# Counterfactual Collaborative Reasoning


Jianchao Ji
Rutgers University
New Brunswick, NJ, US
jianchao.ji@rutgers.edu

Zelong Li
Rutgers University
New Brunswick, NJ, US
zelong.li@rutgers.edu

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

Max Xiong
Rutgers Preparatory School
Somerset, NJ, US
mxiong24@rutgersprep.org

Juntao Tan
Rutgers University
New Brunswick, NJ, US
juntao.tan@rutgers.edu

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

Hao Wang
Rutgers University
New Brunswick, NJ, US
hw488@cs.rutgers.edu

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



## ABSTRACT

Causal reasoning and logical reasoning are two important types of reasoning abilities for human intelligence. However, their relationship has not been extensively explored under machine intelligence context. In this paper, we explore how the two reasoning abilities can be jointly modeled to enhance both accuracy and explainability of machine learning models. More specifically, by integrating two important types of reasoning ability—counterfactual reasoning and (neural) logical reasoning—we propose Counterfactual Collaborative Reasoning (CCR), which conducts counterfactual logic reasoning to improve the performance. In particular, we use recommender system as an example to show how CCR alleviate data scarcity, improve accuracy and enhance transparency. Technically, we leverage counterfactual reasoning to generate "difficult" counterfactual training examples for data augmentation, which—together with the original training examples—can enhance the model performance. Since the augmented data is model irrelevant, they can be used to enhance any model, enabling the wide applicability of the technique. Besides, most of the existing data augmentation methods focus on "implicit data augmentation" over users' implicit feedback, while our framework conducts "explicit data augmentation" over users explicit feedback based on counterfactual logic reasoning. Experiments on three real-world datasets show that CCR achieves better performance than non-augmented models and implicitly augmented models, and also improves model transparency by generating counterfactual explanations.



## CCS CONCEPTS

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

## KEYWORDS

Neural Logic Reasoning; Counterfactual Reasoning; Counterfactual Data Augmentation; Recommender Systems




*WSDM '23, February 27-March 3, 2023, Singapore, Singapore*



ACM ISBN 978-1-4503-9407-9/23/02…$15.00
https://doi.org/10.1145/3539597.3570464

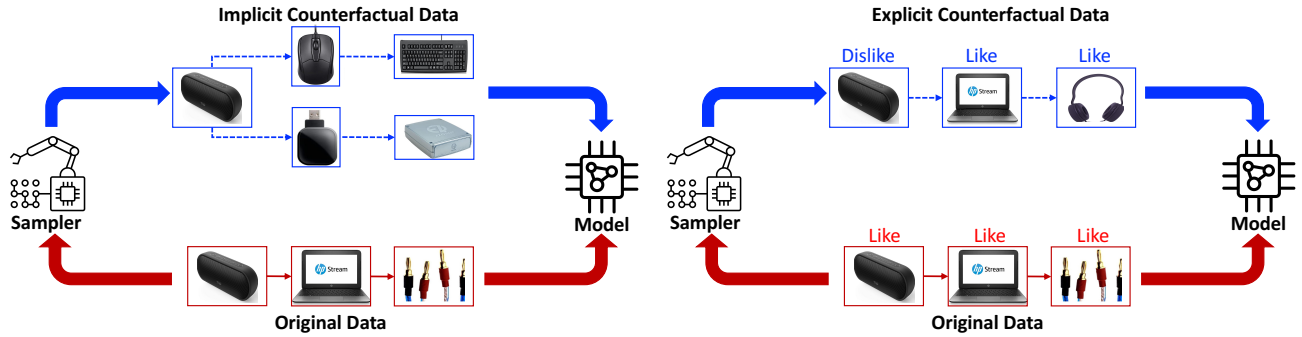
**ACM Reference Format:**
Jianchao Ji, Zelong Li, Shuyuan Xu, Max Xiong, Juntao Tan, Yingqiang Ge, Hao Wang, and Yongfeng Zhang. 2023. Counterfactual Collaborative Reasoning. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining (WSDM '23), February 27-March 3, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3539597.3570464


## 1 INTRODUCTION

Causal reasoning and logical reasoning are two important types of reasoning abilities. In this paper, we explore how the two reasoning abilities can be jointly modeled to enhance both accuracy and explainability of machine learning models. We take recommender system, which is an important prediction task, as an example to demonstrate the benefits of jointly modeling causal and logical reasoning. One important problem in recommendation model training is data scarcity. This is because user interactions are very sparse compared to the vast amount of items in the system, and this is especially true for sequential recommendation models which leverage a few or a few tens of user history interactions to predict the next interaction out of thousands or millions of candidates. One state-of-the-art approach to alleviating the data scarcity problem is counterfactual data augmentation, which conducts counterfactual reasoning over the original training examples to produce a set of informative augmented training examples [11, 18, 53, 54, 60, 63]. The augmented training examples, together with the original training examples, can help to improve the recommendation performance by generating synthetic data to cover the unexplored input space while maintaining correctness of the data as much as possible [54].

Take Figure 1(a) as an example, the original training data shows that the user purchased a speaker, a laptop, and a data cable to connect the speaker to the laptop. Current data augmentation methods perturb one or more of the user's purchase histories to alternative items that can trigger the recommendation result to change. In this example, the laptop is perturbed to a mouse or a USB flash drive, and the recommendation result is changed to a keyboard or an SSD drive, respectively. Thus, two counterfactual examples are created. The key is to perturb the history item to very similar items that can change the output, which creates difficult examples to challenge the recommendation model and help train the model to distinguish such difficult examples for better performance [53].

Although counterfactual data augmentation has achieved success in improving many recommendation models, one major problem with existing counterfactual data augmentation methods is




(a) Toy example for implicit counterfactual data augmentation

(b) Toy example for explicit counterfactual data augmentation

**Figure 1: An illustration of (a) implicit counterfactual data augmentation and (b) explicit counterfactual data augmentation**

that they only consider users' implicit feedback such as click or purchase information for generating augmented counterfactual examples, while the explicit feedback such as users' like or dislike preference on the items are ignored. However, explicit feedback contains rich user preference information that helps to understand user behaviors and make accurate predictions. Take Figure 1(b) as an example, which still shows the speaker, laptop and cable as the three items. If the user liked the speaker and laptop, then the user could indeed purchase the cable as the next item to connect the two. However, if the user disliked the speaker, then the next item could not be the cable but instead a headset as a substitute to the speaker. The example shows that even if the user's interaction history is the same, user's counterfactual preferences on the history items can create informative counterfactual examples to enhance the recommendation model.

There is a reason that existing counterfactual data augmentation methods mostly focus on implicit data augmentation while ignoring the explicit feedback. Usually, counterfactual data augmentation relies on an anchor sequential recommendation model to perturb the history items and generate counterfactual examples. However, most of the existing sequential recommendation models work with implicit feedback and cannot handle explicit feedback in sequential modeling [12, 28, 30, 32, 33, 35, 42, 45, 47, 50, 53, 60]. As a result, the counterfactual reasoning procedure is only able to perturb implicit feedback when generating counterfactual examples. Fortunately, recent advances on neural-symbolic/neural-logic reasoning methods [5, 7, 8, 31, 46] such as neural collaborative reasoning (NCR) [8] shed light on this problem. By learning the logical inverse (NOT) operation, neural logical reasoning is able to model explicit feedback in sequential learning, thus making it possible to conduct counterfactual reasoning over users' explicit like/dislike feedback to generate explicit counterfactual examples for data augmentation.

In this paper, we integrate the two important types of reasoning ability—counterfactual reasoning and logical reasoning, and we propose a novel Counterfactual Collaborative Reasoning (CCR) framework which is able to generate explicit counterfactual examples to enhance the performance of sequential recommendation models. Technically, to create explicit counterfactual data, CCR provides a machine learning-based method to generate minimal changes on the users' historical explicit feedback that can lead to changes in the output under the NCR sampler model (Figure 1(b)). A very desirable feature of the CCR framework is its "augment once, apply to all" property, i.e., for a given dataset, we only need to conduct the data augmentation procedure for once to enrich the dataset, and the enriched dataset can be used to enhance the performance of multiple recommendation models. An additional benefit of the CCR framework is that the data augmentation procedure can naturally produce counterfactual explanations for the recommendation, which not only improves the recommendation performance but also helps to understand the reason of the recommendations.

Experiments on three real-world datasets show that our CCR framework achieves significantly better performance than the models without data augmentation and the models with current existing data augmentation methods for sequential recommendation. Besides, quantitative evaluation results also show that our framework generates reliable explanations for the recommendations.

The key contributions of the paper are as follows:

- To the best of our knowledge, this work is the first to consider explicit counterfactual data augmentation for sequential recommendation. Besides, we demonstrate how logical reasoning and counterfactual reasoning–two of the most important reasoning abilities of humans–can be jointly modeled for better performance and explainability.
- Under the "augment once, apply to all" framework, the generated explicit counterfactual data can improve the performance of multiple sequential recommendation models.
- The data augmentation process not only enhances the recommendation performance but also improves the explainability.
- We conduct experiments on several real-world datasets to analyze both the recommendation performance and the explanation performance.

The following part of the paper is organized as follows: We review related work in Section 2, provide some preliminary knowledge in Section 3, present the details of our CCR framework in Section 4, and discuss the experimental results in Section 5. We finally conclude the work with future research visions in Section 6.

## 2 RELATED WORK

Sequential recommendation is one of the most important types of recommendation models in real-world systems due to their generally good performance. The key idea is to predict the next item based on the user's historical interactions. For example, Markov Chain-based models assume that each of the user's behavior is determined

by the user's most recent behavior thus construct the transition matrix among items based on users' behaviors to predict users' future preference [28, 45]. To relax the most recent behavior assumption, researchers have made efforts to consider longer user behavior records for recommendation. For example, recurrent neural network based models such as GRU4Rec [30], DREAM [64], NARM [35] and other variants [15, 29, 44, 55] embed users' historical behaviors into a latent vector/representation to predict users' future behaviors. Recently, researchers also considered convolutional networks (Caser) [50], memory networks (MANN, KSR) [12, 32], attention mechanism (STAMP, SASRec) [33, 42], self-supervised learning ($S^3$-Rec) [70], bi-directional transformers (BERT4Rec, SSE-PT, XLNet) [14, 47, 56], logical reasoning (NCR) [5, 8, 46], and foundation models (P5) [24] for sequential recommendation.

Compared with non-sequential models, sequential models need higher-quality sequential data for training since the output will be directly influenced by the input sequence [11, 53, 60]. However, the sparsity of real-world data may hinder the performance of the sequential recommendation models. To alleviate this problem, researchers have been exploring counterfactual thinking for data augmentation so as to enhance both the dataset and the model. The data augmentation process uses alternatives to exchange the past behaviors and generate counterfactual examples [16]. Following this idea, counterfactual data augmentation has made several important achievements in recommender systems [11, 53, 60, 63, 65].

However, existing counterfactual data augmentation methods for recommender systems only consider users' implicit feedback and cannot deal with the explicit feedback for counterfactual example generation [53, 63]. Actually, explicit information is also very useful in recommendation, since explicit and implicit feedback exhibit user preference from different perspectives. As a result, considering both types of feedback can enhance sequential modeling performance [9]. For example, neural logic reasoning (NLR) [46], neural collaborative reasoning (NCR) [8], and graph collaborative reasoning (GCR) [5] consider users' like and dislike feedback for sequential modeling.

Explainability is another important perspective for recommender systems research because it improves users' trust and satisfaction and helps the system designers in model debugging [13, 20, 67, 68]. Researchers have explored various types of explanation styles, such as pre-defined templates [52, 68], image visualizations [10, 22], knowledge graph paths [1, 6, 23, 58], feature comparisons [59, 62], reasoning rules [7, 8, 31, 46, 57, 66, 71] and natural language sentences [4, 22, 24, 35–39], and recently, counterfactual reasoning has emerged as an effective method to generate explanations [25, 49, 51, 61]. Explainable AI has also been driving research in computer vision [26], natural language processing [17], AI for Science [41, 48] and algorithmic fairness [19, 21, 40].

## 3 PRELIMINARIES

In this section, we briefly introduce the notations we used in this work as well as some background knowledge.

### 3.1 Sequential Recommendation

Suppose we have a user set $U = \{u_1, u_2, \cdots, u_{|U|}\}$ and an item set $V = \{v_1, v_2, \cdots, v_{|V|}\}$. The user $u_i$ interacted with a sequence of historical items $H_i = \{v_1^i, v_2^i, \cdots, v_n^i\}$. The corresponding feedback

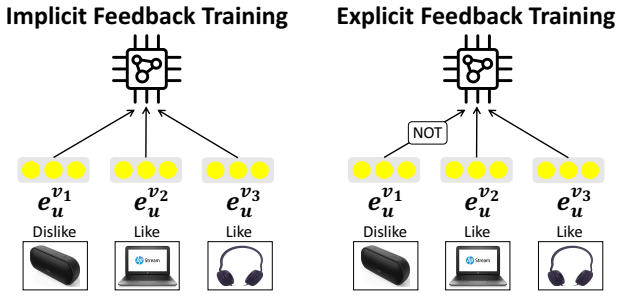



**Figure 2: An illustration of the difference between sequential modeling with implicit feedback and with explicit feedback. The explicit feedback model leverages the logical NOT gate to distinguish between positive and negative feedback.**

of these interacted historical items are $B_i = \{b_1^i, b_2^i, \cdots, b_n^i\}$. We let $b_t^i = 1$ if the user likes the item and $b_t^i = 0$ if the user dislikes the item. A sequential recommendation model $f$ predicts the ranking score $r_{ij}$ for user $u_i$ on item $v_j$ based on $H_i$ and $B_i$:

$$r_{ij} = f(H_i, B_i, v_j) \tag{1}$$

By ranking the candidate items in descending order of the ranking score $r_{ij}$, the recommendation model predicts the user's preferences and produces the recommendation list. Sequential recommendation models can predict users' near future behaviors based on their history behaviors. Sometimes, we not only have users' interacted items from the dataset, but also we can know whether they like each item or not. However, most of the existing sequential recommendation models only consider the interacted histories for recommendation, or in other words, they only use implicit feedback information to train the model and make prediction, which may lose part of the useful information and may hinder the ability to make accurate predictions. One approach to modeling explicit feedback for sequential recommendation is neural collaborative reasoning (NCR) [8], which we briefly introduce in the following.

### 3.2 Sequential Modeling with Explicit Feedback

Neural collaborative reasoning (NCR) [8] models the explicit feedback for sequential recommendation by training the neural logical NOT operator. By applying the NOT operation on history items, the model is able to distinguish users' positive and negative feedback on the historical items, as shown in Figure 2. More specifically, NCR encodes a user-item interaction into an interaction event vector:

$$\boldsymbol{e}_u^v = \boldsymbol{W}_2\phi(\boldsymbol{W}_1\begin{bmatrix}\boldsymbol{u}\\ \boldsymbol{v}\end{bmatrix} + \boldsymbol{b}_1) + \boldsymbol{b}_2 \tag{2}$$

where $\boldsymbol{u}, \boldsymbol{v}$ are user and item latent embedding vectors; $\boldsymbol{W}_1, \boldsymbol{W}_2$ and $\boldsymbol{b}_1, \boldsymbol{b}_2$ are weighted matrices and bias vectors that need to be learned; $\boldsymbol{e}_u^v$ is the encoded event vector that represents the interaction of user $u$ and item $v$, and $\phi(\cdot)$ is the activation function which is rectified linear unit (ReLU) in this work. By introducing neural logical modules: AND ($\wedge$), OR ($\vee$) and NOT ($\neg$), NCR can transform the sequential data into a logical expression. Suppose user $u$'s feedback on item $v_1$ is negative and the feedback on other items are positive, then the explicit reasoning expression is:

$$\neg \boldsymbol{e}_u^{v_1} \wedge \boldsymbol{e}_u^{v_2} \wedge \cdots \wedge \boldsymbol{e}_u^{v_n} \rightarrow \boldsymbol{e}_u^{v_{n+1}} \tag{3}$$

where $\{e_u^{v_1}, e_u^{v_2}, \cdots, e_u^{v_n}\}$ represents the user's history event and $e_u^{v_{n+1}}$ is the next item the user interacts with. To put explicit feedback information into consideration, $e_u^v$ is used to represent that user $u$ interacted with item $v$ with positive feedback and and $\neg e_u^v$ is used to represent that user $u$ interacted with item $v$ with negative feedback. Ideally, the sequential recommendation procedure can predict $e_u^{v_{n+1}}$ based on $\{e_u^{v_1}, e_u^{v_2}, \cdots, e_u^{v_n}\}$. Based on the definition of material implication[1], the expression is equivalent to:

$$(\neg\neg e_u^{v_1} \vee \neg e_u^{v_2} \vee \cdots \vee \neg e_u^{v_n}) \vee e_u^{v_{n+1}} \tag{4}$$

The recommendation score of a candidate item $v_{n+1}$ is calculated based on the similarity between the logical expression and the constant True (T) vector. Based on the score, the model will decide whether the item should be recommend to the user (if the expression is close to True) or not (if the expression is close to False).

## 4 COUNTERFACTUAL COLLABORATIVE REASONING (CCR)

We build an counterfactual collaborative reasoning (CCR) framework to generate explicit counterfactual examples for data augmentation and improve the performance of sequential recommendation models. The main idea of the proposed data augmentation framework is to discover slight changes $\Delta$ on users' explicit feedback via solving a counterfactual optimization problem which will be formulated in the following. Meanwhile, the process of generating explicit counterfactual data can also provide explanations for items in the top-$K$ recommendation.

### 4.1 Explicit Counterfactual Data Sampler

As shown in figure 1(b), besides a sequential recommendation model $\mathcal{A}$, our CCR framework introduces a sampler $\mathcal{S}$ to generate explicit counterfactual examples. Firstly, both $\mathcal{A}$ and $\mathcal{S}$ in our model are pre-trained based on the original dataset. Then, the explicit counterfactual data generated by the sampler will be used to re-optimize the anchor model $\mathcal{A}$. After that, the re-optimized anchor model will provide the final recommendation list for the user.

To generate explicit counterfactual data, we use NCR as the sampler to conduct counterfactual reasoning. This is because NCR can consider the counterfactual of explicit feedback with the help of logical negations ($\neg$). The first step of the sampler is to decide which explicit feedback of a user $u_i$'s historical items should be changed. We use a binary vector $\boldsymbol{\Delta_i} = \{0, 1\}^{|\boldsymbol{B_i}|}$ to represent the intervention, where the the vector size is equal to the size of the user's explicit feedback vector $\boldsymbol{B_i}$. Then, we apply the intervention on $\boldsymbol{B_i}$:

$$\boldsymbol{B_i^*} = (1 - \boldsymbol{B_i}) \odot \boldsymbol{\Delta_i} + \boldsymbol{B_i} \odot (1 - \boldsymbol{\Delta_i}) \tag{5}$$

For each $\delta_t \in \boldsymbol{\Delta_i}$, if $\delta_t = 1$, then the corresponding feedback is reversed; otherwise, the feedback remains the same. For example, if $\boldsymbol{B_i} = [0, 1, 1]$ and $\boldsymbol{\Delta_i} = [1, 1, 0]$, it means that the user's feedback on the first and second items should be reversed, thus $\boldsymbol{B_i^*} = [1, 0, 1]$. To decide which feedback should be changed, we design an optimization function for $\boldsymbol{\Delta_i}$:

$$\boldsymbol{\Delta_i} = \text{argmin}_{\boldsymbol{\Delta_i}} \, \|\boldsymbol{\Delta_i}\|_0 + \alpha \cdot \mathcal{S}(\boldsymbol{v_{n+1}} \mid \boldsymbol{H_i}, \boldsymbol{B_i^*}) \tag{6}$$

[1]Material Implication ($\rightarrow$) can be represented as: $x \rightarrow y \Leftrightarrow \neg x \vee y$

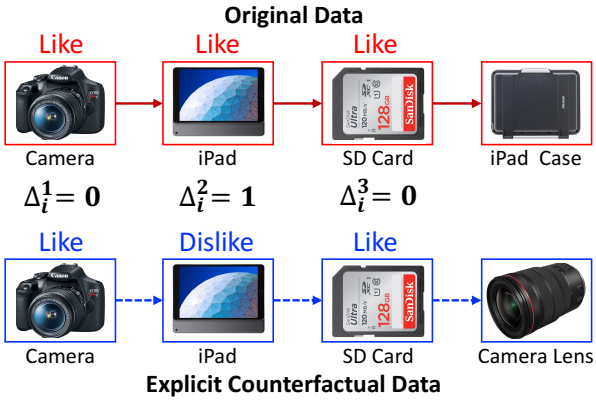



**Figure 3: An illustration of explicit counterfactual data generation progress. In the original data, the user likes the iPad and then they purchased an iPad case which is compatible with the iPad. Suppose $\Delta_i^2 = 1$ and the user's explicit feedback on iPad is changed, then the sampler will generate a new item as the next item.**

where $\|\boldsymbol{\Delta_i}\|_0$ is the zero-norm of the intervention vector $\boldsymbol{\Delta_i}$ that represents the amount of changed feedback, $\alpha$ is a hyper-parameter, and $\boldsymbol{v_{n+1}}$ is the item embedding vector of the ground-truth next item. $\mathcal{S}(\boldsymbol{v_{n+1}} \mid \boldsymbol{H_i}, \boldsymbol{B_i^*})$ is the ranking score of the sampler model for user $u_i$ on item $v_{n+1}$ under the counterfactual user feedback $\boldsymbol{B_i^*}$. In Eq.(6), the first term aims to minimize the amount of intervened feedback between the original data and the explicit counterfactual data. The second term tries to find the explicit feedback that can alter the output the sequence, i.e., the ranking score of item $v_{n+1}$ under the counterfactual feedback is decreased so that a new item appears as the output. However, the $\boldsymbol{\Delta_i}$ is not differentiable since it is discrete. Thus, we will introduce a relaxed optimization method later.

To get the new next item, we take the history items $\boldsymbol{H_i}$ and the intervened feedback $\boldsymbol{B_i^*}$ into the sampler to derive the next item $\hat{\boldsymbol{v}}_{n+1}$ that the user may interact with:

$$\hat{\boldsymbol{v}}_{n+1} = \text{argmax}_{v \in I} \, \mathcal{S}(\boldsymbol{v} \mid \boldsymbol{H_i}, \boldsymbol{B_i^*}) \tag{7}$$

where $I$ is a set of items in the dataset, which can be the whole item set $V$ or another set involving prior knowledge. Finally, an explicit counterfactual sequence is generated as $(H_i, B_i^*, \hat{v}_{n+1})$. Together with the original sequence $(H_i, B_i, v_{n+1})$, the anchor model $\mathcal{A}$ will be optimized over the augmented training dataset.

The intuition of the explicit counterfactual data augmentation procedure is that the sampler model creates "difficult" examples that leads to changes in the next item even if the historical feedback is just slightly changed. Such difficult examples, once included as augment examples to enhance the training data, will help the training of sequential recommendation models to distinguish the influence of minor changes in users' historical feedback [53]. Take Figure 3 as an example, in the original sequential data, user purchased an iPad case as the next item since it is compatible with the iPad that the user already purchased and the user actually likes the iPad. However, if the user dislike the iPad, then the next item would not be an iPad case but could be a camera lens since it is compatible with the camera that the user liked before.

## 4.2 Relaxed Optimization

One difficulty in optimizing Eq.(6) is that the binary vector $\mathbf{\Delta_i}$ and the first term of the optimization function $\|\mathbf{\Delta_i}\|_0$ are not differentiable since they are discrete. In the implementation, we relax $\mathbf{\Delta_i}$ to a real-valued vector and relax the $\ell_0$-norm $\|\mathbf{\Delta_i}\|_0$ to $\ell_1$-norm $\|\mathbf{\Delta_i}\|_1$ to make the intervention vector $\mathbf{\Delta_i}$ learnable. This method has been shown to be effective in prior research [2, 3] and helps to minimize the number of changed explicit feedback in the sequence.

$$\mathbf{\Delta_i} = \text{argmin}_{\mathbf{\Delta_i}} \|\mathbf{\Delta_i}\|_1 + \alpha \cdot \mathcal{S}(v_{n+1} \mid \boldsymbol{H_i}, \boldsymbol{B_i^*}) \tag{8}$$

As mentioned before, the sampler $\mathcal{S}$ is implemented with NCR to accommodate the explicit feedback in the sequence. More specifically, for each user-item interaction event $e$ in the history $H_i$, suppose $\boldsymbol{e}$ is the corresponding event vector for the event (which could be negated if this event is a negative feedback in the original data), and suppose event $e$'s corresponding value in the intervention vector is $\delta_e$, then the intervened event vector $\boldsymbol{e}^*$ is:

$$\boldsymbol{e}^* = \neg \boldsymbol{e} \cdot \delta_e + \boldsymbol{e} \cdot (1 - \delta_e) \tag{9}$$

These intervened event vectors constitute the counterfactual history $\{\boldsymbol{H_i}, \boldsymbol{B_i^*}\}$ for the NCR sampler to calculate the ranking score of item $v_{n+1}$, as shown in Eq.(8). After optimization, the values in the learned intervention vector $\mathbf{\Delta_i}$ may not be exactly equal to 0 or 1. As a result, a threshold will be applied to binarize $\mathbf{\Delta_i}$ as the final output. In the experiments, we set the threshold as 0.5, i.e., for those elements in $\mathbf{\Delta_i}$ larger than 0.5, we set them to 1, otherwise, we set them to 0. Finally, the binarized intervention vector $\mathbf{\Delta_i}$ is used to generate the new next item $\hat{v}_{n+1}$ for the explicit counterfactual example according to Eq.(7).

## 4.3 Reduce Noisy Examples

As mentioned above, for the generated explicit counterfactual data, the next interaction item is selected based on Eq.(7). However, since no sampler model is perfectly accurate, it may generate inaccurate predictions. Both of the accurate and inaccurate counterfactual examples generated by the sampler will be used to re-optimized the anchor model $\mathcal{A}$, as a result, if we do not set some constraints to reduce the amount of inaccurate counterfactual examples, the performance of the re-optimized anchor model may be harmed. Inspired by [53], we set a confidence parameter $\kappa \in [0, 1)$ to mitigate this issue. We accept the generated explicit counterfactual data only when $\mathcal{S}(\hat{v}_{n+1} \mid \boldsymbol{H_i}, \boldsymbol{B_i^*}) > \kappa$. This means that we will only accept a counterfactual example when the sampler is sufficiently confident of it. Otherwise, the model will discard the example.

## 4.4 Learning Algorithm

To make the whole process clear, we summarize the learning algorithm of our framework in Algorithm 1. At first, we train both the sampler $\mathcal{S}$ and the anchor model $\mathcal{A}$ based on the original dataset $T$. Then for each training example in $T$, the sampler will learn the intervention vector $\Delta_i$ and generate counterfactual data based on Eq.(7). If the sampler has enough confidence of the generated counterfactual data, it will be added into the counterfactual dataset $T_c$. When the model finishes the data augmentation process, the anchor model $\mathcal{A}$ will be re-optimized based on $T_c \cup T$ to provide the final recommendations for each user.

**Algorithm 1** CCR Learning Algorithm

```
Input: the original dataset T
Input: Pre-train sampler S and anchor model A
Initialize the counterfactual dataset T_c = ∅
for each training example from T do
    Randomly initialize the intervention vector Δ_i
    Learn Δ_i by Eq.(8)
    Generate new sequence (H_i, B_i^*, v̂_{n+1}) based on Eq.(7)
    if S(v̂_{n+1} | H_i, B_i^*) > κ then
        T_c ← T_c ∪ (H_i, B_i^*, v̂_{n+1})
    end if
end for
Re-optimize A based on T ∪ T_c
Output: the final recommendations based on re-optimized A
```

## 4.5 Counterfactual Explanations

During the process of generating explicit counterfactual data, our framework can also provide explanations to show why the model recommends the item to the user. Previous counterfactual explanation methods for recommendation [25, 49, 51] mostly focus on implicit counterfactual explanation based on implicit behaviors. However, one contribution of our work is that our framework can generate explicit counterfactual explanations.

In Eq.(8), we are trying to explore an intervention vector $\mathbf{\Delta_i}$. Because of the first term in Eq.(8), only a few explicit feedback will be changed. Meanwhile, the second term of Eq.(8) will penalize the probability of interacting with the current item. Therefore, only the most essential history items' feedback will be changed. These items can be used to generate counterfactual explanations for the recommended item. Take Figure 4 as an example, since $\Delta_i^2 = 1$, the corresponding item is the counterfactual explanation: the iPad is the reason for recommending the iPad case, because if the user disliked the iPad, we would not have recommended the iPad case but would have recommended the camera lens instead. We store all explanations in the set $E$.

To evaluate if our explanation correctly explains the recommended item, inspired by recent work on counterfactual explainable recommendation [48], we use Probability of Necessity (PN) and Probability of Sufficiency (PS) to evaluate our explanations. In logic and mathematics, if $X$ happens then $Y$ will happen, we say $X$ is a sufficient condition for $Y$. Similarly, if $X$ does not happen then $Y$ will not happen, we say $X$ is a necessary condition for $Y$.

*4.5.1* ***Probability of Necessity.*** Suppose a set of items $E_{ij} \subset V$ constitute the explanation for the recommended item $v_j$ to user $u_i$. The idea of the PN score is: if the items in $E_{ij}$ are reversed (for explicit explanation) or removed (for implicit explanation), then what is the probability that item $v_j$ would not be recommended for user $u_i$. We calculate the percentage of the generated explanations that meet the above PN condition:

$$\text{PN} = \frac{\sum_{\text{u}_\text{i} \in \text{U}} \sum_{\text{v}_\text{j} \in \text{R}_\text{i,K}} \text{PN}_\text{ij}}{\sum_{\text{u}_\text{i} \in \text{U}} \sum_{\text{v}_\text{j} \in \text{R}_\text{i,K}} \text{I}(\text{E}_\text{ij} \neq \emptyset)}, \ \text{PN}_\text{ij} = \begin{cases} 1, \text{ if } \text{v}_\text{j} \notin \text{R}^*_\text{i,K} \\ 0, \text{ else} \end{cases} \tag{10}$$

where $\text{R}_\text{i,K}$ is the original top-K recommendation list for user $u_i$. Let $v_j \in \text{R}_\text{i,K}$ be a recommended item that our model has a nonempty

explanation $E_{ij} \neq \emptyset$. Then for the original sequence data, we intervene (reverse or remove) the item(s) in $E_{ij}$ and get the new recommendation list $R^{*}_{i,K}$ for user $u_i$ from the recommendation model. $I(E_{ij} \neq \emptyset)$ is an identity function: when $E_{ij} \neq \emptyset$, $I(E_{ij} \neq \emptyset) = 1$. Otherwise, $I(E_{ij} \neq \emptyset) = 0$.

*4.5.2* ***Probability of Sufficiency.*** Similar to the definition of PN, the idea of PS score is: if the items in $E_{ij}$ are maintained while other items are reversed (for explicit explanation) or removed (for implicit explanation), then what is the probability that item $v_j$ would still be recommended for user $u_i$. We calculate the percentage of the generated explanations that meet the above PS condition:

$$\mathrm{PS} = \frac{\sum_{u_i \in U} \sum_{v_j \in R_{i,K}} \mathrm{PS}_{ij}}{\sum_{u_i \in U} \sum_{v_j \in R_{i,K}} (E_{ij} \neq \emptyset)}, \ \mathrm{PS}_{ij} = \begin{cases} 1, & \text{if } v_j \in R'_{i,K} \\ 0, & \text{else} \end{cases} \tag{11}$$

where $R'_{i,K}$ is the new recommendation list after the intervention is applied, and other notations have similar meanings as above.

## 5 EXPERIMENTS

In this section, we conduct experiments on three real-world datasets and compare the results of (1) the original sequential recommendation model without data augmentation, (2) models with implicit counterfactual data augmentation, and (3) models with our Counterfactual Collaborative Reasoning (CCR) framework. Furthermore, the counterfactual explanation results show our framework's ability to generate higher quality explanations.

### 5.1 Dataset

We use three real-world datasets in the experiments.

**ML100K** [27]: The MovieLens-100K (ML100K) dataset stores users' preference for various movies. It contains 100,000 movie ratings from 1 to 5 stars of 943 users to 1,682 movies.

**Amazon** [43]: This is the Amazon e-commerce dataset. We take **Movies** & **TV** and **Electronics** datasets as two examples for experiments. Movies & TV contains 123,961 users, 50,053 products and 1,697,533 product ratings. Electronics contains 192,404 users, 63,002 products and 1,689,188 product ratings.

Some basic statistic of the datasets can be found in Table 1. We consider 1-3 ratings as negative feedback with label as 0, and 4-5 ratings as positive feedback with label as 1. We use positive Leave-One-Out [8, 69] to create the training, validation and testing dataset. For each user, we put the last positive interaction and its following negative interactions into the testing set, and we put the last but one positive interaction and its following negative interactions into the validation set. Then, we put all of the rest interactions into the training set. If a user has less than 5 interactions, we put all of the interactions into the training set to avoid cold-start.

### 5.2 Baselines

We consider both standalone sequential recommendation models and implicit data augmentation methods for comparison:

**GRU4Rec** [30]: GRU4Rec is a sequential recommendation model based on Recurrent Neural Networks (RNN).

**STAMP** [42]: STAMP is a sequential recommendation model based on attention mechanism, which can capture users' long-term and short-term preferences for recommendation.

| Dataset | #users | #items | #interaction | Density |
|---|---|---|---|---|
| ML100K | 943 | 1,682 | 100,000 | 6.30% |
| Movies & TV | 123,961 | 50,053 | 1,697,533 | 0.027% |
| Electronics | 192,404 | 63,002 | 1,689,188 | 0.014% |

**Table 1: Basic statistics of the datasets**

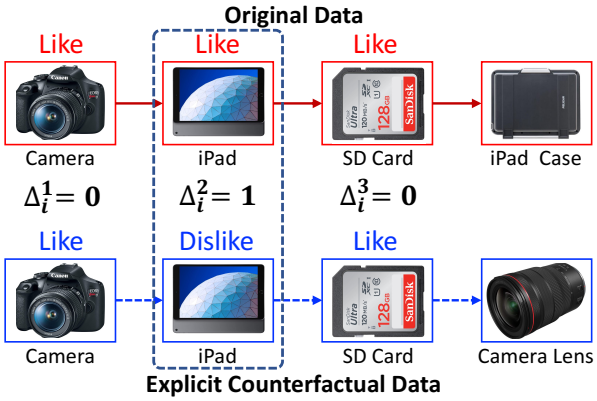



**Figure 4: An example of generating counterfactual explanations. Since $\Delta_i^2 = 1$, the corresponding item is the counterfactual explanation for the recommended item.**

**SASRec** [33]: SASRec is a sequential recommendation model based on self-attention mechanism

**NCR** [8]: NCR is a sequential recommendation model based on neural logical reasoning, which captures the logical relationship between user-item interactions for recommendation.

**CASR** [53]: CASR is a state-of-the-art implicit counterfactual data augmentation method for sequential modeling.

Both CASR and our CCR frameworks can be applied on all of the four recommendation models.

### 5.3 Implementation Details

The learning rate is searched in [0.0001,0.001,0.01,0.1] for all methods. We apply ReLU as the activation function. For all methods, the embedding size is 64. We optimize the methods using mini-batch [34] and the batch size is 128. The hyper-parameter $\alpha$ is searched in $[10^{-3},10^{-2},10^{-1},1,10^{1},10^{2},10^{3}]$, and finally set to 10 for the results we report in the paper. The confidence parameter $\kappa$ is searched from 0 to 1. The influence of different $\kappa$ on the performance will be discussed in the following experiments. We tune each model's parameters to its own best performance on the validation set. For both CASR (implicit counterfactual data augmentation) and CCR (explicit counterfactual data augmentation), we generate one counterfactual example for each sequence in the training set.

### 5.4 Evaluation Metrics

We use Normalized Discounted Cumulative Gain at rank $K$ (NDCG@K) and Hit Ratio at rank $K$ (HR@K) to evaluate recommendation performance. To evaluate the explanation performance, we use Probability of Necessity (PN), Probability of Sufficiency (PS) and their harmonic mean $\mathrm{F_{NS}} = \frac{2 \cdot \mathrm{PN} \cdot \mathrm{PS}}{\mathrm{PN}+\mathrm{PS}}$. For each user-item pair in the validation set and the test set, we randomly sample 100 irrelevant items and rank all of these 101 items for recommendation.

| Dataset | ML100K | | | | Movies & TV | | | | Electronics | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metric | NDCG@5 | NDCG@10 | HR@5 | HR@10 | NDCG@5 | NDCG@10 | HR@5 | HR@10 | NDCG@5 | NDCG@10 | HR@5 | HR@10 |
| STAMP | 0.342 | 0.402 | 0.503 | 0.665 | 0.406 | 0.427 | 0.521 | 0.657 | 0.301 | 0.341 | 0.412 | 0.542 |
| CASR-STAMP | 0.351 | 0.406 | 0.511 | 0.676 | 0.412 | 0.445 | 0.532 | 0.661 | 0.307 | 0.349 | 0.425 | 0.553 |
| CCR-STAMP | **0.365*** | **0.418*** | **0.527*** | **0.693*** | **0.435*** | **0.463*** | **0.552*** | **0.689*** | **0.329*** | **0.364*** | **0.453*** | **0.570*** |
| GRU4Rec | 0.340 | 0.403 | 0.502 | 0.672 | 0.411 | 0.431 | 0.538 | 0.661 | 0.312 | 0.354 | 0.432 | 0.554 |
| CASR-GRU4Rec | 0.349 | 0.411 | 0.509 | 0.680 | 0.414 | 0.453 | 0.542 | 0.671 | 0.326 | 0.369 | 0.447 | 0.560 |
| CCR-GRU4Rec | **0.370*** | **0.429*** | **0.522*** | **0.689*** | **0.428*** | **0.466*** | **0.557*** | **0.695*** | **0.344*** | **0.386*** | **0.476*** | **0.581*** |
| SASRec | 0.348 | 0.411 | 0.508 | 0.678 | 0.412 | 0.456 | 0.543 | 0.667 | 0.322 | 0.357 | 0.439 | 0.558 |
| CASR-SASRec | 0.357 | 0.415 | 0.518 | 0.685 | 0.420 | 0.461 | 0.549 | 0.673 | 0.335 | 0.365 | 0.450 | 0.575 |
| CCR-SASRec | **0.376*** | **0.427*** | **0.531*** | **0.701*** | **0.438*** | **0.479*** | **0.562*** | **0.699*** | **0.358*** | **0.390*** | **0.471*** | **0.592*** |
| NCR | 0.359 | 0.412 | 0.514 | 0.680 | 0.415 | 0.457 | 0.551 | 0.673 | 0.332 | 0.366 | 0.441 | 0.557 |
| CASR-NCR | 0.362 | 0.419 | 0.518 | 0.689 | 0.417 | 0.458 | 0.555 | 0.682 | 0.339 | 0.374 | 0.451 | 0.569 |
| CCR-NCR | **0.376*** | **0.434*** | **0.535*** | **0.705*** | **0.433*** | **0.472*** | **0.568*** | **0.702*** | **0.354*** | **0.395*** | **0.469*** | **0.588*** |

**Table 2: Experimental results on Normalize Discounted Cumulative Gain (NDCG) and Hit Ratio (HR). For each model, we present the performance of the original model, the results of applying implicit counterfactual data augmentation method CASR on the model, and the results of applying our CCR method on the model. Bold numbers represent best performance. We use * to indicate that the performance is significant better than other baselines. The significance is at 0.05 level on paired t-test.**

| Dataset | ML100K | | | | | | Movies & TV | | | | | | Electronics | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Top N | N=1 | | | N=5 | | | N=1 | | | N=5 | | | N=1 | | | N=5 | | |
| Metric | PN% | PS% | $F_{NS}$% | PN% | PS% | $F_{NS}$% | PN% | PS% | $F_{NS}$% | PN% | PS% | $F_{NS}$% | PN% | PS% | $F_{NS}$% | PN% | PS% | $F_{NS}$% |
| CASR-STAMP | 25.6 | 7.7 | 11.8 | 22.1 | 12.6 | 11.8 | 29.2 | 17.8 | 22.2 | 18.4 | 17.1 | 17.7 | 30.9 | 8.7 | 13.6 | 24.9 | 12.1 | 16.3 |
| CASR-GRU4Rec | 21.7 | 8.2 | 11.9 | 18.4 | 13.1 | 15.3 | 23.9 | 3.5 | 6.1 | 19.8 | 8.4 | 11.8 | 22.4 | 9.9 | 13.8 | 20.6 | 14.0 | 16.7 |
| CASR-SASRec | 23.9 | 9.3 | 13.4 | 19.3 | 13.5 | 15.8 | 25.7 | 18.2 | 21.3 | 17.4 | 18.9 | 18.1 | 25.3 | 10.2 | 14.5 | 19.4 | 14.5 | 16.6 |
| CASR-NCR | 17.2 | 32.5 | 22.5 | 14.7 | 36.8 | 21.0 | 19.4 | 38.6 | 25.8 | 10.8 | 44.3 | 17.4 | 19.9 | 39.0 | 26.3 | 16.3 | 40.3 | 23.2 |
| CountER-STAMP | **53.2** | 17.0 | 25.8 | **38.3** | 26.4 | 31.2 | **58.6** | 36.8 | 45.2 | **47.9** | 43.8 | 45.8 | **59.6** | 18.6 | 28.3 | **48.1** | 27.5 | 34.9 |
| CountER-GRU4Rec | 40.8 | 19.1 | 26.0 | 34.5 | 29.5 | 31.8 | 46.1 | 6.9 | 12.0 | 41.3 | 14.9 | 21.9 | 43.5 | 19.2 | 26.6 | 39.7 | 29.6 | 33.9 |
| CountER-SASRec | 45.3 | 21.9 | 29.5 | 36.0 | 30.1 | 32.7 | 48.3 | 39.7 | 43.5 | 40.5 | 45.5 | 42.8 | 50.7 | 23.9 | 32.4 | 41.2 | 36.6 | 38.7 |
| CountER-NCR | 34.7 | 52.4 | 41.7 | 28.1 | 54.5 | 37.1 | 42.7 | 53.7 | 47.5 | 34.9 | 59.0 | 43.8 | 36.7 | 52.3 | 43.2 | 32.2 | 57.8 | 41.3 |
| CCR | 42.1 | **60.3** | **49.6** | 36.1 | **66.7** | **46.8** | 50.1 | **64.7** | **56.5** | 41.9 | **73.2** | **53.3** | 45.8 | **74.8** | **56.8** | 41.0 | **79.6** | **54.1** |

**Table 3: Results on PN, PS and $F_{NS}$. Bold numbers are best performance. All numbers are percentage numbers with % omitted. When CCR achieves the best result, its improvements against the best baseline are significant at $p < 0.01$.**

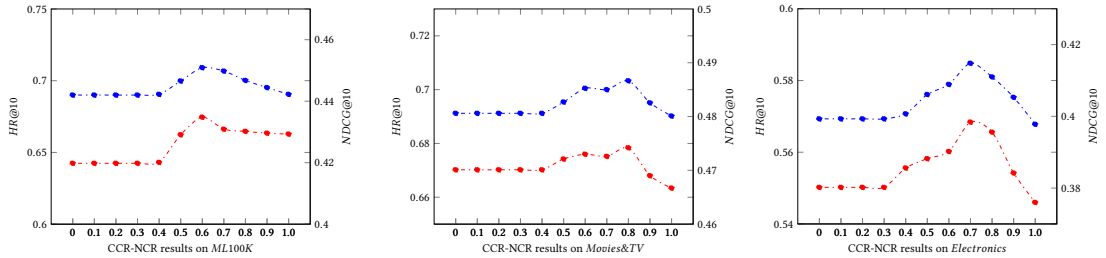



**Figure 5: Performance on HR@10 (Blue Line) and NDCG@10 (Red Line) on different $\kappa$ with different datasets.**

## 5.5 Compatible with the Baselines

One issue in comparison with baselines is that our framework can generate explanations in the counterfactual generation progress while the baseline methods can not. We use two approaches to make the baselines compatible for explanation evaluation.

First, we apply CASR on each of the four recommendation models to generate explanations. Since CASR manually selects one item to intervene, we intervene each item in each example's history and select the one that achieves the highest $F_{NS}$ score as the CASR explanation. Second, to get stronger explanations for each model, we use our framework as a guideline to tell the baseline methods how many items they should use to generate explanations. Then, we apply the counterfactual explainable recommendation (CountER) framework [49] to each of the recommendation model (STAMP, GRU4Rec, SASRec, NCR) to generate explanations for them: based on the number of items, we search all of the combinations of users' history items

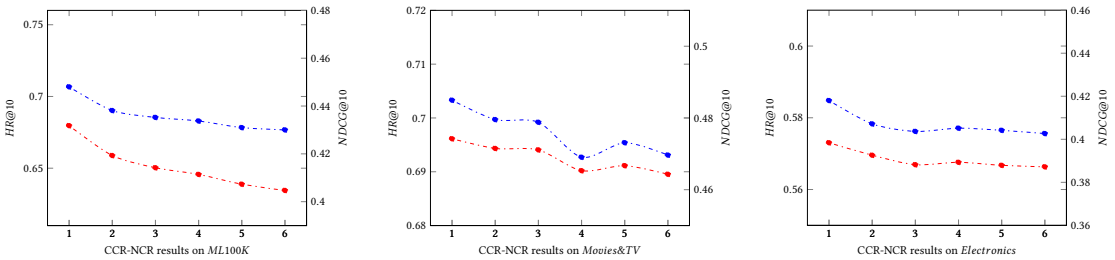



**Figure 6: Performance on HR@10 (Blue Line) and NDCG@10 (Red Line) with different iterations.**

as candidate explanations and take the one that gives highest $F_{NS}$ score as the explanation of the model under CountER framework. Finally, since the CCR framework works in the "augment once, apply to all" paradigm and directly produces explicit explanation with NCR sampler during data augmentation, we directly take its output explanation for evaluation.

## 5.6 Performance on Recommendation

The experimental results on NDCG and Hit Ratio (HR) are shown in Table 2. Based on the results, we have following observations.

First and most importantly, compared with the original model and the model under implicit counterfactual data augmentation, our CCR framework achieves significantly better performance than the baseline methods on all of these three datasets. Compared with the original model, CCR can get better results based on the the generated explicit counterfactual data, which alleviates the data scarcity and encodes informative examples into the training dataset. Compared with the implicit counterfactual data augmentation method CASR, the explicit counterfactual data generated by CCR are more effective since CCR takes advantages of the explicit feedback.

## 5.7 Performance on Explanation

The experimental results on explanation are shown in Table 3. Based on the experiment results, we have following observations.

In terms of the overall explanation performance ($F_{NS}$), CountER-based explanations are better than CASR-based explanations. This is understandable since CountER uses the optimal number of items from CCR to generate explanations while CASR selects one and only one item as explanation. Furthermore, by considering explicit feedback, CCR generates even better explanations than CountER-based methods.

Besides, by considering explicit feedback, the CCR explanations are better than CASR explanations on both PN and PS and thus better overall explanation quality $F_{NS}$. This means that the CCR augmented examples have higher quality since they are more sufficient and necessary, and when these better examples are used to augment the dataset, it helps CCR to achieve better recommendation performance than CASR.

## 5.8 Impact of Hyper-Parameters

*5.8.1* ***Impact of*** $\kappa$. In the generation process of the explicit counterfactual data, we have a confidence parameter $\kappa$. We accept the generated counterfactual data only when the ranking score of the data is larger than $\kappa$. The results on the influence of $\kappa$ is shown in Figure 5. From the figure, we can see that our framework will have the best performance when $\kappa$ is set around 0.7. When $\kappa$ is very small, it does not change the performance of the framework because all of the generated data can pass the constraint of $\kappa$. When $\kappa$ is too big, the performance will decrease because only a few counterfactual data can pass through the constraint and thus we cannot get enough counterfactual data to re-optimize the anchor model.

## 5.9 Impact of Iterative Re-optimization

Since the re-optimized anchor model $\mathcal{A}'$ achieves better performance than the original anchor model $\mathcal{A}$, a natural idea is that if we use the re-optimized anchor model to generate a set of new augmented data and optimize $\mathcal{A}'$ again to $\mathcal{A}''$ in a boosting way, whether the performance can be even better. We use NCR as an example anchor model to test the idea. Unfortunately, as we can see in Figure 6, the performance on HR and NDCG decreases with the number of rounds of iterative re-optimization. This observation is consistent with previous research [53]. The reason is that since the sampler is not perfectly accurate, the generated counterfactual examples can contain noise and such noise is learned into the anchor model. As a result, multiple rounds of augmentation and re-optimization may propagate such noise and thus decrease the performance. This means that even though data augmentation can improve the model performance, it cannot boost the model performance infinitely.

# 6 CONCLUSION

In this paper, we propose a Counterfactual Collaborative Reasoning (CCR) framework, which integrates the power of logical reasoning and counterfactual reasoning and generates explicit counterfactual data to enhance the performance of sequential recommendation models. Experiments on three real-world datasets verified the effectiveness of the framework. Furthermore, a unique advantage of the CCR framework is that it can also generate explicit counterfactual explanations to better understand the user behavior sequence. In this work, we take recommender system as an example to explore the joint ability of logical and counterfactual reasoning, which are two important types of reasoning abilities for machine learning. On the other hand, they can also be considered to improve other intelligent tasks beyond recommendation, such as vision and language processing tasks, which we will explore in the future.

**Acknowledgment**. This work was supported in part by NSF IIS 1910154, 2007907, 2046457, 2127918 and CCF 2124155. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

# REFERENCES


[1] Q. Ai, V. Azizi, X. Chen, and Y. Zhang. 2018. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms* (2018).

[2] Emmanuel J Candes, Justin K Romberg, and Terence Tao. 2006. Stable signal recovery from incomplete and inaccurate measurements. *CPAM* (2006).

[3] Emmanuel J Candes and Terence Tao. 2005. Decoding by linear programming. *IEEE transactions on information theory* 51, 12 (2005), 4203–4215.

[4] Hanxiong Chen, Xu Chen, Shaoyun Shi, and Yongfeng Zhang. 2019. Generate natural language explanations for recommendation. In *EARS 2019 at SIGIR*.

[5] Hanxiong Chen, Yunqi Li, Shaoyun Shi, Shuchang Liu, He Zhu, and Yongfeng Zhang. 2022. Graph collaborative reasoning. In *WSDM*. 75–84.

[6] Hongxu Chen, Yicong Li, Xiangguo Sun, Guandong Xu, and Hongzhi Yin. 2021. Temporal meta-path guided explainable recommendation. In *WSDM*. 1056–1064.

[7] H. Chen, Y. Li, H. Zhu, and Y. Zhang. 2022. Learn Basic Skills and Reuse: Modularized Adaptive Neural Architecture Search (MANAS). In *CIKM*. 169–179.

[8] Hanxiong Chen, Shaoyun Shi, Yunqi Li, and Yongfeng Zhang. 2021. Neural collaborative reasoning. In *Proceedings of the Web Conference 2021*. 1516–1527.

[9] Shulong Chen and Yuxing Peng. 2018. Matrix factorization for recommendation with explicit and implicit feedback. *Knowledge-Based Systems* 158 (2018), 109–117.

[10] X. Chen, H. Chen, H. Xu, Y. Zhang, Y. Cao, Z. Qin, and H. Zha. 2019. Personalized fashion recommendation with visual explanations based on multimodal attention network: Towards visually explainable recommendation. In *SIGIR*. 765–774.

[11] X. Chen, Z. Wang, H. Xu, J. Zhang, Y. Zhang, X. Zhao, et al. 2022. Data Augmented Sequential Recommendation based on Counterfactual Thinking. In *TKDE*.

[12] X. Chen, H. Xu, Y. Zhang, J. Tang, Y. Cao, Z. Qin, and H. Zha. 2018. Sequential recommendation with user memory networks. In *WSDM*. 108–116.

[13] Xu Chen, Yongfeng Zhang, and Ji-Rong Wen. 2022. Measuring "Why" in Recommender Systems: a Comprehensive Survey on the Evaluation of Explainable Recommendation. *arXiv preprint arXiv:2202.06466* (2022).

[14] Gabriel de Souza Pereira Moreira, Sara Rabhi, Jeong Min Lee, Ronay Ak, and Even Oldridge. 2021. Transformers4Rec: Bridging the Gap between NLP and Sequential/Session-Based Recommendation. In *RecSys*. 143–153.

[15] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. 2017. Sequential user-based recurrent neural network recommendations. In *RecSys*. 152–160.

[16] Kai Epstude and Neal J Roese. 2008. The functional theory of counterfactual thinking. *Personality and social psychology review* 12, 2 (2008), 168–192.

[17] Amir Feder, Nadav Oved, Uri Shalit, and Roi Reichart. 2021. Causalm: Causal model explanation through counterfactual language models. *Comp. Ling.* (2021).

[18] T. Fu, X. Wang, M. Peterson, S. Grafton, M. Eckstein, and W. Wang. 2020. Counterfactual vision-and-language navigation via adversarial path sampler. In *ECCV*.

[19] Zuohui Fu, Yikun Xian, Ruoyuan Gao, Jieyu Zhao, Qiaoying Huang, et al. 2020. Fairness-aware explainable recommendation over knowledge graphs. In *SIGIR*.

[20] Yingqiang Ge, Shuchang Liu, Zuohui Fu, Juntao Tan, Zelong Li, Shuyuan Xu, Yunqi Li, Yikun Xian, and Yongfeng Zhang. 2022. A survey on trustworthy recommender systems. *arXiv:2207.12515* (2022).

[21] Yingqiang Ge, J. Tan, Y. Zhu, Y. Xia, J. Luo, S. Liu, Z. Fu, S. Geng, Z. Li, and Y. Zhang. 2022. Explainable Fairness in Recommendation. In *SIGIR*.

[22] Shijie Geng, Z. Fu, Y. Ge, L. Li, G. de Melo, and Y. Zhang. 2022. Improving Personalized Explanation Generation through Visualization. In *ACL*. 244–255.

[23] Shijie Geng, Z. Fu, J. Tan, Y. Ge, G. De Melo, and Y. Zhang. 2022. Path Language Modeling over Knowledge Graphsfor Explainable Recommendation. In *WWW*.

[24] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as Language Processing (RLP): A Unified Pretrain, Personalized Prompt & Predict Paradigm (P5). In *RecSys*.

[25] Azin Ghazimatin, Oana Balalau, Rishiraj Saha Roy, and Gerhard Weikum. 2020. Prince: Provider-side interpretability with counterfactual explanations in recommender systems. In *WSDM*. 196–204.

[26] Yash Goyal, Ziyan Wu, Jan Ernst, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Counterfactual visual explanations. In *ICML*. PMLR, 2376–2384.

[27] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *TIIS* 5, 4 (2015), 1–19.

[28] Ruining He and Julian McAuley. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *ICDM*. IEEE, 191–200.

[29] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *CIKM*. 843–852.

[30] Balázs Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. 2016. Session-based recommendations with recurrent neural networks. *ICLR* (2016).

[31] Wenyue Hua and Yongfeng Zhang. 2022. System 1 + System 2 = Better World: Neural-Symbolic Chain of Logic Reasoning. In *EMNLP*.

[32] Jin Huang, X. Zhao, H. Dou, J. Wen, and E. Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *SIGIR*.

[33] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*. IEEE, 197–206.

[34] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations* (2015).

[35] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *CIKM*. 1419–1428.

[36] Lei Li, Yongfeng Zhang, and Li Chen. 2020. Generate neural template explanations for recommendation. In *CIKM*. 755–764.

[37] Lei Li, Yongfeng Zhang, and Li Chen. 2021. Extra: Explanation ranking datasets for explainable recommendation. In *SIGIR*. 2463–2469.

[38] Lei Li, Yongfeng Zhang, and Li Chen. 2021. Personalized Transformer for Explainable Recommendation. In *ACL*. 4947–4957.

[39] Lei Li, Yongfeng Zhang, and Li Chen. 2022. Personalized prompt learning for explainable recommendation. *arXiv preprint arXiv:2202.07371* (2022).

[40] Yunqi Li, Hanxiong Chen, Shuyuan Xu, Yingqiang Ge, Juntao Tan, Shuchang Liu, and Yongfeng Zhang. 2022. Fairness in Recommendation: A Survey. In *arXiv*.

[41] Zelong Li, Jianchao Ji, and Yongfeng Zhang. 2022. From Kepler to Newton: Explainable AI for Science Discovery. In *ICML 2022 AI for Science*.

[42] Qiao Liu, Yifu Zeng, R. Mokhosi, and H. Zhang. 2018. STAMP: short-term attention/memory priority model for session-based recommendation. In *KDD*.

[43] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *EMNLP*.

[44] Massimo Quadrana, Alexandros Karatzoglou, et al. 2017. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *RecSys*.

[45] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW*.

[46] Shaoyun Shi, Hanxiong Chen, Weizhi Ma, Jiaxin Mao, Min Zhang, and Yongfeng Zhang. 2020. Neural logic reasoning. In *CIKM*. 1365–1374.

[47] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*. 1441–1450.

[48] Juntao Tan, Shijie Geng, Zuohui Fu, Yingqiang Ge, Shuyuan Xu, Yunqi Li, and Yongfeng Zhang. 2022. Learning and Evaluating Graph Neural Network Explanations based on Counterfactual and Factual Reasoning. In *WWW*. 1018–1027.

[49] Juntao Tan, Shuyuan Xu, Yingqiang Ge, Yunqi Li, Xu Chen, and Yongfeng Zhang. 2021. Counterfactual explainable recommendation. In *CIKM*. 1784–1793.

[50] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*. 565–573.

[51] Khanh Hiep Tran, Azin Ghazimatin, and Rishiraj Saha Roy. 2021. Counterfactual Explanations for Neural Recommenders. In *SIGIR*. 1627–1631.

[52] Nan Wang, Hongning Wang, Yiling Jia, and Yue Yin. 2018. Explainable recommendation via multi-task learning in opinionated text data. In *SIGIR*. 165–174.

[53] Zhenlei Wang, Jingsen Zhang, Hongteng Xu, Xu Chen, Yongfeng Zhang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Counterfactual data-augmented sequential recommendation. In *SIGIR*. 347–356.

[54] Qingsong Wen, Liang Sun, Fan Yang, Xiaomin Song, Jingkun Gao, et al. 2021. Time series data augmentation for deep learning: A survey. *IJCAI* (2021).

[55] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *WSDM*. 495–503.

[56] Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James Sharpnack. 2020. SSE-PT: Sequential recommendation via personalized transformer. In *RecSys*.

[57] Yikun Xian, Zuohui Fu, et al. 2020. CAFE: Coarse-to-fine neural symbolic reasoning for explainable recommendation. In *CIKM*.

[58] Yikun Xian, Z. Fu, S. Muthukrishnan, G. De Melo, and Y. Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *SIGIR*.

[59] Yikun Xian, Tong Zhao, Jin Li, Jim Chan, Andrey Kan, Jun Ma, Xin Luna Dong, Christos Faloutsos, George Karypis, Shan Muthukrishnan, and Yongfeng Zhang. 2021. Ex3: Explainable attribute-aware item-set recommendations. In *RecSys*.

[60] Kun Xiong, Wenwen Ye, X. Chen, Y. Zhang, X. Zhao, B. Hu, Z. Zhang, and J. Zhou. 2021. Counterfactual Review-based Recommendation. In *CIKM*.

[61] Shuyuan Xu, Yunqi Li, Shuchang Liu, Zuohui Fu, Yingqiang Ge, Xu Chen, et al. 2021. Learning causal explanations for recommendation. In *CSR*.

[62] Aobo Yang, Nan Wang, Renqin Cai, Hongbo Deng, and Hongning Wang. 2022. Comparative Explanations of Recommendations. In *WWW*. 3113–3123.

[63] Mengyue Yang, Q. Dai, Z. Dong, X. Chen, X. He, and J. Wang. 2021. Top-N Recommendation with Counterfactual User Preference Simulation. In *CIKM*.

[64] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A dynamic recurrent model for next basket recommendation. In *SIGIR*. 729–732.

[65] Shengyu Zhang, D. Yao, Z. Zhao, T. Chua, and F. Wu. 2021. Causerec: Counterfactual user sequence synthesis for sequential recommendation. In *SIGIR*.

[66] Wei Zhang, Junbing Yan, Z. Wang, and J. Wang. 2022. Neuro-Symbolic Interpretable Collaborative Filtering for Attribute-based Recommendation. In *WWW*.

[67] Yongfeng Zhang, Xu Chen, et al. 2020. Explainable recommendation: A survey and new perspectives. *Foundations and Trends® in Information Retrieval* (2020).

[68] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*. 83–92.

[69] Wayne Xin Zhao, Junhua Chen, Pengfei Wang, Qi Gu, and Ji-Rong Wen. 2020. Revisiting Alternative Experimental Settings for Evaluating Top-N Item Recommendation Algorithms. In *CIKM*. 2329–2332.

[70] Kun Zhou, Hui Wang, et al. 2020. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *CIKM*.

[71] Yaxin Zhu, Yikun Xian, Zuohui Fu, Gerard de Melo, and Yongfeng Zhang. 2021. Faithfully explainable recommendation via neural logic reasoning. In *NAACL*.