Debiased Representation Learning in Recommendation via Information Bottleneck

DUGANG LIU, Shenzhen University, China
PENGXIANG CHENG, HONG ZHU, and ZHENHUA DONG, Huawei Noah's Ark Lab, China
XIUQIANG HE, Tencent FIT, China
WEIKE PAN and ZHONG MING, Shenzhen University, China

How to effectively mitigate the bias of feedback in recommender systems is an important research topic. In this article, we first describe the generation process of the biased and unbiased feedback in recommender systems via two respective causal diagrams, where the difference between them can be regarded as the source of system-induced biases. We then define this difference as a confounding bias and propose a new perspective on debiased representation learning to alleviate it. Specifically, for the case with biased feedback alone, we derive the conditions that need to be satisfied to obtain a debiased representation from the causal diagrams. Then, we propose a novel framework called debiased information bottleneck (DIB) to optimize these conditions and then find a tractable solution for it. The proposed framework constrains the model to learn a biased embedding vector with independent biased and unbiased components in the training phase, and uses only the unbiased component in the test phase to deliver more accurate recommendations. We further propose a variant of DIB by relaxing the independence between the biased and unbiased components. Finally, we conduct extensive experiments on a public dataset and a real product dataset to verify the effectiveness of the proposed framework.

CCS Concepts: • **Information systems → Recommender systems;**

Additional Key Words and Phrases: Confounding bias, causal diagrams, recommender systems, information bottleneck

5

The authors acknowledge the support of National Natural Science Foundation of China Grants No. 61836005, No. 62272315, and No. 62172283.

This work is an extension of our previous work [26]. Compared with our previous work, we have added the following new contents in this article: (1) We have developed a new variant as a complement to DIB [26] under relaxed conditions (i.e., rDIB) in Section 4.3; (2) we have included new experimental results and associated analyses (i.e., new results in Figures 6(b), 7(a), 7(b), 8, 9, 10(a), 10(b), 10(c), 10(d), 10(e) and 11, and new results in Tables 6 and 7 in Section 5; (3) we have added more descriptions to further improve the presentation of the article (i.e., some new examples for different variables in Section 3.3, a formal definition of the studied problem and a new visualization example in Section 3.4, and some guidance on application of DIB in Section 4.2.4); and (4) we have made many other improvements throughout the whole article.

Authors' addresses: D. Liu, W. Pan (corresponding author), and Z. Ming (corresponding author), Shenzhen University, 3688# Nanhai Avenue, Shenzhen, Guangdong 518060 China; emails: dugang.ldg@gmail.com, {panweike, mingz}@szu.edu.cn; P. Cheng, H. Zhu, and Z. Dong, Huawei Noah's Ark Lab, Bantian Street, Shenzhen, Guangdong 518129 China; emails: {chengpengxiang1, zhuhong8, dongzhenhua}@huawei.com; X. He, Tencent FIT, 33# Haitian Second Road, Shenzhen, Guangdong 518060, China; email: xiuqianghe@tencent.com.

2770-6699/2023/01-ART5 $15.00
https://doi.org/10.1145/3568030

ACM Reference format:
Dugang Liu, Pengxiang Cheng, Hong Zhu, Zhenhua Dong, Xiuqiang He, Weike Pan, and Zhong Ming. 2023. Debiased Representation Learning in Recommendation via Information Bottleneck. *ACM Trans. Recomm. Syst.* 1, 1, Article 5 (January 2023), 27 pages.
https://doi.org/10.1145/3568030

1 INTRODUCTION

As a feedback loop system, a recommender system is associated with various biases during the interaction between the users and the system, such as position bias [3, 45], selection bias [30, 35], and popularity bias [1, 7]. Ignoring these biases will cause a recommendation model to converge to a biased sub-optimal solution, and have harmful effects on a recommender system and the users, such as filter bubbles [18], echo chambers [12], and unfairness [11, 13]. Therefore, how to effectively alleviate the bias of the feedback data collected in a recommender system is an important problem. In addition, since most of the existing recommendation scenarios have their own relatively complete models for deployment, a good debiasing framework needs to be independent of a specific architecture.

The previous works solving the bias problem in recommender systems mainly include the following four lines: heuristic-based methods [28, 51], inverse propensity score-based methods [35, 52, 53], unbiased data augmentation methods [6, 25, 46, 54], and some theoretical tools-based methods [33, 34]. The first line assumes that user feedback depends on certain specific factors and models this relationship, such as item features [14, 22] and public opinions [24, 27]. The second line uses the inverse propensity score as the sample weight to adjust the biased feedback distribution. The third line introduces a special uniform data as an unbiased target data to guide the training of the biased feedback. The last line aims to couple certain theoretical tools with the bias problem, and uses these theoretical tools to design some debiasing models, such as information bottleneck and causal inference techniques [44, 47, 48, 50]. However, most methods ignore the bias generation process, and thus may only be applicable to a certain type of bias problem.

In this article, inspired by [15, 20], we first describe the generation process of the biased feedback and unbiased feedback in recommender systems via two respective causal diagrams, where the difference between them can be viewed as the source of system-induced biases. We define this difference as a *confounding bias*, which can be viewed as the total offset on the observed feedback labels due to the system-induced biases. In order to simplify and match the main models in the recommendation field, we generally assume that the confounding bias will be reflected in the embedding representation of a recommendation model trained with a biased feedback data. This is because they will blindly learn the offset on the feedback labels due to the confounding bias. Based on this assumption, we propose a new perspective on debiased representation learning to alleviate the confounding bias. Further, in scenarios where only biased feedback is available, we propose a novel framework called **debiased information bottleneck** (**DIB**) based on information theory as a solution for debiased representation learning. Moreover, we also propose a variant of DIB (i.e., **relaxed debiased information bottleneck** (**rDIB**)) that considers relaxation of the optimization conditions adopted in DIB.

Specifically, the proposed framework is based on our observations in the causal diagrams of the feedback generation process described above. In the training phase, we constrain the model to learn a special *biased embedding vector*, including a biased component responsible for the effect of the confounding bias, and an unbiased component responsible for the effect of the user's true preference. In other words, a biased component is expected to capture the total offset on the observed

feedback labels due to the confounding bias, and an unbiased component is expected to capture the ideal part in the user's feedback label. We derive the conditions that need to be satisfied to obtain such a desired biased embedding vector and then use them as the optimization objective of DIB. Furthermore, a variant of DIB can be obtained by relaxing the independence condition between the biased and unbiased components, i.e., allowing a shared part between the two components. To remove the influence of the confounding bias in the test phase, we only retain the unbiased component in the embedding vector in the process of recommending items, i.e., a *debiased embedding vector*.

The proposed framework has better interpretability because it is directly derived from the causal diagram of the bias generation process. The proposed framework can also be easily integrated with most existing recommendation models, since it does not depend on a specific architecture. In addition, the proposed framework can be used to solve a more general bias problem, since the confounding bias is essentially the effect of a mixture of system-induced biases. Finally, we conduct extensive experiments on a public dataset and a real product dataset to verify the effectiveness of the proposed method, including standard unbiased tests, ablation studies, and some in-depth analysis of the proposed method.

2 RELATED WORK

In this article, we focus on designing information bottleneck-based solutions to mitigate a confounding bias from the perspective of debiased representation learning. Therefore, we firstly review some related work about debiasing methods in recommender systems, and then briefly introduce the information bottleneck and its applications in machine learning.

2.1 Debiasing in Recommender Systems

Solving the bias problem in recommender systems is an important topic that has gradually received more attention by both the researchers and practitioners from the academia and industry. The existing works on debiasing in recommender systems can be categorized into four lines, including heuristic-based methods, inverse propensity score-based methods [35, 52, 53], unbiased data augmentation methods [6, 25, 46, 54], and some theoretical tools-based methods [36, 42, 49, 55]. A heuristic-based method links a user's feedback with different specific factors, such as item features [14, 22], user ratings [28, 51], and public opinions [24, 27], and uses some probabilistic graphical models for modeling. An inverse propensity score-based method corrects the biased feedback distribution by introducing some sample weights. An unbiased data augmentation method guides the model training on biased feedback by introducing an unbiased data collected by a special uniform policy. Since the collection process of an unbiased data is very expensive in practice, some recent methods directly perform debiasing on one single biased data with some theoretical tools, such as information bottleneck [47], positive-unlabeled learning [34], asymmetric tri-training [33], and causal inference techniques [44, 48, 50]. Our DIB falls into the fourth line, but differs significantly from the existing works. On the one hand, we are the first to analyze the generation process of feedback events through the lens of causal inference, thereby identifying the source of system-induced biases and addressing them. This means that our DIB is expected to be able to mitigate a mixture of system-induced biases suffered by feedback events, whereas the vast majority of the existing works aim to address a specific bias. On the other hand, our DIB is designed based on the idea of debiased representation learning, and it only needs to manipulate the representation of the integrated backbone model without changing the other aspects. Conversely, most of the existing works require a custom model architecture or training method. Therefore, our DIB is of better scalability and flexibility.

2.2 Information Bottleneck

The information bottleneck method is a technique in information theory for finding the best trade-off between accuracy and complexity [40]. It has been regarded as a theoretical foundation of deep learning and been applied to many fields, such as robust or invariant representation learning [10, 29], disentangled representation learning [2], compressed representation learning [9], causal inference [31], and feature detection [37]. The works most relevant to ours in this line are [8, 17]. They focus on learning disentangled representation of texts through customized information bottleneck loss when the labels that only indicate the style information of texts are given. This representation includes independent style embedding and content embedding, which is similar to our expectation that the biased and unbiased components in the biased embedding vector are as independent as possible. However, these methods are not applicable to recommender systems since the labels in the recommendation field indicate a mixture of user preferences and biases. In addition, as far as we know, there is only one work that considers the use of information bottleneck to solve the bias problem in recommender systems. That work is based on contrastive analysis of feature embedding, which proposes a counterfactual variational information bottleneck method to solve the selection bias in recommender systems [47]. In contrast to this work, our DIB can effectively mitigate a mixture of system-induced biases, and act on arbitrary layers of the backbone model to effectively discriminate between the biased and unbiased components.

3 PRELIMINARIES

3.1 Notations

Let $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_d$ be a d-dimensional observed feature space and $\mathcal{Y} = \{0, 1\}$ be a label space. In this article, we focus on solving the bias problem in recommender systems. Specifically, suppose we have the following N events

$$(x^1, y^1), \ldots, (x^N, y^N), \tag{1}$$

where $x^i = (x^i_1, \ldots, x^i_d) \in \mathcal{X}$ and $y^i \in \mathcal{Y}$ are the feature vector and the label of the i-th event, respectively. According to the nature of feedback in a typical recommender system, we assume that the label y satisfies

$$y = \begin{cases} 1 & \text{the event was displayed and clicked,} \\ 0 & \text{the event was displayed but not clicked.} \end{cases} \tag{2}$$

An event represents an interaction between a user and a system. For example, a commercial system recommends a movie or displays an advertisement to a user, and the user provides some corresponding feedback. In particular, when no side information is available, i.e., only the user ID u and the item ID i are provided, an event can be simplified to $(x = (u, i), y)$.

The feedback events are used to train the recommendation model, which evaluates the users' preferences on the item set as accurately as possible by learning a decision function $\hat{y}(x) \in \{-\infty, +\infty\}$. In practice, the decision function is usually implemented based on a low-rank model and a neural network model, which are also included in our experiments. Both of them apply an embedding vector z^* as a representation of the input x. Therefore, we can describe the decision function as a mapping process from the representation z^* to the label $\hat{y}$, which passes through one or more hidden layers h_j. This process can be formalized as a Markov chain of successive representations [47], $y \to x \to z^* \to h_1 \to \cdots \to h_L \to \hat{y}$.

3.2 Information Theory

In this subsection, we briefly introduce some necessary background knowledge about information theory so that the reader can easily understand the description of the method and derivations

in Section 4, including mutual information and its relationship with information bottleneck and entropy.

Mutual information is a metric to calculate the correlation between two random variables (such as a and b), and is denoted as $I(a;b)$. The larger the mutual information, the higher the correlation between the random variables a and b. When the mutual information is 0, the random variables a and b are considered to be independent of each other. Mutual information also has some special properties, such as non-negativity $I(a;b) \geq 0$ and symmetry $I(a;b) = I(b;a)$. In real applications, mutual information $I(a;b)$ can be transformed into the form of KL divergence or entropy, i.e., $I(a;b) = \mathbb{E}_b\left[D_{\mathrm{KL}}(p(a|b) \parallel p(a))\right]$ and $I(a;b) = H(b) - H(b|a)$. An important application of mutual information is the information bottleneck [40], and can be expressed as

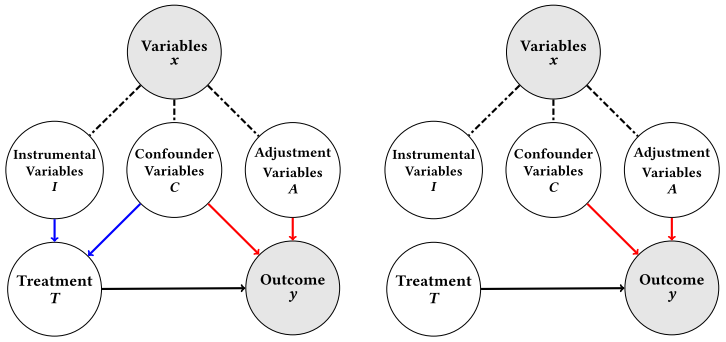
$$\min I(x;z) - \beta I(z;y), \tag{3}$$

where z is the representation obtained by encoding x. Based on the definition of mutual information, we can find that the goal of the information bottleneck is to maintain the correlation between z and y while reducing that between x and z, so as to achieve an optimal balance between accuracy and complexity. The correlation between variables a and b can be further denoted as conditional mutual information $I(a;b|c)$ given a third variable c. Similarly, we can relate conditional mutual information $I(a;b|c)$ to entropy, i.e., $I(a;b|c) = H(a|c) - H(a|c,b)$. In addition, we can build a bridge between conditional mutual information $I(a;b|c)$ and mutual information $I(a;b)$ via the chain rule of mutual information, i.e., $I(a;b) = I(a;c) - I(a;c|b) + I(a;b|c)$ [39].

3.3 Confounding Bias

To better understand the source of system-induced biases and address them in a targeted manner, in Figure 1(a), we show the generation process of feedback events in recommender systems from the perspective of causal inference. We first give definitions and examples of some key symbols used in Figure 1(a), and then describe the generation process in detail.

- The input variables x are equivalent to the feature vector, which may include user features, item features, context features, and so on. For example, one of the simplest input variables is to include only the user ID u and the item ID i, i.e., $x = (u, i)$.
- The outcome y is equivalent to the user's feedback label for the displayed item. For example, a click feedback can be denoted as $y = 1$, and a non-click feedback as $y = 0$. Note that the user's feedback label is usually provided by two parts, which are the user's true preferences and the offset caused by a mixture of system-induced biases.
- The treatment T in recommender systems can be thought of as referring to a recommendation strategy; that is, which items are selected by the system to combine with the current item, and in which layout and arrangement these items are displayed to the current user. In other words, it reflects the presentation strategy of the previous recommendation model for the current user and item. For example, a user sees a group of items in a nine-box grid that contains a specific item, while most of the other items are popular and are ranked at the top. Obviously, it is difficult to quantify the treatment T accurately.
- The instrumental variables I are a subset of features that only affect the treatment T, i.e., they lead to different recommendation strategies, resulting in different offsets on the feedback labels and have no impact on the user's true preferences. For example, the feature "device size" will affect the presentation of recommended items, but generally does not reflect a user's preferences.
- The confounder variables C are a subset of features that affect both the treatment T and the user's true preferences, i.e., they also lead to different recommendation strategies, and have



(a) Biased Feedback (b) Unbiased Feedback

Fig. 1. (a) A causal diagram of biased feedback, where x and y are known variables, blue arrows indicate the indirect effect, and red arrows indicate the direct effect. (b) A causal diagram of unbiased feedback where the indirect effect is truncated.

an impact on the user's true preferences while bringing different offsets on the feedback labels. For example, the feature "income level" will affect the choice of displayed items, and to a certain extent, it can reflect the user's preferences.

— The adjustment variables A are a subset of features that only affect the user's true preferences, i.e., they largely reflect the ideal part of the user's feedback label that is provided by true preferences. For example, they might be contextual features like weather and mood that only influence user decisions.

Based on the above definitions, Figure 1(a) means that the input variables x (i.e., the feature vector) can be divided into three non-overlapping parts, including some instrumental variables I, some confounder variables C, and some adjustment variables A. As two special variables that affect the treatment T, the instrumental variables I and the confounder variables C can determine the treatment T, and they have an indirect effect on the outcome y (i.e., the label) through the path $\{I, C\} \rightarrow T \rightarrow y$. An indirect effect can be seen as the total offset on the user's feedback label produced by a mixture of system-induced biases. As two special variables that affect the user's true preferences, the confounder variables C and the adjustment variables A have a direct effect on y through the path $\{C, A\} \rightarrow y$. A direct effect can be seen as the ideal part in the user's feedback label, which is generated by the user's true preferences, i.e., the ground-truth feedback label. Similar causal diagrams can also be found in previous works on treatment effect estimation [15, 20], and we follow their assumptions, i.e., *strong ignorability* [32].

Different recommendation strategies will produce different indirect effects on feedback events by influencing the treatment T. This will lead to the inherent variability in the feedback events, and make most recommendation models that aim to minimize the error of the observed feedback not have good generalizability. Conversely, because the direct effect does not depend on the treatment T, it can be considered as the stable and true user preferences. This means that if we can cut off the indirect effect, i.e., we have a special strategy that only depends on the direct effect, then the collected feedback events are relatively stable and unbiased. We show the generation process of this unbiased feedback event in Figure 1(b). By comparing Figure 1(a) and (b), we call the bias brought by the recommendation strategy the *confounding bias*. Since the recommendation strategy depends only on the deployed system, the confounding bias is essentially the total offsets in the observed feedback labels resulting from a mixture of system-induced biases, e.g., the feedback labels may suffer from both the position bias and popularity bias.

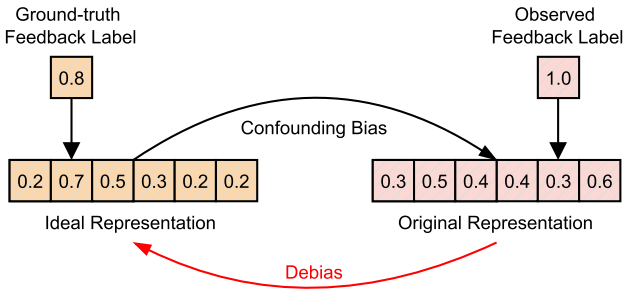


Fig. 2. An example of debiased representation learning. The original representation obtained by a traditional recommendation model is corrupted by the confounding bias due to blindly learning the total offsets on the observed feedback labels resulting from the confounding bias. The debiased representation learning aims to remove the influence of the confounding bias to obtain the ideal representation by restoring the ground-truth feedback labels.

Definition 3.1 (Confounding Bias). Suppose that the variables x, the outcome y, the indirect effect $\{I, C\} \rightarrow T \rightarrow y$, and the direct effect $\{C, A\} \rightarrow y$ from x to y are given. Confounding bias refers to the total offsets of the observed feedback labels in recommender systems due to the indirect effects.

3.4 Problem Formulation

In practice, it may be difficult to directly group the variables x to obtain the instrumental variables I, the confounder variables C, and the adjustment variables A. Instead, since the goal of most recommendation models is to learn an accurate embedding representation vector derived from the variables x, we make a general assumption, which is intuitively reasonable. The representation z^* learned by a traditional recommendation model, which is a good proxy for the variables x, naturally suffers from the biased variables in x. In other words, a traditional recommendation model will blindly learn the total offsets on feedback labels due to the confounding bias, resulting in z^* being biased. Since the correspondence between the biased variables and the semantics of the dimension in z^* is difficult to be obtained, z^* is also indistinguishable.

Assumption 3.2. *The confounding bias will be reflected in the embedding representation learned by a traditional recommendation model, since it will blindly learn the offsets on the observed feedback labels produced by the confounding bias. And the bias will usually corrupt all the dimensions, i.e., the original representation z^* is biased and indistinguishable.*

Based on this assumption, we can equate the solution to the bias problem in recommender systems, especially the system-induced biases, as how to remove the corruption caused by the confounding bias from the embedding representation. We call this new perspective *debiased representation learning*, a schematic diagram of which is shown in Figure 2. On the other hand, although previous works have shown that the unbiased feedback can be obtained through a uniform policy [6, 25, 46, 54], i.e., for user requests, the system randomly samples some items from a candidate set, and displays them after some random arrangement, feedback events can only be collected within a particularly limited network traffic as this policy would hurt the users' experiences and the revenue of the platform. This means a more engaging scenario is that only the biased feedback is available. Therefore, in this article, we focus on mitigating the confounding bias through debiased representation learning when only biased feedback is available.

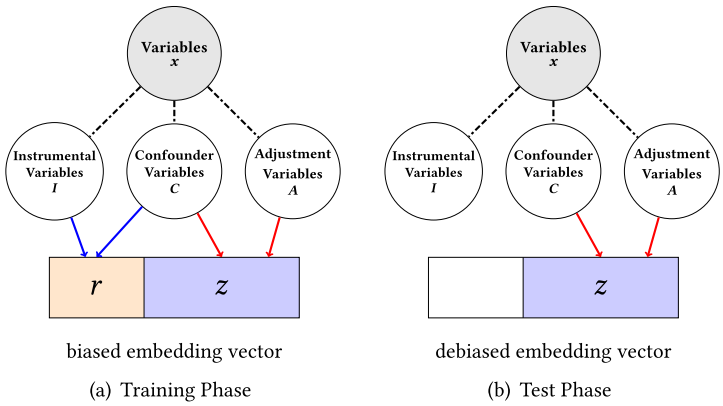


Fig. 3. (a) In the training phase, the model obtains a special biased embedding vector from the variables x, which includes a biased component r and an unbiased component z. (b) In the test phase, the biased component r is discarded, and the unbiased component z is used in the recommendation process.

4 THE PROPOSED METHOD

4.1 Debiased Information Bottleneck

We illustrate the main idea of the proposed framework in Figure 3. In the training phase (i.e., Figure 3(a)), we constrain the model to obtain a special biased representation vector from the variables x, which includes two independent components, i.e., a biased component r and an unbiased component z. The biased component r is responsible for the indirect effect, while the unbiased component z is responsible for the direct effect. This means that we expect a biased component r to capture the total offsets in the observed feedback label caused by a mixture of system-induced biases, and an unbiased component z to capture the ideal part of the feedback label generated by the user's true preferences. Note that this special biased embedding vector, i.e., $[r, z]$, is easier to distinguish the influence of the confounding bias than the original representation z^*, since r explicitly encodes the confounding bias and z avoids the effect of the confounding bias. In the test phase (i.e., Figure 3(b)), we discard the biased component and only use the unbiased component for more accurate recommendation.

From Figure 3(a), we can see that in order to achieve a more accurate recommendation, we must meet at least the following four conditions: (1) to avoid the influence of the biased variables, the unbiased component z should not overfit the variables x; (2) due to the role of the direct effect, the unbiased component z needs to predict the label y as accurately as possible; (3) the biased component r and the unbiased component z must be as independent as possible to get a better distinction, i.e., $z \perp r$; and (4) due to the role of the indirect effect, the biased component r is also beneficial to predict the label y to a certain extent. Note that we have not constrained the relationship between the biased component r and the variables x, because the degree of dependence between them is determined by the strength of the bias in the feedback data, and blindly optimizing it may lead to bad results. Inspired by information theory, we can then derive the required objective function to be minimized from the above analysis:

$$\mathcal{L}_{DIB} := \underbrace{\beta I(z;x)}_{①} - \underbrace{I(z;y)}_{②} + \underbrace{\gamma I(z;r)}_{③} - \underbrace{\alpha I(r;y)}_{④}, \tag{4}$$

where term ① is a compression term that describes the mutual information between the variables x and the unbiased embedding z; term ② is an accuracy term that describes the performance of the

unbiased embedding z; term ③ is a de-confounder penalty term, which describes the dependency degree between the biased embedding r and the unbiased embedding z; and term ④ is similar to term ②, which is used for the potential gain from the biased embedding r. Note that β, γ, and α are the weight parameters. Since terms ① and ② in Equation (4) are similar to a standard information bottleneck, we call the proposed framework a *debiased information bottleneck*, or DIB for short. By optimizing $\mathcal{L}_{DIB}$, we expect to get the desired biased and unbiased components, and can then prune the confounding bias. Note that Equation (4) is a necessary optimization objective to drive the model toward the debiased embedding vector, and introducing more reasonable targets based on Equation (4) may help to obtain a more accurate debiased embedding vector, which we leave to future work.

4.2 A Tractable Optimization Framework

$\mathcal{L}_{DIB}$ is clearly an intractable optimization function, especially because of the key term $I(z;r)$ used to induce the desired embedding separation. Next, we discuss the optimization of the de-confounder penalty term $I(z;r)$ and the compression term $I(z;x)$, and derive an upper bound of Equation (4). Finally, we describe a tractable solution for the objective function $\mathcal{L}_{DIB}$ according to the upper bound.

4.2.1 The De-Confounder Penalty Term. Based on the chain rule of mutual information, we have the following equation about the de-confounder penalty term ③ in Equation (4):

$$I(z;r) = I(z;y) - I(z;y|r) + I(z;r|y)\,. \tag{5}$$

We further inspect the term $I(z;r|y)$ in Equation (5). Since the distribution of z depends solely on the variables x, and x is affected by y, we have $H(z|y,r) = H(z|y)$, where $H(\cdot|\cdot)$ denotes the conditional entropy [17, 29]. Combining the properties of mutual information, we have

$$I(z;r|y) = H(z|y) - H(z|y,r) = H(z|y) - H(z|y) = 0. \tag{6}$$

By substituting Equation (6) into Equation (5), we have

$$I(z;r) = I(z;y) - I(z;y|r)\,. \tag{7}$$

Since the term $I(z;y|r)$ in Equation (7) is still difficult to be calculated, we use the form of conditional entropy to further simplify it,

$$I(z;r) = I(z;y) - H(y|r) + H(y|z,r)\,. \tag{8}$$

Finally, combining Equations (4) and (8), we can rewrite the objective function $\mathcal{L}_{DIB}$ in Equation (4) as follows:

$$\begin{aligned}\mathcal{L}_{DIB} &= \beta I(z;x) - I(z;y) + \gamma I(z;r) - \alpha I(r;y)\\ &= \beta I(z;x) - I(z;y) + \gamma\left[I(z;y) - H(y|r) + H(y|z,r)\right] - \alpha I(r;y)\\ &= \beta I(z;x) - (1-\gamma) I(z;y) - \gamma H(y|r) + \gamma H(y|z,r) - \alpha I(r;y)\,.\end{aligned} \tag{9}$$

4.2.2 The Compression Term. We can find that only the compression term $I(z;x)$ is related to the variables x in Equation (9). To optimize it directly, we describe a simple and precise expression of this mutual information using a method similar to that in [17, 47]. First, based on the relationship between mutual information and **Kullback-Leibler** (**KL**) divergence, the compression term $I(z;x)$ can be calculated as follows:

$$I(z;x) = \mathbb{E}_x\left[D_{\mathrm{KL}}(p(z|x) \parallel p(z))\right] = \sum_x p(x) \sum_z p(z|x) \log p(z|x) - \sum_z p(z) \log p(z). \tag{10}$$

However, the marginal probability $p(z) = \sum_x p(z|x)p(x)$ is usually difficult to be calculated in practice. We use variational approximation to address this issue, i.e., we use a variational distribution $q(z)$ instead of $p(z)$. According to Gibbs' inequality, we know that the KL divergence is non-negative. Therefore, we can derive an upper bound of Equation (10),

$$\begin{aligned} & D_{\mathrm{KL}}\left(p\left(z\right) \parallel q\left(z\right)\right) \geq 0 \\ & \Rightarrow -\sum_z p(z) \log p(z) \leq -\sum_z p(z) \log q(z) \\ & \Rightarrow D_{\mathrm{KL}}\left(p\left(z|x\right) \parallel p\left(z\right)\right) \leq D_{\mathrm{KL}}\left(p\left(z|x\right) \parallel q\left(z\right)\right). \end{aligned} \tag{11}$$

Similar to most previous works [23], we can assume that the posterior $p(z|x)$ is a Gaussian distribution (i.e., $p(z|x) = \mathcal{N}(\mu(x), \mathrm{diag}\{\sigma^2(x)\})$), where $\mu(x)$ is the encoded embedding of the variables x and $\mathrm{diag}\{\sigma^2(x)\}$ is the diagonal matrix indicating the variance. Through the reparameterization trick, the embedding z can be generated according to $z = \mu(x) + \epsilon \odot \sigma(x)$, where $\epsilon \sim \mathcal{N}(0, I)$. Obviously, if we fix $\sigma(x)$ to be an all-zero matrix, z will reduce to a deterministic embedding. On the other hand, the prior $q(z)$ is assumed to be a standard Gaussian variational distribution, i.e., $q(z) = \mathcal{N}(0, I)$. Finally, we can rewrite the above upper bound,

$$D_{\mathrm{KL}}\left(p\left(z|x\right) \parallel q\left(z\right)\right) = \frac{1}{2}\left\|\mu\left(x\right)\right\|_2^2 + \frac{1}{2}\sum_d \left(\sigma_d^2 - \log \sigma_d^2 - 1\right), \tag{12}$$

where σ_d^2 is an element in $\mathrm{diag}\{\sigma^2(x)\}$, i.e., $\mathrm{diag}\{\sigma^2(x)\} = \{\sigma_d^2\}_{d=1}^D$. This means that for a deterministic embedding z, we can optimize this upper bound by directly applying the ℓ_2-norm regularization on the embedding vector z, which is equivalent to optimizing the compression term $I(z; x)$. Note that the compression term in previous works acts on the entire biased representation z^*, and we only compress the unbiased component z of the representation.

4.2.3 Solution. For the mutual information $I\left(z; y\right)$ in Equation (9), we have $I\left(z; y\right) = H\left(y\right) - H\left(y|z\right)$. Since $H\left(y\right)$ is a positive constant and can be ignored, we have the following inequality:

$$I\left(z; y\right) \geq -H\left(y|z\right). \tag{13}$$

This inequality also applies to the mutual information $I\left(r; y\right)$ in Equation (9). Combining Equations (12) and (13), we can rewrite Equation (9) as follows:

$$\begin{aligned} \mathcal{L}_{DIB} &= \beta I\left(z; x\right) - \left(1-\gamma\right) I\left(z; y\right) - \gamma H\left(y|r\right) + \gamma H\left(y|z, r\right) - \alpha I\left(r; y\right) \\ &\leq \beta \left\|\mu(x)\right\|_2^2 + \left(1-\gamma\right) H(y|z) - \left(\gamma - \alpha\right) H(y|r) + \gamma H(y|z, r). \end{aligned} \tag{14}$$

Finally, we get a tractable solution for $\mathcal{L}_{DIB}$,

$$\hat{\mathcal{L}}_{DIB} = \underbrace{\left(1-\gamma\right) H(y|z)}_{(a)} - \underbrace{\left(\gamma - \alpha\right) H(y|r)}_{(b)} + \underbrace{\gamma H(y|z, r)}_{(c)} + \underbrace{\beta \left\|\mu(x)\right\|_2^2}_{(d)}, \tag{15}$$

where $0 < \alpha < \gamma < 1$. Let $\hat{y}_r$, $\hat{y}_z$, and $\hat{y}_{z,c}$ be the predicted labels generated by the biased component r, the unbiased component z, and the biased embedding vector $[z, r]$ as shown in Figure 3(a), respectively. The final objective function also contains four terms: term (a) denotes the cross entropy between $\hat{y}_z$ and y; term (b) denotes the cross entropy between $\hat{y}_r$ and y; term (c) denotes the cross entropy between $\hat{y}_{z,r}$ and y; and term (d) is the regularization term used to improve the robustness of the embedding representation. A complete optimization process of DIB is shown in Algorithm 1.

ALGORITHM 1: Debiased Information Bottleneck (DIB)

Input: Observed feedback events $\{(x, y)\}$; hyper-parameters α, β, γ;

1: Initialize the parameters of the model θ (including the biased embedding representation r and the unbiased embedding representation z);
2: **repeat**
3: Calculate term (c) in Equation (15) by the concatenation of z and r, i.e., $H(y|z, r)$;
4: Calculate term (a) in Equation (15) by z, i.e., $H(y|z)$ (to keep the dimensions consistent, z is concatenated with a zero vector with the same dimension of r);
5: Calculate term (b) in Equation (15) by r, i.e., $H(y|r)$ (similar to the above operation);
6: Calculate term (d) in Equation (15) by applying a regularization to z;
7: Update the model parameters θ via stochastic gradient descent based on $\hat{\mathcal{L}}_{DIB}$ (i.e., Equation (15));
8: **until** training loss stops to decrease.

4.2.4 Remarks. We include discussions about Equation (15) from two perspectives in order to intuitively show why our DIB is effective in mitigating the confounding bias.

— Note that we often have no specific knowledge of a mixture of system-induced biases, and it is also difficult to identify them from the collected feedback events in practice. Fortunately, we can try to infer them by the peculiar nature of bias information. Recall that the observed feedback labels contain two sources of information, i.e., a user's true preferences and the offset generated by the confounding bias. Since the confounding bias does not produce a large offset, the user's preferences are usually dominant. This means that without the help of the user's true preferences, the bias information is not able to estimate the user's feedback label well, and when combined with the user's true preferences, there will be an overfitting to the user's feedback label. Since we expect z and r to encode the user's true preferences and the bias information, respectively, the above two cases can be considered corresponding to terms (b) and (c) in Equation (15). Combined with terms (a) and (d) that ensure the effect of z, Equation (15) is expected to distinguish the bias information under reasonable parameter settings.

— The confounding bias can also be seen as a distribution shift since the offset on the feedback labels produced by the confounding bias results in a different distribution than the ground-truth feedback labels. This means that we can also interpret debiased representation learning in terms of stable learning or robust learning. Let us consider that r encodes noisy information that is detrimental to the estimation of the feedback labels, and the term (b) of Equation (15) constrains its range of variation. The terms (a) and (c) in Equation (15) constrain z not only to estimate the feedback labels well by itself, but also to estimate them as well as possible in the presence of noise. With this, z is expected to perform relatively well under different distributions, i.e., we can think that it is encoding the user's true preferences and is stable and unbiased under varying degrees of the confounding bias.

When setting the weights of Equation (15) in practice, two aspects need to be considered. Firstly, since the unbiased component z is responsible for the user's true preferences and is used for prediction, term (a) needs a larger weight to ensure model effectiveness, i.e., γ is set to a smaller value. For example, we search for the best value from 0.001 to 0.2 in the experiments. Secondly, term (b) needs a positive weight because the biased component r needs to confuse the unbiased component z during model training, i.e., $\gamma - \alpha > 0 \Rightarrow \alpha < \gamma$. The proportion of the dimensions occupied by the biased component r and the unbiased component z in the biased representation

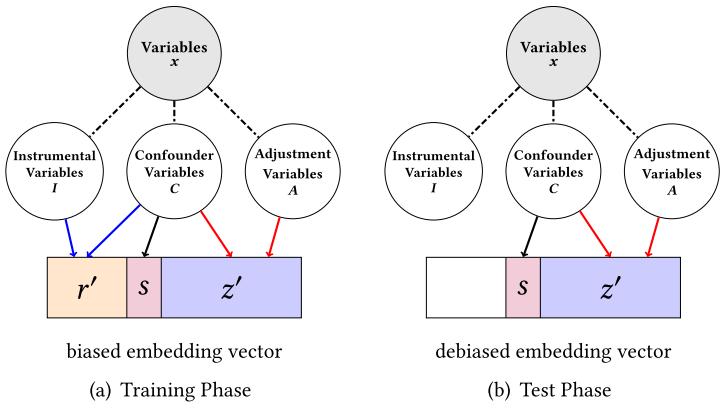


Fig. 4. (a) In the training phase, the biased component r consists of an independent part r' and a shared part s, while the unbiased component z consists of an independent part z' and the same shared part s as that of r. (b) In the test phase, the independent part r' of the biased component r is discarded, and the unbiased component z, including z' and s, is used in the recommendation process.

vector shown in Figure 3(a) is adjustable, where intuitively the latter should have a higher proportion. We set them to the same proportion (i.e., 50%) in our experiments for simplicity.

The proposed DIB is compatible with most existing recommendation models since it does not depend on a specific architecture. In particular, when applying DIB to neural network–based recommendation models in practice, it is optional to use the output of a layer in the network as the biased representation vector. For example, it can be defined in the feature encoding layer, feature interaction layer, or prediction layer. For simplicity, we use the form defined on the feature encoding layer in our experiments. In addition, DIB is also suitable for scenarios where no side information is available. Since the user activity and item popularity are potential influencing factors of recommendation strategies, the features related to the users and items may have the property of the confounder variables C, that is, there are still some indirect effects that need to be removed.

4.3 Relaxed Debiased Information Bottleneck

In Section 4.1, we expect the biased representation vector to be split into two mutually independent components, i.e., a biased component r and an unbiased component z. This may be overly restrictive in some cases, especially with some indistinguishable confounding variables. Therefore, in this section, we propose a variant of DIB by relaxing this constraint, which is thus called rDIB. Specifically, we allow a shared part between the biased component r and the unbiased component z during the training phase. This means that the biased component r consists of an independent part r' and a shared part s, while the unbiased component z consists of an independent part z' and the same shared part s as that of r. In the test phase, we still only use the unbiased component z for more accurate recommendations, which is equivalent to discarding the independent part r' of the biased component r. We illustrate the main idea of rDIB in Figure 4.

Based on Figure 4 and the analysis in Section 4.1, we can derive a desired objective function to be minimized,

$$\mathcal{L}_{rDIB} := \underbrace{\beta I\left(z', s; x\right)}_{①} - \underbrace{I\left(z', s; y\right)}_{②} + \underbrace{\gamma I\left(z'; r'\right)}_{③} - \underbrace{\alpha I\left(r', s; y\right)}_{④}. \tag{16}$$

Comparing Equation (16) with Equation (4), we can find that the difference is that a shared part s acts on the unbiased component z and the biased component r in the first two terms and the

ALGORITHM 2: Relaxed Debiased Information Bottleneck (rDIB)

Input: Observed feedback events $\{(x, y)\}$; hyper-parameters α, β, γ;

1: Initialize the parameters of the model θ (including independent biased embedding representation r', independent unbiased embedding representation z', and shared embedding representation s);
2: **repeat**
3: Obtain the biased embedding vector by concatenating the three embedding representations, i.e., $[r', s, z']$;
4: Calculate term (c) in Equation (20) by masking the shared part s with $\mathbf{0}$, i.e., $[r', \mathbf{0}, z']$ used in $H(y|z', r')$;
5: Calculate term (a) in Equation (20) by masking r' and the concatenation of s and r' with $\mathbf{0}$, respectively, i.e., $[\mathbf{0}, s, z']$ used in $H(y|z', s)$ and $[\mathbf{0}, \mathbf{0}, z']$ used in $H(y|z')$;
6: Calculate term (b) in Equation (20) by masking z' and the concatenation of s and z' with $\mathbf{0}$, respectively, i.e., $[r', s, \mathbf{0}]$ used in $H(y|r', s)$ and $[r', \mathbf{0}, \mathbf{0}]$ used in $H(y|r')$;
7: Calculate term (d) in Equation (20) by applying a regularization to $[\mathbf{0}, s, z']$;
8: Update the model parameters θ via stochastic gradient descent based on $\hat{\mathcal{L}}_{rDIB}$ (i.e., Equation (20));
9: **until** training loss stops to decrease.

fourth term, respectively, and the third term applies only to the independent parts of the two components. $\mathcal{L}_{rDIB}$ is also an intractable optimization function. And by referring to the description in Section 4.2, we next derive its upper bound as a tractable solution. Firstly, similar to the procedure in Section 4.2.1, we can obtain the equation corresponding to term ③ in Equation (16),

$$I\left(z'; r'\right) = I\left(z'; y\right) - H\left(y|r'\right) + H\left(y|z', r'\right). \tag{17}$$

Combining Equations (17) and (16), we can rewrite the objective function $\mathcal{L}_{rDIB}$ of Equation (16) as follows:

$$\begin{aligned} \mathcal{L}_{rDIB} &= \beta I\left(z', s; x\right) - I\left(z', s; y\right) + \gamma I\left(z'; r'\right) - \alpha I\left(r', s; y\right) \\ &= \beta I\left(z', s; x\right) - I\left(z', s; y\right) + \gamma \left[I\left(z'; y\right) - H\left(y|r'\right) + H\left(y|z', r'\right)\right] - \alpha I\left(r', s; y\right). \end{aligned} \tag{18}$$

Secondly, referring to the procedure in Section 4.2.2, we can replace the first term of Equation (18) by directly applying the ℓ_2-norm regularization to the unbiased component z (i.e., $[s, z']$),

$$\mathcal{L}_{rDIB} = \beta \left\|\mu(x)\right\|_2^2 - I\left(z', s; y\right) + \gamma \left[I\left(z'; y\right) - H\left(y|r'\right) + H\left(y|z', r'\right)\right] - \alpha I\left(r', s; y\right). \tag{19}$$

Finally, we use the inequality in Equation (13) to further convert the mutual information terms in Equation (19) into the corresponding entropy form, and then obtain a tractable solution for $\mathcal{L}_{rDIB}$,

$$\hat{\mathcal{L}}_{rDIB} = \underbrace{H(y|z', s) - \gamma H(y|z')}_{(a)} - \underbrace{\gamma H(y|r') + \alpha H(y|r', s)}_{(b)} + \underbrace{\gamma H(y|z', r')}_{(c)} + \underbrace{\beta \left\|\mu(x)\right\|_2^2}_{(d)}. \tag{20}$$

We can observe that the difference from Equation (15) is that both the unbiased and biased components include two optimization terms that consider only the independent part and the shared part, respectively. A complete optimization process of rDIB is shown in Algorithm 2.

5 EXPERIMENTS

In this section, we conduct comprehensive experiments with the aim of answering the following five key questions.

— RQ1: How does the proposed framework perform in an unbiased evaluation?

Table 1. Statistics of the Datasets

	Yahoo! R3		Product	
	#Feedback	P/N	#Feedback	P/N
Training set	254,713	67.02%	4,160,414	2.21%
Validation set	56,991	67.00%	897,449	2.15%
Test set	54,000	9.64%	225,762	1.03%

P/N represents the ratio between the numbers of positive and negative feedback.

— RQ2: What is the role of each term in the proposed framework?
— RQ3: What is the effect of the proposed variant (i.e., rDIB)?
— RQ4: What are the characteristics of the proposed framework?
— RQ5: How does the proposed framework perform in real-world recommendation scenarios?

5.1 Experiment Setup

5.1.1 Datasets. In order to evaluate the performance of the model in mitigating the confounding bias, we need a test set that includes unbiased feedback events to simulate the ideal distribution. We consider both a public dataset and a real-world product dataset in the experiments, where the statistics are described in Table 1.

— **Yahoo! R3:** This dataset contains ratings collected from two different sources on Yahoo! Music services, containing 15,400 users and 1,000 songs. The user set consists of ratings provided by users' subjective choices and is therefore considered to be biased. The random set consists of ratings collected during an online survey, i.e., each of the first 5,400 users is required to provide ratings on 10 randomly displayed songs, which can be considered unbiased. We binarize each rating by checking whether it is larger than 3 and then obtain some positive feedback (i.e., $y = 1$) and negative feedback (i.e., $y = 0$). Note that the missing entries are treated as negative feedback. For the user set, we randomly split it into two subsets from the user level: 80% of each user's feedback for training and the remaining 20% for validation to tune the hyper-parameters. The random set is used as the test set for evaluating our method. Note that the random set with unbiased property is not available during the training phase.
— **Product:** This is a large-scale dataset for CTR prediction, which includes 2 weeks of user click records from a real-world advertising system. This data covers 122 displayed advertisements and approximately 300,000 users. Similarly, it contains a user set and a random set. The user set is recorded when the system is using several traditional ranking-oriented recommendation policies, and the random set is recorded using a uniform policy. Next, we randomly split the user set into two subsets in the same way as that for Yahoo! R3, i.e., 80% feedback is used as the training set, and the rest is used as the validation set. The entire random set is used as the test set for evaluating our method. Note that the random set with unbiased property is also unavailable during the training phase.

5.1.2 Evaluation Metrics. For all experiments, we employ four evaluation metrics that are widely used in recommender systems, including the **area under the ROC curve** (**AUC**), **precision** (**P@K**), **recall** (**R@K**), and **normalized discounted cumulative gain** (**nDCG**). We set the maximum length of the recommendation list as 50, and choose AUC as our main evaluation metric because it is one of the most important metrics in industry and previous works on debiasing. We report the results of P@K and R@K when K is 5 and 10, and the results of nDCG when K is 50. More experimental results about different values of K can be found in the open source link.

The candidate items to be recommended for a user are from the set of items that have not been interacted by the user.

5.1.3 Baselines. Since most debiasing methods are integrated into some existing recommendation models, in order to evaluate the generalization of our method, we adopt two of the most common recommendation models as the backbones, i.e., **matrix factorization** (**MF**) [19] and **neural collaborative filtering** (**NCF**) [16]. In addition, among the four lines of debiasing methods summarized in Section 2.1, the heuristic-based methods are usually inefficient, and the unbiased data augmentation methods are not suitable for the scenario where only a biased data is available for training in this article. Therefore, we choose the representative methods in the other two lines as the baselines. Note that based on different backbones, each of these baselines has two corresponding versions.

— **MF [19]:** This is one of the most classic recommendation models. The user-item interaction matrix is factorized to learn the representation of the users and items, and their inner products are used to predict the user preferences for the items.
— **NCF [16]:** This is one of the most classic recommendation models combined with neural networks. Compared with the linearity of matrix factorization, neural networks are used to model the nonlinear relationship between users and items to further improve the recommendation performance.
— **IPS [35]:** This is a representative method based on the **inverse propensity score** (**IPS**) in debiasing recommendation. The estimated propensity score is used as the weight of the training feedback to adjust the biased distribution of the feedback. We estimate the propensity score of item i for any user via the relative item popularity,

$$P_{*,i} = \left(\frac{\sum_{u \in \mathcal{U}} O_{u,i}}{max_{i \in \mathcal{I}} \sum_{u \in \mathcal{U}} O_{u,i}} \right)^{\eta}, \tag{21}$$

where $O_{u,i}$ indicates the observation status of user u for item i, i.e., $O_{u,i} = 1$ means that the user u observes the item i, while $O_{u,i} = 0$ is the opposite, and η is a control weight used to adapt to different datasets. Similar methods have also been adopted in previous works [21, 34].
— **SNIPS [38]:** The above method based on the inverse propensity score may face the problem of high variance in practice. To solve this problem, a self-normalized inverse propensity score is proposed and can be written as

$$\mathcal{L}_{SNIPS} = \frac{\sum_{(u,i):O_{u,i}=1} \frac{\mathcal{L}\left(\hat{Y}_{u,i}, Y_{u,i}\right)}{P_{u,i}}}{\sum_{(u,i):O_{u,i}=1} \frac{1}{P_{u,i}}}, \tag{22}$$

where $P_{u,i}$ is calculated according to Equation (21), and $\mathcal{L}(\hat{Y}_{u,i}, Y_{u,i})$ is the cross-entropy loss of the observed feedback. $Y_{u,i}$ and $\hat{Y}_{u,i}$ are the true and predicted label of user u for item i, respectively.
— **AT [33]:** For the selection bias in recommender systems, an upper bound of the generalization error is derived and optimized by an asymmetric tri-training method. Two pre-models are used to generate a pseudo-label set, and the target model and the two pre-models are then retrained on this set. This process is repeated until the debiasing performance of the target model is improved the most.
— **Relevance [34]:** Based on the idea of positive unlabeled learning, an unbiased estimation is derived for binary feedback modeling in recommender systems. The inverse propensity score is estimated and weighted differently for positive feedback and negative feedback.

Table 2. Hyper-Parameters and their Values Tuned in the Experiments

Name	Range	Functionality
$rank$	$\{5, 10, \ldots, 195, 200\}$	Embedding dimension
β	$\{1e^{-5}, 1e^{-4}, \ldots, 1e^{-1}, 1\}$	Regularization
bs	$\{2^7, 2^8, \ldots, 2^{13}, 2^{14}\}$	Batch size
lr	$\{1e^{-4}, 5e^{-4}, \ldots, 5e^{-2}, 1e^{-1}\}$	Learning rate
γ	$[0.001, 0.2]$	Loss weight
α	$[0.0001, \min\{\gamma\}]$	Loss weight

The objective function can be represented as follows:

$$\hat{\mathcal{L}}_{Rel} = \frac{1}{|\mathcal{D}|} \sum_{(u,i)\in\mathcal{D}} \left[\frac{Y_{u,i}}{P_{u,i}} \delta^{(1)}_{u,i} + \left(1 - \frac{Y_{u,i}}{P_{u,i}}\right) \delta^{(0)}_{u,i} \right], \tag{23}$$

where $\delta^{(1)}_{u,i}$ and $\delta^{(0)}_{u,i}$ denote the positive loss and the negative loss, respectively, and $P_{u,i}$ is calculated by Equation (21).

— **CVIB [47]:** It is an alternative framework for debiasing learning without an unbiased feedback data, which is called the information-theoretic counterfactual variational information bottleneck. By separating the task-aware mutual information term in standard information bottleneck into a factual part and a counterfactual part, it derives a contrastive information regularizer and an additional output confidence penalty to improve the learning balance between the observed and unobserved data.

5.1.4 Implementation Details. We implement all the methods on TensorFlow 1.4[1] with the Adam optimizer. By evaluating the performance on AUC on the validation set, we use the hyper-parameter search library *Optuna* [4] to accelerate the tuning process of all the methods. We also adopt an early stopping mechanism with a patience of 5 to avoid overfitting to the training set. All methods are run on a cluster with 2 Intel(R) Xeon(R) E5-2680 and 8 Tesla P100 GPU. The range of the values of the hyper-parameters is shown in Table 2. Note that the source codes and results are available at https://github.com/dgliu/TORS_DIB.

5.2 RQ1: Comparison Results of Unbiased Evaluation

The comparison results are reported in Tables 3 and 4. For the results of using matrix factorization as the backbone model shown in Table 3, the proposed framework consistently outperforms all the baselines on all the metrics across the two datasets of Yahoo! R3 and Product. In addition, we have the following main observations: (1) The debiasing method is usually better than the basic baseline, i.e., matrix factorization, except for IPS-MF on Yahoo! R3. This may be due to the inaccurate estimation of the propensity score when the data size is small. (2) By avoiding the estimation of the propensity score and providing theoretical completeness, theoretical tools-based methods (i.e., CVIB-MF, AT-MF, and Relevance-MF) usually achieve better and more stable performance. (3) For the important baseline CVIB-MF whose objective function is also induced by information theory, its improvement over MF on Yahoo! R3 is small, while the proposed framework is better and more robust on both datasets.

For the results of using neural collaborative filtering as the backbone shown in Table 4, the proposed framework outperforms all the baselines in most cases. Specifically, we have the following

[1] https://www.tensorflow.org.

Table 3. Comparison Results of Unbiased Evaluation using MF as the Backbone Model, where the Best Results and the Second Best Results are Marked in Bold and Underlined, Respectively

Yahoo! R3						
Method	**AUC**	**nDCG**	**P@5**	**P@10**	**R@5**	**R@10**
MF	0.7081	0.0341	0.0043	0.0043	0.0123	0.0273
IPS-MF	0.7040	0.0259	0.0031	0.0033	0.0091	0.0182
SNIPS-MF	0.7124	0.0390	0.0057	0.0048	0.0182	0.0293
CVIB-MF	0.7086	0.0488	0.0080	0.0075	0.0253	0.0471
AT-MF	0.7290	0.0676	0.0113	0.0101	0.0345	0.0629
Relevance-MF	0.7469	0.0843	0.0159	0.0131	0.0526	0.0863
DIB-MF	**0.7602**	**0.0932**	**0.0177**	**0.0151**	**0.0566**	**0.0960**
Product						
Method	**AUC**	**nDCG**	**P@5**	**P@10**	**R@5**	**R@10**
MF	0.6936	0.0324	0.0085	0.0079	0.0407	0.0752
IPS-MF	0.7125	0.0408	0.0095	0.0105	0.0456	0.1019
SNIPS-MF	0.7098	0.0403	0.0092	0.0104	0.0438	0.1003
CVIB-MF	0.7143	0.0521	0.0106	0.0109	0.0520	0.1076
AT-MF	0.7191	0.0443	0.0110	0.0113	0.0532	0.1092
Relevance-MF	0.6709	0.0656	0.0157	0.0142	0.0776	0.1386
DIB-MF	**0.7365**	**0.0819**	**0.0198**	**0.0173**	**0.0965**	**0.1688**

AUC is the main evaluation metric.

Table 4. Comparison Results of Unbiased Evaluation using NCF as the Backbone Model, where the Best Results and the Second Best Results are Marked in Bold and Underlined, Respectively

Yahoo! R3						
Method	**AUC**	**nDCG**	**P@5**	**P@10**	**R@5**	**R@10**
NCF	0.7251	0.0350	0.0037	0.0036	0.0097	0.0203
IPS-NCF	0.7221	0.0322	0.0032	0.0039	0.0085	0.0219
SNIPS-NCF	0.7230	0.0310	0.0030	0.0031	0.0087	0.0175
CVIB-NCF	0.7265	0.0347	0.0036	0.0030	0.0105	0.0177
AT-NCF	0.7139	0.0333	0.0031	0.0033	0.0084	0.0179
Relevance-NCF	0.6867	0.0507	0.0071	0.0064	0.0233	0.0408
DIB-NCF	**0.7553**	**0.0686**	**0.0108**	**0.0101**	**0.0339**	**0.0630**
Product						
Method	**AUC**	**nDCG**	**P@5**	**P@10**	**R@5**	**R@10**
NCF	0.7211	0.1465	0.0174	0.0136	0.0865	0.1343
IPS-NCF	0.7284	0.1463	**0.0176**	0.0135	**0.0870**	0.1330
SNIPS-NCF	0.7257	0.1454	0.0173	0.0135	0.0856	0.1323
CVIB-NCF	0.7291	0.1440	0.0149	0.0137	0.0732	0.1350
AT-NCF	0.6814	0.1464	0.0156	0.0139	0.0774	0.1375
Relevance-NCF	0.6653	0.1404	0.0156	0.0145	0.0763	0.1423
DIB-NCF	**0.7345**	**0.1483**	0.0175	**0.0163**	0.0865	**0.1613**

AUC is the main evaluation metric.

observations: (1) when K takes a relatively small value on Product, the proposed framework is slightly worse than IPS-NCF on both precision and recall (i.e., P@5 and R@5). However, when K takes a larger value, the proposed framework has obvious performance advantages (i.e., P@10 and R@10). This means that the proposed framework still has better overall performance. (2) Even with the use of the techniques to reduce variance, the incorrect estimation of the propensity score of a small dataset will be further exacerbated. (3) AT and Relevance may not be well adapted to neural network–based models because they rely on some specific type of loss function rather than the cross-entropy loss function.

5.3 RQ2: Ablation Studies

To understand the respective effects of the last three terms in Equation (15) on model training, we conduct ablation studies on Yahoo! R3 and Product. The results are shown in Table 5. In the upper half of the sub-tables w.r.t. the two different datasets in Table 5, we show the results using matrix factorization as the backbone. For the results on Yahoo! R3, we can see that removing any term will hurt the performance in most cases, and removing term (d) leads to the worst performance. This may be due to the small size of Yahoo! R3, which causes the term (d) to play a more important role in avoiding overfitting of the model to the training set. Conversely, we can find that removing the term (d) on the larger Product dataset has less impact on most metrics except on AUC.

We present the results using neural collaborative filtering as the backbone model in the lower half w.r.t. each dataset in Table 5. We note that removing the term (d) will bring the greatest performance degradation on different data scales. One possible reason is that the neural network–based model introduces more parameters, and these parameters need to be well learned under the constraints of regularization terms. There are some unexpected cases in Table 5, i.e., when K takes a small value, the full version of the proposed framework has a slight disadvantage on a few metrics. This may be due to the noise caused by only considering AUC in parameter tuning. In general, all terms in the proposed framework can synergistically produce the greatest gain.

5.4 RQ3: Analysis of Variants

In this section, we conduct a comparative analysis of DIB and its variant rDIB. As shown in Table 6, we report the results of DIB and rDIB on Yahoo! R3 using matrix factorization and neural collaborative filtering as the backbone models. We can observe that rDIB consistently outperforms DIB on all metrics when matrix factorization is used as the backbone model. Considering neural collaborative filtering as the backbone model, rDIB also outperforms DIB in most cases, except on precision and recall when K takes small values. Overall, by relaxing the relationship between the biased and unbiased components, rDIB can further improve the model's recommendation performance. One possible reason is that the additional shared part s between the biased component r and the unbiased component z can accommodate the acceptability of the bias by different users.

5.5 RQ4: Model Analysis

In this section, we conduct some more in-depth analysis on the proposed framework to better understand the properties of the proposed framework.

5.5.1 Visualization of the Biased Component r and the Unbiased Component z. A key question is whether the biased component r and the unbiased component z in the proposed framework have a pattern that meets the expectation, i.e., they gradually become independent in training. To answer this question, at several training points, we use the *t-SNE* [41] method to visualize the biased and unbiased components. The results are shown in Figure 5. We can see that as the training progresses, there is a clear separation between the biased component in orange and the unbiased component

Table 5. Results of the Ablation Studies Using MF and NCF as the Backbone Models, where the Best Results and the Second Best Results are Marked in Bold and Underlined, Respectively

Yahoo! R3						
Method	AUC	nDCG	P@5	P@10	R@5	R@10
DIB-MF	**0.7602**	**0.0932**	**0.0177**	**0.0151**	0.0566	**0.0960**
w/o term (b)	0.7505	0.0893	0.0175	0.0138	0.0563	0.0909
w/o term (c)	0.7545	0.0915	0.0173	0.0142	**0.0569**	0.0937
w/o term (d)	0.7342	0.0769	0.0144	0.0117	0.0478	0.0737
DIB-NCF	**0.7553**	**0.0686**	**0.0108**	**0.0101**	**0.0339**	**0.0630**
w/o term (b)	0.7326	0.0553	0.0089	0.0081	0.0271	0.0489
w/o term (c)	0.7373	0.0592	0.0097	0.0093	0.0292	0.0585
w/o term (d)	0.7243	0.0597	0.0102	0.0099	0.0318	0.0603
Product						
Method	AUC	nDCG	P@5	P@10	R@5	R@10
DIB-MF	**0.7365**	**0.0819**	**0.0198**	**0.0173**	**0.0965**	**0.1688**
w/o term (b)	0.7173	0.0546	0.0126	0.0115	0.0614	0.1112
w/o term (c)	0.7156	0.0511	0.0109	0.0113	0.0527	0.1083
w/o term (d)	0.6809	0.0719	0.0178	0.0153	0.0867	0.1504
DIB-NCF	**0.7345**	**0.1483**	0.0175	**0.0163**	0.0865	**0.1613**
w/o term (b)	0.7274	0.1474	0.0181	0.0131	0.0901	0.1294
w/o term (c)	0.7276	0.1474	**0.0181**	0.0131	**0.0901**	0.1294
w/o term (d)	0.7133	0.1438	0.0177	0.0123	0.0876	0.1214

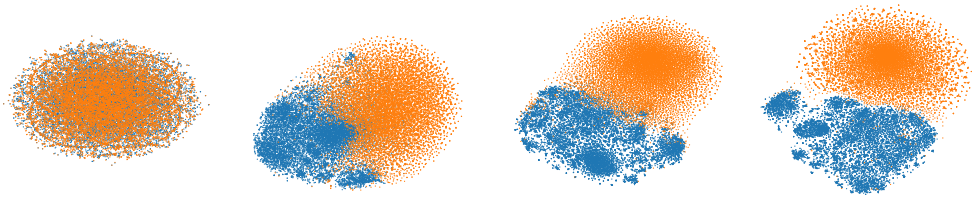
AUC is the main evaluation metric.

Table 6. Comparison Results of DIB and Its Variant rDIB using MF and NCF as the Backbone Models on Yahoo! R3, where the Best Results are Marked in Bold

Method	AUC	nDCG	P@5	P@10	P@50	R@5	R@10	R@50
DIB-MF	0.7602	0.0932	0.0177	0.0151	0.0083	0.0566	0.0960	0.2526
rDIB-MF	**0.7611**	**0.0977**	**0.0180**	**0.0155**	**0.0084**	**0.0585**	**0.0998**	**0.2565**
improvement	0.12%	4.83%	1.69%	2.65%	1.21%	3.36%	3.96%	1.54%
DIB-NCF	0.7553	0.0686	0.0108	0.0101	0.0068	**0.0339**	**0.0630**	0.2046
rDIB-NCF	**0.7605**	**0.0706**	0.0108	**0.0103**	**0.0071**	0.0336	0.0621	**0.2149**
improvement	0.69%	2.92%	0.00%	1.98%	4.41%	−0.88%	−1.43%	5.03%

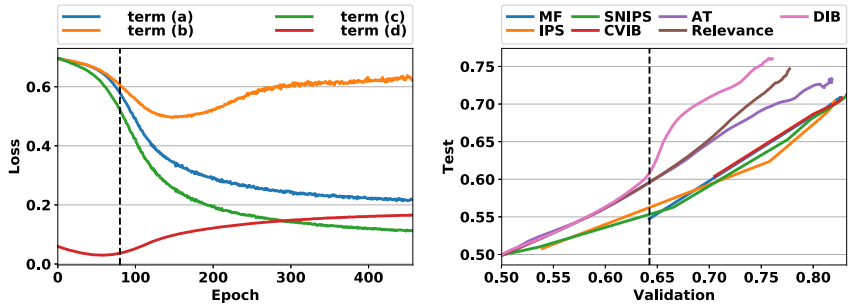
in blue. This verifies the effectiveness of the proposed framework. We also note that the unbiased component is looser than the biased component, and a small part of the unbiased component is still mixed with the biased component. This may be because the unbiased component needs to learn the personalized differences between the users, among whom there are still some users who are difficult to be distinguished.

5.5.2 Visualization of the Loss of Each Term in $\mathcal{L}_{DIB}$. We next analyze the change trend of the corresponding loss for each term in $\mathcal{L}_{DIB}$. The results are shown in Figure 6(a). Early in training ($Epoch \leq 80$), the model is tuned from an initialization to a transition state, and then is fine-tuned as the components are separated. A sufficient reduction in the loss of term (a) guarantees the basic performance of the proposed method (i.e., the performance corresponding to the



(a) Initialization (b) Epoch=100 (c) Epoch=200 (d) Convergence

Fig. 5. Dynamic visualization of the unbiased component z and the biased component r as training progresses, in which the orange and blue points are used to denote the biased component r and the unbiased component z, respectively.



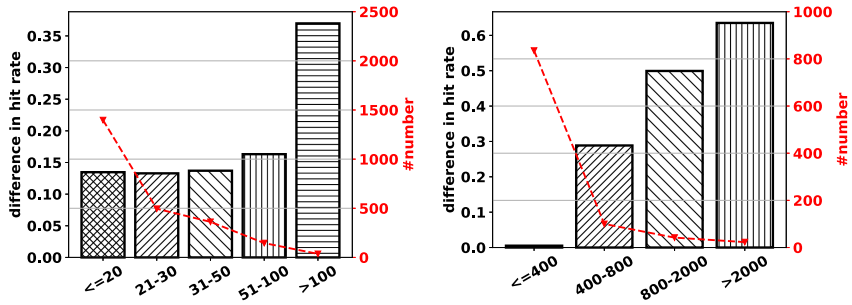
(a) Loss (b) AUC results

Fig. 6. (a) The change trend of the corresponding loss to each term of $\mathcal{L}_{DIB}$ during the training progresses, where the black dotted line indicates $Epoch = 80$. (b) During training, the AUC results of each method on the validation set and the corresponding AUC results on the test set, where the black dotted line indicates the result of DIB on the validation set at the 80th epoch.

unbiased component z). By comparing the trends of term (b) and term (c), we can see that using the biased component r alone will lead to a very poor result, but as long as it is combined with the unbiased component z, the best result can be achieved. This is reasonable, because the biased component itself does not reflect a user's preferences well, but the combination with the unbiased component will lead to an over-approximation effect on the observed feedback. The above training trends are in line with our analysis in Section 4.2.4 about the effectiveness of our DIB. In addition, the trend of term (d) indicates that the compression term is helpful in learning a more ideal unbiased component throughout the training process.

5.5.3 Performance Trends of All Methods on Validation Set and Test Set. In our situation, only biased feedback is available in the training phase and the validation set with biased properties and the test set with unbiased properties are different in distribution. Therefore, a good debiasing method should avoid over-approximation to the validation set. We show the paired AUC results on validation and test sets of all methods during training in Figure 6(b). We can find that DIB achieves a better balanced result on the validation and test sets, especially after the biased and unbiased components are clearly separated. This again verifies the effectiveness of our DIB for debiasing through the introduced biased and unbiased components. Furthermore, it also matches our analysis in Section 4.2.4 for why our DIB is effective.

5.5.4 Sources of Performance Gains in DIB. We further analyze the sources of performance gains in DIB. First, all users (or items) are grouped according to their activity (or popularity). For



(a) User activity (b) Item popularity

Fig. 7. (a) Differences on hit rate between DIB and the base model (i.e., MF) in different user groups, defined by the number of users contained in each group. (b) Differences on hit rate between DIB and the base model (i.e., MF) in different item groups according to the number of items contained in each group.

Table 7. In Different User Groups and Item Groups, the Coverage of Hit Rate of DIB and the Base Model, where the Coverage Rate Refers to the Proportion of Users and Items with a Hit Rate Greater Than 0

	User activity					**Item popularity**			
Method	<=20	21–30	31–50	51–100	>100	<=400	400–800	800–2,000	>2,000
Base	0.1582	0.1818	0.1934	0.1986	0.0556	0.1341	0.1500	0.3810	0.8696
DIB	**0.3508**	**0.3677**	**0.3950**	**0.4521**	**0.6667**	**0.1653**	**0.7700**	**0.9762**	**1.0000**

each user (or item) group, we calculate the mean of the difference on hit rate between DIB and the base model (i.e., MF). A user's hit rate is the proportion of positive items in the test set that are correctly recommended to that user, and an item's hit rate is similarly defined. The results are shown in Figure 7(a) and (b). We can observe that DIB achieves significant performance improvements across all user groups and most item groups, especially for users with high activity and items with high popularity. Next, we count the coverage of the hit rate of DIB and the base model in different user (or item) groups, where the coverage rate refers to the proportion of users (or items) with a hit rate greater than 0. The results are shown in Table 7, and we can find that DIB has a clear advantage in coverage across all groups. In particular, we notice that DIB also has a coverage improvement in the first group of item popularity, and this means that in Figure 7(b), DIB's slight improvement in terms of the average hit rate for this group is due to the hit rate being amortized across more items. From the above results, we can find that our DIB can effectively obtain the unbiased embeddings of the users and items, so that it can more accurately capture a user's true preferences on an item.

5.5.5 Analysis of Recommendation Distribution and Utility. Next, we analyze the recommendation distribution and the corresponding utility of our DIB and other baselines. We first show the recommendation distribution and utility on the randomized data in Yahoo! R3 in Figure 8(a), where popular items are defined as the top 20% most frequent items in the dataset, and the rest are the unpopular items. We can find that although the global recommendation distribution is almost uniform, i.e., the probability of popular and unpopular items being shown is also 2:8, the utility of popular items is significantly higher than that of unpopular items. This means that many users' decision-making behaviors are still popularity driven. Therefore, blindly reducing one particular bias, such as the popularity bias, can be counterproductive. We show the recommendation distribution and utility of our DIB and other baselines in Figures 8(b) and (c). We can find that

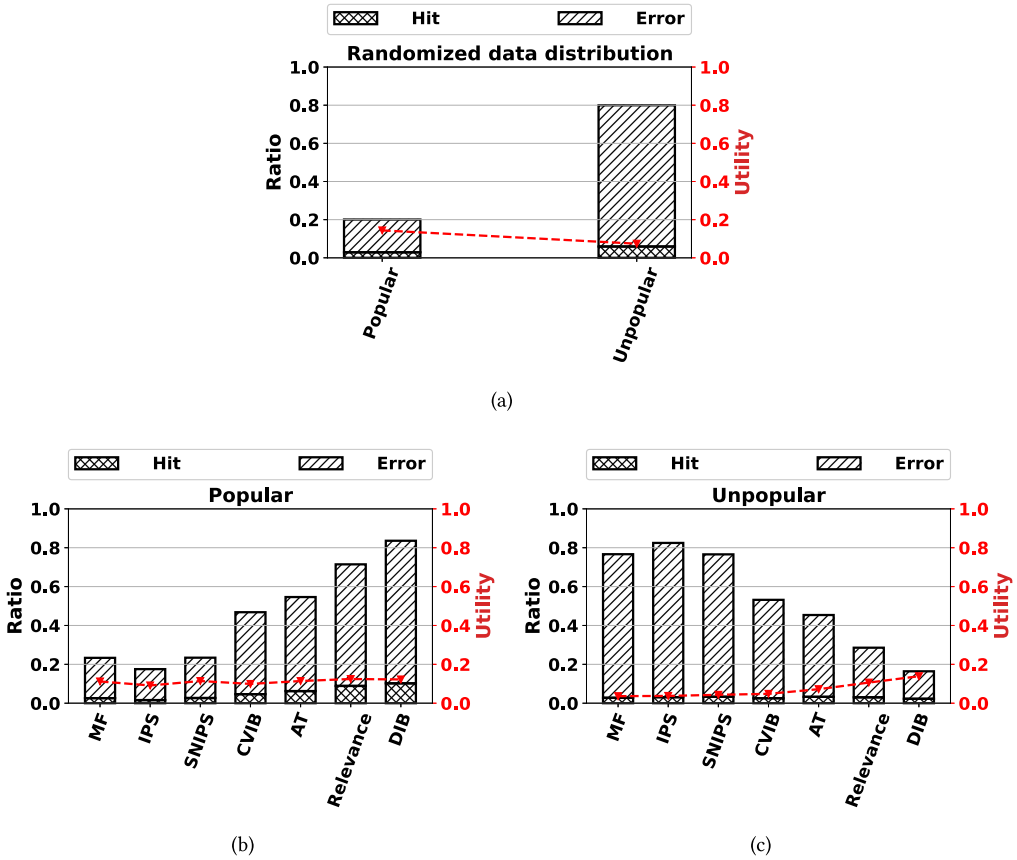


Fig. 8. On Yahoo! R3, recommendation distribution and utility of the randomized data (a), and recommendation distribution and utility of different methods on its popular items (b) and unpopular items (c).

AT, Relevance, and our DIB, which perform better in unbiased tests, largely retain a user's popularity-oriented preferences instead of suppressing them blindly. In addition, the utility of all methods on popular items is not significantly different. On the other hand, although AT, Relevance, and our DIB show a smaller fraction of unpopular items, they have a clear utility boost, in particular of our DIB. This means that unpopular items are more susceptible to the offset on the feedback label caused by the confounding bias, while our DIB can more accurately identify a user's true preferences on unpopular items, i.e., the confounding bias is more effectively mitigated.

5.5.6 Analysis of Item Embedding. Many of the above results have shown that our DIB can obtain unbiased embeddings of the users and items to more accurately capture a user's true preferences on an item. Therefore, we take the obtained item embeddings as an example to further analyze the difference between our DIB and other baselines. More specifically, inspired by previous works showing that the norms of parameters associated with different categories can reflect the bias between them [5], we compute the norms of item embeddings obtained in each method separately. Note that we have normalized the norms of all item embeddings within each method for ease of comparison. The results are shown in Figure 9, and the x-axis has been arranged in ascending order. We can find that a traditional recommendation model (i.e., MF) focuses on extremely popular items and is confused on other items, which are more susceptible to the confounding bias.

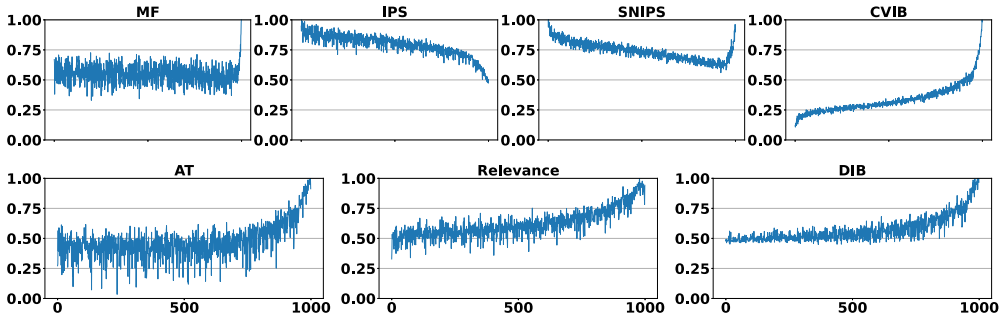
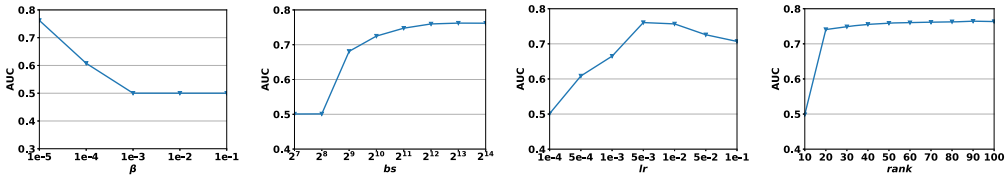
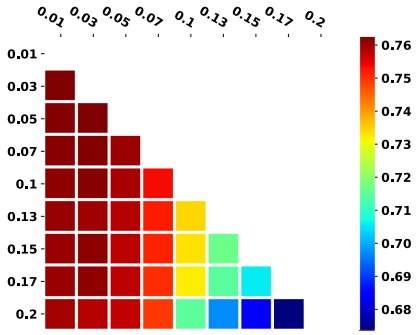


Fig. 9. Visualization of the corresponding norms for the obtained item embedding representations.



(a) Regularization weight (b) Batch size (c) Learning rate (d) Embedding dimension



(e) Loss weight

Fig. 10. In the process of parameter search, the results of DIB with different values of parameters, including (a) regularization weight β, (b) batch size bs, (c) learning rate lr, (d) embedding dimension $rank$, and (e) loss weights γ and α.

IPS over-corrects the distribution, and SNIPS maintains the dominance of popular items by reducing the variance of estimates. Methods with better performance, i.e., AT, Relevance, and our DIB, aim to promote a smoother curve, where most items have a similar norm and grow gently with the popularity of the items. In particular, benefiting from the idea of debiased representation learning, our DIB can obtain a trend with less fluctuation.

5.5.7 Sensitivity Analysis of Parameters. Finally, we analyze the results of DIB under parameters with different values during parameter search. We first consider the regularization weight β, batch size bs, learning rate lr, and embedding dimension $rank$ in Table 2 and report the results in Figures 10(a), (b), (c), and (d), respectively. We can find that a smaller regularization weight, a larger batch size, and a moderate learning rate are better choices. In addition, the impact of the

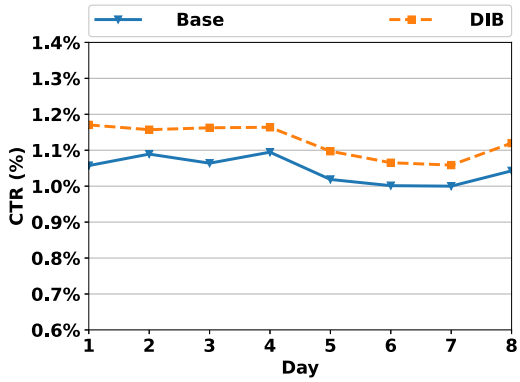


Fig. 11. The results of A/B testing of the original baseline model and the baseline model integrated with DIB in one cycle.

embedding dimension on DIB is milder. We then consider the two most important parameters in DIB, i.e., the loss weights γ and α, and the results are shown in Figure 10(e). The y-axis and x-axis represent the values of γ and $\gamma - \alpha$, respectively. We can observe that the performance of DIB depends on the relationship between γ and $\gamma - \alpha$, where a better choice is for $\gamma - \alpha$ to take a small value, i.e., constrain γ and α to have similar values.

5.6 RQ5: Results in Real-World Recommendation Scenarios

In this section, we describe the online results of deploying DIB in a real-world commercial recommender system. The scenario is video recommendation in a news feed module on a commercial browser. The adopted training data involves 120 million records of 20 million users in the system, and a feature set with more than 100 fields is designed. We integrate DIB by taking a baseline model deployed online in the system as the backbone model. This baseline model is based on the main architecture of DCN-V2 [43] with some self-developed small modified modules. DCN-V2 is one of the commonly used models for click-through rate prediction in industrial recommendations, which contains a cross network and a deep network, and can learn the feature interactions efficiently. We randomly select 2% of online users as subjects for A/B testing, of which 1% are divided into the experimental group and the rest are divided into the control group. The control group uses the existing baseline model, while the experimental group uses the baseline model with integrated DIB. We show the results of A/B testing in one cycle in Figure 11[2] and can find that DIB can consistently and effectively improve the performance of the deployed baseline model.

6 CONCLUSIONS AND FUTURE WORK

In this article, we describe the generation process of the biased and unbiased feedback in recommender systems via two respective causal diagrams, and then define a set of system-induced biases resulting from their differences as confounding bias. When only the biased feedback is available, we analyze the conditions that need to be met to alleviate the confounding bias, and then propose a new perspective on debiased representation learning to solve these conditions. Specifically, we propose a DIB framework and its variant under relaxed conditions to perform this optimization process based on the guidance of information theory. Moreover, we derive a tractable solution for the proposed framework and its variant. We conduct extensive experiments on a public dataset

[2]Note that due to confidentiality requirements, we scale the actual metrics.

and a real product dataset to verify the effectiveness of the proposed framework, including unbiased evaluation, ablation studies, analysis of the variant, in-depth analysis of the model, and A/B testing.

For future works, we plan to extend the proposed framework to scenarios where multiple biased data are available, such as cross-domain recommendation and multi-task optimization. We are also interested in how to extend the proposed framework with an additional small unbiased data, which will further expand the applicability of the framework in debiasing recommendation. Finally, we will also investigate other solutions for debiased representation learning.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their expert and constructive comments and suggestions.

REFERENCES

[1] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2017. Controlling popularity bias in learning-to-rank recommendation. In *Proceedings of the 11th ACM Conference on Recommender Systems*. 42–46.

[2] Alessandro Achille and Stefano Soatto. 2018. Information dropout: Learning optimal representations through noisy computation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 12 (2018), 2897–2905.

[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 12th ACM International Conference on Web Search and Data Mining*. 474–482.

[4] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2623–2631.

[5] Shaden Alshammari, Yu-Xiong Wang, Deva Ramanan, and Shu Kong. 2022. Long-tailed recognition via weight balancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6897–6907.

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

[7] Rocío Cañamares and Pablo Castells. 2018. Should I follow the crowd?: A probabilistic analysis of the effectiveness of popularity in recommender systems. In *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*. 415–424.

[8] Pengyu Cheng, Martin Renqiang Min, Dinghan Shen, Christopher Malon, Yizhe Zhang, Yitong Li, and Lawrence Carin. 2020. Improving disentangled text representation learning with information-theoretic guidance. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 7530–7541.

[9] Bin Dai, Chen Zhu, Baining Guo, and David Wipf. 2018. Compressing neural networks using the variational information bottleneck. In *Proceedings of the 35th International Conference on International Conference on Machine Learning*. 1135–1144.

[10] Marco Federici, Anjan Dutta, Patrick Forré, Nate Kushman, and Zeynep Akata. 2020. Learning robust representations via multi-view information bottleneck. In *Proceedings of the 8th International Conference on Learning Representations*.

[11] Ruoyuan Gao and Chirag Shah. 2020. Counteracting bias and increasing fairness in search and recommender systems. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 745–747.

[12] Yingqiang Ge, Shuya Zhao, Honglu Zhou, Changhua Pei, Fei Sun, Wenwu Ou, and Yongfeng Zhang. 2020. Understanding echo chambers in e-commerce recommender systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2261–2270.

[13] Yingqiang Ge, Xiaoting Zhao, Lucia Yu, Saurabh Paul, Diane Hu, Chu-Cheng Hsieh, and Yongfeng Zhang. 2022. Toward Pareto efficient fairness-utility trade-off in recommendation through reinforcement learning. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining*. 316–324.

[14] Prem Gopalan, Jake M. Hofman, and David M. Blei. 2015. Scalable recommendation with hierarchical Poisson factorization. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*. 326–335.

[15] Negar Hassanpour and Russell Greiner. 2020. Learning disentangled representations for counterfactual regression. In *Proceedings of the 8th International Conference on Learning Representations*.

[16] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 27th International Conference on World Wide Web*. 173–182.

[17] Ayush Jaiswal, Rob Brekelmans, Daniel Moyer, Greg Ver Steeg, Wael AbdAlmageed, and Premkumar Natarajan. 2019. Discovery and separation of features for invariant representation learning. https://arxiv.org/abs/1912.00646.

[18] Ray Jiang, Silvia Chiappa, Tor Lattimore, András György, and Pushmeet Kohli. 2019. Degenerate feedback loops in recommender systems. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*. 383–390.
[19] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.
[20] Kun Kuang, Peng Cui, Bo Li, Meng Jiang, Shiqiang Yang, and Fei Wang. 2017. Treatment effect estimation with data-driven variable decomposition. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. 140–146.
[21] Dawen Liang, Laurent Charlin, and David M. Blei. 2016. Causal inference for recommendation. In *Workshop on Causation: Foundation to Application Co-located with the 32nd Conference on Uncertainty in Artificial Intelligence*.
[22] Dawen Liang, Laurent Charlin, James McInerney, and David M. Blei. 2016. Modeling user exposure in recommendation. In *Proceedings of the 25th International Conference on World Wide Web*. 951–961.
[23] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 27th International Conference on World Wide Web*. 689–698.
[24] Chen Lin, Dugang Liu, Yanghua Xiao, and Hanghang Tong. 2020. Spiral of silence and its application in recommender systems. *IEEE Transactions on Knowledge and Data Engineering* 34, 6 (2020), 2934–2947.
[25] Dugang Liu, Pengxiang Cheng, Zhenhua Dong, Xiuqiang He, Weike Pan, and Zhong Ming. 2020. A general knowledge distillation framework for counterfactual recommendation via uniform data. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 831–840.
[26] Dugang Liu, Pengxiang Cheng, Hong Zhu, Zhenhua Dong, Xiuqiang He, Weike Pan, and Zhong Ming. 2021. Mitigating confounding bias in recommendation via information bottleneck. In *Proceedings of the 15th ACM Conference on Recommender Systems*. 351–360.
[27] Dugang Liu, Chen Lin, Zhilin Zhang, Yanghua Xiao, and Hanghang Tong. 2019. Spiral of silence in recommender systems. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*. 222–230.
[28] Benjamin M. Marlin and Richard S. Zemel. 2009. Collaborative prediction and ranking with non-random missing data. In *Proceedings of the 3rd ACM Conference on Recommender Systems*. 5–12.
[29] Daniel Moyer, Shuyang Gao, Rob Brekelmans, Greg Ver Steeg, and Aram Galstyan. 2018. Invariant representations without adversarial training. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 9102–9111.
[30] 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 29th International Conference on World Wide Web*. 1863–1873.
[31] Sonali Parbhoo, Mario Wieser, and Volker Roth. 2018. Causal deep information bottleneck. https://arxiv.org/abs/1807.02326v1.
[32] 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.
[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 ACM 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 *Proceedings of the 33rd International Conference on Machine Learning*. 1670–1679.
[36] Zihua Si, Xueran Han, Xiao Zhang, Jun Xu, Yue Yin, Yang Song, and Ji-Rong Wen. 2022. A model-agnostic causal learning framework for recommendation using search data. In *Proceedings of the 31st International Conference on World Wide Web*.
[37] Yixin Su, Rui Zhang, Sarah Erfani, and Zhenghua Xu. 2021. Detecting beneficial feature interactions for recommender systems. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*. 4357–4365.
[38] Adith Swaminathan and Thorsten Joachims. 2015. The self-normalized estimator for counterfactual learning. In *Proceedings of the 29th International Conference on Neural Information Processing Systems*. 3231–3239.
[39] Thomas M. Cover and Joy A. Thomas. 2006. *Elements of Information Theory*. Wiley-Interscience.
[40] Naftali Tishby, Fernando C Pereira, and William Bialek. 1999. The information bottleneck method. In *Proceedings of the 37th Annual Allerton Conference on Communications, Control and Computing*. 368–377.
[41] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, 11 (2008), 2579–2605.
[42] Qi Wan Wan, Xiangnan He, Xiang Wang, Jiancan Wu, Wei Guo, and Ruiming Tang. 2022. Cross pairwise ranking for unbiased item recommendation. In *Proceedings of the 31st International Conference on World Wide Web*.
[43] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 2021. DCN V2: Improved deep and cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the 30th International Conference on World Wide Web*. 1785–1797.

[44] Wenjie Wang, Fuli Feng, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. 2021. Clicks can be cheating: Counterfactual recommendation for mitigating clickbait issue. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1288–1297.

[45] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. 2018. Position bias estimation for unbiased learning to rank in personal search. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. 610–618.

[46] Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. 2021. Combating selection biases in recommender systems with a few unbiased ratings. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 427–435.

[47] Zifeng Wang, Xi Chen, Rui Wen, Shao-Lun Huang, Ercan E. Kuruoglu, and Yefeng Zheng. 2020. Information theoretic counterfactual learning from missing-not-at-random feedback. In *Advances in Neural Information Processing Systems*. 1854–1864.

[48] 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 and Data Mining*. 1791–1800.

[49] Teng Xiao and Suhang Wang. 2022. Towards unbiased and robust causal ranking for recommender systems. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining*. 1158–1167.

[50] Shuyuan Xu, Yingqiang Ge, Yunqi Li, Zuohui Fu, Xu Chen, and Yongfeng Zhang. 2021. Causal collaborative filtering. https://arxiv.org/abs/2102.01868.

[51] Haiqin Yang, Guang Ling, Yuxin Su, Michael R. Lyu, and Irwin King. 2015. Boosting response aware model-based collaborative filtering. *IEEE Transactions on Knowledge and Data Engineering* 27, 8 (2015), 2064–2077.

[52] Jiangxing Yu, Hong Zhu, Chih-Yao Chang, Xinhua Feng, Bowen Yuan, Xiuqiang He, and Zhenhua Dong. 2020. Influence function for unbiased recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1929–1932.

[53] 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.

[54] Shuxi Zeng, Murat Ali Bayir, Joel Pfeiffer, Denis Charles, and Emre Kiciman. 2021. Causal transfer random forest: Combining logged data and randomized experiments for robust prediction. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 211–219.

[55] Ziwei Zhu and James Caverlee. 2022. Fighting mainstream bias in recommender systems via local fine tuning. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining*. 1497–1506.

Received 31 March 2022; revised 10 August 2022; accepted 25 September 2022