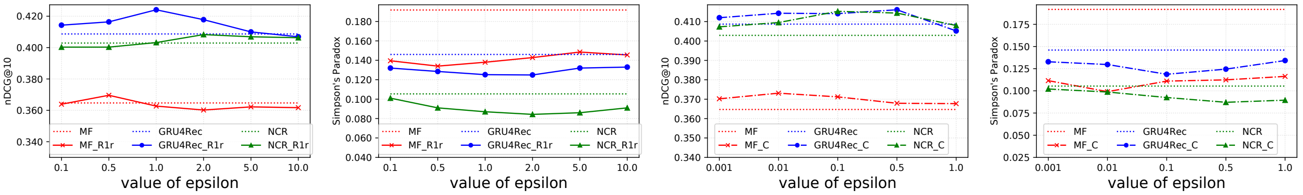
(a) Discrete Recommendation (b) Discrete Simpson's Paradox (c) Continuous Recommendation (d) Continuous Simpson's Paradox

Figure 4: Recommendation results (nDCG@10) and Simpson's paradox results (grouped by gender) on ML100k with different counterfactual selection parameters. (a) and (b) are discrete versions with R1r heuristic rule under parameter k. (c) and (d) are continuous versions under parameter ϵ_2.



(a) Discrete Recommendation (b) Discrete Simpson's Paradox (c) Continuous Recommendation (d) Continuous Simpson's Paradox

Figure 5: Recommendation results (nDCG@10) and Simpson's paradox results (grouped by gender) on ML100k with different counterfactual constraint weight ω. (a) and (b) are discrete versions with R1r heuristic rule. (c) and (d) are continuous versions.



(a) Discrete Recommendation (b) Discrete Simpson's Paradox (c) Continuous Recommendation (d) Continuous Simpson's Paradox

Figure 6: Recommendation results (nDCG@10) and Simpson's paradox results (grouped by gender) on ML100k with different counterfactual constraint threshold ϵ (ϵ_1 for continuous version). (a) and (b) are discrete versions with R1r heuristic rule. (c) and (d) are continuous versions.

	ML100k	Coat Shopping
MF	0.52	0.33
GRU4Rec	0.84	0.18
NCR	0.43	0.67

Table 4: Pearson correlation coefficient to measure the relationship between recommendation improvement and Simpson's paradox mitigation.

have Simpson's paradox. The Simpson's paradox performance is shown in Table 3, and more results of Simpson's paradox mitigation are provided in Section 5.6. We can see that our CCF framework mitigates Simpson's paradox in most cases while improving recommendation performance. In contrast, other causal frameworks may improve the recommendation performance but not necessarily mitigate the Simpson's paradox since they are designed through other perspectives. We calculate the Pearson correlation coefficient between recommendation improvement (average of improvements on nDCG@10 and Hit@1) and Simpson's paradox mitigation (average of improvements on gender and age) in Table 4. The positive correlation indicates that mitigating Simpson's paradox will help improve recommendation performance.

As we mentioned before, CausE improves performance by splitting the observational training data into approximately randomized data, IPS improves performance by re-weighting the observational training data, DCF improves performance by reconstructing a substitute confounder, DICE improves performance by adopting separate embeddings for interest and conformity to disentangle them, and MACR improves performance by eliminating the popularity bias through removing the direct effect between item properties and the ranking score. All these frameworks only consider the real-world examples though with different techniques, however, the CCF framework not only considers real-world examples but also involves counterfactual examples, which helps to mitigate the Simpson's paradox for making better decision and improving the recommendation performance.

5.4 Analyzing Counterfactual Examples

In this section, we aim to answer research questions **RQ3** and **RQ4**. We first dig into the difference between different heuristic rules. We then show the necessity of the selection process after generation.

5.4.1 ***Difference between Heuristic Rules.*** In this section, we focus on the discrete versions of CCF and discuss the effect of different heuristic rules. Among the heuristic rules in Table 2, K1 and D1 generate much fewer counterfactual histories than R1, because K1 and D1 are limited by the number of interactions in the user's real history, while R1 can replace each interacted item with a large number of possible items. As a result, it is more difficult for K1 and D1 to get satisfied counterfactual examples in the selection process

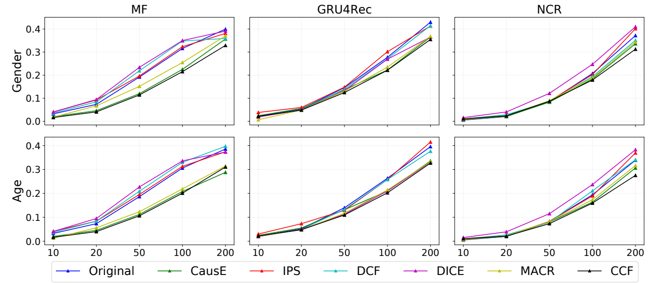


Figure 7: The percentage of item pairs showing paradox when grouping by gender (above) and age (below) on ML100k. x-axis is the number of items recommended by each user.

when k is small, where k is the top-k selection threshold introduced in Section 4.3.2. Considering the difficulty of generating qualified counterfactual examples, the R1 rules are intuitively better than D1 and K1. This is consistent with the experimental results.

5.4.2 ***Counterfactual Example Selection.*** The selection is based on parameter k (Section 4.3.2) and ϵ_2 in Eq.(12) for CCF discrete and continuous version, respectively. We first examine the discrete versions. We plot the recommendation performance on nDCG@10 and the Simpson's paradox results under different k on Movielens-100k in Figure 4(a-b), where users are grouped by gender and the R1r rule is used. Other rules and datasets have similar observations. We see that when k is properly chosen, our framework will mitigate Simpson's paradox and improve the performance. However, when k is too large—such as $k = 100$ so that all the generated counterfactual examples are selected—the counterfactual constraint will mislead the causal preference estimation and lead to relatively more paradox thus hurt the performance. This observation is consistent with the theory of conditional intervention (Section 4.2).

For continuous version, we plot recommendation performance and Simpson's paradox results under different ϵ_2 in Figure 4(c-d). When ϵ_2 is small, the counterfactual embedding $\mathbf{x'}$ is very close to the real embedding $\mathbf{x}$ (Eq.(10)(12)), therefore, the estimation of preference after applying CCF has no much difference from the original preference, because the counterfactual constraint in Eq.(10)(12) is easily satisfied. In contrast, if ϵ_2 is too large, $\mathbf{x'}$ will be too far away from the real embedding $\mathbf{x}$, and if we force their predictions to be close, causal preference will not be correctly estimated thus the performance will decrease.

5.5 Analyzing Counterfactual Constraints

There are two important parameters for the counterfactual constraint—parameter ω in Eq.(11) and (12), and parameter ϵ in Eq.(11) (or ϵ_1 in Eq.(12)). In this section, we provide the answers to **RQ5**. We will discuss the two parameters separately in the following.

5.5.1 ***Counterfactual Constraint Weight.*** Given loss function as Eq.(11) and (12), the larger the counterfactual constraint weight ω, the more likely the results will follow the constraint. We tune the ω while keeping other parameters fixed. The results of nDCG@10 are shown in Figure 5(a)(c). The results of Simpson's paradox (grouped by gender) are provided in Figure 5(b)(d). We see that in most cases the performance would first getting better and then worse, meaning that the constraint is useful for recommendation and Simpson's paradox mitigation but it also requires a good balance with the original loss. When ω is too small, the constraint has little effect on the total loss, leading to only slight improvement or even slight decrease considering the larger model complexity. In contrast, if ω is too large, the constraint loss will dominate the total loss, and thus the recommendation performance is significantly decreased since the original loss does not take too much effect. Meanwhile, in this case, the causal preference may not be accurately estimated thus hurt Simpson's paradox mitigation. Overall, the weight needs to be carefully specified in practice, and compared with an overly large weight, a relatively smaller weight would be preferred.

5.5.2 ***Counterfactual Constraint Threshold.*** The counterfactual constraint threshold (i.e. ϵ in Eq.(9) and ϵ_1 in Eq.(10)) controls how rigorous the constraint is. We plot nDCG@10 with different threshold in Figure 6. We see that the performance first getting better and then worse and finally tend to be flat when the threshold is large enough. When the threshold is too small, the constraint would be too tight and it makes the model less capable of handling the potential errors in counterfactual examples. When the threshold is too large, we are actually applying no constraint, since the difference between real and counterfactual examples' prediction would always be smaller than the threshold, and thus the L_c in Eq.(11) and (12) would be 0 in most cases. As a result, the performance becomes relatively flat when the threshold is large enough.

5.6 Influence of Recommendation Length on Simpson's Paradox Mitigation

Figure 7 shows the percentage of item pairs that have paradox, with different K ranging from 10 to 200 and users are grouped by gender or age. The figure shows three base recommendation models under all frameworks, where CCF is the continuous version. Coat dataset has similar observations. From the results, we can see that CCF framework is able to reduce paradox compared with the original model in the most cases. Additionally, CCF (i.e., the black line in Figure 7) is almost always the lowest line in all sub-figures, showing that CCF achieves the best performance compared with other frameworks in terms of mitigating paradox.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a causal framework (CCF) for mitigating Simpson's Paradox in recommendation. We provided a conditional intervention approach to estimating the $P(y|u, do(v))$ and proposed a flexible counterfactual constrained learning framework which is applicable to many recommendation models. Experiments show that CCF helps to mitigate Simpson's paradox and improve the performance of the matching-, sequential- and reasoning-based models. The CCF framework is flexible and can be extended in various dimensions in the future, such as extending the causal graph to more complicated graphs for more complex recommendation scenarios. We will explore these possibilities in the future.

ACKNOWLEDGMENT

This work was supported in part by NSF IIS-1910154, IIS-2007907, IIS-2046457 and IIS-2127918. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering* 17, 6 (2005), 734–749.

[2] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. 2018. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms* 11, 9 (2018), 137.

[3] Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W Bruce Croft. 2018. Unbiased learning to rank with unbiased propensity estimation. In *SIGIR*. 385–394.

[4] Nazanin Alipourfard, Peter G Fennell, and Kristina Lerman. 2018. Can you Trust the Trend? Discovering Simpson's Paradoxes in Social Data. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 19–27.

[5] Nazanin Alipourfard, Peter G Fennell, and Kristina Lerman. 2018. Using Simpson's paradox to discover interesting patterns in behavioral data. In *Twelfth international AAAI conference on web and social media*.

[6] Colin R Blyth. 1972. On Simpson's paradox and the sure-thing principle. *J. Amer. Statist. Assoc.* 67, 338 (1972), 364–366.

[7] Stephen Bonner and Flavian Vasile. 2018. Causal embeddings for recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 104–112.

[8] Hanxiong Chen, Shaoyun Shi, Yunqi Li, and Yongfeng Zhang. 2021. Neural Collaborative Reasoning. In *Proceedings of the 30th Web Conference (WWW)*.

[9] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *WSDM*. 108–116.

[10] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

[11] Michael D Ekstrand, John T Riedl, and Joseph A Konstan. 2011. *Collaborative filtering recommender systems*. Now Publishers Inc.

[12] Azin Ghazimatin, Oana Balalau, Rishiraj Saha Roy, and Gerhard Weikum. 2020. PRINCE: Provider-side Interpretability with Counterfactual Explanations in Recommender Systems. In *WSDM*. 196–204.

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

[14] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *ICLR*.

[15] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative metric learning. In *In WWW*. 193–201.

[16] Ziniu Hu, Yang Wang, Qu Peng, and Hang Li. 2019. Unbiased lambdamart: an unbiased pairwise learning-to-rank algorithm. In *WWW*. 2830–2836.

[17] Amir H Jadidinejad, Craig Macdonald, and Iadh Ounis. 2021. The Simpson's Paradox in the Offline Evaluation of Recommendation Systems. *ACM Transactions on Information Systems (TOIS)* 40, 1 (2021), 1–22.

[18] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. 2010. *Recommender systems: an introduction*. Cambridge University Press.

[19] Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. 2020. A Re-visit of the Popularity Baseline in Recommender Systems. *SIGIR* (2020).

[20] Thorsten Joachims, Adith Swaminathan, and Maarten de Rijke. 2018. Deep learning with logged bandit feedback. In *ICLR*.

[21] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased learning-to-rank with biased feedback. In *WSDM*. 781–789.

[22] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. 2010. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *RecSys*. 79–86.

[23] Joseph A Konstan, Bradley N Miller, David Maltz, Jonathan L Herlocker, Lee R Gordon, and John Riedl. 1997. GroupLens: applying collaborative filtering to Usenet news. *Commun. ACM* 40, 3 (1997), 77–87.

[24] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[25] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 1419–1428.

[26] Yunqi Li, Hanxiong Chen, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. 2021. Towards Personalized Fairness based on Causal Notion. *SIGIR* (2021).

[27] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 7, 1 (2003).

[28] 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*. 831–840.

[29] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*. ACM.

[30] Rishabh Mehrotra, James McInerney, Hugues Bouchard, Mounia Lalmas, and Fernando Diaz. 2018. Towards a fair marketplace: Counterfactual evaluation of the trade-off between relevance, fairness & satisfaction in recommendation systems. In *Proceedings of the 27th acm international conference on information and knowledge management*. 2243–2251.

[31] Eric Neufeld. 1995. Simpson's paradox in artificial intelligence and in real life. *Computational intelligence* 11, 1 (1995), 1–10.

[32] Harrie Oosterhuis and Maarten de Rijke. 2020. Policy-aware unbiased learning to rank for top-k rankings. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 489–498.

[33] Harrie Oosterhuis, Rolf Jagerman, and Maarten de Rijke. 2020. Unbiased learning to rank: counterfactual and online approaches. In *Companion Proceedings of the Web Conference 2020*. 299–300.

[34] Judea Pearl. 2000. Causality: Models, reasoning and inference. *Cambridge University Press* (2000).

[35] Judea Pearl, Madelyn Glymour, and Nicholas P Jewell. 2016. *Causal inference in statistics: A primer*. John Wiley & Sons.

[36] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*. IEEE, 995–1000.

[37] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *UAI* (2012).

[38] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: an open architecture for collaborative filtering of netnews. In *CSCW*. 175–186.

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

[40] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*. 285–295.

[41] Masahiro Sato, Janmajay Singh, Sho Takemori, Takashi Sonoda, Qian Zhang, and Tomoko Ohkuma. 2019. Uplift-based evaluation and optimization of recommenders. In *RecSys*. 296–304.

[42] Masahiro Sato, Sho Takemori, Janmajay Singh, and Tomoko Ohkuma. 2020. Unbiased learning for the causal effect of recommendation. In *RecSys*. 378–387.

[43] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative filtering recommender systems. In *The adaptive web*. Springer, 291–324.

[44] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as treatments: Debiasing learning and evaluation. In *ICML*.

[45] Rahul Sharma, Minakshi Kaushik, Sijo Arakkal Peious, Markus Bertl, Ankit Vidyarthi, Ashwani Kumar, and Dirk Draheim. 2022. Detecting Simpson's Paradox: A Step Towards Fairness in Machine Learning. In *European Conference on Advances in Databases and Information Systems*. Springer, 67–76.

[46] Galit Shmueli and Inbal Yahav. 2018. The forest or the trees? Tackling Simpson's paradox with classification trees. *Production and Operations Management* 27, 4 (2018), 696–716.

[47] Edward H Simpson. 1951. The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society: Series B (Methodological)* 13, 2 (1951), 238–241.

[48] Juntao Tan, Shuyuan Xu, Yingqiang Ge, Yunqi Li, Xu Chen, and Yongfeng Zhang. 2021. Counterfactual explainable recommendation. *CIKM* (2021).

[49] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 565–573.

[50] Khanh Hiep Tran, Azin Ghazimatin, and Rishiraj Saha Roy. 2021. Counterfactual Explanations for Neural Recommenders. *SIGIR* (2021).

[51] Julius von Kügelgen, Luigi Gresele, and Bernhard Schölkopf. 2021. Simpson's paradox in Covid-19 case fatality rates: a mediation analysis of age-related causal effects. *IEEE transactions on artificial intelligence* 2, 1 (2021), 18–27.

[52] Pengfei Wang, Hanxiong Chen, Yadong Zhu, Huawei Shen, and Yongfeng Zhang. 2019. Unified Collaborative Filtering over Graph Embeddings. In *SIGIR*. 155–164.

[53] Wenjie Wang, Fuli Feng, Xiangnan He, Xiang Wang, and Tat-Seng Chua. 2021. Deconfounded recommendation for alleviating bias amplification. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1717–1725.

[54] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. 2018. Position bias estimation for unbiased learning to rank in personal search. In *WSDM*. 610–618.

[55] Yixin Wang, Dawen Liang, Laurent Charlin, and David M Blei. 2020. Causal Inference for Recommender Systems. In *Fourteenth ACM Conference on Recommender Systems*. 426–431.

[56] Zhenlei Wang, Jingsen Zhang, Hongteng Xu, Xu Chen, Yongfeng Zhang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Counterfactual data-augmented sequential recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 347–356.

[57] Tianxin Wei, Fuli Feng, Jiawei Chen, Ziwei Wu, Jinfeng Yi, and Xiangnan He. 2021. Model-agnostic counterfactual reasoning for eliminating popularity bias in recommender system. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1791–1800.

[58] Xinwei Wu, Hechang Chen, Jiashu Zhao, Li He, Dawei Yin, and Yi Chang. 2021. Unbiased Learning to Rank in Feeds Recommendation. In *WSDM*. 490–498.

[59] Xu Xie, Zhaoyang Liu, Shiwen Wu, Fei Sun, Cihang Liu, Jiawei Chen, Jinyang Gao, Bin Cui, and Bolin Ding. 2021. CausCF: Causal Collaborative Filtering for

RecommendationEffect Estimation. *CIKM* (2021).

[60] Kun Xiong, Wenwen Ye, Xu Chen, Yongfeng Zhang, Wayne Xin Zhao, Binbin Hu, Zhiqiang Zhang, and Jun Zhou. 2021. Counterfactual Review-based Recommendation. *CIKM* (2021).

[61] Chenguang Xu, Sarah M Brown, and Christan Grant. 2018. Detecting Simpson's paradox. In *The Thirty-First International Flairs Conference*.

[62] Shuyuan Xu, Juntao Tan, Zuohui Fu, Jianchao Ji, Shelby Heinecke, and Yongfeng Zhang. 2022. Dynamic Causal Collaborative Filtering. In *CIKM*.

[63] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems.. In *IJCAI*, Vol. 17. Melbourne, Australia, 3203–3209.

[64] Longqi Yang, Yin Cui, Yuan Xuan, Chenyang Wang, Serge Belongie, and Deborah Estrin. 2018. Unbiased offline recommender evaluation for missing-not-at-random implicit feedback. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 279–287.

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

[66] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 974–983.

[67] Bowen Yuan, Jui-Yang Hsia, Meng-Yuan Yang, Hong Zhu, Chih-Yao Chang, Zhenhua Dong, and Chih-Jen Lin. 2019. Improving ad click prediction by considering non-displayed events. In *CIKM*. 329–338.

[68] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *KDD*. 353–362.

[69] Shengyu Zhang, Dong Yao, Zhou Zhao, Tat-Seng Chua, and Fei Wu. 2021. Causerec: Counterfactual user sequence synthesis for sequential recommendation. In *SIGIR*. 367–377.

[70] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.

[71] Yongfeng Zhang, Qingyao Ai, Xu Chen, and W Bruce Croft. 2017. Joint representation learning for top-n recommendation with heterogeneous information sources. In *CIKM*. 1449–1458.

[72] Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. 2021. Causal Intervention for Leveraging Popularity Bias in Recommendation. *WWW* (2021).

[73] Lei Zheng, Vahid Noroozi, and Philip S. Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *WSDM*.

[74] Yu Zheng, Chen Gao, Xiang Li, Xiangnan He, Depeng Jin, and Yong Li. 2021. Disentangling User Interest and Conformity for Recommendation with Causal Embedding. *WWW* (2021).