# Deconfounded Causal Collaborative Filtering

SHUYUAN XU, Rutgers University, New Brunswick, NJ, US
JUNTAO TAN, Rutgers University, New Brunswick, NJ, US
SHELBY HEINECKE, Salesforce Research, Palo Alto, CA, US
VENA JIA LI, Meta, Menlo Park, CA, US
YONGFENG ZHANG, Rutgers University, New Brunswick, NJ, US

Recommender systems may be confounded by various types of confounding factors (also called confounders) that may lead to inaccurate recommendations and sacrificed recommendation performance. Current approaches to solving the problem usually design each specific model for each specific confounder. However, real-world systems may include a huge number of confounders and thus designing each specific model for each specific confounder could be unrealistic. More importantly, except for those "explicit confounders" that experts can manually identify and process such as item's position in the ranking list, there are also many "latent confounders" that are beyond the imagination of experts. For example, users' rating on a song may depend on their current mood or the current weather, and users' preference on ice creams may depend on the air temperature. Such latent confounders may be unobservable in the recorded training data. To solve the problem, we propose Deconfounded Causal Collaborative Filtering (DCCF). We first frame user behaviors with unobserved confounders into a causal graph, and then we design a front-door adjustment model carefully fused with machine learning to deconfound the influence of unobserved confounders. Experiments on real-world datasets show that our method is able to deconfound unobserved confounders to achieve better recommendation performance.

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

Additional Key Words and Phrases: Recommender Systems; Deconfounded Recommendation; Causal Analysis

## 1 INTRODUCTION

Recommender systems occupy an expanding role in daily decision making, such as e-commerce, video streaming and social media. Most of the existing recommendation models learn from the collective historical interactions to achieve good recommendation performance. However, the collected data may have been affected by various types of confounders, i.e., variables that affect both treatment assignment (which item to be exposed to the user) and outcome (whether the user likes the item) [18, 27, 44], which may lead to spurious correlation [40, 51]. As a result, the collected data may not represent the accurate preference of users. Therefore, recommendation algorithms that are trained with confounded data may result in inaccurate recommendations and sacrificed performance.

The existence of confounders naturally leads to a question: can we develop models to mitigate the influence of confounders so as to estimate more accurate user-item preferences? To answer this question, current approaches usually identify one specific confounder based on researchers' expertise. Then, they design specific models to tackle the pre-identified confounding problem. For instance, many models are proposed to handle the popularity confounders [17, 47, 58, 60, 61], click confounders [35, 40, 41], exposure confounders [1, 12, 34, 55], etc. Existing efforts have performed well on combating these manually identified confounders. However, real-world systems may involve a huge number of confounders, which makes it unrealistic to design each specific model for each

Authors' addresses: Shuyuan Xu, Rutgers University, New Brunswick, NJ, US, shuyuan.xu@rutgers.edu; Juntao Tan, Rutgers University, New Brunswick, NJ, US, juntao.tan@rutgers.edu; Shelby Heinecke, Salesforce Research, Palo Alto, CA, US, shelby.heinecke@salesforce.com; Vena Jia Li, Meta, Menlo Park, CA, US, vena900620@gmail.com; Yongfeng Zhang, Rutgers University, New Brunswick, NJ, US, yongfeng.zhang@rutgers.edu.

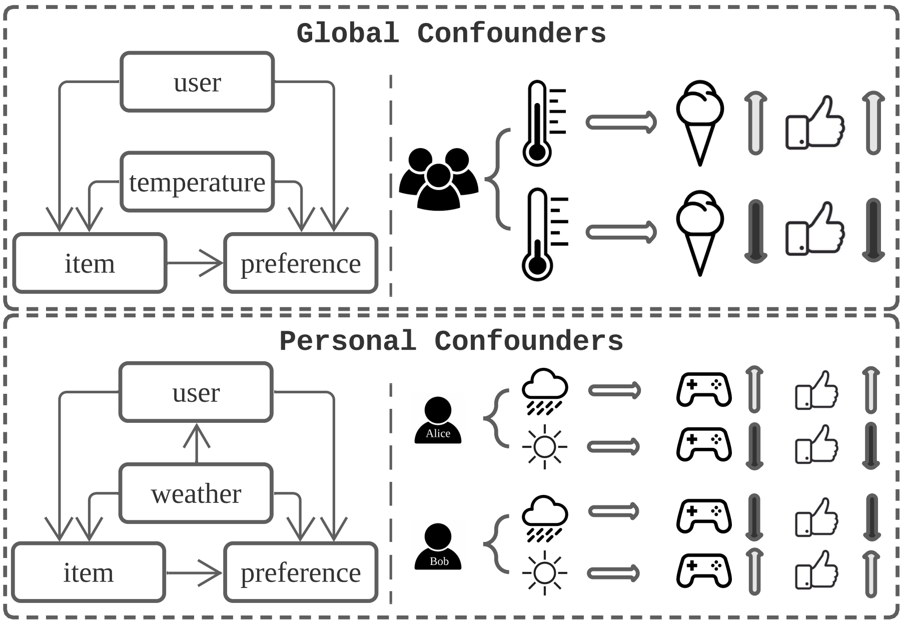


Fig. 1. Examples of global and personal confounders.

specific confounder. Moreover, the confounders are not limited to the "explicit confounders" that can be manually identified by experts based on their knowledge, but also include many "latent confounders" that can be hardly identified or measured. For example, a user's interaction and rating with music recommendations can be affected by his/her current mood or spirit, however, the mood or spirit cannot be quantitatively measured or logged in the training data. The large number of confounders and the existence of latent confounders make it important to develop "universal" deconfounded recommendation models that are agnostic to certain types or certain numbers of confounders. It is worth mentioning that deconfounding is similar with debiasing in some cases but they are different concepts. Confounding requires the confounders to affect both treatments (exposures) and outcomes (feedbacks), while biases such as exposure bias or selection bias only result in non-uniform exposures [51]. More specifically, if a variable affects both exposures and feedbacks, it creates the discrepancy between the observed and true feedbacks, which is the confounding issue. Otherwise, if a variable only leads to non-uniform exposures, it is the normal biasing issues.

Broadly speaking, the confounders can be classified into global confounders and personal confounders, as shown in Figure 1. Global confounders are those that influence preferences of all users in a consistent way, while personal confounders are those that affect different users differently. For example, the air temperature can be a global confounder for users' preferences on ice creams—during normal times, some users tend to like ice creams while others tend to dislike ice creams, however, an extremely high temperature may increase both groups' preference on ice creams. On the other hand, the weather can be a personal confounder for users' preferences on game consoles, which means it may have different effects for different users. For example, some users preferences on game consoles might decrease in sunny days while increasing in rainy days. This is because they choose to take outdoor activities in sunny days but stay at home in rainy days. However, for other users, their preference on game consoles may increase in sunny days because they have certain skin disease and avoid going outside when it's sunny. This complex nature of confounders makes it challenging to design deconfounded recommendation models.

Fortunately, causal inference techniques—especially front-door adjusted models—make it possible to design a unified deconfounding model for both global and personal unobserved confounders [27, 51]. Actually, the essence of recommendation is trying to answer a "what if" question: what would happen if we recommend a certain item to a target user? Inspired by causal collaborative filtering [50, 52, 53], this "what if" question can be represented as $P(y|u, do(v))$, where $u, v$ is a user-item pair and $y$ is the estimated preference score. In particular, the user, item, and preference can be formulated into a causal graph, while unobserved global or personal confounders present possible connections to the user, item and preference nodes (Figure 1). Technically, we identify inherent item features as a mediator variable $M$ between item and preference that is independent from the influence of confounders (Figure 3, more details later). Based on mediator analysis, we are able to estimate the deconfounded user-item preference $P(y|u, do(v))$ based on front-door adjustments rooted in causal theory [27], where $u, v$ is a pair of user and item, $y$ is the preference score between them and the $do$-operation is used to model the interventions. Intuitively, the deconfounded preference is used to represent the causal preference if we intervene to recommend item $v$ instead of passively observing item $v$ in training data. Considering the large scale of items in recommender systems, it is impractical to strictly apply the front-door adjustment into the model, therefore, we design a sample-based approach to make it calculable (more details will be shown in Section 4). In the experiments, we compare our model with both state-of-the-art deconfounded recommendation algorithms and traditional association-based recommendation models. The results show that our deconfounded algorithm achieves better recommendation performance than all of the baselines. In summary, we list our key contributions as follows:

- We design a causal graph to describe the data generation process and represent the effects of unobserved confounders in the recommendation scenario.
- To mitigate the effects of unobserved confounders which are not directly measurable, we adapt the front-door adjustment into the proposed deconfounded causal recommnedation model.
- We design a sample-based approach integrated with exposure models to make the front-door adjustment calculable despite the large item space.
- Experiments on three real-world datasets show that our deconfounded model outperforms existing deconfounded recommendation models and traditional association-based models.

The remainder of this paper is organized as follows. We discuss the related works in Section 2. In Section 3, we introduce some notations, basic concepts and theorems to help readers gain a better understanding of the fundamentals. We introduce our proposed model in Section 4. In Section 5, we experiment on real-world datasets and make discussions. Finally, we conclude the work and discuss future directions in Section 6.

## 2 RELATED WORK

Deconfounded recommender systems aim to remove or reduce the unwanted effect of confounders, which is important to the accurate estimation of user preferences [51]. Among the large number of methods mitigating the confounding effects, causal technique plays an important role. Inverse Propensity Score (IPS) based method is an important approach for deconfounding. The basic idea is to re-weight the observations with inverse propensity scores so as to mitigate the influence of selection bias [35, 41], exposure bias [34, 45, 55], position bias [3, 11, 15, 39], etc. IPS-based methods are well used approaches for deconfounded learning with observational data, but they still suffer from some known issues. For example, the performance of IPS-based methods highly depends on the accurate estimation of propensity scores [33] and usually suffers from the high variance of the propensity scores [5].

Besides IPS, leveraging causal graph is a powerful approach to deconfounded recommendation. Many existing methods construct a specific causal graph based on certain assumptions to incorporate specific confounders into the data generation process, and then apply causal inference techniques to mitigate the confounding bias. Just to name a few as examples, Zhang et al. [58] assumed that item popularity affects the item exposure and user interaction, and constructed a causal graph incorporated with item popularity to estimate the user-item preferences; Qiu et al. [29] observed visual bias in visual-aware recommendation and that users' attention to visual features does not always reflect the real preference, and thus developed a causal graph to remove the effect of visual confounders; Li et al. [19] noticed that users' sensitive features may lead to unfair recommendations and thus developed causal graphs to deconfound the influence of sensitive features in recommendation; Li et al. [18] achieved causal feature selection in factorization machines so as to enhance the robustness of recommendation when the distributions of training data and testing data are different; Wang et al. [40] identified that user clicks may be influenced by item popularity and proposed a deconfounding method to alleviate the amplification of popularity bias. Many other research works are conducted to address different types of confounders, including but not limited to item popularity [2, 9, 17, 47, 60, 61], item exposure [1, 12, 34, 43, 44, 55], user selection [21, 26, 35, 40, 41], and ranking positions [3, 4, 13, 25, 28, 48]. Furthermore, causal and counterfactual reasoning has also been adopted to tackle many other issues in recommender systems such as explainability [14, 37, 38], fairness [19], robustness [7, 18], adversarial attacks [7], data augmentation [6, 46, 49], and evaluating intelligent systems [10].

However, due to the large number of confounders and the existence of unobserved latent confounders, it is unrealistic to design each specific model to tackle each specific confounder. Some recent works took advantage of a subset of unbiased data as supervision to remove confounding effects [57]. However, unbiased data may not always exist and collecting unbiased data can be difficult since the randomized recommendations involved may hurt the user experience. Some recent deconfounded methods rely on certain user information such as user social network in [8]. However, user information may be sensitive and not available sometimes. A recent work by Zhu et al. considers click behavior as mediator and designs a multi-task learning framework that simultaneously learns the probability of click and purchase to mitigate confounding effects [59]. Shang et al. [36] designed a confounder policy to handle the hidden confounder for environment reconstruction on reinforcement learning based recommendation. However, this model is not suitable for general settings.

As a result, a broad-spectrum deconfounded recommendation model to mitigate the confounding effects based on observed data is urgently needed. As we will show later, we incorporate the unobserved confounders into the causal graph and leverage front-door adjustments to mitigate the effect of confounders, which is able to deconfound recommender systems based on observed data without the need to enumerate confounders or collect unbiased datasets.

## 3 PRELIMINARIES

As we introduced in Section 1, our model is able to estimate the deconfounded user-item preference $P(y|u, do(v))$ by leveraging the designed causal graph. We will first introduce some basic concepts in causal theory.

Definition 1. *(Causal Graph) [27, p.35] A causal graph is a directed acyclic graph (DAG) $\mathcal{G} = (U, E)$, which captures the direct causal relationships among the variables.*

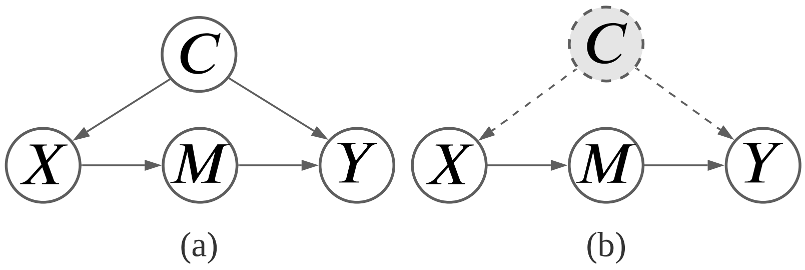


Fig. 2. $X$ represents treatment variable, $Y$ represents outcome variable, $M$ represents mediator variable, $C$ represents confounder variable. Figure (a) shows variable $C$ as observed and measurable, while in Figure (b), variable $C$ is unobserved or unmeasurable.

The key component of our deconfounded estimation is *do*-operation, which models the interventions, in the estimated preference $P(y|u, do(v))$. We introduce intervention as follows:

Definition 2. *(Intervention) [27, p.55] We distinguish between cases where a variable $X$ takes a value $x$ naturally and cases where we fix $X = x$ by denoting the later $do(X = x)$. So $P(Y = y|X = x)$ is the probability that $Y = y$ conditioned on finding $X = x$, while $P(Y = y|do(X = x))$ is the probability that $Y = y$ when we intervene to make $X = x$. Similarly, we write $P(Y = y|do(X = x), Z = z)$ to denote the conditional probability of $Y = y$, given $Z = z$, in the distribution created by the intervention $do(X = x)$.*

Given a causal graph, there are some methods to estimate the desired interventional probabilities in causal theory. The back-door adjustment and the front-door adjustment are two fundamental theorems in causal inference that can be used to estimate the deconfounded user-item preference using the observational data. However, according to our designed causal graph (will introduce in Section 4.1), the back-door adjustment is not applicable for our causal graph because the confounders are unobserved. We will first introduce the back-door adjustment and the front-door adjustment in causal inference. Additionally, for better understanding, we will mathematically illustrate why back-door adjustment is not applicable in our setting and why we use front-door adjustment instead to estimate the deconfounded user-item preference.

Definition 3. *(Back-door Criterion) [27, p.61] Given an ordered pair of variables $(X, Y)$ in a causal graph $\mathcal{G}$, a set of variables $Z$ satisfies the back-door criterion with respect to $(X, Y)$ if $Z$ satisfies the following conditions:*

- *No node in $Z$ is a descendant of $X$;*
- *$Z$ blocks every path between $X$ and $Y$ that contains an arrow into $X$.*

With the help of a set of variables that satisfy the back-door criterion, we can adjust for the effect of measured confounders. We take the causal graph in Figure 2(a) as an example. Considering the treatment variable $X$ and the outcome variable $Y$, we want to estimate the effect of $X$ on $Y$, which is denoted as $P(Y = y|do(X = x))$. Due to the existence of confounder $C$, we cannot conclude that $P(Y = y|do(X = x)) = P(Y = y|X = x)$. However, since variable $C$ satisfies the back-door criterion, we use it to adjust the effect, in other words, we are accounting for and measuring all confounders [27, 31]. Therefore, we compute $P(Y = y|do(X = x))$ (abbreviated as $P(y|do(x))$ later) as follows:

$$P(Y = y|do(X = x)) = \sum_{c} P(Y = y|X = x, C = c)P(C = c) \tag{1}$$

The above equation is based on the assumption that the confounder variable, which also satisfies the backdoor criterion, is measurable. However, in recommender systems, there may exist various unobserved or unmeasurable confounding variables (as mentioned in Section 1). To address this setting, we introduce the front-door criterion.

Definition 4. *(Front-door Criterion) [27, p.69] Given an ordered pair of variables $(X, Y)$ in a causal graph $\mathcal{G}$, a set of variables $Z$ satisfies the front-door criterion with respect to $(X, Y)$ if $Z$ satisfies the following conditions:*

- *$Z$ intercepts all directed paths from $X$ to $Y$.*
- *There is no unblocked back-door path from $X$ to $Z$.*
- *$X$ blocks all back-door paths from $Z$ to $Y$.*

Given a set of variables that satisfies the front-door criterion, we can identify the causal effect with unobserved confounders [27].

Definition 5. *(Front-door Adjustment) [27, p.69] If a set of variables $Z$ satisfy the front-door criterion related to an ordered pair of variables $(X, Y)$, and if $P(x, z) > 0$, then the causal effect of $X$ on $Y$ is identifiable and is given by*

$$P(y|do(x)) = \sum_z P(z|x) \sum_{x'} P(y|x', z)P(x') \tag{2}$$

We take Figure 2(b) as an example. Although variable $C$ satisfies the back-door criterion, it is not measurable, so the back-door adjustment (Eq.(1)) cannot be applied in this example. Although the back-door adjustment is not applicable here, given that variable $M$ satisfies the front-door criterion, we can use the front-door adjustment to handle unmeasurable or unobserved confounders. Intuitively, the desired effect can be expressed as follows

$$P(y|do(x)) = \sum_m P(m|do(x))P(y|do(m)) \tag{3}$$

Since the only parent node of variable $M$ is $X$ (thus $P(m|do(x)) = P(m|x)$ [27]), and variable $X$ satisfies the back-door criterion with respect to $(M, Y)$ so that we can apply back-door adjustment for $P(y|do(m))$, therefore, Eq.(3) can be further derived as

$$P(y|do(x)) = \sum_m P(m|x) \sum_{x'} P(y|x', m)P(x') \tag{4}$$

We can see Eq.(4) is exactly the front-door adjustment as Eq.(2).

## 4 THE DECONFOUNDED MODEL

In this section, we first introduce our designed causal graph depicting user behaviors with unobserved confounders. We then elaborate on our Deconfounded Causal Collaborative Filtering (DCCF) model. The primary notation used in this section is detailed in Table 1.

### 4.1 The Causal Graph

We formulate the process of user-system interaction as a causal graph shown in Figure 3. We explain the rationality of our designed causal graph from the view of data generation as follows:

- Edges $\{U, C\} \longrightarrow V$ denote that the user will determine which item to interact with, and the unobserved confounder may affect user's decision.
- Edge $V \longrightarrow M$ denotes that the item will determine the inherent item features. Here, the inherent features of the items that are designed as is from the beginning and will never change. For example, a silver macbook has a color as silver, brand as Apple, memory as 8G, OS as Macintosh. Thus the inherent item feature is solely determined by the item $V$ and will not be directly influences by user $U$ and confounders $C$.

| Symbol | Definition |
|---|---|
| $U, u$ | The user variable and the corresponding specific values |
| $V, v$ | The item variable and the corresponding specific values |
| $M, m$ | The inherent item features and the corresponding specific values |
| $Y, y$ | The preference variable and the corresponding specific values |
| $C$ | The confounder variable |
| $\mathcal{U}, \mathcal{V}$ | The user set and the item set |
| $\mathbf{u}$,$\mathbf{v}$ | The representation of user $u$ and item $v$ |
| $\mathbf{U}$, $\mathbf{V}$ | The representation matrix of users and items |
| $m_v$, $\mathbf{m}_v$ | The inherent item feature of item $v$ and the latent representation of the inherent item feature of item $v$ |
| $D_{uv}$ | The dimension of the latent representation of users and items |
| $D_m$ | The dimension of the latent representation of inherent item features |

Table 1. Notations

- Edge $C \rightarrow U$ denotes that the unobserved confounders may affect user's preference. For example, as we exemplified in Section 1, weather, as a confounder, may personally affect users and lead to different preference.
- Edges $\{U, M, C\} \rightarrow Y$ denote that the user preference score is determined by user $U$, inherent item feature $M$ and unobserved confounders $C$. Specifically, the user's preference on a certain item may be affected by inherent item features (e.g., whether the functional attributes of the item conform to user preferences) and the confounding factors (e.g., the weather, the mood, etc.).

The effect of global confounders (e.g., the air temperature as mentioned in Section 1) is represented by edge $C \rightarrow V$ and edge $C \rightarrow Y$. The effect of personal confounders (e.g. the mood as mentioned in Section 1) is represented by edge $C \rightarrow U$, edge $C \rightarrow V$ and edge $C \rightarrow Y$.

Setting variable of inherent item features as the mediator is intuitive and reasonable because inherent item features such as the color or brand of products and the genre or director of movies are inherent and fixed properties of an item, which are not influenced by users or external confounders. The inherent item features are not directly affected by users or confounders. Considering a general example, a user will click an item during browsing the website, where the inherent item features may not visible to users at this stage (for example, amazon homepage recommendations only show an image without any text information about the inherent item features). After clicking the item, based on the detailed inherent item features, combining with the effect of unobserved confounders and personal preference, the user will decide like or not, purchase or not, etc. Our model aims to estimate the user's true preference of an item that is decided by the inherent item features and not affected by confounders.

In order to achieve our purpose during implementation, it is crucial to carefully select the inherent item features to serve as the mediator. Not all item features can be considered as inherent features since the mediator should not be directly affected by users or confounders. Specifically, we only select objective features, including features related to the appearance and functionality of the item, such as item description, color, and other similar features. Some other item features, such as sales rank and related products, are generated based on user's subjective reviews or user interactions and cannot be selected as the mediator. In real-world scenarios, it may not always be feasible for the selected mediator to completely intercept the causal pathways from the items to the preference variable, resulting in the possibility of direct effects from items to preference. Such a situation violates the front-door criterion and may have a negative impact on the deconfounding performance. However, if the mediator is chosen appropriately, it can capture a significant portion of the causal effect, making the direct effect weak and negligible in predicting preference. Hence, the impact on performance may not be significant. We will show some experimental results to illustrate it in Section 5.

It is worth clarifying that the causal graph is used for describing how the collected data was generated. We ultimately build a recommendation model that can leverage this causal graph to produce better estimations of user preferences.

## 4.2 Estimated Preference Score

In this section, we estimate user preference from a causal view, which aims to answer a *what if* question: what would be the user's preference on an item if we intervene to recommend the item to the user. In particular, we estimate a user's preference if a certain item is recommended to the user, which can be represented as $P(y|u, do(v))$. The estimation of the preference score $P(y|u, do(v))$ is based on the designed causal graph in Figure 3. From the causal graph as shown in Figure 3, it seems that the desired estimation can be calculated by the back-door adjustment [27, p.61] (as Eq.(1)) since a set of variables $\{C, U\}$ satisfies the back-door. However, the back-door adjustment is not applicable because variable $C$ is unobserved or unmeasurable. Therefore, considering the existence of unobserved confounders, we apply front-door adjustment [27, p.69] (following Eq.(2)) to estimate user's preference on items. Concretely, We estimate the preference score of a $(u, v)$ pair as follows:

$$P(y|u, do(v)) = \sum_{m} P(m|v) \sum_{v'} P(v'|u) P(y|u, v', m) \tag{5}$$

where $y$ represents the value of preference score and $m$ represents a specific value of inherent item features $M$. Here $m$ integrates all inherent features of an item. For example, an item has inherent features including color, brand, etc. $m$ represents a inherent feature profile which integrates the specific value of each inherent features, such as color as silver, brand as Apple, etc.

To utilize the item features into model prediction, we represent the item features into the continuous space, then we can rewrite the preference score as follows:

$$P(y|u, do(v)) = \int_{\mathbf{m}} P(\mathbf{m}|v) \sum_{v'} P(v'|u) P(y|u, v', \mathbf{m}) d\mathbf{m} \tag{6}$$

We denote the described feature of item $v$ as $\mathbf{m}_v$, and consider the conditional distribution of item feature as $P(\mathbf{m}|v) \sim \mathcal{N}(\mathbf{m}_v, \sigma_m)$. Using distribution instead of deterministic value is intuitive and reasonable in practice, because the feature of individual product may slightly differ from the described feature for reasons such as quality control issue.

To calculate the integration over item's inherent feature, we apply Monte Carlo Sampling. More specifically, we sample $d$ values of $\mathbf{m}$ (i.e., $\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_d$) based on the distribution of $P(\mathbf{m}|v)$. Therefore, we can convert the calculation in Eq.(6) as follows:

$$P(y|u, do(v)) = \frac{1}{d} \sum_{j=1}^{d} \sum_{v'} P(v'|u) P(y|u, v', \mathbf{m}_j) \tag{7}$$

Real-world systems may include a large scale of items, therefore, it is unrealistic to involve all items to estimate the preference of a user-item pair. To tackle this issue, we modify the original front-door adjustment to a sample-based one that is more suitable and realistic for the recommendation scenario. Specifically, we randomly sample a set of items while calculating $P(y|u, do(v))$ as a trade-off between efficiency and deconfounding performance (more discussion and results are provided in Section 5.7). The item set for a user-item pair $(u, v)$ is represented as $R(u, v)$ (including sampled items and item $v$), and the number of sampled items is $n$, which is a hyper-parameter to be selected.

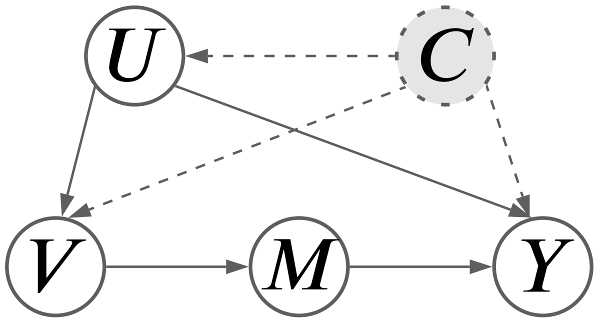


Fig. 3. The meaning of variables are: $U$ represents user, $V$ represents exposed item, $M$ represents mediator (i.e., the inherent item features), $Y$ represents user's preference, and $C$ represents unobserved confounders. Dashed nodes represent unobserved variables and dashed arrows represent causal relations pointing from unobserved variables.

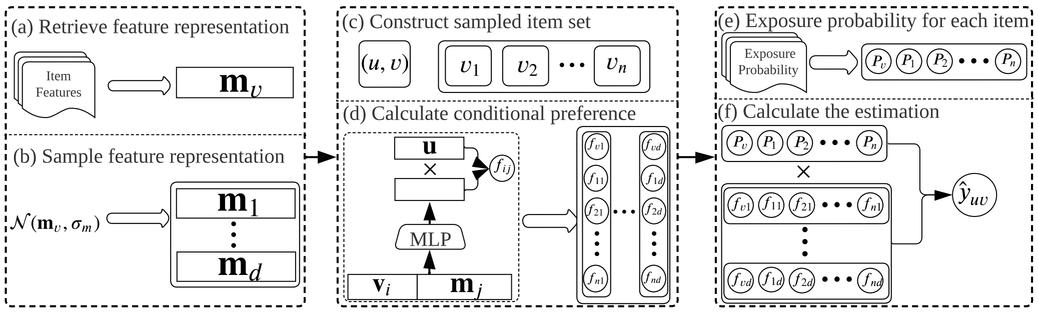

(a) Retrieve feature representation
Item Features
$\mathbf{m}_v$
(b) Sample feature representation
$\mathcal{N}(\mathbf{m}_v, \sigma_m)$
$\mathbf{m}_1$
$\mathbf{m}_d$
(c) Construct sampled item set
$(u, v)$
$v_1$ $v_2$ ⋯ $v_n$
(d) Calculate conditional preference
$\mathbf{u}$
×
$f_{ij}$
MLP
$\mathbf{v}_i$ $\mathbf{m}_j$
$f_{v1}$ $f_{11}$ $f_{21}$ $f_{n1}$
$f_{vd}$ $f_{1d}$ $f_{2d}$ $f_{nd}$
(e) Exposure probability for each item
Exposure Probability
$P_v$ $P_1$ $P_2$ ⋯ $P_n$
(f) Calculate the estimation
$P_v$ $P_1$ $P_2$ ⋯ $P_n$
×
$f_{v1}$ $f_{11}$ $f_{21}$ ⋯ $f_{n1}$
$f_{vd}$ $f_{1d}$ $f_{2d}$ ⋯ $f_{nd}$
$\hat{y}_{uv}$

Fig. 4. The process of estimating preference $\hat{y}_{uv}$. $\mathbf{m}_v$ is the feature representation for item $v$ described in the dataset; $\mathbf{m}_1 \cdots \mathbf{m}_d$ are sampled inherent item feature from $\mathcal{N}(\mathbf{m}_v, \sigma_m)$; $v_1, v_2 \cdots v_n$ are $n$ sampled items; $f_{ij}$ is the conditional preference $f(u, v_i, m_j)$ of user $u$ on item $v_i$ with inherent item feature $\mathbf{m}_j$ (Eq.(13)); $P_i$ is the exposure probability of item $v_i$ on user $u$ (i.e., the $P^e_{uv_i}$ value according to Eq.(11)); $\hat{y}_{uv}$ is the final preference score between user $u$ and item $v$ (Eq.(14)).

We can rewrite Eq.(7) into a sample-based format.

$$P(y|u, do(v)) = \frac{1}{d} \sum_{j=1}^{d} \sum_{v' \in R(u,v)} P(v'|u) P(y|u, v', \mathbf{m}_j) \tag{8}$$

For each item $v' \in R(u, v)$ in Eq.(8), its exposure probability (i.e. $P(v'|u)$) and conditional preference (i.e. $P(y|u, v', \mathbf{m}_j)$) are two required values. We will introduce how to calculate them respectively.

*4.2.1* ***Calculate the Exposure Probability.*** The casual graph and front-door adjustment enable us to expand the expression of $P(y|u, do(v))$ (as Eq.(8)). However, to obtain the value of $P(y|u, do(v))$ for recommendation, we need to calculate the value of $P(v'|u)$ as exposure probability which is a part of the expanded $P(y|u, do(v))$ expression. The exposure probability $P(v|u)$ represents the probability of an item $v$ being exposed to a certain user $u$. The implicit feedback in the dataset contains binary interactions, which indicates which user interacted with which item. If we take the interacted items as exposed (positive) samples and non-interacted items as unexposed (negative) samples (we will relax this assumption in the following), then we can train an exposure model based on pair-wise learning using the following objective:

$$\min - \sum_{u \in \mathcal{U}} \sum_{v_u^+} \sum_{v_u^-} \ln(\sigma(P^e_{uv_u^+} - P^e_{uv_u^-})) \tag{9}$$

where $\sigma(\cdot)$ is the sigmoid function: $\sigma(x) = \frac{1}{1+e^{-x}}$, $v_u^+$ and $v_u^-$ are positive and negative items for user $u$ respectively, $P^e_{uv}$ is the predicted exposure probability for user-item pair $(u, v)$, which corresponds to $P(v|u)$ in Eq.(8). More specifically, we can apply Matrix Factorization (MF) [16] to calculate $P^e_{uv}$:

$$P^e_{uv} = \mathbf{p}_u \mathbf{q}_v^T + b_u + b_v + b_g, \tag{10}$$

where $\mathbf{p}_u$ and $\mathbf{q}_v$ are embedding representations for user $u$ and item $v$, $b_u$ is the user preference offset term, $b_v$ is item preference offset term, and $b_g$ is global offset term. $\mathbf{p}_u$, $\mathbf{q}_v$, $b_u$, $b_v$, and $b_g$ are all learned in training.

Although we can obtain exposure probabilities based on Eq.(10), it ignores the bias in the observational data, that the non-interacted items may not always represent non-exposed samples. Therefore, the exposure probabilities for some non-interacted items can be underestimated, which leads to inaccurate causal estimation (i.e., Eq.(8)) and further sacrifices the performance.

To address this issue and obtain more accurate exposure probabilities, some efforts have been made, such as doubly robust model [42] and predictive model [20]. For simplicity, we apply the simple IPS-based debiasing technique [34]. Concretely, we use propensity scores to correct the predicted probability.

$$P^e_{uv} = (\mathbf{p}_u \mathbf{q}_v^T + b_u + b_v + b_g)/p_{uv} \tag{11}$$

where $p_{uv}$ is the propensity score for user-item pair $(u, v)$ (calculated as Eq.(18)). Theoretically, Eq.(11) will return more accurate and unbiased exposure probabilities [55]. We will discuss the empirical influence of different exposure models in the experiments section (Section 5.7).

*4.2.2* ***Calculate the Conditional Preference.*** To estimate the desired value of $P(y|u, do(v))$, the conditional preference $P(y|u, v', \mathbf{m}_j)$ is an essential component. Recall that $u$ represents a certain user, $v'$ represents a specific item, and $\mathbf{m}_j$ represents the representation of inherent item feature. We design a function $f : U, V, M \longrightarrow Y$ which takes users, items and inherent item features as input and returns preference scores such that $P(y|u, v', \mathbf{m}_j)$ is proportional to the corresponding conditional preference score.

$$P(y|u, v', \mathbf{m}_j) \propto f(u, v', \mathbf{m}_j) \tag{12}$$

For a triple $(u, v', \mathbf{m}_j)$, we first use a Multi-Layer Perceptron (MLP) to encode the item embedding and latent representation of inherent item features into a vector, and then use the dot product between the obtained vector and the user embedding to obtain the preference score.

$$f(u, v', \mathbf{m}) = \mathbf{u} \cdot \phi(\mathbf{W}_\ell \phi(\mathbf{W}_{\ell-1} \phi(\ldots \phi(\mathbf{W}_1 \begin{bmatrix} \mathbf{v}' \\ \mathbf{m}_j \end{bmatrix}) \ldots)))^T \tag{13}$$

where $\mathbf{u}, \mathbf{v}' \in \mathbb{R}^{D_{uv}}$ are the user and item latent embedding vectors in $D_{uv}$-dimensional space (and it is not same as $\mathbf{p}_u$ and $\mathbf{q}_v$ in Eq.(11) since $\mathbf{p}_u$ and $\mathbf{q}_v$ are used to calculate the exposure probability while $\mathbf{u}$ and $\mathbf{v}'$ are used to calculate the deconfounded preference); $\mathbf{m}_j \in \mathbb{R}^{D_m}$ is the representation of inherent item features (i.e., latent representation that integrates all inherent features of item $v$) in $D_m$-dimensional space ($D_{uv}$ and $D_m$ are not necessarily equal), which is retrieved from data; $\ell$ is the number of layers in MLP; $\mathbf{W}_i, i = 1, \ldots, \ell$ are weight matrices to be learned; and $\phi(\cdot)$ is the rectified linear unit (ReLU) activation function: $\phi(x) = \max(0, x)$.

**Algorithm 1** **D**econfounded **C**ausal **C**ollaborative **F**iltering (**DCCF**)

**Input:**
Observed interactions as user-item pair $(u, v)$ in the training data;
The inherent item feature representation $\mathbf{m}$ for all items in $D_m$-dimensional space;
L2-norm regularization weight: $\lambda_\Theta$;
The number of sampled items for each user-item pair: $n$
**Output:**
user representations $\mathbf{U}$; item representations $\mathbf{V}$; MLP network $\mathbf{W}_i$

```
1:  Train the exposure model and obtain $P(v|u)$ as Eq.(11)
2:  Random initialization of $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}_i$
3:  while not converged do
4:      for each batch in training set do
5:          $Loss \leftarrow 0$
6:          for each training sample $(u, v)$ in the batch do
7:              $R(u,v) \leftarrow v \cup$ Randomly sample $n$ items
8:              $\mathbf{m}_j|_{j=1}^{d} \leftarrow$ Sample inherent item features $d$ times following distribution $\mathcal{N}(\mathbf{m}_v, \sigma_m)$
9:              $\hat{y}_{uv} \leftarrow \frac{1}{d}\sum_{j=1}^{d}\sum_{v'\in R(u,v)} P(v'|u) f(u, v', \mathbf{m}_j)$
10:             $v^- \leftarrow$ Obtain negative sample
11:             $R(u,v^-) \leftarrow v^- \cup$ Randomly sample $n$ items
12:             $\mathbf{m}'_j|_{j=1}^{d} \leftarrow$ Sample inherent item features $d$ times following distribution $\mathcal{N}(\mathbf{m}_{v^-}, \sigma_m)$
13:             $\hat{y}_{uv^-} \leftarrow \frac{1}{d}\sum_{j=1}^{d}\sum_{v'\in R(u,v^-)} P(v'|u) f(u, v', \mathbf{m}'_j)$
14:             $Loss \leftarrow Loss + \ln(\sigma(\hat{y}_{uv} - \hat{y}_{uv^-}))$
15:         end for
16:         $Loss \leftarrow Loss + \lambda_\Theta \, ||\Theta||_2^2$
17:         Update $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}_i$
18:     end for
19: end while
20: return $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}_i$
```

*4.2.3* ***Calculate the Expectation.*** Combined Eq.(8) with Eq.(12), we have our estimation as follows:

$$P(y|u, do(v)) = \frac{1}{d}\sum_{j=1}^{d}\sum_{v'\in R(u,v)} P(v'|u) f(u, v', \mathbf{m}_j) \tag{14}$$

where $P(v'|u)$ is obtained by Eq.(11) and $f(u, v', m_v)$ is calculated by Eq.(13). The calculation of $\hat{y}_{uv}$ is shown in Figure 4.

## 4.3 Model Learning

In this section, we will introduce details about model learning, i.e., how to learn the estimation as Eq.(14). In this work, we use the pair-wise learning-to-rank algorithm [30] for training. Suppose we observe user $u$ interacted with item $v_i$ in the dataset, The estimated preference $\hat{y}_{ui}^{+}$ is obtained from Eq.(14). We sample another item $v_j \in \mathcal{V}$ that user $u$ did not interact with as a negative sample. The estimated preference for negative sample $v_j$ is represented as $\hat{y}_{uj}^{-}$. The difference between two estimated preferences is noted as $\hat{y}_{uij}$.

$$\hat{y}_{uij} = \hat{y}_{ui}^{+} - \hat{y}_{uj}^{-} \tag{15}$$

Given the observed interactions, we randomly sample a negative item for each user-item pair in implementation. Specifically, we use $\mathcal{V}_u^{+}$ to represent the observed items set for user $u$ ($v_i \in \mathcal{V}_u^{+}$)

and $\mathcal{V}_u^-$ to represent the sampled negative item set for user $u$ ($v_j \in \mathcal{V}_u^-$). The recommendation loss function can be written as:

$$\mathcal{L} = -\sum_{u\in\mathcal{U}}\sum_{v_i\in\mathcal{V}_u^+}\sum_{v_j\in\mathcal{V}_u^-} \ln(\sigma(\hat{y}_{uij})) + \lambda_\Theta \|\Theta\|_2^2 \tag{16}$$

where $\Theta$ represents all parameters of the model; $\lambda_\Theta$ is the L2-norm regularization weight; $\sigma(\cdot)$ is the sigmoid function: $\sigma(x) = \frac{1}{1+e^{-x}}$. It is worth mentioning that the representation of inherent item feature $\mathbf{m}$ and exposure probability model to calculate $P(v'|u)$ are pre-trained and fixed during the training of our model. The overall algorithm is provided in Algorithm 1.

## 4.4 Identifiability Issue

In this section, we discuss the identifiability of our estimated causal preference. Generally speaking, identifiability in causal inference indicates whether we can use observed data to estimate target values with consistency [23, 27]. From a view of causal inference in statistics, $(y, u, v', \mathbf{m}_i)$ is unobserved in the data thus the target value $P(y|u, do(v))$ (Eq.(14)) is unidentifiable. Such non-identifiability problems are common problems in recommender systems. The reason is that recommender systems are often estimated from datasets containing a very high portion of missing data. Many works [22, 34, 42] have shown that the missing data is often missed not at random (MNAR) in reality, and recent research [23] has shown that the target value is unidentifiable to any method if MNAR exists. Even if the target values are unidentifiable, in many practical intelligent systems such as recommender systems, we still need to estimate reasonably good values for such unidentifiable variables [40, 56, 58], so as to make it possible to provide practical services for users. One example is to estimate $P(y|u, v')$ where $v'$ is an item that user u never interacted with before. Real-world recommender systems usually have billions of items, but each individual user only has interacted with tens or at most hundreds of them. Even though most of the $(u, v')$ interactions cannot be observed, we still need to estimate the probability of such interactions, because only by doing so can we provide recommendations to users [40, 56, 58]. If we give up any attempt to estimate such unidentifiable variables, then many intelligent services will be impossible in practice.

In our model, the key to estimate such unidentifiable variables is through “collaborative learning”. The basic idea is that, although the target value is unobserved, there could exist some other similar examples whose values are observed, so that we can borrow their values to estimate the target value. By doing so, the examples can “collaborate” with each other so as to help estimate the values of each other, which is the key idea of “collaborative filtering” – one type of collaborative learning techniques. In our model, More specifically, given $(y, u, v, \mathbf{m}_v)$ is observed, $(y, u, v', \mathbf{m}_j)$ is indeed unobserved, but if $v$ and $v'$ share similar features, $\mathbf{m}_v$ and $\mathbf{m}_{v'}$ will also be similar representations learned by language models. Specifically, when we calculate $P(y|u, v', \mathbf{m}_j)$, considering $\mathbf{m}_j$ is similar to $\mathbf{m}_v$ (i.e., $\mathbf{m}_j$ is sampled from $\mathcal{N}(\mathbf{m}_v, \sigma_m)$), if $v'$ shares similar inherent item feature with $v$, then $\mathbf{m}_{v'}$ is also similar to $\mathbf{m}_j$, thus $P(y|u, v', \mathbf{m}_j)$ is similar to observed $P(y|u, v, \mathbf{m}_v)$. On the contrary, if $v'$ is very different from $v$, then $\mathbf{m}_{v'}$ is also different from $\mathbf{m}_j$, which will lead to $P(y|u, v', \mathbf{m}_j)$ being quite different from $P(y|u, v, \mathbf{m}_v)$. The above is a direct implication of the collaborative learning principle in representation learning techniques.

## 5 EXPERIMENTS

In this section, we conduct experiments to demonstrate the efficacy of our deconfounded recommender (DCCF) on real datasets. In particular, we aim to answer the following research questions:

- **RQ1**: How do deconfounded models perform on real-world datasets?
- **RQ2**: How does DCCF perform compared with other deconfounded models?
- **RQ3**: How does the number of sampled items influence the performance and efficiency?

| Dataset | # users | # items | # interactions | Sparsity |
|---|---|---|---|---|
| Electronics | 33602 | 16448 | 788143 | 99.86% |
| CDs and Vinyl | 6867 | 6953 | 249456 | 99.48% |
| Yelp | 21439 | 15914 | 996118 | 99.71% |

Table 2. Statistics about datasets

- **RQ4**: How different exposure models influence the performance?

We will first describe datasets, baselines and implementation details and then provide our results and analysis.

## 5.1 Data Description

*5.1.1* ***Real-world Data***. Our experiments are conducted on the Amazon Review Datasets[1] [24] and Yelp dataset [2]. Amazon Review Datasets include user, item, rating and product metadata spanning from May 1996 to October 2018. More specifically, we experiment with two categories in Amazon Review Datasets, (1) *Electronics* and (2) *CDs and Vinyl*. Yelp dataset includes user, business, rating and business information. For each dataset, we consider ratings $\geq 4$ as positive feedback (likes) and ratings $\leq 3$ as negative feedback (dislikes). We sample 70% of positive interaction as training data, 10% as validation data and the remaining 20% as testing data. The statistics about the datasets are shown in Table 2.

Some other benchmark datasets such as Movielens datasets contains the genre information as inherent feature, however, this is a categorical feature and many movies are of the same genre, and thus it is difficult to distinguish different movies using such information. Therefore, Movielens datasets are not suitable for our setting. Yahoo! R3[3] and Coat Shopping [35] are suitable datasets for evaluating unbiased or deconfounded methods since both two datasets have asked users to rate randomly exposed items. However, in our front-door adjusted method, the inherent item features (cannot be extracted from reviews or interactions) are required to estimate the desired causal preference, and the inherent item features are extracted from image or text information. Yahoo R3 only contains user interaction and rating information and Coat only contains categorical feature (similar to Movielens), and thus Yahoo!R3 and Coat are not suitable for our experiments.

Although the above two datasets (Yahoo! R3 and Coat Shopping) for evaluating deconfounded models are not suitable for our setting, we apply a commonly used splitting strategy to evaluate deconfounded models. Following [5], we apply the skewed splitting strategy (denoted as SKEW) for all datasets, which is commonly used strategy for evaluating unbiased or deconfounded model. It simulates the results of randomized experiment by exposing as uniformly as possible each user to each item in testing set. Concretely, we first calculate the popularity of each item and find the least popular item. For all other items, We use its popularity ratio compared with the least popular item to calculate the probability of being divided into the test set. For example, a 100 times more popular item than the least popular item will have a probability of being sampled in the test set as 1%. To avoid from the least popular item being not available in training, similar to [5], we set the maximum probability for a user-item interaction record to be put into the test set as 0.9. The SKEW dataset is used to test if our model can perform well on randomized experiment results. Additionally, we also apply the traditional random splitting strategy (denoted as RAND), which

[1]https://nijianmo.github.io/amazon/index.html
[2]https://www.yelp.com/dataset
[3]https://webscope.sandbox.yahoo.com/catalog.php?datatype=r

considers each data sample with equal probability to be chosen into the testing set. All data and source code will be released when paper is published.

*5.1.2* ***Synthetic Data.*** The main purposes of using synthetic data are two fold: 1) to show the effectiveness of our model on mitigating the effect of unobserved confounders because the effect of unobserved confounders may not be measured in real-world data; 2) to observe the performance if the mediator cannot "interpret" all causal effect from item to preference. We generate two synthetic datasets **SD1** and **SD2** for the two purposes, respectively. For **SD1**, we follow the data generation process described in our designed causal graph (i.e., Figure 3). More specifically, we use a Matrix Factorization model to generate interactions, and use a neural network which takes users and inherent item features as input to obtain the feedback. For **SD2**, we combine two neural networks, one takes users and inherent item features as inputs to obtain the feedback, and the other takes users and items as inputs to obtain the feedback. In order to simulate a situation where the selected mediator captures most of the causal effects from items to preference, we scale down the output of the second neural network by a small value (e.g., 0.1). For both synthetic datasets, we assume that there are 900 users and 1000 items. The inherent item feature is generated by a neural network. When generating the data for training, we include the effect of confounders (randomly generated). When generating the data for testing, we eliminate the effect of confounders to obtain an unbiased testing set. For each user, we generate 20 interactions for training and 5 interactions for testing.

## 5.2 Baselines

We introduce the baselines used in our experiments as follows.

- **MF** [30]: The matrix factorization model, which uses matrix factorization as the prediction function under the Bayesian personalized ranking framework.
- **DMF** [54]: Deep Matrix Factorization is a deep learning model for recommendation, which filters the user-item interaction matrix through a neural network architecture for prediction.
- **DecRS** [40]: A deconfounded recommendation model for alleviating bias amplification, which identifies the confounder as user historical distribution over item groups, and further applies back-door adjustment to mitigate the impact of the confounder.
- **IPW-MF** [35]: Inverse Propensity Weighting Matrix Factorization for recommendation, which adapts causal inference (inverse propensity weighting, specifically) on the matrix factorization model to handle the selection bias in observational data.
- **DCF** [44]: A deconfounded recommendation model, which uses an exposure model to construct a substitute confounder and then conditions on the substitute confounder for modeling.
- **HCR** [59]: A deconfounded recommendation model, which considers click behavior as mediator and designs a multi-task learning framework that simultaneously learns the probability of click and post-click behavior to mitigate confounding effect.
- **DCCF_NS**: This is a no-item-sampling variant of our model. Specifically, for a user-item pair $(u, v)$, this variant only uses $P(v|u)f(u, v, m)$ in Eq.(13) as the estimated preference. It can be considered as a reweighting method.
- **DCCF_ND**: This is a no-deconfounding (ND) variant of our model. Specifically, for a user-item pair $(u, v)$, this variant only uses $f(u, v, m)$ in Eq.(13) as the estimated preference. It can be considered as our model (DCCF) without the front-door adjustment, which loses the deconfounding ability.

The baselines include both classical association-based recommendation models (MF and DMF) and state-of-the-art deconfounded recommendation models (DecRS, IPW-MF, DCF and HCR), as well as two ablated variants of our own model (DCCF_NS and DCCF_ND). Many recent deconfounded methods such as DecRS mainly focus on mitigating one specific confounder, while our model is used to mitigate the effect of unobserved confounders. Some other deconfounded methods require certain user information such as user social networks [8], which is not suitable as a baseline for our problem setting. Finally, we use **DCCF** to denote the complete version of our model.

## 5.3 Evaluation Metrics

Our model aims to estimate the accurate preference with the existence of unobserved confounders. To evaluate the deconfounding performance, some deconfounded models, which mainly focus on one specific confounder, group items or users by the value of the confounder and compare the performance among groups. For example, Zhang et al. [58] treat item popularity as the confounder, thus the deconfounding performance is evaluated by grouping items by item popularity and calculating the recommendation ratio for each group. In our case, however, it is impossible to group users or items by unobserved confounders.

Since the deconfounded recommendation will obtain more accurate estimation of preference to improve the recommendation performance [44], we evaluate our model and baselines on the top-$K$ recommendation task. We adopt three widely used metrics to evaluate recommendation performance including: *Normalized Discounted Cumulative Gain@K* (nDCG@K for short), *Recall@K* (Rec@K for short), and *Precision@K* (Pre@K for short). The calculation of the three metrics is given as follows:

$$\begin{aligned} nDCG@K &= \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\frac{DCG_u@K}{IDCG_u@K} \\ Rec@K &= \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\frac{|TP_u|}{|TP_u|+|FN_u|} \\ Pre@K &= \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\frac{|TP_u|}{|TP_u|+|FP_u|} \end{aligned} \tag{17}$$

where $DCG_u@K$ is the Discounted Cumulative Gain at position $K$ for user $u$, $IDCG_u@K$ is the Ideal Discounted Cumulative Gain through position $K$ for user $u$, $|TP_u|$ is the number of recommended items that are relevant for user $u$ at position $K$, $|TP_u| + |FN_u|$ is the total number of relevant items for user $u$, and $|TP_u| + |FP_u|$ is the total number of recommended items at position $K$ for user $u$. In our experiment, we refer to [44] and set $K = 5$.

## 5.4 Implementation Details

We use the same training, validation and testing sets for all models, including our models and baseline models. For evaluation, we apply *real-plus-N* [32] to calculate the values of each metric. Concretely, for each user in the validation and testing set, we randomly sample 1000 negative items for ranking evaluation in real-world data and 100 negative items for synthetic data. All recommendation models adopt the Bayesian Personalized Ranking (BPR) [30] framework for pair-wise learning. During the training, we sample one negative item that the user did not interact with for each interacted user-item pair. We optimize the models using mini-batch Adam with a batch size of 128.

To obtain the inherent feature representation of the items (i.e., $\mathbf{m}_v$ in Eq.(13)), we apply a pre-trained transformer-based sentence embedding model[4] to represent the textual information of an item into a dense vector embedding with dimension 768. For the Amazon Review Datasets, the textual information comes from the 'title', 'feature', and 'literal description' of each item, which

[4]we apply the pre-trained paraphrase-distilroberta-base-v1 sentence embedding model in a public transformer implementation: https://github.com/UKPLab/sentence-transformers

are included in the meta-data of the amazon review dataset. For the Yelp dataset, we include the 'name', 'address', 'city', 'state', 'attributes', and 'categories' of the business into the inherent feature, which can be found in the business information in the Yelp dataset. We use the above pre-trained sentence transformer to convert the textual information into separate embeddings and further take the average as the inherent feature representation of the item.

For the hyper-parameters, we perform the grid-search and search the user/item embedding dimension from {32,64,128}, the structure of the neural network is a two-layer MLP with dimension 64 for deep models. For all recommendation models, we search the learning rate from {0.0005,0.001,0.003,0.005}, the $\ell_2$-regularization weight is chosen from {1e-3, 1e-4, 1e-5}. The total number of epoches is set to 100. For a fair comparison, we tune the parameters to the best performance for each model on the validation data based on nDCG@5. For our model, we sample 20 items for each user-item pair for calculating the expectation (i.e., the number of sampled items $n$ in Section 4.2) and sample 2 inherent item feature values for calculating the integration (i.e., the number of sampled item features $d$ in Section 4.2). We set the variance of the distribution of inherent item feature as 0.1 (i.e., $\sigma_m$ in Section 4.2). To obtain the exposure probability calculated based on Eq.(11), we estimate the user independent propensity scores as [34]:

$$p_{*v} = \left( \frac{\sum_{u \in \mathcal{U}} y_{uv}}{\max_{v' \in \mathcal{V}} \sum_{u \in \mathcal{U}} y_{uv'}} \right)^{\eta} \tag{18}$$

Here $y_{uv}$ is an indicator: $y_{uv} = 1$ if the user-item pair $(u, v)$ is observed in the dataset, $y_{uv} = 0$ otherwise. We set $\eta = 0.5$ in the experiment. The running time of our model is about 40 seconds/epoch for *Electronics*, 15 seconds/epoch for *CDs and Vinyl* and 35 seconds/epoch for *Yelp*.

## 5.5 Results and Discussion

In this section, we aim to answer **RQ1** and **RQ2**. For real-world data, the recommendation performance for models on the SKEW splitting datasets are shown in Table 3. The SKEW splitting strategy simulates a test set as a result of randomized experiment, and the test distribution is different from the training distribution. The recommendation performance on the RAND splitting datasets are shown in Table 4. The RAND splitting strategy is the regular splitting strategy that randomly splits data samples into testing set, thus the training and testing distributions are constant. Comparing the RAND and SKEW splitting strategies, we can observe that the recommendation performance with the RAND splitting strategy is consistently better than the performance with the SKEW splitting strategy on all three datasets. This shows that learning recommendation models when the training and testing distributions are different (SKEW) is a more challenging task than under the same train-test distribution (RAND).

The following observations are consistent no matter what the splitting strategy is. First, the recommendation models shown in Table 3 and Table 4 can be categorized into classic association-based models and deconfounded models, where the difference between the two is whether the effects of confounders are mitigated or not. From the results, we can see that the deconfounded recommendation models (DecRS, IPW-MF, DCF, HCR and DCCF) achieve better recommendation performance than classic recommendation models (MF and DMF) in most cases on all three datasets. When averaging across all deconfounded recommendation models on all datasets using the SKEW splitting strategies, the deconfounded models achieve 54.3% relative improvement on nDCG@5 than classic recommendation models, 55.0% on Recall@5, and 51.7% on Precision@5. When averaging across all deconfounded recommendation models on all datasets using the RAND splitting strategies, the deconfounded models achieve 4.9% relative improvement on nDCG@5, 2.9% on Recall@5, and 8.3% on Precision@5. According to this comparison, we can observe that the existence of confounders will lead to inaccurate recommendations and deconfounded recommendation

| Methods | Electronics | | | CDs and Vinyl | | | Yelp | | |
|---|---|---|---|---|---|---|---|---|---|
| | nDCG@5 | Rec@5 | Pre@5 | nDCG@5 | Rec@5 | Pre@5 | nDCG@5 | Rec@5 | Pre@5 |
| MF | 0.0201 | 0.0185 | 0.0170 | 0.1567 | 0.1158 | 0.1344 | 0.1013 | 0.0601 | 0.0980 |
| DMF | 0.0253 | 0.0237 | 0.0205 | 0.1363 | 0.1011 | 0.1164 | 0.1107 | 0.0666 | 0.1065 |
| DecRS | 0.0687 | 0.0652 | 0.0549 | 0.2046 | 0.1550 | 0.1697 | 0.1108 | 0.0676 | 0.1079 |
| IPW-MF | 0.0447 | 0.0398 | 0.0345 | 0.1959 | 0.1434 | 0.1622 | 0.1099 | 0.0666 | 0.1071 |
| DCF | 0.0317 | 0.0308 | 0.0257 | 0.1573 | 0.1162 | 0.1352 | 0.1115 | 0.0679 | 0.1083 |
| HCR | 0.0790 | 0.0751 | 0.0626 | 0.2033 | 0.1548 | 0.1707 | 0.1123 | 0.0680 | 0.1091 |
| DCCF_NS | 0.0429 | 0.0413 | 0.0350 | 0.1581 | 0.1168 | 0.1330 | 0.1087 | 0.0660 | 0.1058 |
| DCCF_ND | 0.0413 | 0.0401 | 0.0329 | 0.1427 | 0.1078 | 0.1210 | 0.0991 | 0.0598 | 0.0961 |
| DCCF (ours) | **0.0934** | **0.0880** | **0.0737** | **0.2289** | **0.1702** | **0.1892** | **0.1184** | **0.0718** | **0.1143** |

Table 3. Recommendation performance for *Electronics*, *CDs and Vinyl*, and *Yelp* using the SKEW splitting strategy. We evaluate on ranking task with metrics nDCG@5, Recall@5 and Precision@5. The best results are highlighted in bold. The improvements are significant at $p < 0.01$.

| Methods | Electronics | | | CDs and Vinyl | | | Yelp | | |
|---|---|---|---|---|---|---|---|---|---|
| | nDCG@5 | Rec@5 | Pre@5 | nDCG@5 | Rec@5 | Pre@5 | nDCG@5 | Rec@5 | Pre@5 |
| MF | 0.1463 | 0.1454 | 0.1027 | 0.2891 | 0.2168 | 0.2382 | 0.2980 | 0.1776 | 0.2734 |
| DMF | 0.0948 | 0.0956 | 0.0681 | 0.1618 | 0.1208 | 0.1395 | 0.2483 | 0.1523 | 0.2355 |
| DecRS | 0.1589 | 0.1431 | 0.1200 | 0.3093 | 0.2361 | 0.2536 | 0.3055 | 0.1846 | 0.2819 |
| IPW-MF | 0.1502 | 0.1385 | 0.1135 | 0.2954 | 0.2197 | 0.2553 | 0.3034 | 0.1832 | 0.2802 |
| DCF | 0.1463 | 0.1340 | 0.1124 | 0.2885 | 0.2175 | 0.2389 | 0.2995 | 0.1817 | 0.2772 |
| HCR | 0.1613 | 0.1487 | 0.1224 | 0.3059 | 0.2320 | 0.2495 | 0.3045 | 0.1842 | 0.2804 |
| DCCF_NS | 0.1542 | 0.1419 | 0.1171 | 0.2886 | 0.2164 | 0.2360 | 0.2999 | 0.1813 | 0.2766 |
| DCCF_ND | 0.1387 | 0.1276 | 0.1045 | 0.2247 | 0.1692 | 0.1882 | 0.2887 | 0.1737 | 0.2665 |
| DCCF (ours) | **0.1742** | **0.1598** | **0.1312** | **0.3237** | **0.2471** | **0.2643** | **0.3073** | **0.1861** | **0.2835** |

Table 4. Recommendation performance for *Electronics*, *CDs and Vinyl*, and *Yelp* using the RAND splitting strategy. We evaluate on ranking task with metrics nDCG@5, Recall@5 and Precision@5. The best results are highlighted in bold. The improvements are significant at $p < 0.01$.

models will improve the recommendation performance by mitigating the effect of confounders. Meanwhile, comparing the improvement on both splitting strategies, the improvement brought by the deconfounded models is more significant on the SKEW datasets than the RAND datasets.

Among deconfounded recommendation models, we can see that our DCCF model achieves the best recommendation performance in most cases. Compared with the strongest deconfounded model in baseline, when averaging across all datasets using the SKEW splitting strategies, our model yields 18.2% improvement on nDCG@5, 16.9% improvement on Recall@5, and 17.2% improvement on Precision@5. When averaging across all datasets using the RAND splitting strategies, our model yields 4.9% improvement on nDCG@5, 5.7% improvement on Recall@5, and 4.4% improvement on Precision@5. These observations imply that by applying the front-door adjustment into the estimation of user's preference, our model is capable of reducing the effect of unobserved confounders and further improving the recommendation performance. Moreover, based on the results, applying the front-door adjustment is more effective than other deconfounded models.

Although the front-door adjustment is effective on reducing the effect of unobserved confounders, it requires inherent item features. We need to confirm that the improved performance is indeed brought by the deconfounding effect instead of the use of inherent item features. Therefore, we

| Methods | SD1 | | | SD2 | | |
|---|---|---|---|---|---|---|
| | nDCG@5 | Rec@5 | Pre@5 | nDCG@5 | Rec@5 | Pre@5 |
| MF | 0.1201 | 0.0833 | 0.1023 | 0.1069 | 0.0654 | 0.1167 |
| DMF | 0.1134 | 0.0837 | 0.0972 | 0.1090 | 0.0663 | 0.1195 |
| DecRS | 0.1219 | 0.0888 | 0.1030 | 0.1138 | 0.0973 | 0.1235 |
| IPW-MF | 0.1224 | 0.0899 | 0.1034 | 0.1176 | 0.0715 | 0.1276 |
| DCF | 0.1287 | 0.0913 | 0.1127 | 0.1254 | 0.0751 | 0.1361 |
| HCR | 0.1392 | 0.1000 | 0.1186 | 0.1334 | 0.0787 | 0.1424 |
| DCCF_NS | 0.1212 | 0.0878 | 0.1066 | 0.0988 | 0.0602 | 0.1072 |
| DCCF_ND | 0.1025 | 0.0748 | 0.0880 | 0.0844 | 0.0521 | 0.0923 |
| DCCF (ours) | **0.1438** | **0.1066** | **0.1216** | **0.1358** | **0.0807** | **0.1466** |

Table 5. Recommendation performance for two synthetic datasets **SD1** and **SD2**. We evaluate on ranking task with metrics nDCG@5, Recall@5 and Precision@5. The best results are highlighted in bold. The improvements are significant at $p < 0.01$.

consider a variant of our model DCCF_ND, which also involves inherent item feature representation into estimation but without the front-door adjustment as deconfounding component. More specifically, unlike DCCF that applies the front-door adjustment with weighted sum over sampled items to obtain the estimation of a user-item pair, DCCF_ND only considers the user-item pair itself when calculating the estimation. From the performance in Table 3 and Table 4, we can see that DCCF_ND is even worse than classic models in many cases. Therefore, the improvement of our model is brought by deconfounding with the front-door adjustment instead of involving inherent item feature representation into calculation. We also include a variant of our model DCCF_NS, which involves exposure probability but without sampled items. It can be considered as a reweighting method, instead of applying the front-door adjustment. From the performance in Table 3 and Table 4, we can see that DCCF_NS is able to improve recommendation performance compared with classic models, and even achieve better performance than deconfounded models in some cases. However, it cannot obtain better performance than our model. Therefore, applying front-door adjustment is more effective than using exposure probability for reweighting.

For synthetic datasets, the recommendation performance for models are shown in Table 5. Dataset **SD1** simulates a scenario where the selected inherent item features completely intercept the causal effect from items to preference, while **SD2** represents a situation where the mediator intercepts a significant portion of the causal effect from items to preference. We can observe that deconfounded models still achieve better performance than classical models. Among different deconfounded models, DecRS and IPW-MF mainly focus on mitigating the effect of item popularity, therefore, they can only achieve lightly improved performance since the confounders in synthetic data are not necessary related to item popularity. The remaining deconfounded models (i.e., DCF, HCR and DCCF) are not restricted to a specific confounder, thus are capable of achieving better deconfounding performance. Our DCCF model achieves the best performance on both synthetic datasets, which illustrate the effectiveness of our model on mitigating the effect of unobserved confounders, even when the mediator cannot fully intercept the causal effect from items to preference.

## 5.6 Influence of the Number of Sampled Items

In this section, we will discuss the influence of the number of sampled items to answer **RQ3**. As we mentioned in Section 4.2, instead of strictly following the front-door adjustment as Eq.(7), we randomly sample a set of items and estimate the preference by the sample-based front-door adjustment (i.e., as Eq.(8)).

| Methods | Electronics | | | CDs and Vinyl | | | Yelp | | |
|---|---|---|---|---|---|---|---|---|---|
| | nDCG@5 | Rec@5 | Pre@5 | nDCG@5 | Rec@5 | Pre@5 | nDCG@5 | Rec@5 | Pre@5 |
| DCCF_Random | 0.0340 | 0.0333 | 0.0272 | 0.0831 | 0.0613 | 0.0714 | 0.0722 | 0.0418 | 0.0695 |
| DCCF_Uniform | 0.0338 | 0.0334 | 0.0272 | 0.0853 | 0.0634 | 0.0723 | 0.0717 | 0.0411 | 0.0691 |
| DCCF_Bias | 0.0687 | 0.0652 | 0.0549 | 0.2053 | 0.1535 | 0.1700 | 0.1106 | 0.0666 | 0.1071 |
| DCCF_Unbias | **0.0846** | **0.0814** | **0.0670** | **0.2167** | **0.1654** | **0.1806** | **0.1124** | **0.0682** | **0.1090** |

Table 6. The recommendation performance of DCCF under different exposure models on three datasets using the SKEW splitting strategy. We report the evaluatation on ranking task with metrics nDCG@5, Recall@5 and Precision@5. The best results are highlighted in bold.

| Methods | Electronics | | | CDs and Vinyl | | | Yelp | | |
|---|---|---|---|---|---|---|---|---|---|
| | nDCG@5 | Rec@5 | Pre@5 | nDCG@5 | Rec@5 | Pre@5 | nDCG@5 | Rec@5 | Pre@5 |
| DCCF_Random | 0.0609 | 0.0581 | 0.0469 | 0.1194 | 0.0867 | 0.1008 | 0.1332 | 0.0799 | 0.1277 |
| DCCF_Uniform | 0.0627 | 0.0598 | 0.0484 | 0.1199 | 0.0862 | 0.1005 | 0.1428 | 0.0860 | 0.1362 |
| DCCF_Bias | 0.1510 | 0.1362 | 0.1139 | 0.2886 | 0.2164 | 0.2360 | 0.2876 | 0.1743 | 0.2673 |
| DCCF_Unbias | **0.1583** | **0.1422** | **0.1194** | **0.2939** | **0.2290** | **0.2428** | **0.2958** | **0.1786** | **0.2733** |

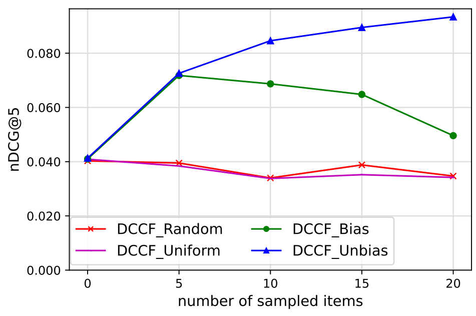
Table 7. The recommendation performance of DCCF under different exposure models on three datasets using the RAND splitting strategy. We report the evaluatation on ranking task with metrics nDCG@5, Recall@5 and Precision@5. The best results are highlighted in bold.

To discuss how the number of sampled items influence the recommendation performance, in Figure 5, we plot nDCG@5 for DCCF (i.e., DCCF_Unbias in Figure 5) with different numbers of sampled items. We can observe that the recommendation performance increases with the increasing of the number of sampled items, but the magnitude of the increment decreases as the number of sampled items increases. Considering the model applies a two layer MLP with dimension $D_{uv}$, the time complexity of calculating Eq.(13) would be $(D_{uv} + D_m)D_{uv} + 2D_{uv}^2$. Suppose $D_m$ is linear to $D_{uv}$, the time complexity is $O(D_{uv}^2)$. For a fixed number of sampled inherent item feature $d$ ($d \ll D_{uv}$), if the number of sampled item $n$ is small enough (i.e., $n \ll D_{uv}$), the time complexity of calculating $P(y|u, do(v))$ is $O(D_{uv}^2)$. However, if $n$ is large (i.e., linear to $D_{uv}$), the time complexity would be $O(D_{uv}^3)$. Additionally, larger $n$ will occupy more memory during the training. Therefore, it is important to choose an appropriate value of $n$ to balance the recommendation performance and efficiency when applying the DCCF model.

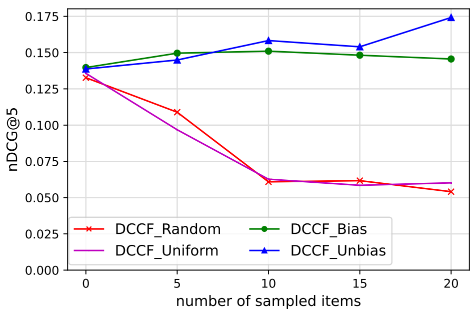
## 5.7 Influence of Exposure Models

In this section, we will discuss the influence of the exposure models to answer **RQ4**. As we mentioned before, the exposure probability, which is required by the front-door adjustment but not available in the feedback data, is an essential component of our model. To empirically show the influence of it, we design three versions of our model with different exposure probabilities. The only difference among them is how to obtain the exposure probability.

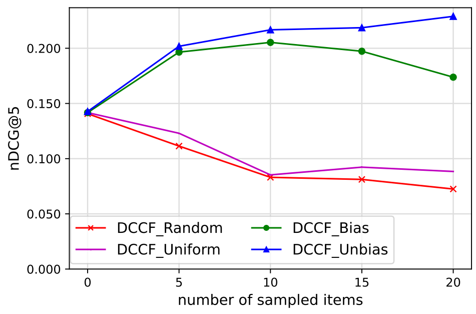
- **DCCF_Random**: In this model, the exposure probabilities are not learned from the data, instead, we randomly generate a matrix to represent the exposure probability.
- **DCCF_Uniform**: In this model, we assume that each item has an equal probability to be exposed for each user.
- **DCCF_Bias**: In this model, we estimate the exposure probability as Eq.(10), and use the pair-wise learning method to train the model based on implicit feedback.



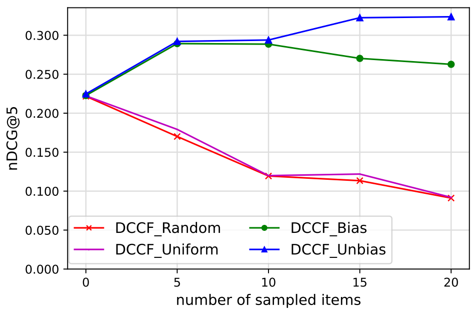
(a) Electronics (SKEW)



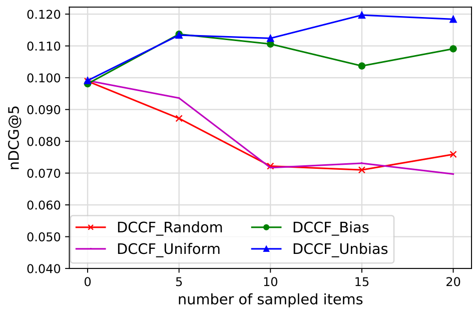
(b) Electronics (RAND)



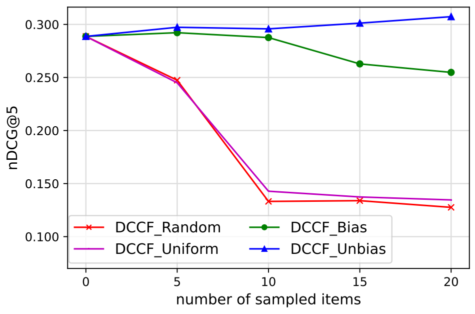
(c) CDs and Vinyl (SKEW)



(d) CDs and Vinyl (RAND)



(e) Yelp (SKEW)



(f) Yelp (RAND)

Fig. 5. nDCG@5 for different exposure probability models under different numbers of sampled items.

- **DCCF_Unbias**: This model is the same as **DCCF** in Table 3 and Table 4. Concretely, we consider the bias in the implicit feedback data, and apply Eq.(11) to obtain unbiased estimation of exposure probabilities.

Following the setting in main experiment, we evaluate three datasets using two splitting strategies on ranking task with metrics nDCG@5, Recall@5 and Precision@5. The recommendation performance on the SKEW splitting strategy is shown in Table 6 and the performance on the RAND splitting strategy is shown in Table 7. For each of the version, we use the same representation of inherent item features and sample 10 items for each pair to calculate the estimated preference.

From the results, we can see that DCCF_Unbias achieves the best performance; DCCF_Random and DCCF_Uniform have the worst performance (DCCF_Random and DCCF_Uniform get similar performance) in most cases. This observation is consistent on both RAND and SKEW splitting strategies. Comparing the four versions with different exposure models shows that accurate exposure probability will lead to accurate estimation under the front-door adjustment. Specifically, the exposure probability is not observed in the feedback data, therefore, we need to learn a model to estimate it from observational feedback data. DCCF_Unbias uses the unbiased estimation and gets the most accurate exposure probability, while DCCF_Random randomly generates exposure probabilities and DCCF_Uniform applyies uniform probabilities, which are irrelevant with the feedback data, and thus will get the least accurate probability. We can see that the ranking of the exposure probability accuracy matches the ranking of recommendation performance.

The exposure model affects our model in several ways, i.e., not only the best recommendation performance, but also the trend of the recommendation performance as the number of sampled item increases. As we mentioned in Section 4 and Eq.(14), we apply the front-door adjustment over a set of sampled items to make it suitable for the recommendation scenario. Ideally, if there exist the ground truth values of exposure probability and we apply them into the calculation, then increasing the number of sampled items may reduce the efficiency but will at least not hurt the performance significantly.

In order to investigate how our model changes with increasing the number of sampled items under different exposure models, we plot nDCG@5 for four versions of the model with different numbers of sampled items, as shown in Figure 5. As we can see from the figure, the performance of DCCF_Unbias increases with the increasing of sample number, the performance of DCCF_Bias increases first and then drops after 5 samples, and the performance of DCCF_Random and DCCF_Uniform is monotonically decreasing. Meanwhile, for a given number of sampled items, the performance among the four models follows the conclusion as Table 6 and Table 7 in most cases, i.e., DCCF_Unbias > DCCF_Bias > DCCF_Random/DCCF_Uniform.

For DCCF_Unbias, since it has the most accurate exposure probability, it will achieve better performance with more sampled items. For DCCF_Bias, the probability is also learned from data as DCCF_Unbias but not accurate enough due to ignoring the bias. Therefore, when the number of sampled items is small, the model can get slightly better performance with the help of sampled items. However, when more sampled items are involved into the estimation, inaccurate probabilities may mislead front-door adjustment and hurt the performance, thus resulting in performance drops. For DCCF_Random/DCCF_Uniform, the sampled items do not provide useful information but instead introduce noises that completely mislead the front-door adjustment, and thus hurt the performance. In summary, accurate exposure probability will improve the performance while inaccurate probability hurts the performance. Therefore, it is important to adopt an accurate exposure model to obtain better performance.

## 6 CONCLUSIONS AND FUTURE WORK

In this paper, we notice that the unobserved confounders may result in inaccurate recommendations. To solve this problem, we propose a deconfounded causal collaborative filtering (DCCF) model based on front-door adjustment. Specifically, we first design a causal graph to depict user behaviors with unobserved confounders. We then apply the front-door adjustment to design the model for mitigating the influence of unobserved confounders. Considering the large scale of items in real-world system, we use the sample-based front-door adjustment to calculate the deconfounded estimation of users' preference. Experiments on real-world datasets with both skewed splitting and random splitting show that our deconfounded model can outperform both classical association-based models and deconfounded recommendation models. Furthermore, the experiments on the influence of exposure models show that the performance of our DCCF model is affected by the accuracy of the learned exposure probabilities. Therefore, a proper exposure model helps improve the recommendation performance of our deconfounded recommendation model.

**Limitations and future work**. Our model treats all confounders as bias terms and removes all of their influence from the recommender system. However, the confounders may not always be harmful from the provider side. In practice, certain confounders can be useful and their effects should be remained in the recommendation. This requires a mixed recommendation strategy that considers partially associative rules and partially causal rules. The causal model used in DCCF could also be adapted to increase the interpretability of the recommender systems, as another important aspect to be considered for recommendation.

## 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] Himan Abdollahpouri and Masoud Mansoury. 2020. Multi-sided exposure bias in recommendation. *arXiv preprint arXiv:2006.15772* (2020).

[2] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. 2020. The connection between popularity bias, calibration, and fairness in recommendation. In *Fourteenth ACM Conference on Recommender Systems*. 726–731.

[3] Aman Agarwal, Ivan Zaitsev, Xuanhui Wang, Cheng Li, Marc Najork, and Thorsten Joachims. 2019. Estimating position bias without intrusive interventions. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 474–482.

[4] Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W Bruce Croft. 2018. Unbiased learning to rank with unbiased propensity estimation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 385–394.

[5] Stephen Bonner and Flavian Vasile. 2018. Causal embeddings for recommendation. In *Proceedings of the 12th ACM conference on recommender systems*. 104–112.

[6] Xu Chen, Zhenlei Wang, Hongteng Xu, Jingsen Zhang, Yongfeng Zhang, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Data Augmented Sequential Recommendation based on Counterfactual Thinking. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[7] Ziheng Chen, Fabrizio Silvestri, Jia Wang, Yongfeng Zhang, and Gabriele Tolomei. 2023. The Dark Side of Explanations: Poisoning Recommender Systems with Counterfactual Examples. In *Proceedings of the 46th International ACM SIGIR conference on Research and Development in Information Retrieval*.

[8] Junruo Gao, Mengyue Yang, Yuyang Liu, and Jun Li. 2021. Deconfounding Representation Learning Based on User Interactions in Recommendation Systems.. In *PAKDD (2)*. Springer, 588–599.

[9] Yingqiang Ge, Shuchang Liu, Ruoyuan Gao, Yikun Xian, Yunqi Li, Xiangyu Zhao, Changhua Pei, Fei Sun, Junfeng Ge, Wenwu Ou, et al. 2021. Towards Long-term Fairness in Recommendation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 445–453.

[10] Yingqiang Ge, Shuchang Liu, Zelong Li, Shuyuan Xu, Shijie Geng, Yunqi Li, Juntao Tan, Fei Sun, and Yongfeng Zhang. 2021. Counterfactual evaluation for explainable ai. *arXiv:2109.01962* (2021).

[11] Ruocheng Guo, Xiaoting Zhao, Adam Henderson, Liangjie Hong, and Huan Liu. 2020. Debiasing grid-based product search in e-commerce. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2852–2860.

[12] Shantanu Gupta, Hao Wang, Zachary C Lipton, and Yuyang Wang. 2021. Correcting Exposure Bias for Link Recommendation. *ICML* (2021).

[13] Ziniu Hu, Yang Wang, Qu Peng, and Hang Li. 2019. Unbiased LambdaMART: An unbiased pairwise learning-to-rank algorithm. In *The World Wide Web Conference*. 2830–2836.

[14] Jianchao Ji, Zelong Li, Shuyuan Xu, Max Xiong, Juntao Tan, Yingqiang Ge, Hao Wang, and Yongfeng Zhang. 2023. Counterfactual Collaborative Reasoning. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 249–257.

[15] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased learning-to-rank with biased feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. 781–789.

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

[17] Adit Krishnan, Ashish Sharma, Aravind Sankar, and Hari Sundaram. 2018. An adversarial approach to improve long-tail performance in neural collaborative filtering. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 1491–1494.

[18] Yunqi Li, Hanxiong Chen, Juntao Tan, and Yongfeng Zhang. 2022. Causal factorization machine for robust recommendation. In *Proceedings of the 22nd ACM/IEEE Joint Conference on Digital Libraries*. 1–9.

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

[20] Benjamin M Marlin and Richard S Zemel. 2009. Collaborative prediction and ranking with non-random missing data. In *Proceedings of the third ACM conference on Recommender systems*. 5–12.

[21] Benjamin M. Marlin, Richard S. Zemel, Sam Roweis, and Malcolm Slaney. 2007. Collaborative Filtering and the Missing at Random Assumption. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence (UAI'07)*. 267–275.

[22] Benjamin M Marlin, Richard S Zemel, Sam T Roweis, and Malcolm Slaney. 2011. Recommender systems: missing data and statistical model estimation. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

[23] Karthika Mohan and Judea Pearl. 2021. Graphical models for processing missing data. *J. Amer. Statist. Assoc.* (2021), 1–16.

[24] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 188–197.

[25] 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.

[26] Zohreh Ovaisi, Ragib Ahsan, Yifan Zhang, Kathryn Vasilaky, and Elena Zheleva. 2020. Correcting for selection bias in learning-to-rank systems. In *Proceedings of The Web Conference 2020*. 1863–1873.

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

[28] Zhen Qin, Suming J Chen, Donald Metzler, Yongwoo Noh, Jingzheng Qin, and Xuanhui Wang. 2020. Attribute-based propensity for unbiased learning in recommender systems: Algorithm and case studies. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2359–2367.

[29] Ruihong Qiu, Sen Wang, Zhi Chen, Hongzhi Yin, and Zi Huang. 2021. CausalRec: Causal Inference for Visual Debiasing in Visually-Aware Recommendation. *arXiv preprint arXiv:2107.02390* (2021).

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

[31] Paul R Rosenbaum and Donald B Rubin. 1983. The central role of the propensity score in observational studies for causal effects. *Biometrika* 70, 1 (1983), 41–55.

[32] Alan Said and Alejandro Bellogín. 2014. Comparative recommender system evaluation: benchmarking recommendation frameworks. In *Proceedings of the 8th ACM Conference on Recommender systems*. 129–136.

[33] Yuta Saito. 2020. Asymmetric Tri-training for Debiasing Missing-Not-At-Random Explicit Feedback. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 309–318.

[34] Yuta Saito, Suguru Yaginuma, Yuta Nishino, Hayato Sakata, and Kazuhide Nakata. 2020. Unbiased recommender learning from missing-not-at-random implicit feedback. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 501–509.

[35] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as treatments: Debiasing learning and evaluation. In *international conference on machine learning*. PMLR.

[36] Wenjie Shang, Yang Yu, Qingyang Li, Zhiwei Qin, Yiping Meng, and Jieping Ye. 2019. Environment reconstruction with hidden confounders for reinforcement learning based recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 566–576.

[37] Juntao Tan, Shijie Geng, Zuohui Fu, Yingqiang Ge, Shuyuan Xu, Yunqi Li, and Yongfeng Zhang. 2022. Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning. In *Proceedings of the ACM Web Conference 2022*. 1018–1027.

[38] Juntao Tan, Shuyuan Xu, Yingqiang Ge, Yunqi Li, Xu Chen, and Yongfeng Zhang. 2021. Counterfactual explainable recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1784–1793.

[39] Ali Vardasbi, Maarten de Rijke, and Ilya Markov. 2020. Cascade model-based propensity estimation for counterfactual learning to rank. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2089–2092.

[40] Wenjie Wang, Fuli Feng, Xiangnan He, Xiang Wang, and Tat-Seng Chua. 2021. Deconfounded Recommendation for Alleviating Bias Amplification. In *KDD*. ACM.

[41] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. Learning to rank with selection bias in personal search. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 115–124.

[42] 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.

[43] Yixin Wang and David M Blei. 2019. The blessings of multiple causes. *J. Amer. Statist. Assoc.* 114, 528 (2019), 1574–1596.

[44] 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.

[45] Zhenlei Wang, Shiqi Shen, Zhipeng Wang, Bo Chen, Xu Chen, and Ji-Rong Wen. 2022. Unbiased Sequential Recommendation with Latent Confounders. In *Proceedings of the ACM Web Conference 2022*. 2195–2204.

[46] 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.

[47] Tianxin Wei, Fuli Feng, Jiawei Chen, Chufeng Shi, Ziwei Wu, Jinfeng Yi, and Xiangnan He. 2021. Model-Agnostic Counterfactual Reasoning for Eliminating Popularity Bias in Recommender System. *KDD* (2021).

[48] Xinwei Wu, Hechang Chen, Jiashu Zhao, Li He, Dawei Yin, and Yi Chang. 2021. Unbiased learning to rank in feeds recommendation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 490–498.

[49] Kun Xiong, Wenwen Ye, Xu Chen, Yongfeng Zhang, Wayne Xin Zhao, Binbin Hu, Zhiqiang Zhang, and Jun Zhou. 2021. Counterfactual review-based recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2231–2240.

[50] Shuyuan Xu, Yingqiang Ge, Yunqi Li, Zuohui Fu, Xu Chen, and Yongfeng Zhang. 2023. Causal Collaborative Filtering. In *Proceedings of The 9th ACM SIGIR / The 13th International Conference on the Theory of Information Retrieval*.

[51] Shuyuan Xu, Jianchao Ji, Yunqi Li, Yingqiang Ge, Juntao Tan, and Yongfeng Zhang. 2023. Causal Inference for Recommendation: Foundations, Methods and Applications. *arXiv:2301.04016* (2023).

[52] Shuyuan Xu, Juntao Tan, Zuohui Fu, Jianchao Ji, Shelby Heinecke, and Yongfeng Zhang. 2022. Dynamic causal collaborative filtering. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2301–2310.

[53] Shuyuan Xu, Da Xu, Evren Korpeoglu, Sushant Kumar, Stephen Guo, Kannan Achan, and Yongfeng Zhang. 2022. Causal Structure Learning with Recommendation System. *arXiv:2210.10256* (2022).

[54] 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.

[55] 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.

[56] Mengyue Yang, Quanyu Dai, Zhenhua Dong, Xu Chen, Xiuqiang He, and Jun Wang. 2021. Top-N Recommendation with Counterfactual User Preference Simulation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2342–2351.

[57] 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 *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 329–338.

[58] 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 *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21)*.

[59] Xinyuan Zhu, Yang Zhang, Fuli Feng, Xun Yang, Dingxian Wang, and Xiangnan He. 2022. Mitigating Hidden Confounding Effects for Causal Recommendation. *arXiv preprint arXiv:2205.07499* (2022).

[60] Ziwei Zhu, Yun He, Xing Zhao, and James Caverlee. 2021. Popularity Bias in Dynamic Recommendation. *KDD* (2021).

[61] Ziwei Zhu, Yun He, Xing Zhao, Yin Zhang, Jianling Wang, and James Caverlee. 2021. Popularity-Opportunity Bias in Collaborative Filtering. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 85–93.