Model-Agnostic Counterfactual Reasoning for Eliminating Popularity Bias in Recommender System

Tianxin Wei[1], Fuli Feng[2*], Jiawei Chen[1], Ziwei Wu[1], Jinfeng Yi[3] and Xiangnan He[1*]
[1]University of Science and Technology of China, [2]National University of Singapore, [3]JD AI Research
rouseau@mail.ustc.edu.cn,fulifeng93@gmail.com,cjwustc@ustc.edu.cn
maggiewuzw@gmail.com,yijinfeng@jd.com,xiangnanhe@gmail.com

ABSTRACT

The general aim of the recommender system is to provide *personalized* suggestions to users, which is opposed to suggesting *popular* items. However, the normal training paradigm, *i.e.*, fitting a recommender model to recover the user behavior data with pointwise or pairwise loss, makes the model biased towards popular items. This results in the terrible Matthew effect, making popular items be more frequently recommended and become even more popular. Existing work addresses this issue with Inverse Propensity Weighting (IPW), which decreases the impact of popular items on the training and increases the impact of long-tail items. Although theoretically sound, IPW methods are highly sensitive to the weighting strategy, which is notoriously difficult to tune.

In this work, we explore the popularity bias issue from a novel and fundamental perspective — cause-effect. We identify that popularity bias lies in the *direct effect* from the item node to the ranking score, such that an item's intrinsic property is the cause of mistakenly assigning it a higher ranking score. To eliminate popularity bias, it is essential to answer the counterfactual question that *what the ranking score would be if the model only uses item property*. To this end, we formulate a causal graph to describe the important cause-effect relations in the recommendation process. During training, we perform multi-task learning to achieve the contribution of each cause; during testing, we perform counterfactual inference to remove the effect of item popularity. Remarkably, our solution amends the learning process of recommendation which is agnostic to a wide range of models — it can be easily implemented in existing methods. We demonstrate it on Matrix Factorization (MF) and LightGCN [20], which are representative of the conventional and SOTA model for collaborative filtering. Experiments on five real-world datasets demonstrate the effectiveness of our method.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Recommendation, Popularity Bias, Causal Reasoning

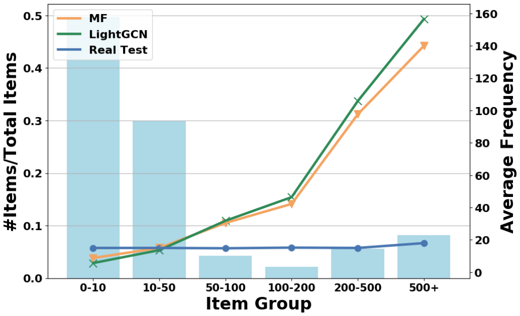


Figure 1: An illustration of popularity bias in recommender system. Items are organized into groups w.r.t. the popularity in the training set wherein the background histograms indicate the ratio of items in each group, and the vertical axis represents the average recommendation frequency.

ACM Reference Format:
Tianxin Wei[1], Fuli Feng[2*], Jiawei Chen[1], Ziwei Wu[1], Jinfeng Yi[3] and Xiangnan He[1*]. 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 (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3447548.3467289

1 INTRODUCTION

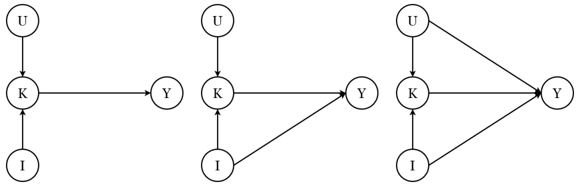
Personalized recommendation has revolutionized a myriad of online applications such as e-commerce [52, 55, 60], search engines [43], and conversational systems [29, 45]. A huge number of recommender models [19, 26, 49] have been developed, for which the default optimization choice is reconstructing historical user-item interactions. However, the frequency distribution of items is never even in the interaction data, which is affected by many factors like exposure mechanism, word-of-mouth effect, sales campaign, item quality, etc. In most cases, the frequency distribution is long-tail, i.e., the majority of interactions are occupied by a small number of popular items. This makes the classical training paradigm biased towards recommending popular items, falling short to reveal the true preference of users [2].

Real-world recommender systems are often trained and updated continuously using real-time user interactions with training data and test data NOT independent and identically distributed (non-IID) [11, 58]. Figure 1 provides an evidence of popularity bias on a real-world Adressa dataset [18], where we train a standard MF and LightGCN [20] and count the frequency of items in the top-K recommendation lists of all users. The blue line shows the item frequency of the real non-IID test dataset, which is what we expect. As can be seen, more popular items in the training data are

[*] Fuli Feng and Xiangnan He are Corresponding Authors.

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467289



(a) User-item matching (b) Incorporating item popularity (c) Incorporating item popularity and user conformity

Figure 2: Causal graph for (a) user-item matching; (b) incorporating item popularity; and (c) incorporating user conformity. I: item. U: user. K: matching features between user and item. Y: ranking score (e.g., the probability of interaction).

recommended much more frequently than expected, demonstrating a severe popularity bias. As a consequence, a model is prone to recommending items simply from their popularity, rather than user-item matching. This phenomenon is caused by the training paradigm, which identifies that recommending popular items more frequently can achieve lower loss thus updates parameters towards that direction. Unfortunately, such popularity bias will hinder the recommender from accurately understanding the user preference and decrease the diversity of recommendations. Worse still, the popularity bias will cause the Matthew Effect [36] — popular items are recommended more and become even more popular.

To address the issues of normal training paradigm, a line of studies push the recommender training to emphasize the long-tail items [7, 31]. The idea is to downweigh the influence from popular items on recommender training, e.g., re-weighting their interactions in the training loss [30, 50], incorporating balanced training data [7] or disentangling user and item embeddings [58]. However, these methods lack fine-grained consideration of how item popularity affects each specific interaction, and a systematic view of the mechanism of popularity bias. For instance, the interactions on popular items will always be downweighted than a long-tail item regardless of a popular item better matches the preference of the user. We believe that instead of pushing the recommender to the long-tail in a blind manner, the key of eliminating popularity bias is to understand how item popularity affects each interaction.

Towards this end, we explore the popularity bias from a fundamental perspective — cause-effect, which has received little scrutiny in recommender systems. We first formulate a causal graph (Figure 2(c)) to describe the important cause-effect relations in the recommendation process, which corresponds to the generation process of historical interactions. In our view, three main factors affect the probability of an interaction: user-item matching, item popularity, and user conformity. However, existing recommender models largely focus on the user-item matching factor [22, 53] (Figure 2(a)), ignoring how the item popularity affects the interaction probability (Figure 2(b)). Suppose two items have the same matching degree for a user, the item that has larger popularity is more likely to be known by the user and thus consumed. Furthermore, such impacts of item popularity could vary for different users, e.g., some users are more likely to explore popular items while some are not. As such, we further add a direct edge from the user node (U) to the ranking score (Y) to constitute the final causal graph (Figure 2(c)). To eliminate popularity bias effectively, it is essential to infer the direct effect from the item node (I) to the ranking score (Y), so as to remove it during recommendation inference.

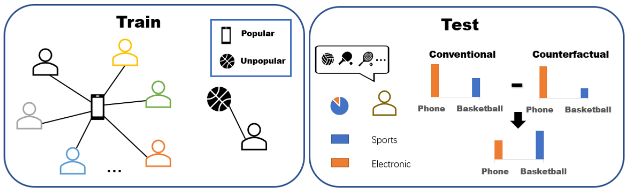


Figure 3: An example of counterfactual inference.

To this end, we resort to causal inference which is the science of analyzing the relationship between a cause and its effect [35]. According to the theory of counterfactual inference [35], the direct effect of $I \longrightarrow Y$ can be estimated by imagining a world where the user-item matching is discarded, and an interaction is caused by item popularity and user conformity. To conduct popularity debiasing, we just deduct the ranking score in the counterfactual world from the overall ranking score. Figure 3 shows a toy example where the training data is biased towards iPhone, making the model score higher on iPhone even though the user is more interested in basketball. Such bias is removed in the inference stage by deducting the counterfactual prediction.

In our method, to pursue a better learning of user-item matching, we construct two auxiliary tasks to capture the effects of $U \longrightarrow Y$ and $I \longrightarrow Y$. The model is trained jointly on the main task and two auxiliary tasks. Remarkably, our approach is model-agnostic and we implement it on MF [28] and LightGCN [20] to demonstrate effectiveness. To summarize, this work makes the following contributions:

- Presenting a causal view of the popularity bias in recommender systems and formulating a causal graph for recommendation.
- Proposing a model-agnostic counterfactual reasoning (MACR) framework that trains the recommender model according to the causal graph and performs counterfactual inference to eliminate popularity bias in the inference stage of recommendation.
- Evaluating on five real-world recommendation datasets to demonstrate the effectiveness and rationality of MACR.

2 PROBLEM DEFINITION

Let $\mathcal{U} = \{u_1, u_2, ...u_n\}$ and $\mathcal{I} = \{i_1, i_2, ...i_m\}$ denote the set of users and items, respectively, where n is the number of users, and m is the number of items. The user-item interactions are represented by $Y \in \mathbb{R}^{n \times m}$ where each entry,

$$y_{ui} = \begin{cases} 1, & \text{if user } u \text{ has interacted with item } i, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The goal of recommender training is to learn a scoring function $f(u, i|\theta)$ from Y, which is capable of predicting the preference of a user u over item i. Typically, the learned recommender model is evaluated on a set of holdout (e.g., randomly or split by time) interactions in the testing stage. However, the traditional evaluation may not reflect the ability to predict user true preference due to the existence of popularity bias in both training and testing. Aiming to focus more on user preference, we follow prior work [7, 30] to perform debiased evaluation where the testing interactions are

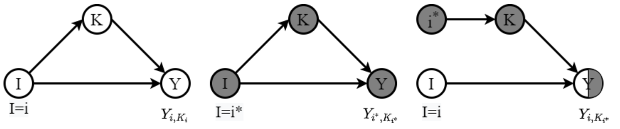


Figure 4: Example of causal graph where I, Y, and K denote cause, effect and mediator variable, respectively. Gray nodes mean the variables are at reference status (e.g., $I = i^*$).

sampled to be a uniform distribution over items. This evaluation also can examine a model's ability in handling the popularity bias.

3 METHODOLOGY

In this section, we first detail the key concepts about counterfactual inference (Section 3.1), followed by the causal view of the recommendation process (Section 3.2), the introduction of the MACR framework (Section 3.3), and its rationality for eliminating the popularity bias (Section 3.4). Lastly, we discuss the possible extension of MACR when the side information is available (Section 3.5).

3.1 Preliminaries

• *Causal Graph.* The *causal graph* is a directed acyclic graph $G = \{V, E\}$, where V denotes the set of variables and E represents the cause-effect relations among variables [35]. In a causal graph, a capital letter (e.g., I) denotes a variable and a lowercase letter (e.g., i) denotes its observed value. An edge means the ancestor node is a cause (I) and the successor node is an effect (Y). Take Figure 4 as an example, $I \longrightarrow Y$ means there exists a direct effect from I to Y. Furthermore, the path $I \longrightarrow K \longrightarrow Y$ means I has an indirect effect on Y via a mediator K. According to the causal graph, the value of Y can be calculated from the values of its ancestor nodes, which is formulated as:

$$Y_{i,k} = Y(I = i, K = k), \tag{2}$$

where $Y(.)$ means the value function of Y. In the same way, the value of the mediator can be obtained through $k = K_i = K(I = i)$. In particular, we can instantiate $K(I)$ and $Y(I, K)$ as neural operators (e.g., fully-connected layers), and compose a solution that predicts the value of Y from input I.

• *Causal Effect.* The *causal effect* of I on Y is the magnitude by which the target variable Y is changed by a unit change in an ancestor variable I [35]. For example, the *total effect* (TE) of $I = i$ on Y is defined as:

$$TE = Y_{i,K_i} - Y_{i^*,K_{i^*}}, \tag{3}$$

which can be understood as the difference between two hypothetical situations $I = i$ and $I = i^*$. $I = i^*$ refers to a the situation where the value of I is muted from the reality, typically set the value as null. K_{i^*} denotes the value of K when $I = i^*$. Furthermore, according to the structure of the causal graph, TE can be decomposed into *natural direct effect* (NDE) and *total indirect effect* (TIE) which represent the effect through the direct path $I \longrightarrow Y$ and the indirect path $I \longrightarrow K \longrightarrow Y$, respectively [35]. NDE expresses the value change of Y with I changing from i^* to i on the direct path $I \longrightarrow Y$, while K is set to the value when $I = i^*$, which is formulated as:

$$NDE = Y_{i,K_{i^*}} - Y_{i^*,K_{i^*}}, \tag{4}$$

where $Y_{i,K_{i^*}} = Y(I = i, K = K(I = i^*))$. The calculation of Y_i, K_{i^*} is a counterfactual inference since it requires the value of the same variable I to be set with different values on different paths (see Figure 4). Accordingly, TIE can be obtained by subtracting NDE from TE as following:

$$TIE = TE - NDE = Y_{i,K_i} - Y_{i,K_{i^*}}, \tag{5}$$

which represents the effect of I on Y through the indirect path $I \longrightarrow K \longrightarrow Y$.

3.2 Causal Look at Recommendation

In Figure 2(a), we first abstract the causal graph of most existing recommender models, where U, I, K, and Y represent user embedding, item embedding, user-item matching features, and ranking score, respectively. The models have two main components: a matching function $K(U, I)$ that learns the matching features between user and item; and the scoring function $Y(K)$. For instance, the most popular MF model implements these functions as an element-wise product between user and item embeddings, and a summation across embedding dimensions. As to its neural extension NCF [22], the scoring function is replaced with a fully-connected layer. Along this line, a surge of attention has been paid to the design of these functions. For instance, LightGCN [20] and NGCF [49] employ graph convolution to perform matching feature learning, ONCF [21] adopts convolutional layers as the scoring function. However, these models discards the user conformity and item popularity that directly affect the ranking score.

A more complete causal graph for recommendation is depicted in Figure 2(c) where the paths $U \longrightarrow Y$ and $I \longrightarrow Y$ represent the direct effects from user and item on the ranking score. A few recommender models follow this causal graph, e.g., the MF with additional terms of user and item biases [28] and NeuMF [22] which takes the user and item embeddings as additional inputs of its scoring function. While all these models perform inference with a forward propagation, the causal view of the inference over Figure 2(a) and Figure 2(c) are different, which are $Y_{K_{u,i}}$ and $Y_{u,i,K_{u,i}}$, respectively. However, the existing work treats them equally in both training and testing stages. For briefness, we use $\hat{y}_{ui}$ to represent the ranking score, which is supervised to recover the historical interactions by a recommendation loss such as the BCE loss [54]:

$$L_O = \sum_{(u,i)\in D} -y_{ui} \log(\sigma(\hat{y}_{ui})) - (1 - y_{ui}) \log(1 - \sigma(\hat{y}_{ui})), \tag{6}$$

where D denotes the training set and $\sigma(\cdot)$ denotes the sigmoid function. $\hat{y}_{u,i}$ means either $Y_{K_{u,i}}$ or $Y_{u,i,K_{u,i}}$. In the testing stage, items with higher ranking scores are recommended to users.

Most of these recommender model suffer from popularity bias (see Figure 1). This is because $\hat{y}_{ui}$ is the likelihood of the interaction between user u and item i, which is estimated from the training data and inevitably biased towards popular items in the data. From the causal perspective, item popularity directly affects $\hat{y}_{ui}$ via $I \longrightarrow Y$, which bubbles the ranking scores of popular items. As such, blocking the direct effect from item popularity on Y will eliminate the popularity bias.

3.3 Model-Agnostic Counterfactual Reasoning

To this end, we devise a model-agnostic counterfactual reasoning (MACR) framework, which performs multi-task learning for recommender training and counterfactual inference for making debiased recommendation. As shown in Figure 5, the framework follows the causal graph in Figure 2(c), where the three branches correspond to the paths $U \longrightarrow Y$, $U\&I \longrightarrow K \longrightarrow Y$, and $I \longrightarrow Y$, respectively. This

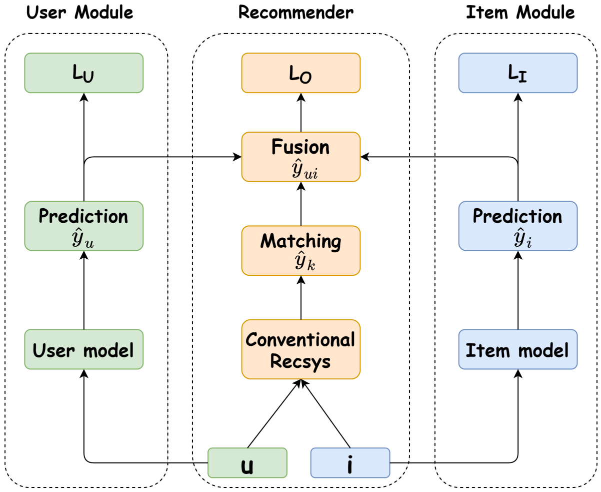


Figure 5: The framework of MACR. The orange rectangles denote the main branch, i.e., the conventional recommender system. The blue and green rectangles denote the user and item modules, respectively.

framework can be implemented over any existing recommender models that follow the structure of $U\&I \longrightarrow K \longrightarrow Y$ by simply adding a user module $Y_u(U)$ and an item module $Y_i(I)$. These modules project the user and item embeddings into ranking scores and can be implemented as multi-layer perceptrons. Formally,

- *User-item matching:* $\hat{y}_k = Y_k(K(U = u, I = i))$ is the ranking score from the existing recommender, which reflects to what extent the item i matches the preference of user u.
- *Item module:* $\hat{y}_i = Y_i(I = i)$ indicates the influence from item popularity where more popular item would have higher score.
- *User module:* $\hat{y}_u = Y_u(U = u)$ shows to what extent the user u would interact with items no matter whether the preference is matched. Considering the situation where two users are randomly recommended the same number of videos, one user may click more videos due to a broader preference or stronger conformity. Such "easy" user is expected to obtain a higher value of $\hat{y}_u$ and can be affected more by item popularity.

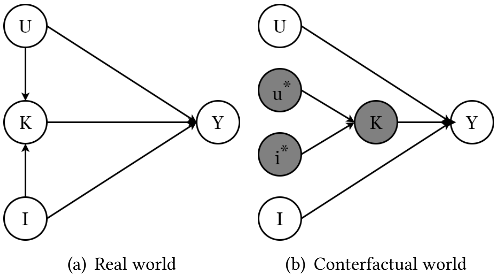
As the training objective is to recover the historical interactions y_{ui}, the three branches are aggregated into a final prediction score:

$$\hat{y}_{ui} = \hat{y}_k * \sigma(\hat{y}_i) * \sigma(\hat{y}_u), \tag{7}$$

where $\sigma(\cdot)$ denotes the sigmoid function. It scales $\hat{y}_u$ and $\hat{y}_i$ to be click probabilities in the range of $[0, 1]$ so as to adjust the extent of relying upon user-item matching (*i.e.* $\hat{y}_k$) to recover the historical interactions. For instance, to recover the interaction between an inactive user and unpopular item, the model will be pushed to highlight the user-item matching, *i.e.* enlarging $\hat{y}_k$.

Recommender Training. Similar to (6), we can still apply a recommendation loss over the overall ranking score $\hat{y}_{ui}$. To achieve the effect of the user and item modules, we devise a multi-task learning schema that applies additional supervision over $\hat{y}_u$ and $\hat{y}_i$. Formally, the training loss is given as:

$$L = L_O + \alpha * L_I + \beta * L_U, \tag{8}$$



(a) Real world (b) Conterfactual world

Figure 6: Comparison between real world and counterfactual world causal graphs in recommender systems.

where α and β are trade-off hyper-parameters. Similar as L_O, L_I and L_U are also recommendation losses:

$$L_U = \sum_{(u,i)\in D} -y_{ui}\log(\sigma(\hat{y}_u)) - (1 - y_{ui})\log(1 - \sigma(\hat{y}_u)),$$

$$L_I = \sum_{(u,i)\in D} -y_{ui}\log(\sigma(\hat{y}_i)) - (1 - y_{ui})\log(1 - \sigma(\hat{y}_i)).$$

Counterfactual Inference. As aforementioned, the key to eliminate the popularity bias is to remove the direct effect via path $I \longrightarrow Y$ from the ranking score $\hat{y}_{ui}$. To this end, we perform recommendation according to:

$$\hat{y}_k * \sigma(\hat{y}_i) * \sigma(\hat{y}_u) - c * \sigma(\hat{y}_i) * \sigma(\hat{y}_u), \tag{9}$$

where c is a hyper-parameter that represents the reference status of $\hat{y}_k$. The rationality of the inference will be detailed in the following section. Intuitively, the inference can be understood as an adjustment of the ranking according to $\hat{y}_{ui}$. Assuming two items i and j with $\hat{y}_{ui}$ slightly lower than $\hat{y}_{uj}$, item j will be ranked in front of i in the common inference. Our adjustment will affect if item j is much popular than i where $\hat{y}_j >> \hat{y}_i$. Due to the subtraction of the second part, the less popular item i will be ranked in front of j.. The scale of such adjustment is user-specific and controlled by $\hat{y}_u$ where a larger adjustment will be conducted for "easy" users.

3.4 Rationality of the Debiased Inference

As shown in Figure 2(c), I influences Y through two paths, the indirect path $I \longrightarrow K \longrightarrow Y$ and the direct path $I \longrightarrow Y$. Following the counterfactual notation in Section 3.1, we calculate the NDE from I to Y through counterfactual inference where a counterfactual recommender system (Figure 6(b)) assigns the ranking score without consideration of user-item matching. As can be seen, the indirect path is blocked by feeding feature matching function $K(U, I)$ with the reference value of I, K_{u^*,i^*}. Formally, the NDE is given as:

$$NDE = Y(U = u, I = i, K = K_{u^*,i^*}) - Y(U = u^*, I = i^*, K = K_{u^*,i^*}),$$

where u^* and i^* denote the reference values of U and I, which are typically set as the mean of the corresponding variables, *i.e.* the mean of user and item embeddings.

According to Equation 3, the TE from I to Y can be written as:

$$TE = Y(U = u, I = i, K = K_{u,i}) - Y(U = u^*, I = i^*, K = K_{u^*,i^*}).$$

Accordingly, eliminating popularity bias can be realized by reducing NDE from TE, which is formulated as:

$$TE - NDE = Y(U = u, I = i, K = K_{u,i}) - Y(U = u, I = i, K = K_{u^*,i^*}), \tag{10}$$

Table 1: Statistics of five different datasets.

	Users	Items	Interactions	Sparsity
Adressa	13,485	744	116,321	0.011594
Globo	158,323	12,005	2,520,171	0.001326
ML10M	69,166	8,790	5,000,415	0.008225
Yelp	31,668	38,048	1,561,406	0.001300
Gowalla	29,858	40,981	1,027,370	0.000840

Recall that the ranking score is calculated according to Equation 7. As such, we have $Y(U = u, I = i, K = K_{u,i}) = \hat{y}_k * \sigma(\hat{y}_i) * \sigma(\hat{y}_u)$ and $Y(U = u, I = i, K = K_{u^*,i^*}) = c * \sigma(\hat{y}_i) * \sigma(\hat{y}_u)$ where c denotes the value $\hat{y}_k$ with $K = K_{u^*,i^*}$. In this way, we obtain the ranking schema for the testing stage as Equation 9. Recall that $TIE = TE - NDE$, the key difference of the proposed counterfactual inference and normal inference is using TIE to rank items rather than TE. Algorithm in Appendix A describes the procedure of our method.

3.5 Discussion

There are usually multiple causes for one item click, such as items' popularity, category, and quality. In this work, we focus on the bias revealed by the interaction frequency. As an initial attempt to solve the problem from the perspective of cause-effect, we ignoring the effect of other factors. Due to the unavailability of side information [39] on such factors or the exposure mechanism to uncover different causes for the recommendation, it is also non-trivial to account for such factors.

As we can access such side information, we can simply extend the proposed MACR framework by incorporating such information into the causal graph as additional nodes. Then we can reveal the reasons that cause specific recommendations and try to further eliminate the bias, which is left for future exploration.

4 EXPERIMENTS

In this section, we conduct experiments to evaluate the performance of our proposed MACR. Our experiments are intended to answer the following research questions:

- **RQ1:** Does MACR outperform existing debiasing methods?
- **RQ2:** How do different hyper-parameter settings (e.g. α, β, c) affect the recommendation performance?
- **RQ3:** How do different components in our framework contribute to the performance?
- **RQ4:** How does MACR eliminate the popularity bias?

4.1 Experiment Settings

Datasets. Five real-world benchmark datasets are used in our experiments: ML10M is the widely-used [6, 40, 59] dataset from MovieLens with 10M movie ratings. While it is an explicit feedback dataset, we have intentionally chosen it to investigate the performance of learning from the implicit signal. To this end, we transformed it into implicit data, where each entry is marked as 0 or 1 indicating whether the user has rated the item; Adressa [18] and Globo [14] are two popular datasets for news recommendation; Also, the datasets Gowalla and Yelp from LightGCN [20] are used for a fair comparison. All the datasets above are publicly available and vary in terms of domain, size, and sparsity. The statistics of these datasets are summarized in Table 1.

Evaluation. Note that the conventional evaluation strategy on a set of holdout interactions does not reflect the ability to predict user's preference, as it still follows the long tail distribution [58]. Consequently, the test model can still perform well even if it only considers popularity and ignores users' preference [58]. Thus, the conventional evaluation strategy is not appropriate for testing whether the model suffers from popularity bias, and we need to evaluate on the debiased data. To this end, we follow previous works [7, 30, 58] to simulate debiased recommendation where the testing interactions are sampled to be a uniform distribution over items. In particular, we randomly sample 10% interactions with equal probability in terms of items as the test set, another 10% as the validation set, and leave the others as the biased training data[1]. We report the all-ranking performance w.r.t. three widely used metrics: Hit Ratio (HR), Recall, and Normalized Discounted Cumulative Gain (NDCG) cut at K.

4.1.1 *Baselines.*

We implement our MACR with the classic MF (MACR_MF) and the state-of-the-art LightGCN (MACR_LightGCN) to explore how MACR boosts recommendation performance. We compare our methods with the following baselines:

- **MF [28]:** This is a representative collaborative filtering model as formulated in Section 3.2.
- **LightGCN [20]:** This is the state-of-the-art collaborative filtering recommendation model based on light graph convolution as illustrated in Section 3.2.
- **ExpoMF [31]:** A probabilistic model that separately estimates the user preferences and the exposure.
- **CausE_MF, CausE_LightGCN [7]:** CausE is a domain adaptation algorithm that learns from debiased datasets to benefit the biased training. In our experiments, we separate the training set into debiased and biased ones to implement this method. Further, we apply CausE into two recommendation models (i.e. MF and LightGCN) for fair comparisons. Similar treatments are used for the following debias strategy.
- **BS_MF, BS_LightGCN [28]:** BS learns a biased score from the training stage and then remove the bias in the prediction in the testing stage. The prediction function is defined as:$\hat{y}_{ui} = \hat{y}_k + b_i$, where b_i is the bias term of the item i.
- **Reg_MF, Reg_LightGCN [2]:** Reg is a regularization-based approach that intentionally downweights the short tail items, covers more items, and thus improves long tail recommendation.
- **IPW_MF, IPW_LightGCN: [30, 42]** IPW Adds the standard Inverse Propensity Weight to reweight samples to alleviate item popularity bias.
- **DICE_MF, DICE_LightGCN: [58]** This is a state-of-the-art method for learning causal embedding to cope with popularity bias problem. It designs a framework with causal-specific data to disentangle interest and popularity into two sets of embedding. We used the code provided by its authors.

As we aim to model the interactions between users and items, we do not compare with models that use side information. We leave out the comparison with other collaborative filtering models, such as NeuMF [22] and NGCF [49], because LightGCN [20] is the state-of-the-art collaborative filtering method at present. Implementation details and detailed parameter settings of the models can be found in Appendix B.

[1] We refer to [7, 30, 58] for details on extracting an debiased test set from biased data.

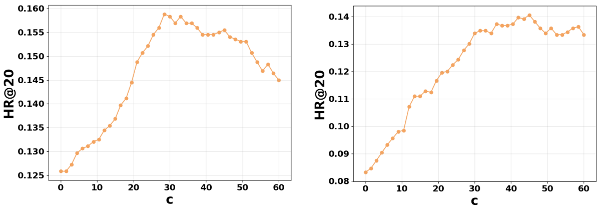
Table 2: The performance evaluation of the compared methods with $K = 20$. Rec means Recall. The bold-face font denotes the winner in that column. Note that the improvement achieved by MACR is significant (p-value << 0.05).

	Adressa			Globo			ML10M			Yelp2018			Gowalla		
	HR	Rec	NDCG	HR	Rec	NDCG	HR	Rec	NDCG	HR	Rec	NDCG	HR	Rec	NDCG
MF	0.111	0.085	0.034	0.020	0.003	0.002	0.058	0.009	0.008	0.071	0.006	0.009	0.174	0.046	0.032
ExpoMF	0.112	0.090	0.037	0.022	0.005	0.003	0.061	0.009	0.008	0.071	0.006	0.009	0.175	0.048	0.034
CausE_MF	0.112	0.084	0.037	0.023	0.005	0.003	0.054	0.008	0.007	0.066	0.005	0.008	0.166	0.045	0.032
BS_MF	0.113	0.090	0.038	0.021	0.005	0.003	0.060	0.009	0.008	0.071	0.006	0.010	0.175	0.046	0.033
Reg_MF	0.093	0.066	0.033	0.019	0.003	0.002	0.051	0.009	0.007	0.064	0.005	0.008	0.161	0.044	0.030
IPW_MF	0.128	0.096	0.039	0.021	0.004	0.003	0.041	0.006	0.005	0.072	0.006	0.010	0.174	0.048	0.033
DICE_MF	0.133	0.098	0.041	0.033	0.007	0.006	0.055	0.011	0.007	0.082	0.008	0.011	0.177	0.052	0.033
MACR_MF	**0.140**	**0.109**	**0.050**	**0.112**	**0.046**	**0.026**	**0.140**	**0.041**	**0.024**	**0.135**	**0.026**	**0.019**	**0.252**	**0.077**	**0.050**
LightGCN	0.123	0.098	0.040	0.017	0.005	0.003	0.038	0.006	0.005	0.061	0.004	0.009	0.172	0.045	0.032
CausE_LightGCN	0.115	0.082	0.037	0.014	0.005	0.003	0.036	0.005	0.005	0.061	0.005	0.009	0.173	0.046	0.033
BS_LightGCN	0.139	0.109	0.047	0.023	0.005	0.004	0.038	0.006	0.005	0.061	0.005	0.009	0.178	0.048	0.035
Reg_LightGCN	0.127	0.098	0.039	0.016	0.005	0.003	0.035	0.005	0.005	0.058	0.004	0.008	0.165	0.045	0.030
IPW_LightGCN	0.139	0.107	0.047	0.018	0.005	0.003	0.037	0.006	0.005	0.071	0.005	0.009	0.174	0.045	0.032
DICE_LightGCN	0.141	0.111	0.046	0.046	0.012	0.008	0.062	0.014	0.009	0.093	0.012	0.013	0.185	0.054	0.036
MACR_LightGCN	**0.158**	**0.127**	**0.052**	**0.132**	**0.059**	**0.030**	**0.155**	**0.049**	**0.029**	**0.148**	**0.031**	**0.018**	**0.254**	**0.077**	**0.051**

4.2 Results (RQ1)

Table 2 presents the recommendation performance of the compared methods in terms of HR@20, Recall@20, and NDCG@20. The bold-face font denotes the winner in that column. Overall, our MACR consistently outperforms all compared methods on all datasets for all metrics. The main observations are as follows:

- In all cases, our MACR boosts MF or LightGCN by a large margin. Specifically, the average improvement of MACR_MF over MF on the five datasets is 153.13% in terms of HR@20 and the improvement of MACR_LightGCN over LightGCN is 241.98%, which are rather substantial. These impressive results demonstrate the effectiveness of our multi-task training schema and counterfactual reasoning, even if here we just use the simple item and user modules. MACR potentially can be further improved by designing more sophisticated models.
- In most cases, LightGCN performs worse than MF, but in regular dataset splits, as reported in [20], LightGCN is usually a performing-better approach. As shown in Figure 1, with the same training set, we can see that the average recommendation frequency of popular items on LightGCN is visibly larger than MF. This result indicates that LightGCN is more vulnerable to popularity bias. The reason can be attributed to the embedding propagation operation in LightGCN, where the influence of popular items is spread on the user-item interaction graph which further amplifies the popularity bias. However, in our MACR framework, MACR_LightGCN performs better than MACR_MF. This indicates that our framework can substantially alleviate the popularity bias.
- In terms of datasets, we can also find that the improvements over the Globo dataset are extremely large. This is because Globo is a large-scale news dataset, and the item popularity distribution is particularly skewed. Popular news in Globo is widely read, while some other unpopular news has almost no clicks. This result indicates our model's capability of addressing popularity bias, especially on long-tailed datasets.
- As to baselines for popularity debias, Reg method [2] have limited improvement over the basic models and even sometimes perform even worse. The reason is that Reg simply downweights popular items without considering their influence on each interaction. CausE also performs badly sometimes as it relies on the debiased training set, which is usually relatively small and the model is hard to learn useful information from. BS and IPW methods can alleviate the bias issue to a certain degree. DICE achieved the best results among the baselines. This indicates the significance of considering popularity as a cause of interaction.



(a) MACR_LightGCN (b) MACR_MF

Figure 7: Effect of c on MACR_LightGCN and MACR_MF w.r.t HR@20.

In Appendix C.1, we also report our experimental results on Adressa dataset w.r.t. different values of K in the metrics for more comprehensive evaluation.

4.3 Case Study

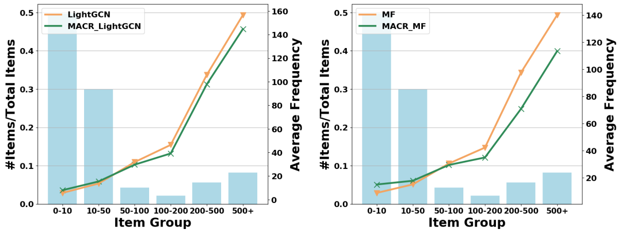
4.3.1 Effect of Hyper-parameters (RQ2). Our framework has three important hyper-parameters, α, β, and c. Due to space limitation, we provide the results of parameter sensitivity analysis of α, β in Appendix C.2.

The hyper-parameter c as formulated in Eq. (9) controls the degree to which the intermediate matching preference is blocked in prediction. We conduct experiments on the Adressa dataset on MACR_LightGCN and MACR_MF and test their performance in terms of HR@20. As shown in Figure 7, taking MACR_LightGCN as an instance, as c varies from 0 to 29, the model performs increasingly better while further increasing c is counterproductive. This illustrates that the proper degree of blocking intermediate matching preference benefits the popularity debias and improves the recommendation performance.

Compared with MACR_MF, MACR_LightGCN is more sensitive to c, as its performance drops more quickly after the optimum.

Table 3: Effect of user and item branch on MACR_MF.

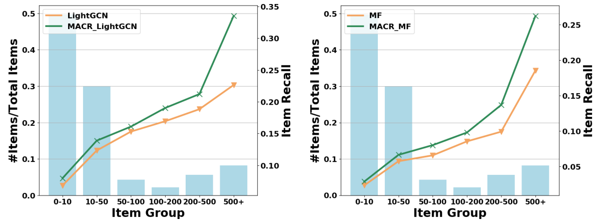
	HR@20	Recall@20	NDCG@20
MACR_MF	**0.140**	**0.109**	**0.050**
MACR_MF w/o user branch	0.137	0.106	0.046
MACR_MF w/o item branch	0.116	0.089	0.038
MACR_MF w/o L_I	0.124	0.096	0.043
MACR_MF w/o L_U	0.138	0.108	0.048



(a) LightGCN

(b) MF

Figure 8: Frequency of different item groups recommended by LightGCN (MF) and MACR_LightGCN (MACR_MF).



(a) LightGCN

(b) MF

Figure 9: Average item recall in different item groups on Adressa.

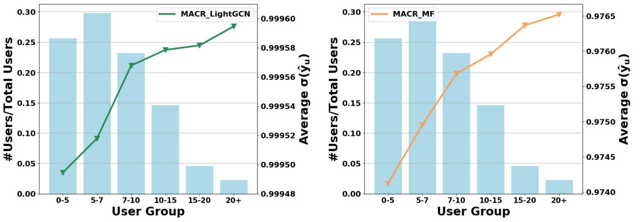
It indicates that LightGCN is more vulnerable to popularity bias, which is consistent with our findings in Section 4.2.

4.3.2 Effect of User Branch and Item Branch (RQ3). Note that our MACR not only incorporates user/item's effect in the loss function but also fuse them in the predictions. To investigate the integral effects of user and item branch, we conduct ablation studies on MACR_MF on the Adressa dataset and remove different components at a time for comparisons. Specifically, we compare MACR with its four special cases: MACR_MF w/o user (item) branch, where user (or item) branch has been removed; MACR_MF w/o L_I (L_U), where we just simply remove L_I (L_U) to block the effect of user (or item) branch on training but retain their effect on prediction.

From Table 3 we can find that both user branch and item branch boosts recommendation performance. Compared with removing the user branch, the model performs much worse when removing the item branch. Similarly, compared with removing L_U, removing L_I also harms the performance more heavily. This result validates that item popularity bias has more influence than user conformity on the recommendation.

Moreover, compared with simply removing L_I and L_U, removing the user/item branch makes the model perform much worse. This result validates the significance of further fusing the item and user influence in the prediction.

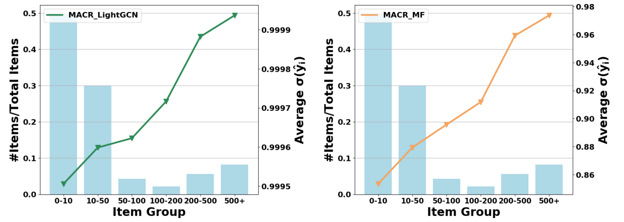
4.3.3 Debias Capability (RQ4). We then investigate whether our model alleviates the popularity bias issue. We compare MACR_MF and MACR_LightGCN with their basic models, MF and LightGCN.



(a) LightGCN

(b) MF

Figure 10: Average $\sigma(\hat{y}_u)$ comparison for different user groups on Adressa.



(a) LightGCN

(b) MF

Figure 11: Average $\sigma(\hat{y}_i)$ comparison for different item groups on Adressa.

As shown in Figure 8, we show the recommendation frequency of different item groups. We can see that our methods indeed reduce the recommendations frequency of popular items and recommend more items that are less popular. Then we conduct in Figure 9 an experiment to show the item recommendation recall in different item groups. In this experiment, we recommend each user 20 items and calculate the item recall. If an item appears N times in the test data, its item recall is the proportion of it being accurately recommended to test users. We have the following findings.

- The most popular item group has the greatest recall increase, but our methods in Figure 8 show the recommendations frequency of popular items is reduced. It means that traditional recommender systems (MF, LightGCN) are prone to recommend more popular items to unrelated users due to popularity bias. In contrast, our MACR reduces the item's direct effect and recommends popular items mainly to suitable users. This confirms the importance of matching users and items for personalized recommendations rather than relying on item related bias.
- The unpopular item group has relatively small improvement. This improvement is mainly due to the fact that we recommend more unpopular items to users as shown in Figure 8. Since these items rarely appear in the training set, it is difficult to obtain a comprehensive representation of these items, so it is difficult to gain a large improvement in our method.

To investigate why our framework benefits the debias in the recommendation, we explore what user branch and item branch, i.e., $\hat{y}_u$ and $\hat{y}_i$, actually learn in the model. We compare $\sigma(\hat{y}_u)$ and $\sigma(\hat{y}_i)$ as formulated in Eq. (7) , which is the output for the specific user u or item i from the user/item model after the sigmoid function, capturing user conformity and item popularity in the dataset. In Figure 10, the background histograms indicate the proportion of users in each group involved in the dataset. The horizontal axis means the user groups with a certain number of interactions. The left vertical axis is the value of the background histograms, which corresponds to the users' proportion in the dataset. The right vertical axis is the value of the polyline, which corresponds to $\sigma(\hat{y}_u)$. All the values are the average values of the users in the groups. As we can see, with the increase of the occurrence frequency of users in the dataset, the sigmoid scores of them also increase. This indicates that the user's activity is consistent with his/her conformity level. A similar phenomenon can be observed in Figure 11 for different item groups. This shows our model's capability of capturing item popularity and users' conformity, thus benefiting the debias.

5 RELATED WORK

In this section, we review existing work on Popularity Bias in Recommendation and Causal Inference in Recommendation, which are most relevant with this work.

5.1 Popularity Bias in Recommendation

Popularity bias is a common problem in recommender systems that popular items in the training dataset are frequently recommended. Researchers have explored many approaches [2, 9, 10, 23, 24, 44, 51, 58] to analyzing and alleviating popularity bias in recommender systems. The first line of research is based on Inverse Propensity Weighting (IPW) [41] that is described in the above section. The core idea of this approach is reweighting the interactions in the training loss. For example, Liang et al. [30] propose to impose lower weights for popular items. Specifically, the weight is set as the inverse of item popularity. However, these previous methods ignore how popularity influence each specific interaction.

Another line of research tries to solve this problem through ranking adjustment. For instance, Abdollahpouri et al. [2] propose a regularization-based approach that aims to improve the rank of long-tail items. Abdollahpouri et al. [3] introduce a re-ranking approach that can be applied to the output of the recommender systems. These approaches result in a trade-off between the recommendation accuracy and the coverage of unpopular items. They typically suffer from accuracy drop due to pushing the recommender to the long-tail in a brute manner. Unlike the existing work, we explore to eliminate popularity bias from a novel cause-effect perspective. We propose to capture the popularity bias through a multi-task training schema and remove the bias via counterfactual inference in the prediction stage.

5.2 Causal Inference in Recommendation

Causal inference is the science of systematically analyzing the relationship between a cause and its effect [35]. Recently, causal inference has gradually aroused people's attention and been exploited in a wide range of machine learning tasks, such as scene graph generation [13, 46], visual explanations [32], vision-language multi-modal learning [34, 37, 47, 56], node classification [15], text classification [38], and natural language inference [16]. The main purpose of introducing causal inference in recommender systems is to remove the bias [4, 5, 8, 25, 42, 48, 57]. We refer the readers to a systemic survey for more details [12].

Inverse Propensity Weighting. The first line of works is based on the Inverse Propensity Weighting (IPW). In [30], the authors propose a framework consisted of two models: one exposure model and one preference model. Once the exposure model is estimated, the preference model is fit with weighted click data, where each click is weighted by the inverse of exposure estimated in the first model and thus be used to alleviate popularity bias. Some very similar models were proposed in [42, 50].

Causality-oriented data. The second line of works is working on leveraging additional debiased data. In [7], they propose to create an debiased training dataset as an auxiliary task to help the model trained in the skew dataset generalize better, which can also be used to relieve the popularity bias. They regard the large sample of the dataset as biased feedback data and model the recommendation as a domain adaption problem. But we argue that their method does not explicitly remove popularity bias and does not perform well on normal datasets. Noted that all these methods are aimed to reduce the user exposure bias.

Causal embedding. Another series of work is based on the probability, in [31], the authors present ExpoMF, a probabilistic approach for collaborative filtering on implicit data that directly incorporates user exposure to items into collaborative filtering. ExpoMF jointly models both users' exposure to an item, and their resulting click decisions, resulting in a model which naturally down-weights the expected, but ultimately un-clicked items. The exposure is modeled as a latent variable and the model infers its value from data. The popularity of items can be added as an exposure covariate and thus be used to alleviate popularity bias. This kind of works is based on probability and thus cannot be generalized to more prevalent settings. Moreover, they ignore how popularity influences each specific interaction. Similar to our work, Zheng et al. [58] also tries to mitigate popularity bias via causal approaches. The difference is that we analyze the causal relations in a fine-grained manner, consider the item popularity, user conformity and model their influence on recommendation. [58] also lacks a systematic view of the mechanism of popularity bias.

6 CONCLUSION AND FUTURE WORK

In this paper, we presented the first cause-effect view for alleviating popularity bias issue in recommender systems. We proposed the model-agnostic framework MACR which performs multi-task training according to the causal graph to assess the contribution of different causes on the ranking score. The counterfactual inference is performed to estimate the direct effect from item properties to the ranking score, which is removed to eliminate the popularity bias. Extensive experiments on five real-world recommendation datasets have demonstrated the effectiveness of MACR.

This work represents one of the initial attempts to exploit causal reasoning for recommendation and opens up new research possibilities. In the future, we will extend our cause-effect look to more applications in recommender systems and explore other designs of the user and item module so as to better capture user conformity and item popularity. Moreover, we would like to explore how to incorporate various side information [39] and how our framework can be extended to alleviate other biases [12] in recommender systems. In addition, we will study the simultaneous elimination of multiple types of biases such as popularity bias and exposure bias through counterfactual inference. Besides, we will explore the combination of causation and other relational domain knowledge [33].

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (U19A2079, 61972372) and National Key Research and Development Program of China (2020AAA0106000).

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *OSDI*. 265–283.

[2] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2017. Controlling popularity bias in learning-to-rank recommendation. In *RecSys*. 42–46.

[3] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2019. Managing popularity bias in recommender systems with personalized re-ranking. In *FLAIRS*.

[4] Aman Agarwal, Kenta Takatsu, Ivan Zaitsev, and Thorsten Joachims. 2019. A general framework for counterfactual learning-to-rank. In *SIGIR*. 5–14.

[5] Alejandro Bellogín, Pablo Castells, and Iván Cantador. 2017. Statistical biases in Information Retrieval metrics for recommender systems. *Inf. Retr. J.* (2017), 606–634.

[6] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2018. Graph convolutional matrix completion. *KDD Deep Learning Day* (2018).

[7] Stephen Bonner and Flavian Vasile. 2018. Causal embeddings for recommendation. In *RecSys*. 104–112.

[8] Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X Charles, D Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. 2013. Counterfactual reasoning and learning systems: The example of computational advertising. *JMLR* (2013), 3207–3260.

[9] Rocío Cañamares and Pablo Castells. 2017. A probabilistic reformulation of memory-based collaborative filtering: Implications on popularity biases. In *SIGIR*. 215–224.

[10] Rocío Cañamares and Pablo Castells. 2018. Should I follow the crowd? A probabilistic analysis of the effectiveness of popularity in recommender systems. In *SIGIR*. 415–424.

[11] Allison J. B. Chaney, Brandon M. Stewart, and Barbara E. Engelhardt. 2018. How Algorithmic Confounding in Recommendation Systems Increases Homogeneity and Decreases Utility. In *RecSys*. 224–232.

[12] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2020. Bias and Debias in Recommender System: A Survey and Future Directions. *arXiv preprint arXiv:2010.03240* (2020).

[13] Long Chen, Hanwang Zhang, Jun Xiao, Xiangnan He, Shiliang Pu, and Shih-Fu Chang. 2019. Counterfactual critic multi-agent training for scene graph generation. In *ICCV*. 4613–4623.

[14] Gabriel de Souza Pereira Moreira, Felipe Ferreira, and Adilson Marques da Cunha. 2018. News session-based recommendations using deep neural networks. In *Workshop on Deep Learning at RecSys*. 15–23.

[15] Fuli Feng, Weiran Huang, Xiangnan He, Xin Xin, Qifan Wang, and Tat-Seng Chua Chua. 2021. Should Graph Convolution Trust Neighbors? A Simple Causal Inference Method. In *SIGIR*.

[16] Fuli Feng, Jizhi Zhang, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. 2021. Empowering Language Understanding with Counterfactual Reasoning. In *ACL-IJCNLP Findings*.

[17] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*. 249–256.

[18] Jon Atle Gulla, Lemei Zhang, Peng Liu, Özlem Özgöbek, and Xiaomeng Su. 2017. The Adressa dataset for news recommendation. In *WI*. 1042–1048.

[19] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. 2015. Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings.. In *AAAI*. 123–125.

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

[21] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. 2018. Outer product-based neural collaborative filtering. In *IJCAI*. 2227–2233.

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

[23] Amir Jadidinejad, Craig Macdonald, and Iadh Ounis. 2019. How Sensitive is Recommendation Systems' Offline Evaluation to Popularity?. In *REVEAL Workshop at RecSys*.

[24] Dietmar Jannach, Lukas Lerche, Iman Kamehkhosh, and Michael Jugovac. 2015. What recommenders recommend: an analysis of recommendation biases and possible countermeasures. *User Model User-adapt Interact* (2015), 427–491.

[25] Ray Jiang, Silvia Chiappa, Tor Lattimore, András György, and Pushmeet Kohli. 2019. Degenerate feedback loops in recommender systems. In *AIES*. 383–390.

[26] Santosh Kabbur, Xia Ning, and George Karypis. 2013. Fism: factored item similarity models for top-n recommender systems. In *KDD*. 659–667.

[27] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *ICLR*.

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

[29] Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. 2020. Estimation-action-reflection: Towards deep interaction between conversational and recommender systems. In *WSDM*. 304–312.

[30] Dawen Liang, Laurent Charlin, and David M Blei. 2016. Causal inference for recommendation. In *Workshop at UAI*.

[31] Dawen Liang, Laurent Charlin, James McInerney, and David M Blei. 2016. Modeling user exposure in recommendation. In *WWW*. 951–961.

[32] Álvaro Parafita Martínez and Jordi Vitrià Marca. 2019. Explaining Visual Models by Causal Attribution. In *ICCV Workshop*. 4167–4175.

[33] Liqiang Nie, Yongqi Li, Fuli Feng, Xuemeng Song, Meng Wang, and Yinglong Wang. 2020. Large-scale question tagging via joint question-topic embedding learning. *ACM TOIS* 38, 2 (2020), 1–23.

[34] Yulei Niu, Kaihua Tang, Hanwang Zhang, Zhiwu Lu, Xian-Sheng Hua, and Ji-Rong Wen. 2021. Counterfactual VQA: A Cause-Effect Look at Language Bias. In *CVPR*.

[35] Judea Pearl. 2009. *Causality*. Cambridge Uiversity Press.

[36] Matjaž Perc. 2014. The Matthew effect in empirical data. *J R Soc Interface* (2014), 20140378.

[37] Jiaxin Qi, Yulei Niu, Jianqiang Huang, and Hanwang Zhang. 2020. Two causal principles for improving visual dialog. In *CVPR*. 10860–10869.

[38] Chen Qian, Fuli Feng, Lijie Wen, Chunping Ma, and Pengjun Xie. 2021. Counterfactual Inference for Text Classification Debiasing. In *ACL-IJCNLP*.

[39] Steffen Rendle. 2010. Factorization machines. In *ICDM*. 995–1000.

[40] Steffen Rendle, Li Zhang, and Yehuda Koren. 2019. On the difficulty of evaluating baselines: A study on recommender systems. *arXiv preprint arXiv:1905.01395* (2019).

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

[42] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as treatments: Debiasing learning and evaluation. In *ICML*. 1670–1679.

[43] Xuehua Shen, Bin Tan, and ChengXiang Zhai. 2005. Implicit user modeling for personalized search. In *CIKM*. 824–831.

[44] Wenlong Sun, Sami Khenissi, Olfa Nasraoui, and Patrick Shafto. 2019. Debiasing the human-recommender system feedback loop in collaborative filtering. In *Companion of WWW*. 645–651.

[45] Yueming Sun and Yi Zhang. 2018. Conversational recommender system. In *SIGIR*. 235–244.

[46] Kaihua Tang, Yulei Niu, Jianqiang Huang, Jiaxin Shi, and Hanwang Zhang. 2020. Unbiased scene graph generation from biased training. In *CVPR*. 3716–3725.

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

[48] Wenjie Wang, Fuli Feng, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. 2021. " Click" Is Not Equal to" Like": Counterfactual Recommendation for Mitigating Clickbait Issue. In *SIGIR*.

[49] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR*. 165–174.

[50] Yixin Wang, Dawen Liang, Laurent Charlin, and David M Blei. 2018. The deconfounded recommender: A causal inference approach to recommendation. *arXiv preprint arXiv:1808.06581* (2018).

[51] Tianxin Wei, Ziwei Wu, Ruirui Li, Ziniu Hu, Fuli Feng, Xiangnan He, Yizhou Sun, and Wei Wang. 2020. Fast Adaptation for Cold-Start Collaborative Filtering with Meta-Learning. In *ICDM*. 661–670.

[52] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A neural influence diffusion model for social recommendation. In *SIGIR*. 235–244.

[53] Jun Xu, Xiangnan He, and Hang Li. 2020. Deep Learning for Matching in Search and Recommendation. *Found. Trends Inf. Ret.* 14 (2020), 102–288.

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

[55] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *KDD*. 974–983.

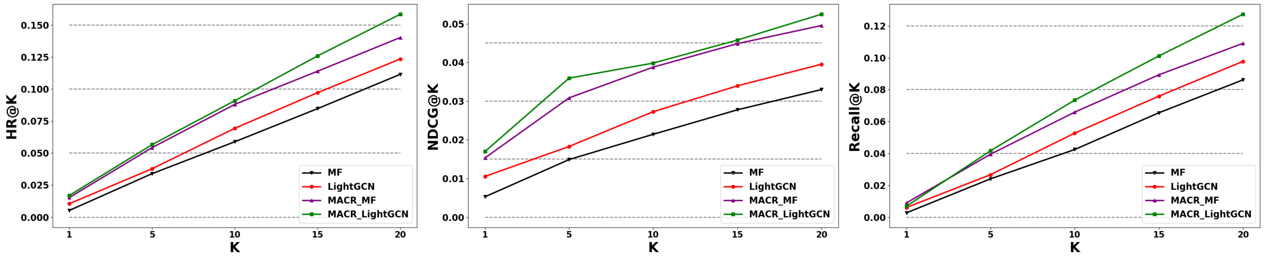
[56] Shengyu Zhang, Tan Jiang, Tan Wang, Kun Kuang, Zhou Zhao, Jianke Zhu, Jin Yu, Hongxia Yang, and Fei Wu. 2020. DeVLBert: Learning Deconfounded Visio-Linguistic Representations. In *ACMMM*. 4373–4382.

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

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

[59] Yin Zheng, Bangsheng Tang, Wenkui Ding, and Hanning Zhou. 2016. A Neural Autoregressive Approach to Collaborative Filtering. In *ICML*. 764–773.

[60] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *KDD*. 1059–1068.



(a) HR@K (b) NDCG@K (c) Recall@K

Figure 12: Top-K recommendation performance on Adressa datasets w.r.t. HR@K, NDCG@K and Recall@K.

A INFERENCE PROCEDURE

Algorithm 1 describes the procedure of our method and traditional recommendation system.

Algorithm 1 Inference

Input: Backbone recommender Y_k, Item module Y_i, User module Y_u, User u, Item i, Reference status c.

Output: $\hat{y}_{ui}$

```
/* Model Agnostic Counterfactual Reasoning */
$\hat{y}_k = Y_k(K(u,i))$;
$\hat{y}_i = Y_i(i)$;
$\hat{y}_u = Y_i(u)$;
if $Is\_Training$ then
    $\hat{y}_{ui} = \hat{y}_k * \sigma(\hat{y}_i) * \sigma(\hat{y}_u)$;
else
    $\hat{y}_{ui} = \hat{y}_k * \sigma(\hat{y}_i) * \sigma(\hat{y}_u) - c * \sigma(\hat{y}_i) * \sigma(\hat{y}_u)$;
end if
/* Traditional Recommender */
$\hat{y}_{ui} = Y_k(K(u,i))$;
```

B IMPLEMENTATION DETAILS

We implement MACR in Tensorflow [1]. The embedding size is fixed to 64 for all models and the embedding parameters are initialized with the Xavier method [17]. We optimize all models with Adam [27] except for ExpoMF which is trained in a probabilistic manner as per the original paper [31]. For all methods, we use the default learning rate of 0.001 and default mini-batch size of 1024 (on ML10M and Globo, we increase the mini-batch size to 8192 to speed up training). Also, we choose binarized cross-entropy loss for all models for a fair comparison. For the LightGCN model, we utilize two layers of graph convolution network to obtain the best results. For the Reg model, the coefficient for the item-based regularization is set to 1e-4 because it works best. For DICE, we keep all the optimal setting in their paper except replacing the regularization term $L_{discrepancy}$ from *dCor* with another option - *L*2. Because with our large-scale dataset, computing *dCor* will be out of memory for the 2080Ti GPU. It is also suggested in their paper. For ExpoMF, the initial value of μ is tuned in the range of $\{0.1, 0.05, 0.01, 0.005, 0.001\}$ as suggested by the author. For CausE, as their model training needs one biased dataset and another debiased dataset, we split 10% of the train data as we mentioned in Section 4.1 to build an additional debiased dataset for it. For our MACR_MF and MACR_LightGCN, the trade-off parameters α and β in Eq. (8) are both searched in the range of $\{1e-5, 1e-4, 1e-3, 1e-2\}$ and set to 1e-3 by default. The c in Eq. (9) is tuned in the range of $\{20, 22, ..., 40\}$. The number of training epochs is fixed to 1000. The L2 regularization coefficient is set to 1e-5 by default.

Table 4: Effect of α on MACR_MF.

	HR@20	Recall@20	NDCG@20
1e-5	0.133	0.104	0.045
1e-4	0.139	0.108	0.049
1e-3	**0.140**	**0.109**	**0.050**
1e-2	0.137	0.108	0.048

Table 5: Effect of β on MACR_MF.

	HR@20	Recall@20	NDCG@20
1e-5	0.139	0.108	0.049
1e-4	0.139	0.109	0.049
1e-3	**0.140**	**0.109**	**0.050**
1e-2	0.139	0.108	0.049

C SUPPLEMENTARY EXPERIMENTS

C.1 Metrics with different Ks

Figure 12 reports our experimental results on Adressa dataset w.r.t. HR@K, NDCG@K and Recall@K where $K = \{1, 5, 10, 15, 20\}$. It shows the effectiveness of MACR which can improve MF and LightGCN on different metrics with a large margin. Due to space limitation, we show the results on the Adressa dataset only, and the results on the other four datasets show the same trend.

C.2 Effect of hyper-parameters

As formulated in the loss function Eq. (8), α is the trade-off hyperparameter which balances the contribution of the recommendation model loss and the item model loss while β is to balance the recommendation model loss and the user loss. To investigate the benefit of item loss and user loss, we conduct experiments of MACR_MF on the typical Adressa dataset with varying α and β respectively. In particular, we search their values in the range of {1e-5, 1e-4, 1e-3, 1e-2}. When varying one parameter, the other is set as constant 1e-3. From Table 4 and Table 5 we have the following findings:

- As α increases from 1e-5 to 1e-3, the performance of MACR will become better. This result indicates the importance of capturing item popularity bias. A similar trend can be observed by varying β from 1e-5 to 1e-3 and it demonstrates the benefit of capturing users' conformity.
- However, when α or β surpasses a threshold (1e-3), the performance becomes worse with a further increase of the parameters. As parameters become further larger, the training of the recommendation model will be less important, which brings the worse results.