KDCRec: Knowledge Distillation for Counterfactual Recommendation via Uniform Data

Dugang Liu, Pengxiang Cheng, Zinan Lin, Jinwei Luo, Zhenhua Dong, Xiuqiang He, Weike Pan, and Zhong Ming

Abstract—The bias problems in recommender systems are an important challenge. In this paper, we focus on solving the bias problems via uniform data. Previous works have shown that simple modeling with a uniform data can alleviate the bias problems and improve the performance. However, the uniform data is usually few and expensive to collect in a real product. In order to use the valuable uniform data more effectively, we propose a novel and general knowledge distillation framework for counterfactual recommendation with four specific methods, including label-based distillation, feature-based distillation, sample-based distillation and model structure-based distillation. Moreover, we discuss the relation between the proposed framework and the previous works. We then conduct extensive experiments on both public and product datasets to verify the effectiveness of the proposed four methods. In addition, we explore and analyze the performance trends of the proposed methods on some key factors, and the changes in the distribution of the recommendation lists. Finally, we emphasize that counterfactual modeling with uniform data is a rich research area, and list some interesting and promising research topics worthy of further exploration. Note that the source codes are available at https://github.com/dgliu/TKDE_KDCRec.

Index Terms—Counterfactual, bias, recommender systems, knowledge distillation, uniform data

1 Introduction

Recommender Systems as a feedback loop system may suffer from the bias problems such as popularity bias [2], [3], previous model bias [4], [5], [6] and position bias [7], [8]. Previous studies have shown that models and evaluation metrics that ignore the biases do not reflect the true performance of a recommender system, and that explicitly handling of the biases may help improve the performance [6], [7], [9]. Most of the previous works to solve the bias problems of recommender systems can be classified as heuristic-based, counterfactual learning-based [9], [10] and some theoretical tools-based methods [11], [12]. The former mainly makes certain assumptions about the data being missing not at random (MNAR) [5], [13], while the latter two mainly use the inverse propensity score (IPS) [10], [14] and the causal inference techniques [11], [12], respectively.

A recent work has shown that a uniform data can alleviate the previous model bias problem [6]. But the uniform data is usually few and expensive to collect in real recommender systems. To collect a uniform data, we must intervene in the system by using a uniform logging policy instead of a stochastic recommendation policy, that is, for each user's request, we do not use the recommendation model for item delivery, but instead randomly select some items from all the candidate items and rank them with a uniform distribution. The uniform data can then be regarded as a good unbiased agent because it isolates the source of bias at the system level as much as possible. However, the uniform logging policy would hurt the users' experiences and the revenue of the platform. This means that it is necessary to constrain the uniform data collection within a particularly small traffic (e.g., 1%).

In this article, we focus on how to solve the bias problems in a recommender system with a uniform data. Since biased data and uniform data can be regarded as the treatment group and the control group in the field of causal counterfactual [15], respectively, we call the studied problem as counterfactual recommendation via uniform data. Along the line of [6], in the conference version of this work [1], we conduct empirical studies on a real advertising system and a public dataset to validate the usefulness of the uniform data, where the uniform data is simply combined with the non-uniform data for training models. We have observed that such a simple method can alleviate the bias and improve the performance, which motivates us to study more advanced methods that can make better use of the uniform data. Although there are many ways to extract information or knowledge from a uniform data, in this paper we focus on knowledge distillation because of its simplicity and flexibility.

To use the few and valuable uniform data more effectively, we propose a general knowledge distillation framework for counterfactual recommendation (KDCRec), which enables

• *Dugang Liu, Zinan Lin, Jinwei Luo, Weike Pan, and Zhong Ming are with the College of Computer Science and Software Engineering and the National Engineering Laboratory for Big Data System Computing Technology, Shenzhen University, Shenzhen 518060, China. E-mail: {dugang.ldg, lzn87591}@gmail.com, luojinwei2016@email.szu.edu.cn, {panweike, mingz}@szu.edu.cn.*
• *Pengxiang Cheng, Zhenhua Dong, and Xiuqiang He are with Huawei Noah's Ark Lab, Shenzhen 518010, China. E-mail: {chengpengxiang1, dongzhenhua, hexiuqiang1}@huawei.com.*

Manuscript received 20 December 2020; revised 7 April 2022; accepted 6 August 2022. Date of publication 17 August 2022; date of current version 21 June 2023. This work was supported by the National Natural Science Foundation of China under Grants 61836005 and 62172283.
(Corresponding authors: Weike Pan and Zhong Ming.)
Recommended for acceptance by H. Lee.
Digital Object Identifier no. 10.1109/TKDE.2022.3199585

uniform data modeling with four methods, i.e., label-based distillation, feature-based distillation, sample-based distillation and model structure-based distillation. Each one is based on a specific concern, i.e., label-based distillation focuses on using the imputed labels as a carrier to provide useful debiasing guidance; feature-based distillation aims to filter out the representative unbiased features; sample-based distillation considers mutual learning and alignment of the information of the uniform and non-uniform data; and model structure-based distillation constrains the training of the models from the perspective of embedded representation.

2 Related Work

Since we study how to apply knowledge distillation techniques for counterfactual recommendation to solve the bias problems, we first review some related works on general knowledge distillation, and then summarize some related work on debiasing recommendation.

2.1 Knowledge Distillation

Hinton's work first proposes the concept of knowledge distillation [16]. By introducing soft targets related to teacher networks as part of the objective function, the training of student networks is guided to achieve knowledge transfer [17]. A series of follow-up works develop different distillation structures (e.g., multiple teachers [18] and cascade distillations [19]) and different forms of knowledge (e.g., alignment of the hidden layers [20] or the relation between the hidden layers [21]). Some recent works are no longer limited to model structure, but consider sample-based knowledge distillation [22], [23]. In this article, we further expand the definition of distillation to include label-based and feature-based forms. The marriage of knowledge distillation and recommender systems has also attracted the attention of the researchers [24], [25], [26]. Most of these works focus on using knowledge distillation to extract some useful knowledge from some auxiliary models to enhance the performance or interpretability of the target recommendation model. In this paper, we focus on using knowledge distillation to solve the bias problems in recommender systems.

2.2 Debiasing Recommendation

Recommender systems generate various bias problems in the feedback loop between the users and system, and how to effectively alleviate these biases is a key issue to be addressed [27]. Based on the availability of a uniform data, the existing methods for debiasing recommendation can be mainly classified into two routes, i.e., debiasing recommendation without uniform data and debiasing recommendation with uniform data.

The former mainly considers the debiasing process directly on the biased log data, and can further be categorized into three classes, including heuristic-based, counterfactual learning-based, and some theoretical tools-based methods. The heuristic-based method links a user's feedback with different specific factors to model the non-random missing data, such as item features [28], [29] and user ratings [5], [30]. The counterfactual learning-based method adopts the inverse propensity score and the doubly robust techniques, and estimates the sample weights on the log data to correct the biased feedback distribution [10], [14], [31]. Moreover, some recent works consider introducing some theoretical tools to integrate and solve the bias problems, such as regularization methods [32], information bottleneck [33], [34], positive-unlabeled learning [35], asymmetric tri-training [36], disentangled representation learning [37], and causal inference techniques [11], [12], [38]. Due to the lack of target knowledge, most methods on this route require more complex designs for a specific bias.

The latter additionally introduces a uniform data as the target knowledge to guide the training of a debiasing model, and can be divided into the following three categories: 1) use a uniform data to estimate sample weights with some inverse propensity score techniques, or train an imputation model for data augmentation with some doubly robust techniques [10], [31], [39], [40]; 2) design a multi-stage training framework that alternately uses the log data and the uniform data to learn debiasing parameters [41], [42], [43]; 3) use the log data and the uniform data to jointly train the two models and constrain their proximity in some way [1], [44]. Existing works only consider the use of the uniform data from a specific perspective. In this paper, we propose a general and flexible debiasing framework based on the idea of knowledge distillation, which can provide a systematic guidance on how to effectively use the uniform data from four different perspectives. In particular, the main ideas of most existing methods can be linked to the proposed framework. A detailed discussion can be found in Section 3.5. It is worth mentioning that the counterfactual learning method is used in both routes because of its theoretical insight and generalization, but the propensity score suffers from serious high variance and is thus difficult to be accurately estimated in practice.

3 The Proposed Framework

In the pilot experiments of the conference version [1], we have observed that training the model on a simply combined uniform data and biased data can alleviate the bias problems. To provide a systematic guide on how to effectively use the uniform data, we propose a general **K**nowledge **D**istillation framework for **C**ounterfactual **R**ecommendation in this section, **KDCRec** for short. Fig. 1 shows the overview of the framework of our KDCRec. In our framework, the uniform data can be modeled with four different methods, including label-based distillation, feature-based distillation, sample-based distillation, and model structure-based distillation. Note that we use a general definition of distillation in the study rather than the past knowledge distillation approaches such as considering the level of sample [22], [23] and model structure [16], [20]. These methods are based on different concerns to mine the potentially useful knowledge from the uniform data, which will be used to improve the learning of the biased data.

Next, we will introduce the four methods in turn as different modules. More specifically, in each module, we will give a formal definition of the corresponding method, and list some practical solutions under the guidance of the definition. Note that this section is an extended version of the framework described in [1]. We omit the introduction of some old strategies in [1] and add some new and more

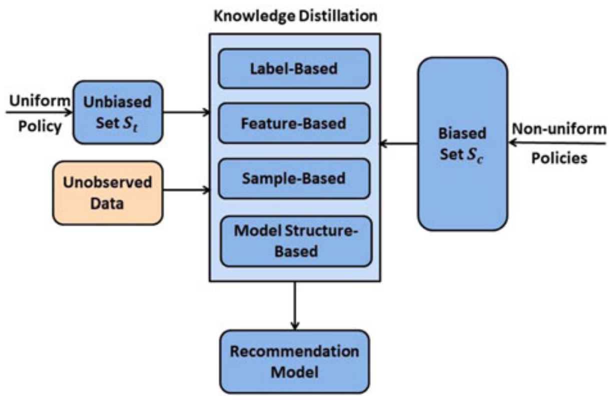


Fig. 1. Overview of the KDCRec framework. The scale of the biased set S_c is much larger than that of the unbiased set S_t. Since the unobserved data is only used in some modules, we distinguish it from S_c and S_t using a different color.

effective improvement strategies. To further distinguish the new strategy from the old strategy, we will add "NEW" as a mark to the new strategy. Please refer to [1] for an introduction to all the old strategies. Note that in the experiments we have two forms to obtain the desired model. The first is to pre-train a model M_c on non-uniform data, and then perform debiasing through fine-tuning. The second is to train a model from initialization and debias it during the training process. M_k is used to denote this model to distinguish it from M_c. For ease of description, the main notations are listed in Table 1.

TABLE 1
The Main Notations and Explanations

Symbol	Meaning
$\mathcal{D}$	the whole set of data, i.e., $\mathcal{D} = \mathcal{S}_t \cup \mathcal{S}_c \cup \text{unobserved data}$
S_t	the uniform data
S_c	the non-uniform data
M_t	a model pre-trained on S_t in a supervised manner
M_c	a model pre-trained on S_c in a supervised manner
M_k	an initialized model trained on $S_c \cup S_t$
(i,j)	a sample associated with user i and item j
y_{ij}	the true label for the sample (i,j)
$\hat{y}^c_{ij}$	the predicted label of M_c for the sample (i,j)
$\hat{y}^t_{ij}$	the predicted label of M_t for the sample (i,j)
$\hat{y}^k_{ij}$	the predicted label of M_k for the sample (i,j)
$\mathcal{W}$	model parameters
ℓ	an arbitrary loss function
λ	the parameters of the regularization
α	a tunable parameter that controls the importance of loss
τ	temperature parameter

3.1 Label-Based Module

Models trained on a non-uniform data S_c tend to produce biased predictions, while predictions from a uniform data S_t are more unbiased. An intuitive idea is that when training a model on S_c, the model may receive the imputed labels produced by S_t to correct the bias of its own predictions. Based on this idea, we develop the following formal definition of label-based distillation. Note that on the premise of using the imputed labels, we can also include the labels of S_t. We emphasize the use of the imputed labels to avoid confusion with other distillation methods.

Definition 1 (D1). *A method can be classified as* label-based distillation *if and only if the training of a non-uniform data S_c can benefit from the imputed labels produced by a uniform data S_t.*

Solutions. We will use the three strategies adopted in our experiments as examples to illustrate how label-based distillation can be realized.

- *Variant 1 of Bridge Strategy (NEW, Bridge-var1 for short).* Let $\mathcal{D}$ denote the whole set of data, including the non-uniform data S_c, the uniform data S_t and the unobserved data. Recalling the Bridge strategy in [1], we consider a scenario where only the models M_c and M_t are used for joint debiasing training, and then M_c is used for recommendation. Different from the original version [1], we consider introducing a third model M_k to learn from S_c and S_t, and make recommendations after debiasing training. Similarly, to make full use of the available knowledge, we also randomly sample an auxiliary set S_a from $\mathcal{D}$ as a bridge in each iterative training, and expect the predicted output of M_c, M_t and M_k on S_a to be close. Note that most of the samples in S_a are unobserved data because of the data sparsity in recommender systems. Since the knowledge of S_c, S_t, M_c, and M_t is comprehensively considered and balanced, we expect that this strategy can reduce the bias of M_k. The final objective function of this strategy is,

$$\min_{\mathcal{W}_c,\mathcal{W}_t,\mathcal{W}_k} \frac{1}{|S_c|}\sum_{(i,j)\in S_c} \ell\Big(y_{ij},\hat{y}^k_{ij}\Big) + \frac{1}{|S_t|}\sum_{(i,j)\in S_t} \ell\Big(y_{ij},\hat{y}^k_{ij}\Big) + \frac{1}{|S_a|}\sum_{(i,j)\in S_a}\left[\frac{1}{2}\ell\Big(\hat{y}^c_{ij},\hat{y}^k_{ij}\Big) + \frac{1}{2}\ell\Big(\hat{y}^t_{ij},\hat{y}^k_{ij}\Big)\right] + \lambda_c R(\mathcal{W}_c) + \lambda_t R(\mathcal{W}_t) + \lambda_k R(\mathcal{W}_k), \quad (1)$$

where $\mathcal{W}_c$, $\mathcal{W}_t$ and $\mathcal{W}_k$ denote the parameters of M_c, M_t and M_k, respectively, and $\ell(\cdot,\cdot)$ is an arbitrary loss function. And y_{ij}, $\hat{y}^c_{ij}$, $\hat{y}^t_{ij}$ and $\hat{y}^k_{ij}$ denote the true label, and the predicted labels of M_c, M_t and M_k for the sample (i,j), respectively, where (i,j) is associated with user i and item j. Note that $R(\cdot)$ is the regularization term, and λ_c, λ_t and λ_k are the parameters of the regularization.
- *Variant 2 of Bridge Strategy (NEW, Bridge-var2 for short).* When M_c contains serious bias problems, the incorrect knowledge provided by M_c may weaken the effectiveness of the Bridge-var1 strategy. For this reason, we consider only using the knowledge in M_t to guide M_k. However, due to the small scale of S_t, M_t may make inaccurate prediction for S_a. This means that directly constraining the alignment of M_k and M_t on label may lead to incorrect training. In order to address this issue, we propose to introduce some additional weights for each sample (S_c and S_a) based on the similarity of the parameters in M_t and M_k, where a higher similarity means a higher weight. The final objective function of this strategy is,

$$\min_{\mathcal{W}_t,\mathcal{W}_k} \frac{1}{|S_c|}\sum_{(i,j)\in S_c} \alpha_{ij}\ell\left(y_{ij},\hat{y}_{ij}^k\right) + \frac{1}{|S_t|}\sum_{(i,j)\in S_t} \ell\left(y_{ij},\hat{y}_{ij}^k\right) + \frac{1}{|S_a|}\sum_{(i,j)\in S_a} \alpha_{ij}\ell\left(\hat{y}_{ij}^t,\hat{y}_{ij}^k\right) + \lambda_t R(\mathcal{W}_t) + \lambda_k R(\mathcal{W}_k), \quad (2)$$

where $\alpha_{ij} = \text{sigmoid}(\frac{cos(\mathcal{W}_t^i,\mathcal{W}_k^i)+cos(\mathcal{W}_t^j,\mathcal{W}_k^j)}{2})$ is a parameter used to control the importance of each sample loss, $cos(\cdot,\cdot)$ denotes the cosine similarity, and $\mathcal{W}^i$ and $\mathcal{W}^j$ are the embedding related to user i and item j, respectively.

- *Variant of Refine Strategy (NEW, Refine-var for short).* The bias of S_c may be reflected in the labels, resulting in models trained on these labels being biased. For example, when generating samples for modeling, all the observed positive feedback are usually labeled as 1, and all the observed negative feedback are labeled as -1. However, in real applications, they should fit a preference distribution. With S_t, we expect to be able to better infer the true distribution of the labels on S_c and then refine them. Recalling the Refine strategy in [1], we use the imputed label generated by M_t to refine the label of S_c. However, previous works show that a combined data is more effective than S_t alone [1]. Instead of obtaining a model M_t pre-trained on S_t [1], we want to obtain a model M_t' pre-trained on a small combined data. More specifically, we first randomly sample a small subset S_c' from S_c, and then obtain a small combined data $S_t \cup S_c'$ and a remaining biased data $S_c \backslash S_c'$. We then use M_t' to predict all the samples in $S_c \backslash S_c'$. These imputed labels are combined with the original labels of $S_c \backslash S_c'$ through a weighting parameter, which are then used to train a more unbiased model M_c. Note that in order to avoid the distribution difference between the imputed labels and the original labels, we need to normalize the imputed labels. The final objective function of this strategy is,

$$\min_{\mathcal{W}_c} \frac{1}{|S_c \backslash S_c'|}\sum_{(i,j)\in S_c\backslash S_c'} \ell\left(y_{ij} + \alpha N\left(\hat{y}_{ij}^{t'}\right), \hat{y}_{ij}^c\right) + \lambda_c R(\mathcal{W}_c), \quad (3)$$

where α is a tunable parameter that controls the importance of the imputed labels produced by M_t', $\hat{y}_{ij}^{t'}$ denotes the predicted labels of M_t', and $N(\cdot)$ denotes a normalization function.

3.2 Feature-Based Module

Previous studies find that some features correlate with labels, but the correlation is not a causal relation [45]. For example, from 1999 to 2009, the correlation between "the number of people who drowned by falling into a pool" and "the number of films Nicolas Cage appeared in" is 66.6% [46]. But as we know, if Nicolas Cage does not appear in any film in a year, the number of people who drown in a pool may still not be 0. Hence, we need to learn some causal and stable features. The feature-based module can be divided into two steps, i.e., stable feature selection and biased data correction. First, we filter out causal and stable

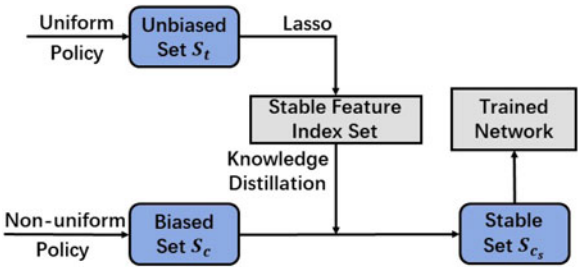


Fig. 2. Illustration of pre-feature selection strategy.

features via a uniform data through some methods. Then, we need to employ the stable features to train a teacher model that can be used to guide the biased model. Hence, we develop the following formal definition of feature-based distillation.

Definition 2 (D2). *A method can be classified as* feature-based distillation *if and only if the training of a non-uniform data S_c can benefit from the representative causal and stable features produced by a uniform data S_t.*

Solutions. We will use the two strategies adopted in our experiments as examples to illustrate how feature-based distillation can be realized.

- *Pre-Feature Selection Strategy (NEW, PFS for short).* Due to the unbiased nature of a uniform data, performing a feature selection process on it is more likely to get the desired stable feature index set. We first adopt a classic feature selection method on S_t to obtain the stable feature index set. Note that we use the Lasso method [47] for feature selection in the experiments. This stable feature index set is then used to perform pre-feature selection on S_c to remove pseudo-correlated features that are not conducive to model training. Finally, we train a debiased recommendation model on a stable set S_{c_s}. This strategy is illustrated in Fig. 2.
- *Influential Feature Selection Strategy (NEW, IFS for short).* The influence function (IF) is an important concept in robust statistics [48], which can be extended to measure the sample-wise influence [49] and the feature-wise influence [50] on validation loss. Sample-wise influence has been used in previous works to estimate the propensity score of a sample in S_c, where S_t is used as the validation set [39]. If a change in a sample in S_c leads to a large change in validation loss, the sample has greater influence and is considered to be closer to be unbiased, otherwise it has less influence and is more biased. Similarly, in this strategy, we use S_t as the validation set and use feature-wise influence to guide the training of a feature selection model built using S_c, where S_t and S_c are used together to compute the influence loss in the trained network. If the selection of a feature leads to a greater change in the verification loss, the feature has a greater influence and is retained. Otherwise, the feature should be filtered. This strategy is shown in Fig. 3. And finally, we obtain a stable feature set S_{c_s} for model training.

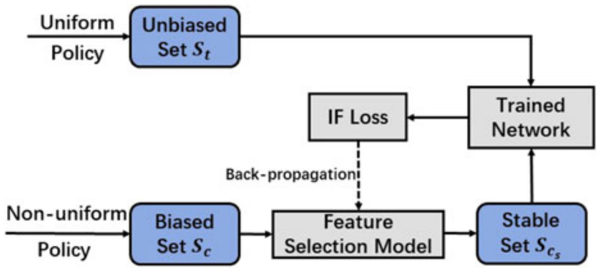


Fig. 3. Illustration of influential feature selection strategy.

3.3 Sample-Based Module

In a real recommender system with a stochastic logging policy, the probability of an item being recommended is different, and the probability of a user making a choice is also different. This means that model M_c may treat some items and users unfairly, because the samples in S_c lack support for these items and users. This unfairness can be corrected to some extent by directly considering the samples in S_t during the training process of M_c. Because the uniform logging policy corresponding to S_t increases the probability of the less popular items being selected, and M_c can weigh this difference between S_c and S_t. Based on this idea, we develop the following formal definition of sample-based distillation,

Definition 3 (D3). *A method can be classified as* sample-based distillation *if and only if a uniform data S_t is directly applied to help learning on all the samples without generating some imputed labels.*

Solutions. We will use the three strategies adopted in our experiments as examples to illustrate how sample-based distillation can be realized.

- *Causal Embedding Strategy (CausE for short).* The causal embedding method [44] first considers the scenario of training M_c and M_t simultaneously. It designs an additional alignment term to explicitly represent the learning of M_c for M_t. Causal embedding defines this alignment term as the pairwise difference between the parameters of M_c and M_t, which is then included in the object function to be minimized. When the value of the alignment term becomes small, it means that M_c learns the causal information contained in S_t, which helps correct the bias in learning on S_c. Note that it is difficult to dynamically optimize the differences between all the parameters of two complex models such as neural networks, so we use two low-rank models to implement this strategy in our experiments. The final objective function is,

$$\min_{\mathcal{W}_c,\mathcal{W}_t} \frac{1}{|S_c|}\sum_{(i,j)\in S_c} \ell\Big(y_{ij},\hat{y}^c_{ij}\Big) + \frac{1}{|S_t|}\sum_{(i,j)\in S_t} \ell\Big(y_{ij},\hat{y}^t_{ij}\Big) + \lambda_c R(\mathcal{W}_c) + \lambda_t R(\mathcal{W}_t) + \lambda^{CausE}_{tc}\|\mathcal{W}_t - \mathcal{W}_c\|^2_F, \tag{4}$$

  where λ^{CausE}_{tc} is the regularization parameter for the alignment term of M_c and M_t.
- *Weighted Sample Strategy (NEW, WeightS for short).* How to effectively combine the sample in S_c and S_t to train the model M_k? Inspired by modeling of heterogeneous implicit feedback [51], we add a confidence parameter to each sample of S_c and S_t to indicate whether it is unbiased. Naturally, the confidence of the samples in S_t is set to 1, and the confidence of the samples in S_c has two schemes to be used. The first scheme is a global setting, i.e., we set a confidence value in advance for all the samples of S_c. The second scheme is a local setting, i.e., each sample of S_c has a confidence value that needs to be learned by M_k. From the experimental results in [1], we find that the local setting does not perform well on a large-scale dataset. One possible reason is that as the scale of S_c increases, it becomes more difficult to accurately learn a large number of weights without referring to other information. We suggest introducing M_t and designing local weights according to the similarity of the parameters in M_t and M_k. When M_t and M_k have similar patterns on some samples, due to the unbiased nature of M_t, we can think that these samples are more likely to be unbiased (i.e., greater weight). The confidence of each sample is related to the corresponding loss function. The final objective function of this strategy is,

$$\min_{\mathcal{W}_k} \frac{1}{|S_c|}\sum_{(i,j)\in S_c} \alpha_{ij}\ell\Big(y_{ij},\hat{y}^k_{ij}\Big) + \frac{1}{|S_t|}\sum_{(i,j)\in S_t} \ell\Big(y_{ij},\hat{y}^k_{ij}\Big) + \lambda_k R(\mathcal{W}_k), \tag{5}$$

  where α_{ij} is a parameter used to control the confidence that we believe the sample (i,j) is unbiased. When considering the global setting, α_{ij} shares a parameter value that we preset for all the samples in S_c, but in the local setting, $\alpha_{ij} = \text{sigmoid}\Big(\frac{cos(\mathcal{W}^i_t,\mathcal{W}^i_k)+cos(\mathcal{W}^j_t,\mathcal{W}^j_k)}{2}\Big)$.
- *Delayed Combination Strategy (Delay for short).* Instead of introducing a confidence parameter, we propose a strategy called delayed combination. This strategy directly applies the data of S_c and S_t to the training of M_k in an alternative manner. Specifically, in the S_c step of each iteration, M_k is trained on the data of s batches from S_c. In the S_t step, we randomly sample one batch of data from S_t to train M_k. We repeat these two steps until all the data of S_c are used. The batch ratio is set to $s:1$, which can better ensure that M_k learns the information of S_c and the correction under the guidance of S_t. The final objective function of this strategy is,

$$\begin{cases} \min_{\mathcal{W}_k} \frac{1}{|S_c|}\sum_{(i,j)\in S_c} \ell\Big(y_{ij},\hat{y}^k_{ij}\Big) + \lambda_c R(\mathcal{W}_k), & S_c \text{ step}. \\ \min_{\mathcal{W}_k} \frac{1}{|S_t|}\sum_{(i,j)\in S_t} \ell\Big(y_{ij},\hat{y}^k_{ij}\Big) + \lambda_c R(\mathcal{W}_k), & S_t \text{ step}. \end{cases} \tag{6}$$

3.4 Model Structure-Based Module

Finally, we return to the model itself through considering how to directly use the pre-trained model M_t to help the learning of M_c. This is the most commonly adopted distillation strategy in existing works [16]. In order to help M_c with the guidance from M_t, we assume that some embedded representations of M_c correspond to some embedded representations of M_t. We constrain the selected embedded representations in M_c to be similar to their corresponding

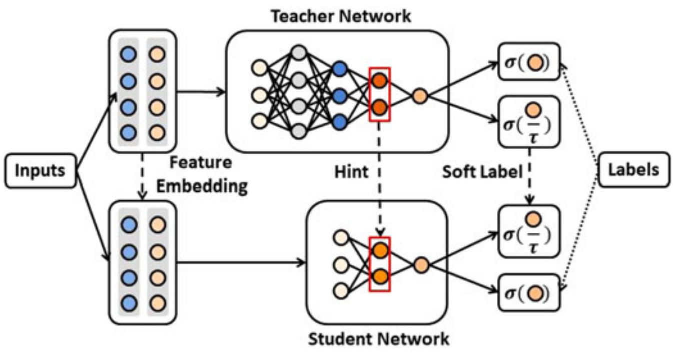


Fig. 4. Illustration of three types of model structure-based distillations, including feature embedding, hint and soft label. We use dotted arrows to indicate the matched pairs considered by different types of distillations, and $\sigma(\cdot)$ is an activation function.

embedded representations in M_t. As a result, M_c will have a similar pattern to M_t and thus may benefit from it. Note that the selected embedded representations of M_c and M_t do not necessarily have the same index. For example, suppose A is a 4-layer network and B is an 8-layer network, we may specify that each layer of A corresponds to an even layer of B, namely 2, 4, 6 and 8. Based on this idea, we develop the following formal definition of model structure-based distillation. For the sake of discussion, as shown in Fig. 4, we classify all the embedded representations into three types with different functions.

Definition 4 (D4). *A method can be classified as* model structure-based distillation *if and only if instead of using the labels and data, the embedded representation trained on a uniform data S_t is used to help the learning of a non-uniform data S_c.*

Solutions. We will use the three strategies adopted in our experiments as examples to illustrate how model structure-based distillation can be realized.

- *Feature Embedding Strategy (FeatE for short).* Feature embedding are the embedded representations that are directly connected to the users and items. In a neural network, it is usually the result of a one-hot coding after a *lookup* operation; and in a low-rank model, it is the users' preference vector U_i and the items' attribute vector V_j. As a special example, we think that the feature embedding of the autoencoder refers to the weights related to the number of items in the first layer and the last layer of the network. It is unreasonable to directly match the feature embedding in M_c with the that in M_t, because M_t may not learn sufficiently on these user- and item-related embedded representations due to the small data size. We thus propose the following two alternatives to use the feature embedding in M_t, including initialization of M_c, and concatenation with the parameters of M_c,

  Initialization. We have three options to choose the type of feature embedding as the initialization of M_c, including using only user-related, only item-related, and both. In addition, if we know which of the user-related and item-related ones is trained better, we can further use the information from M_t by setting their update steps to 1 (for the better one) and s (for the other, >1), respectively. We call it FeatE-a.

  Concatenation. After the parameters of M_c are randomly initialized, the feature embedding of M_t will be concatenated with these parameters to form new parameters to train M_c. Note that the features embedded of M_t in the parameters will not be updated during the training process. We call it FeatE-c.
- *Hint Strategy.* Hint refers to the hidden layer in a neural network, also known as feature map [20]. They contain higher-order non-linear relations between users or items. Note that in the experiments we must use deep neural networks to implement this strategy. After we specify hint for alignment in M_c and M_t, we explicitly model the difference between the two hints on the objective function of M_c. The final objective function of this strategy is,

$$\min_{\mathcal{W}_c} \frac{1}{|S_c|} \sum_{(i,j)\in S_c} \ell\Big(y_{ij}, \hat{y}^c_{ij}\Big) + \lambda_c R(\mathcal{W}_c) + \lambda^{hint}_{tc} \big\| y^{hint}_t - y^{hint}_c \big\|^2_F, \tag{7}$$

  where y^{hint}_c and y^{hint}_t are the output of M_c and M_t on their respective designated hint layers.
- *Soft Label Strategy.* Previous works have shown that training the student network to mimic the output of the teacher network on hard-labeled objectives does not bring much useful information to the student network [16]. But, by introducing some softmax and temperature operations to relax the label, training the student network to keep the same output as the teacher network on a soft label will result in a significant improvement [16]. We follow a similar setup in this strategy. Note that in the experiments we must also use deep neural networks to implement this strategy. The final objective function of this strategy is,

$$\min_{\mathcal{W}_c} \frac{\alpha}{|\mathcal{D}|} \sum_{(i,j)\in \mathcal{D}} \ell\left(\mathrm{softmax}\left(\frac{\hat{y}^c_{ij}}{\tau}\right), \mathrm{softmax}\left(\frac{\hat{y}^t_{ij}}{\tau}\right)\right) + \frac{1}{|S_c|} \sum_{(i,j)\in S_c} \ell\Big(y_{ij}, \hat{y}^c_{ij}\Big) + \lambda_c R(\mathcal{W}_c), \tag{8}$$

  where τ a is a temperature parameter, and α is a tunable parameter that controls the importance of the soft labels.

3.5 Summary and Remarks

Based on the above description, we can see that different strategies exhibit their own characteristics about how to make use of S_t. One possible confusion is between CausE and FeatE. Although they both operate on the feature embeddings, they are different in terms of the available training samples and training methods. According to the different definitions of the two modules to which they belong, CausE cannot use some imputed labels, while FeatE does not have this restriction. In addition, the feature embedding in CausE is used to constrain the alignment of M_c and M_t, so as part of the additional loss term to measure the distance between the two. FeatE aims to use M_t's feature embedding to help M_c's training, so it is more flexible in usage. For example, we can use the proposed Initialization and Concatenation operations.

Although we introduce the four distillation methods in different modules, their are closely related. This means that we can design new strategies with different combinations of the four distillation methods, such as the combination of sample-based distillation and label-based distillation. Note that most of the works in debiasing recommendations with uniform data can be incorporated into our framework. For example, label-based distillation includes a direct method for learning an imputation model and its variants, which is a module that most methods can be equipped with. Sample-based distillation includes the IPS techniques [9], [10], and the combination of sample-based and label-based distillation includes doubly robust techniques [14], [52], which are the ideas adopted by most methods. Moreover, they are also related to the types of knowledge (instance, feature and model) and strategies (adaptive, collective and integrative) in transfer learning [17], [53].

In addition, we must keep in mind the different considerations when using these four distillation methods. Although label-based and sample-based distillations are easy to implement, they need to consider the potential factors on the label and sample that may affect the model, such as the differences in sample size and label distributions. The difference in label distributions is passed on to the distributions of the predicted labels, so that the strategy of directly using the predicted labels may lead to poor results. The difference in data size means that M_c in a rough strategy can almost ignore the guided information from S_t. Feature-based distillation relies on the effectiveness of the method used to filter out the causal and stable features. However, the current research in this direction is still not sufficient. Model structure-based distillation requires only the model itself without regarding to other potential factors. But it is not easy to design an effective distillation structure or to select some good embedded representations.

4 Empirical Evaluation

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

- *RQ1*: How do the proposed methods perform against the baselines in an unbiased evaluation?
- *RQ2*: How do some key factors affect the performance of the proposed methods?
- *RQ3*: How do some specific factors affect the performance of the proposed new strategies?
- *RQ4*: What impact does the proposed methods have on the item distribution of the recommendation lists?

4.1 Experiment Setup

4.1.1 Datasets

To evaluate the performance of the proposed framework in an ideal unbiased scenario, the selected dataset must have a uniform subset for training and test. We consider the following datasets in the experiments, where the statistics are described in Table 2.

TABLE 2
Statistics of the Datasets

	Yahoo! R3		Product	
	#Feedback	P/N	#Feedback	P/N
S_c	311,704	67.02%	29,255,580	2.12%
S_t	2,700	9.36%	20,751	1.57%
S_{va}	2,700	8.74%	20,751	1.42%
S_{te}	48,600	9.71%	373,522	1.48%

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

- *Yahoo! R3* [5]: This dataset contains ratings collected from two different sources on Yahoo! Music services, involving 15,400 users and 1,000 songs. The Yahoo! user set consists of ratings supplied by users during normal interactions, i.e., users pick and rate items as they wish. This can be considered as a stochastic logging policy by following [9], [10], and thus the user set is biased. The Yahoo! random set consists of ratings collected during an online survey, when each of the first 5,400 users is asked to provide ratings on ten songs. The random set is different because the songs are randomly selected by the system instead of by the users themselves. The random set corresponds to a uniform logging policy and can be considered as the ground truth without bias. We binarize the ratings based on a threshold $\epsilon = 3$. Hence, a rating $r_{ij} > \epsilon$ is considered as a positive feedback (i.e., label $y_{ij} = 1$), otherwise, it is considered as a negative feedback (i.e., label $y_{ij} = -1$). The Yahoo! user set is used as a training set in a biased environment (S_c). For the Yahoo! random set, we randomly split the user-item interactions into three subsets: 5% for training in an unbiased environment (S_t), 5% for validation to tune the hyper-parameters (S_{va}), and the rest 90% for test (S_{te}).
- *Product*: This is a large-scale dataset for CTR prediction, which includes three weeks of users' click records from a real-world advertising system. The first two weeks' samples are used for training and the next week's samples for test. The records containing the exposed items clicked by a user are considered as the positive feedback (i.e., label $y_{ij} = 1$), and the records containing the exposed but not clicked items are randomly sampled as the negative feedback (i.e., label $y_{ij} = -1$). There exists two policies in this dataset: non-uniform policy and uniform policy which are defined in Section 3.1. We can thus separate this dataset into two parts, i.e., a uniform data and a non-uniform data. Note that in order to reduce the influence of position bias in the uniform data to ensure better unbiasedness, we only filter samples at positions 1 and 2. The non-uniform data contains around 29 million records and 2.8 million users, which is directly used as a training set named as S_c. Next, we randomly split the uniform data into three subsets by the same way as that of Yahoo! R3, i.e., 5% as training set (S_t), 5% as validation set (S_{va}), and the rest as test set (S_{te}).

4.1.2 Evaluation Metrics

We employ five 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 choose AUC as our main evaluation metric because it is one of the most important metrics in the industry and previous works on debiasing. We set the maximum length of a recommendation list to 50, and the candidate set considered for recommendation is the set of items that the user has not interacted with. We report the results of P@K and R@K when K is 5, and the results of nDCG when K is 50. More results about different values of K can be found in https://github.com/dgliu/TKDE_KDCRec.

4.1.3 Baselines

To demonstrate the effectiveness of our proposed framework, we include the following baselines which are widely used in recommendation scenarios.

Low-Rank Baselines:

Biased Matrix Factorization (biasedMF). We first consider the case where the proposed framework is implemented using a low-rank model. We use biased matrix factorization (biasedMF) [54] as the baseline, which is one of the most classic basic models in recommender systems. In this method, a user i's preference for an item j is formalized as $\hat{Y}_{ij} = U_i^T V_j + bu_i + bv_j$. We directly learn the user, item and bias representations using the square loss. All strategies in the framework are implemented when M_c and M_t are a biasedMF model.

Inverse Propensity Score Matrix Factorization (IPSMF). To test and compare the performance of the propensity-based causal inference, we use a representative counterfactual-based recommendation method as the second low-rank baseline, i.e., IPSMF [10]. Note that we estimate the propensity scores via the naïve Bayes estimator [10].

Neural Networks Baselines:

AutoEncoder (AE). We next consider the case where the proposed framework is implemented using a neural network model. We choose the autoencoder as the baseline to include more model choices. Except for the hint and soft label strategies where we use a five-layer autoencoder, we use the original three-layer autoencoder by default. All strategies in the framework are also implemented when M_c and M_t are an autoencoder model.

Multilayer Perceptron (MLP). In feature-based distillation, we need to introduce and encode the user features, item features, and context features, AE is thus not suitable. Hence, we use MLP as a baseline in feature-based distillation. More specifically, we use an MLP that includes one input layer, two hidden layers, and one output layer, where the hidden layers use Relu as the activaion function.

4.1.4 Implementation Details

We implement all the methods on TensorFlow[1]. We perform grid search to tune the hyper-parameters for the candidate methods by evaluating the AUC on the validation set S_{va}. We set the embedded dimension $d \in \{10, 50, 100, 200\}$, the regularization $\lambda \in \{1e^{-5}, 1e^{-4} \cdots 1e^{-1}\}$, the loss weighting $\alpha \in \{0.1, 0.2 \cdots 0.9, 1.0\}$, the batch size $l \in \{2^5, 2^6 \cdots 2^9\}$, the alternating steps $s \in \{1, 3, 5 \cdots 19, 20\}$ and the temperature $\tau \in \{2, 5, 10, 20\}$. The tuning history and optimal parameter file of each method can be found in the link of the source codes.

1. https://www.tensorflow.org

4.2 RQ1: Comparison Results

The comparison results are shown in Table 3. Because an additional baseline MLP is required in feature-based distillation as described in Section 4.1.3, we list its results separately. In addition, since Yahoo! R3 does not provide feature information, we only report the results on Product. As shown in the tables, our methods perform better than all the compared methods in most cases. More specifically, we have the following observations: 1) The sample combination of S_c and S_t improves the performance in all cases. The method using only S_t has an uncompetitive result on Yahoo! R3, but an improved AUC on Product, which is expected because Product contains a large uniform data S_t. The propensity-based method achieves superior performance on Yahoo except on AUC, but the results on Product are degraded on all metrics. One reason is that S_c and S_t of Product have a close ratio between the positive and negative feedback, which weakens the effect of the propensity scores. 2) The trends on AUC and other metrics are consistent in most cases. 3) The improvements brought by all the proposed strategies vary in different model implementations and different data scales. It means that each strategy's ability to use S_t depends on distinct scenarios. In general, when the data scale is small, the label-based strategy can maintain better performance, followed by the sample-based strategy and the model-based strategy (feature-based strategies are excluded due to different baselines for comparison). When the data scale is large, the difference between different strategies will be reduced, especially when the neural network model is used as the skeleton. We will conduct in-depth study on some strategies separately in the future. In addition, we also verify that the proposed new strategies are better than or supplementary to the original strategies, and the comparison results between them can be found in the appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2022.3199585.

4.3 RQ2: Impact of Key Factors

In this section, we explore and analyze some key factors that may affect the performance of the proposed framework. We only report experimental results on Yahoo! R3. We first consider the difference in P/N value between S_c and S_t. The difference in P/N value between S_c and S_t may be caused by some bias problems such as selection bias. Different degrees of difference obviously have different effects on the proposed strategies. To simulate the differences, we first set the size of the subset sampled from S_c to N (for example, N = 100,000 for Yahoo! R3). We control the proportion of positive samples included in the subset to 10%, 30%, 50%, 70% and 90%, which correspond to 5 different P/N value scenarios (i.e., the P/N value of the biased subset is $\frac{1}{9}$, $\frac{3}{7}$, $\frac{5}{5}$, $\frac{7}{3}$ and $\frac{9}{1}$). Finally, we use the new biased subset and S_t to retrain the proposed framework. The results are shown in Figs. 5a and 5b.

When using MF to implement the various proposed strategies, as the proportion of positive samples in the biased subset increases, the performance shows a trend of first increasing

TABLE 3
Comparison Results of Unbiased Evaluation

Yahoo! R3

		Low Rank (MF)				Neural Nets (AE)			
Module	Strategy	AUC	NDCG	P@5	R@5	AUC	NDCG	P@5	R@5
Baseline	Base (S_c)	+0.00%	+0.00%	+0.00%	+0.00%	+0.00%	+0.00%	+0.00%	+0.00%
	Uniform (S_t)	-21.76%	-15.46%	-29.03%	-27.59%	-24.90%	-29.19%	-57.14%	-52.14%
	Combine (∪)	+0.27%	+3.78%	+16.13%	+19.54%	+0.38%	+7.03%	+6.12%	-0.71%
	Propensity (∪)	-0.78%	+59.11%	+141.94%	+162.07%	—	—	—	—
Label	Refine-var (∪)	**+2.08%**	**+98.28%**	**+219.35%**	**+265.52%**	+1.35%	**+41.89%**	**+69.39%**	**+95.71%**
	Bridge-var1 (∪)	+0.80%	+21.65%	+32.26%	+28.74%	**+1.84%**	+23.78%	+18.37%	+32.14%
	Bridge-var2 (∪)	+1.06%	+47.08%	+54.84%	+87.36%	+1.68%	+20.00%	+10.20%	+15.00%
Sample	Delay (∪)	+0.74%	+19.59%	+29.03%	+28.74%	+0.49%	+3.51%	+10.20%	+5.71%
	CausE (∪)	+0.00%	-1.03%	+0.00%	+0.00%	—	—	—	—
	WeightS-l (∪)	+0.77%	+36.08%	+90.32%	+124.14%	+0.88%	+0.00%	-12.24%	-12.14%
	WeightS-g (∪)	+0.55%	+19.93%	+25.81%	+27.59%	+0.96%	+12.43%	+14.29%	+16.43%
Model Structure	FeatE-a (∪)	+0.60%	+25.77%	+22.58%	+21.84%	+0.61%	+5.68%	+2.04%	-11.43%
	FeatE-c (∪)	+0.33%	+8.25%	+16.13%	+19.54%	+0.14%	+7.03%	-8.16%	-12.86%
	Hint (∪)[a]	—	—	—	—	+1.04%	-27.84%	-63.51%	-61.84%
	Soft Label (∪)[a]	—	—	—	—	+1.09%	+4.64%	+6.76%	+8.21%

Product

		Low Rank (MF)				Neural Nets (AE)			
Module	Strategy	AUC	NDCG	P@5	R@5	AUC	NDCG	P@5	R@5
Baseline	Base (S_c)	+0.00%	+0.00%	+0.00%	+0.00%	+0.00%	+0.00%	+0.00%	+0.00%
	Uniform (S_t)	+0.68%	-29.83%	-22.50%	-34.98%	+2.65%	-35.40%	-30.16%	-35.66%
	Combine (∪)	+0.62%	+2.08%	+3.75%	-19.95%	+2.30%	+6.07%	+5.56%	+8.65%
	Propensity (∪)	-9.17%	-6.49%	-5.00%	-0.74%	—	—	—	—
Label	Refine-var (∪)	+2.45%	-0.39%	+32.50%	+13.79%	+4.55%	+6.30%	+3.97%	+7.98%
	Bridge-var1 (∪)	**+9.76%**	**+23.48%**	**+80.00%**	**+47.66%**	+4.56%	+6.41%	+3.97%	+7.98%
	Bridge-var2 (∪)	+2.17%	+10.38%	+0.00%	-5.30%	**+7.40%**	**+14.09%**	**+16.67%**	+14.97%
Sample	Delay (∪)	+1.62%	+7.13%	+22.51%	+10.10%	+6.05%	+13.40%	+7.94%	+11.22%
	CausE (∪)	+3.85%	+2.72%	+22.51%	-10.34%	—	—	—	—
	WeightS-l (∪)	+1.07%	+5.19%	+0.00%	-5.30%	-0.35%	-1.03%	-2.38%	-0.47%
	WeightS-g (∪)	+6.21%	+5.71%	+8.75%	-2.09%	+5.59%	+11.57%	+8.73%	+12.09%
Model Structure	FeatE-a (∪)	+0.72%	+7.13%	+3.75%	+2.83%	+4.37%	+10.77%	+11.90%	**+15.05%**
	FeatE-c (∪)	+0.54%	+6.87%	+3.75%	+2.83%	+5.07%	+9.39%	+5.56%	+11.97%
	Hint (∪)[a]	—	—	—	—	+2.80%	+1.34%	+1.55%	+2.21%
	Soft Label (∪)[a]	—	—	—	—	+3.84%	+2.46%	+3.88%	+3.84%

[a]Note that since these strategies rely on deep networks, we use the deep version of the base strategy as a reference to report the results, which is Deep AutoEncoder.

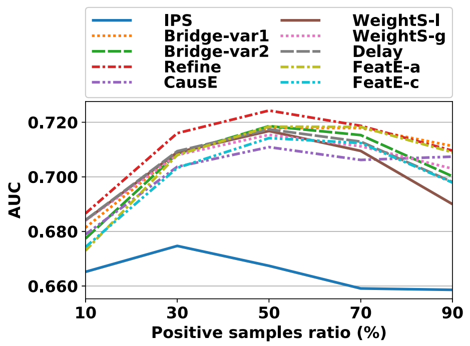
Product

		Neural Nets			
Module	Strategy	AUC	NDCG	P@5	R@5
Feature	Base (S_c)	+0.00%	+0.00%	+0.00%	+0.00%
	Uniform (S_t)	-9.32%	-6.53%	-21.95%	-20.45%
	Combine (∪)	+0.36%	+42.71%	+39.02%	+34.94%
	PFS (∪)	+3.09%	+57.79%	+44.51%	+51.30%
	IFS (∪)	**+3.93%**	**+78.89%**	**+56.10%**	**+70.26%**

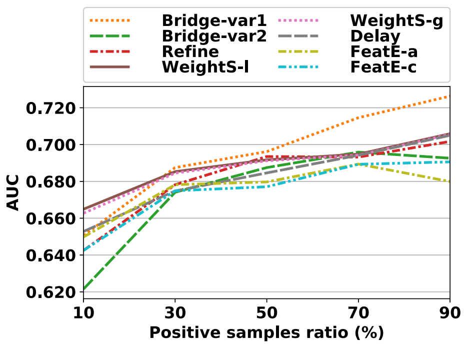
(∪) in the Strategy column indicates that the used data is $S_c \cup S_t$. The best results and the two suboptimal results are marked in bold and underlined, respectively. AUC is the main evaluation metric.

and then decreasing, and most of them reach a peak when the proportion is 50%. When the proportion of positive samples is 10%, the P/N value of the biased subset is close to the P/N value of S_t, which damages the performance of the proposed strategies. Because in this paper we only consider the feedback information, small differences in label information lead to a strategy not being able to extract useful knowledge from S_t well. However, when the difference between the positive sample ratio of the biased data and S_t is too large (i.e., the right side of Fig. 5a), the role of these useful knowledge will be reduced to a certain extent. Note that existing IPS methods are more susceptible to this effect due to excessive correction of the sample distribution. We can observe similar results when using AE to implement the various suggested strategies, as shown in Fig. 5b. An interesting observation is that when the proportion of positive samples in the biased subset is too different from S_t, the strategies implemented using AE have better robustness, and some strategies still maintain a growing trend.

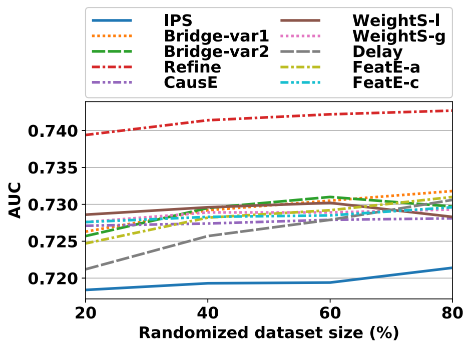
Next, we consider another key factor affecting the performance of the proposed framework, i.e., the size of S_t. Since the core of the proposed framework is to learn some useful unbiased knowledge from S_t, different sizes of S_t obviously have different effects on the proposed strategy. To simulate



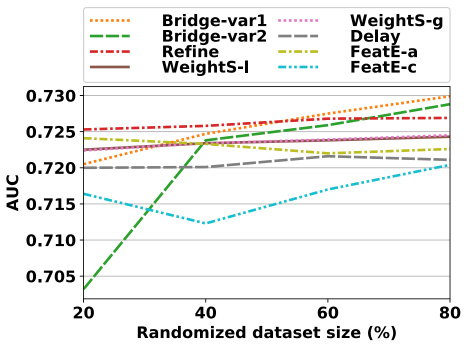
(a) Low Rank



(b) Neural Nets



(c) Low Rank



(d) Neural Nets

Fig. 5. The analysis results of the key factors on Yahoo! R3, where (a) and (b) are considering that S_c has different positive sample ratios, and (c) and (d) are considering that S_t has different data sizes.

the different sizes of S_t, we randomly sample a subset of S_t according to the ratio of 20%, 40%, 60% and 80%, which corresponds to four different size scenarios. Finally, we retrain the proposed framework using S_c and the new unbiased subset. The results are shown in Figs. 5c and 5d.

When using MF to implement the various proposed strategies, the performance improves as the scale of S_t increases. A small number of strategies show slight declines on the right side of Fig. 5c. This may mean that these strategies have reached the upper limit of their use of S_t. Note that the existing IPS method cannot obtain a large gain from the scale of S_t. This may be because the scale of S_t does not cause a large change in its own distribution. Similarly, we can observe similar results when using AE to implement the these strategies, as shown in Fig. 5d. The difference from Fig. 5c is that there is no longer a significantly better strategy. This may be because a smaller-scale uniform data in the neural network limits the performance improvement. In our experiments, the original number of S_t is 2,700, and the corresponding number on the x-axis of Figs. 5c and 5d is 540 to 2,160.

4.4 RQ3: Analysis Results of the Proposed New Strategies

Next, we analyze and discuss some specific factors for the proposed new strategies. As described in the Bridge-var2 strategy and WeightS-l strategy (in Sections 3.1 and 3.3), we use the form of *Sigmoid* mode to construct the similarity of model M_k and model M_t in parameters, i.e., $\mathrm{sigmoid}\left(\frac{cos(\mathcal{W}_t^i,\mathcal{W}_k^i)+cos(\mathcal{W}_t^j,\mathcal{W}_k^j)}{2}\right)$. Naturally, these two strategies may have different performance when using different modes to construct similarity. In the experiments we consider three modes, including *Sigmoid*, *Exp* and *Norm*. *Exp* mode can be described as $\exp(\frac{cos(\mathcal{W}_t^i,\mathcal{W}_k^i)+cos(\mathcal{W}_t^j,\mathcal{W}_k^j)}{2})$, and *Norm* mode can be described as $\frac{(\frac{cos(\mathcal{W}_t^i,\mathcal{W}_k^i)+cos(\mathcal{W}_t^j,\mathcal{W}_k^j)}{2})+1}{2}$. The results are shown in Table 4, from which we observe that *Sigmoid* mode and *Exp* mode are better choices under different conditions.

For the Refine-var strategy, the two key factors are the number of samples from S_c and the mode of sampling. In the experiments, we first fix the number of samples from S_c to $[200, 400, \ldots, 2400, 2600, 2700]$, and then retrain and evaluate the model. Note that we constrain the upper limit of the number of biased subsets to be the same as the number of S_t (i.e., 2,700), because combining with a too large biased subset may damage the unbiased knowledge of S_t. We show the results in Fig. 6. We use the dotted line to represent the result of the original Refine strategy, i.e., only use S_t to train the

TABLE 4
The Results of Bridge-Var2 Strategy and WeightS-L Strategy in Different Similarity Modes

		Low Rank (MF)	Neural Nets (AE)
Strategy	Mode	AUC	AUC
Bridge-var2	*Sigmoid*	0.7352	**0.7307**
	Exp	**0.7359**	0.7247
	Norm	0.6849	0.7306
WeightS-l	*Sigmoid*	**0.7331**	0.7226
	Exp	0.7314	**0.7249**
	Norm	0.6744	0.7221

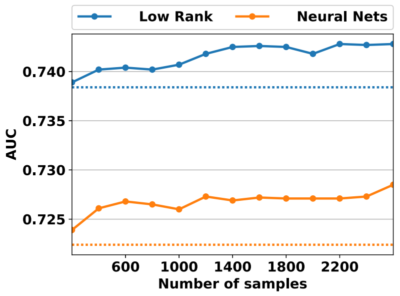


Fig. 6. The results of Refine-var strategy when sampling different sample sizes from S_c.

TABLE 5
The Results of Refine-Var Strategy in Different Sampling Modes

		Low Rank (MF)	Neural Nets (AE)
Strategy	Mode	AUC	AUC
Refine-var	*Random*	0.7426	0.7283
	In-user	0.7432	0.7277
	Out-user	0.7419	**0.7285**
	Head-item	**0.7435**	0.7282
	Tail-item	0.7357	0.7170

imputation model. We observe that as the number of samples from S_c increases, the performance of the Refine-var strategy increases. In addition, we also notice that this upward trend is not linear, i.e., the performance may remain flat within a certain number of biased subsets.

As described in Refine-var strategy, we use a random mode in the previous experiments to sample a biased subset from S_c. We then control the sampling modes, including *In-user*, *Out-uesr*, *Head-item* and *Tail-item* modes. In the *In-user* mode, we only sample from the overlapping user sets between S_c and S_t, while the *Out-uesr* mode is the opposite. We sort the items in descending order according to their popularity, and divide the items into head items and tail items at a ratio of 3:7. In the *Head-item* mode, we constrain to sample only the samples containing the head items from S_c, while the *Tail-item* mode is the opposite. The results are shown in Table 5. We observe that except for the *Tail-item* mode, the other four modes have similar results. This means that enhancing or supplementing the user information in S_t and enhancing the information of popular items in S_t are beneficial to mining some unbiased knowledge.

4.5 RQ4: Item Distribution of the Recommendation Lists

Finally, we conduct a preliminary experiment from the perspective of item distribution to analyze how the proposed strategy can effectively improve the debiasing performance. We sort the items in Yahoo! R3 in descending order according to the corresponding popularity, and then regard the top 20% of the items as popular items, and the rest as unpopular items. We show the distribution of S_t in Fig. 7a, from which we can find that although the probability of the popular and unpopular items being recommended is indeed even (e.g., popular items account for 20% of the total items, and the probability of being recommended also accounts for 20%), but the utility (i.e., the probability of hit divided by the probability of being recommended) brought by popular items is still higher. We only report the results of using MF as a skeleton model on Yahoo! R3.

Combining Figs. 7b and 7c we can observe: 1) To approximate the uniform distribution of S_{va}, MF trained on S_c, which suffers from the popularity bias, tends to capture the global distribution of popular and unpopular items similar to Fig. 7a (i.e., the ratio of 2:8). But unreasonably displaying too many unpopular items may not bring much benefit. 2) Uniform trained on S_t can avoid being misled by the global distribution and capture the property that popular items have higher utility. Therefore, in order to better sort popular items in S_{va}, it retains a slightly more proportion of popular items than MF. However, due to the limitation of the data size, it cannot further accurately identify users' preferences on popular and unpopular items. Combine is a trade-off between those two, and its effectiveness is shown in Table 3. 3) The proposed strategies also play a role in adjusting the proportion of popular items. But different from the baseline, they can maintain a good utility on both the adjusted two parts of the item, especially on the unpopular items. We have also observed that as the degree of adjustment increases, i.e., the proportion of popular items is higher, the two best strategies in Table 3 (i.e., Refine and Bridge-var2) can have a significant utility advantage in both parts of the items. The above results show that the proposed strategy can effectively integrate the information of the log data and uniform data, which can more accurately learn users' preferences on unpopular items, and maintain high utility on popular items. We believe this will produce a more ideal recommendation policy.

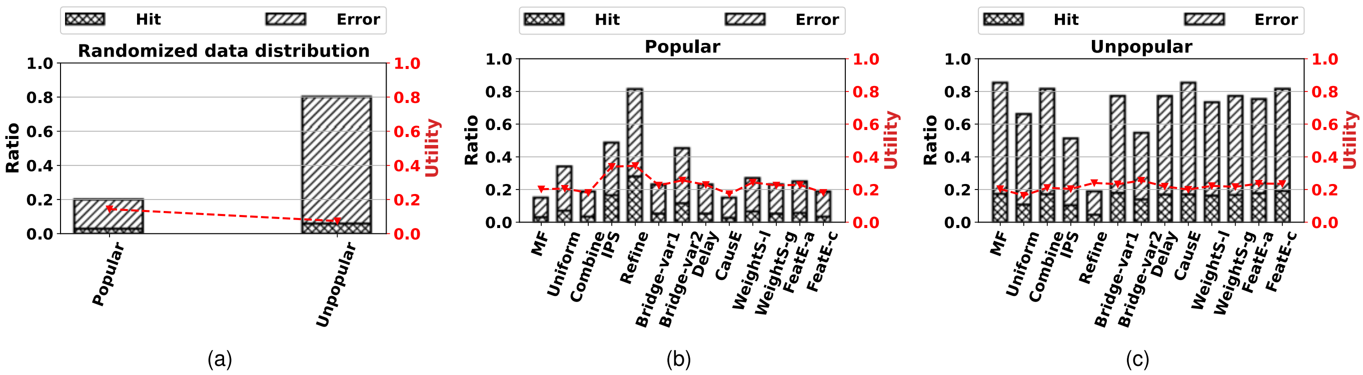


Fig. 7. Item distribution and corresponding utility of different strategies on Yahoo! R3.

5 Future Directions

Counterfactual recommendation via a uniform data is still a rich research field. In this section, we discuss some interesting and promising future directions: 1) *Label-Based Module.* The distribution difference between S_c and S_t is a key factor affecting this module. It is necessary to design strategies that are more robust to different degrees of difference, such as voting using multiple imputation models. 2) *Feature-Based Module.* The current approach only makes use of the feature information in each sample to learn the stable features, while the label in each sample from S_t is more stable and unbiased. So how to filter out the stable features with both labels and features in S_t is another interesting research question. 3) *Sample-Based Module.* The difference between the data size of S_t and that of S_c is a challenge for sample-based methods. One promising direction is to use the information in S_t to filter out a more unbiased subset from S_c, or use the information in S_c to perform data augmentation on S_t. Instead of using the label information, another promising direction is that we can consider modeling the preference ranking relation between S_t and S_c. 4) *Model Structure-Based Module.* The current distillation structure selection methods are based on enumeration or empirical methods. How to effectively design a good distillation structure is another promising direction, for which AutoML has the potential to find a reasonable model structure based on S_t.

In addition, there are also many other directions closely related to the framework. For example, the visualization or interpretation of the useful information (or knowledge) learned from S_t; further exploration of the results at a micro level, i.e., the impact on each user or each item; and the relation between the size of S_t and the performance of the model. In addition, we would like to further investigate the trade-off of training on S_c introduced by S_t and gain more theoretical insights into why it is effective. These theoretical insights can also inspire us to design better distillation strategies.

6 Conclusion

In this work, motivated by the observation that simply modeling with a uniform data can alleviate the bias problems, we propose a novel and general knowledge distillation framework for counterfactual recommendation via uniform data, i.e., KDCRec, including label-based, feature-based, sample-based and model structure-based distillations. We conduct extensive experiments on both public and product datasets, demonstrating that the proposed four methods can achieve better performance over the baseline models. In addition, we also explore and analyze the performance changes of the proposed methods on some key factors. Finally, we discuss some promising directions worthy of further exploration.

References

[1] D. Liu, P. Cheng, Z. Dong, X. He, W. Pan, and Z. Ming, "A general knowledge distillation framework for counterfactual recommendation via uniform data," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 831–840.

[2] H. Abdollahpouri, R. Burke, and B. Mobasher, "Controlling popularity bias in learning-to-rank recommendation," in *Proc. 11th ACM Conf. Recommender Syst.*, 2017, pp. 42–46.

[3] R. Cañamares and P. Castells, "Should I follow the crowd?: A probabilistic analysis of the effectiveness of popularity in recommender systems," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2018, pp. 415–424.

[4] J. J. Heckman, "Sample selection bias as a specification error," *Econometrica: J. Econometric Soc.*, vol. 47, pp. 153–161, 1979.

[5] B. M. Marlin and R. S. Zemel, "Collaborative prediction and ranking with non-random missing data," in *Proc. 3rd ACM Conf. Recommender Syst.*, 2009, pp. 5–12.

[6] D. C. Liu et al., "Related pins at pinterest: The evolution of a real-world recommender system," in *Proc. Web Conf.*, 2017, pp. 583–592.

[7] X. Wang, N. Golbandi, M. Bendersky, D. Metzler, and M. Najork, "Position bias estimation for unbiased learning to rank in personal search," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, 2018, pp. 610–618.

[8] A. Agarwal, I. Zaitsev, X. Wang, C. Li, M. Najork, and T. Joachims, "Estimating position bias without intrusive interventions," in *Proc. 12th ACM Int. Conf. Web Search Data Mining*, 2019, pp. 474–482.

[9] L. Yang, Y. Cui, Y. Xuan, C. Wang, S. Belongie, and D. Estrin, "Unbiased offline recommender evaluation for missing-not-at-random implicit feedback," in *Proc. 12th ACM Conf. Recommender Syst.*, 2018, pp. 279–287.

[10] T. Schnabel, A. Swaminathan, A. Singh, N. Chandak, and T. Joachims, "Recommendations as treatments: Debiasing learning and evaluation," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 1670–1679.

[11] W. Wang, F. Feng, X. He, H. Zhang, and T.-S. Chua, "Clicks can be cheating: Counterfactual recommendation for mitigating clickbait issue," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2021, pp. 1288–1297.

[12] T. Wei, F. Feng, J. Chen, Z. Wu, J. Yi, and X. He, "Model-agnostic counterfactual reasoning for eliminating popularity bias in recommender system," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2021, pp. 1791–1800.

[13] D. Liu, C. Lin, Z. Zhang, Y. Xiao, and H. Tong, "Spiral of silence in recommender systems," in *Proc. 12th ACM Int. Conf. Web Search Data Mining*, 2019, pp. 222–230.

[14] B. Yuan et al., "Improving ad click prediction by considering non-displayed events," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, 2019, pp. 329–338.

[15] J. Pearl and D. Mackenzie, *The Book of Why: The New Science of Cause and Effect*. New York, NY, USA: Basic Books, Inc., 2018.

[16] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[17] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[18] T. Fukuda, M. Suzuki, G. Kurata, S. Thomas, J. Cui, and B. Ramabhadran, "Efficient knowledge distillation from an ensemble of teachers," in *Proc. Annu. Conf. Int. Speech Commun. Assoc.*, 2017, pp. 3697–3701.

[19] H. Bagherinezhad, M. Horton, M. Rastegari, and A. Farhadi, "Label refinery: Improving ImageNet classification through label progression," 2018, *arXiv:1805.02641*.

[20] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "FitNets: Hints for thin deep nets," 2014, *arXiv:1412.6550*.

[21] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4133–4141.

[22] T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros, "Dataset distillation," 2018, *arXiv:1811.10959*.

[23] N. Passalis and A. Tefas, "Learning deep representations with probabilistic knowledge transfer," in *Proc. 15th Eur. Conf. Comput. Vis.*, 2018, pp. 268–284.

[24] C. Xu et al., "Privileged features distillation at taobao recommendations," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 2590–2598.

[25] J. Tang and K. Wang, "Ranking distillation: Learning compact ranking models with high performance for recommender system," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 2289–2298.

[26] Y. Zhang, X. Xu, H. Zhou, and Y. Zhang, "Distilling structured knowledge into embeddings for explainable and accurate recommendation," in *Proc. 13th Int. Conf. Web Search Data Mining*, 2020, pp. 735–743.

[27] J. Chen, H. Dong, X. Wang, F. Feng, M. Wang, and X. He, "Bias and debias in recommender system: A survey and future directions," 2020, *arXiv:2010.03240*.

[28] P. Gopalan, J. M. Hofman, and D. M. Blei, “Scalable recommendation with hierarchical poisson factorization,” in *Proc. 31st Conf. Uncertainty Artif. Intell.*, 2015, pp. 326–335.
[29] D. Liang, L. Charlin, J. McInerney, and D. M. Blei, “Modeling user exposure in recommendation,” in *Proc. Web Conf.*, 2016, pp. 951–961.
[30] H. Yang, G. Ling, Y. Su, M. R. Lyu, and I. King, “Boosting response aware model-based collaborative filtering,” *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 8, pp. 2064–2077, Aug. 2015.
[31] X. Wang, R. Zhang, Y. Sun, and J. Qi, “Doubly robust joint learning for recommendation on data missing not at random,” in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 6638–6647.
[32] L. Boratto, G. Fenu, and M. Marras, “Connecting user and item perspectives in popularity debiasing for collaborative recommendation,” *Inf. Process. Manage.*, vol. 58, no. 1, 2021, Art. no. 102387.
[33] Z. Wang, X. Chen, R. Wen, S.-L. Huang, E. Kuruoglu, and Y. Zheng, “Information theoretic counterfactual learning from missing-not-at-random feedback,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 1854–1864.
[34] D. Liu et al., “Mitigating confounding bias in recommendation via information bottleneck,” in *Proc. 15th ACM Conf. Recommender Syst.*, 2021, pp. 351–360.
[35] Y. Saito, S. Yaginuma, Y. Nishino, H. Sakata, and K. Nakata, “Unbiased recommender learning from missing-not-at-random implicit feedback,” in *Proc. 13th Int. Conf. Web Search Data Mining*, 2020, pp. 501–509.
[36] Y. Saito, “Asymmetric tri-training for debiasing missing-not-at-random explicit feedback,” in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 309–318.
[37] Y. Zheng, C. Gao, X. Li, X. He, Y. Li, and D. Jin, “Disentangling user interest and conformity for recommendation with causal embedding,” in *Proc. Web Conf.*, 2021, pp. 2980–2991.
[38] S. Xu, Y. Ge, Y. Li, Z. Fu, X. Chen, and Y. Zhang, “Causal collaborative filtering,” 2021, *arXiv:2102.01868*.
[39] J. Yu et al., “Influence function for unbiased recommendation,” in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 1929–1932.
[40] T. Schnabel and P. N. Bennett, “Debiasing item-to-item recommendations with small annotated datasets,” in *Proc. 14th ACM Conf. Recommender Syst.*, 2020, pp. 73–81.
[41] X. Wang, R. Zhang, Y. Sun, and J. Qi, “Combating selection biases in recommender systems with a few unbiased ratings,” in *Proc. 14th ACM Int. Conf. Web Search Data Mining*, 2021, pp. 427–435.
[42] J. Chen et al., “AutoDebias: Learning to debias for recommendation,” in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2021, pp. 21–30.
[43] S. Zeng, M. A. Bayir, J. J. Pfeiffer III, D. Charles, and E. Kiciman, “Causal transfer random forest: Combining logged data and randomized experiments for robust prediction,” in *Proc. 14th ACM Int. Conf. Web Search Data Mining*, 2021, pp. 211–219.
[44] S. Bonner and F. Vasile, “Causal embeddings for recommendation,” in *Proc. 12th ACM Conf. Recommender Syst.*, 2018, pp. 104–112.
[45] Y. He, Z. Shen, and P. Cui, “Towards non-IID image classification: A dataset and baselines,” *Pattern Recognit.*, vol. 110, 2021, Art. no. 107383.
[46] M. J. Albers, *Introduction to Quantitative Data Analysis in the Behavioral and Social Sciences.* Hoboken, NJ, USA: Wiley, 2017.
[47] V. Fonti and E. Belitser, “Feature selection using lasso,” *VU Amsterdam Res. Paper Bus. Analytics*, vol. 30, pp. 1–25, 2017.
[48] P. J. Huber, *Robust Statistics*, Berlin, Germany: Springer, 2011.
[49] P. W. Koh and P. Liang, “Understanding black-box predictions via influence functions,” in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1885–1894.
[50] J. Sliwinski, M. Strobel, and Y. Zick, “Axiomatic characterization of data-driven influence measures for classification,” in *Proc. 33rd AAAI Conf. Artif. Intell.*, 2019, pp. 718–725.
[51] W. Pan, H. Zhong, C. Xu, and Z. Ming, “Adaptive Bayesian personalized ranking for heterogeneous implicit feedbacks,” *Knowl.-Based Syst.*, vol. 73, pp. 173–180, 2015.
[52] M. Dudík, J. Langford, and L. Li, “Doubly robust policy evaluation and learning,” in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 1097–1104.
[53] W. Pan, “A survey of transfer learning for collaborative recommendation with auxiliary data,” *Neurocomputing*, vol. 177, pp. 447–453, 2016.
[54] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

Dugang Liu received the MS degree in computer science from Xiamen University, Xiamen, China, in 2019. He is currently working toward the PhD degree with the National Engineering Laboratory for Big Data System Computing Technology and the College of Computer Science and Software Engineering, Shenzhen University, China. His research interests include recommender systems and counterfactual machine learning. He has published papers in refereed conferences and journals such as WSDM, SIGIR, RecSys, *IEEE Transactions on Knowledge and Data Engineering*, etc.

Pengxiang Cheng received the MS degree in computer science and technology from the Harbin Institude of Technology, Harbin, China, in 2019. He is currently a researcher with Recommendation and Search Lab of Huawei Noah’s Ark Lab. His research interests include counterfactual machine learning in the area of recommendation and search.

Zinan Lin received the BS degree from the College of Mechatronics and Control Engineering, Shenzhen University, Shenzhen, China, in 2020. He is currently working toward the MS degree with the National Engineering Laboratory for Big Data System Computing Technology and the College of Computer Science and Software Engineering, Shenzhen University, China. His research interests include recommender systems and transfer learning.

Jinwei Luo received the BS degree from the College of Mechatronics and Control Engineering, Shenzhen University, Shenzhen, China, in 2020. He is currently working toward the MS degree with the National Engineering Laboratory for Big Data System Computing Technology and the College of Computer Science and Software Engineering, Shenzhen University, China. His research interests include recommender systems and transfer learning.

Zhenhua Dong received the PhD degree in computer science from Nankai University, China, in 2012. He is currently a principal researcher of Huawei Noah’s Ark Lab. His research interests include recommender system, counterfactual learning, causal information retrieval, and their applications. He has published papers in refereed conferences and journals such as AAAI, CIKM, ICDE, RecSys, SIGIR, WWW, *IEEE Transactions on Knowledge and Data Engineering*, etc.

Xiuqiang He received the PhD degree in computer science and engineering from the Hong Kong University of Science and Technology, Kowloon, Hong Kong, China, in 2010. He is currently the director of Recommendation and Search Lab and principal researcher of Huawei Noah’s Ark Lab. His research interests include machine learning algorithms in the area of recommendation and search. He has published more than 80 refereed international conference and journal papers.

Weike Pan received the PhD degree in computer science and engineering from the Hong Kong University of Science and Technology, Kowloon, Hong Kong, China, in 2012. He is currently an associate professor with the National Engineering Laboratory for Big Data System Computing Technology and the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include transfer learning, federated learning, recommender systems and machine learning. He has published more than 60 refereed international conference and journal papers. He was the recipient of the ACM TiiS 2016 Best Paper Award and SDM 2013 Nominee of Best Paper Award.

Zhong Ming received the PhD degree in computer science and technology from Sun Yat-Sen University, Guangzhou, China, in 2003. He is currently a professor with the National Engineering Laboratory for Big Data System Computing Technology and the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include software engineering and web intelligence. He has published more than 200 refereed international conference and journal papers (including more than 35 ACM/IEEE Transactions papers). He was the recipient of the ACM TiiS 2016 Best Paper Award and some other best paper awards.