Counterfactual Data-Augmented Sequential Recommendation

Zhenlei Wang[1,2,#], Jingsen Zhang[1,3,#], Hongteng Xu[1,2], Xu Chen[1,2,*], Yongfeng Zhang[4], Wayne Xin Zhao[1,2], Ji-Rong Wen[1,2]
[1]Beijing Key Laboratory of Big Data Management and Analysis Methods
{[2]Gaoling School of Artificial Intelligence, [3]School of Information} Renmin University of China, Beijing 100872, China
[4]Department of Computer Science, Rutgers University
wang.zhenlei@foxmail.com, zhangjingsen@ruc.edu.cn, hongtengxu@ruc.edu.cn, successcx@gmail.com, zhangyf07@gmail.com, batmanfly@gmail.com, jrwen@ruc.edu.cn

ABSTRACT

Sequential recommendation aims at predicting users' preferences based on their historical behaviors. However, this recommendation strategy may not perform well in practice due to the sparsity of the real-world data. In this paper, we propose a novel counterfactual data augmentation framework to mitigate the impact of the imperfect training data and empower sequential recommendation models. Our framework is composed of a sampler model and an anchor model. The sampler model aims to generate new user behavior sequences based on the observed ones, while the anchor model is leveraged to provide the final recommendation list, which is trained based on both observed and generated sequences. We design the sampler model to answer the key counterfactual question: "what would a user like to buy if her previously purchased items had been different?". Beyond heuristic intervention methods, we leverage two learning-based methods to implement the sampler model, and thus, improve the quality of the generated sequences when training the anchor model. Additionally, we analyze the influence of the generated sequences on the anchor model in theory and achieve a trade-off between the information and the noise introduced by the generated sequences. Experiments on nine real-world datasets demonstrate our framework's effectiveness and generality.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Recommendation System,Counterfactual Data Augmentation

ACM Reference Format:
Zhenlei Wang[1,2,#], Jingsen Zhang[1,3,#], Hongteng Xu[1,2], Xu Chen[1,2,*], Yongfeng Zhang[4], and Wayne Xin Zhao[1,2], Ji-Rong Wen[1,2]. 2021. Counterfactual Data-Augmented Sequential Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3404835.3462855

∗ Corresponding author, # Co-first author.

SIGIR '21, July 11–15, 2021, Virtual Event, Canada.

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3462855

1 INTRODUCTION

In the past few years, many sequential recommendation models have been proposed and achieved encouraging performance on various application scenarios. These models leverage sophisticated mechanisms to predict users' preferences from their historical behaviors. For example, the representative FPMC model [24] captures the sequential user-item interactions by a two-order Markov chain. Recently, more cutting-edge models like recurrent neural networks (RNNs) [19, 21], convolutional neural networks (CNNs) [28, 31] and memory networks [4, 15] are applied to model the long-term dependencies among user behaviors. Essentially, the sequential recommendation models above capture the joint distribution of two or more items based on their occurrence in a sequence, which makes them require much more high-quality sequential data for training compared with the non-sequential models. Unfortunately, this demand conflicts with the *sparse* nature of the real-world data — N items have $N!$ permutations, but the sequences observed in practice can only cover extremely few of them. Figure 1 illustrates this problem for the E-commerce scenarios. We may observe that a user purchased a camera, a camera battery, and a battery charger over time. Generally, purchasing the camera is likely to trigger different subsequent items, such as a camera lens and lens cleaner, a roll film and a photo album, and so on. Although such potential sequences are intuitively reasonable and informative for training sequential recommendation models, they may not be recorded for various reasons (*e.g.*, the security and privacy issues and the limitations on time, human resource, and budgets). From the perspective of causal inference [2, 3, 6], these unrecorded sequences are *counterfactual* data, which provide answers to a key counterfactual question: **"What would a user like to buy if her previously purchased items had been different?"**. As a significant complementary resource of the observed sequences, the counterfactual sequences can more comprehensively reveal the user preference[1] and thus help training a sequential recommendation model with good precision and generalization power. Motivated by the analysis above, we propose a novel framework to achieve **C**ounterfactual Data-**A**ugmentation **S**equential **R**commendation (called **CASR** for short). The CASR framework enriches sequential data and improves

[1]As shown in Figure 1, when replacing the camera battery in the sequence with a camera lens, the sequence ends with a lens cleaner rather than the battery charger.

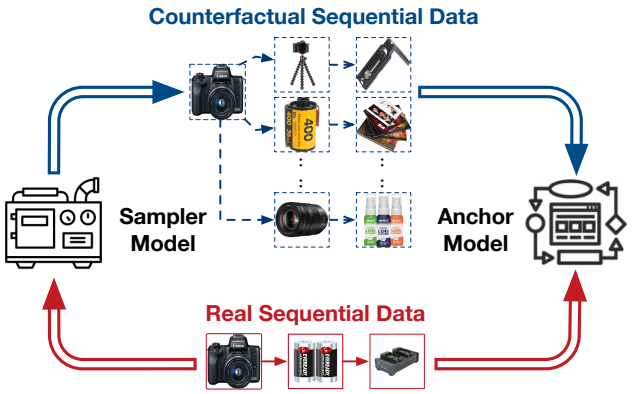

Figure 1: An illustration of our framework for counterfactual data-augmented sequential recommendation.

sequential recommendation models. As illustrated in Figure 1, the CASR framework is composed of a sampler model and an anchor model. Driven by the aforementioned counterfactual question, the sampler model generates counterfactual sequences from the real ones. The anchor model provides the final recommendation list and is trained based on both real and counterfactual sequences. We first implement a naive sampler model that generates new sequences by substituting the items of the real sequences randomly. Beyond this heuristic method, we further develop two types of learning-based sampler models called data- and model-oriented methods, respectively. The data-oriented method is able to generate new sequences which are near the decision boundaries, while the model-oriented method aims at maximizing the information of the new sequences provided to the anchor model. Based on the theory of probably approximately correct (PAC) learning [25], we analyze the relation between the number of generated sequences and the noise level of the sampler model. Accordingly, we design a simple but effective controlling mechanism for the proposed framework to achieve a trade-off between the noise and the information in the generated sequences. Our CASR framework is applicable to most existing recommendation systems, in which the sampler and anchor models can be implemented by various sequential architectures. The experiments on nine real-world datasets demonstrate the usefulness of our framework — from the results, we can observe consistent improvement comparing with the state-of-the-art sequential models.

In a summary, the main contributions of this paper can be concluded as follows: (1) We propose a novel framework to empower sequential recommendation models with counterfactual data. To the best of our knowledge, our work makes the first attempt to leverage the idea of counterfactual thinking in the recommendation domain. (2) We implement the above idea in three ways, developing a heuristic and two learning-based samplers to generate counterfactual sequences. (3) We theoretically analyze the noisy information in the counterfactual data and propose a simple but effective strategy to ensure the informativeness of the generated sequences. (4) We conduct extensive experiments based on nine real-world datasets to verify our framework's effectiveness and generality.

2 THE CASR FRAMEWORK

2.1 Sequential Recommendation

Suppose that we have a user set $\mathcal{U}$ and an item set $\mathcal{I}$. We denote $\mathcal{T}$ as the user-item interaction set[2]. Usually, $\mathcal{T}$ is organized as $\{(\{u_i, t_i^1, t_i^2, ..., t_i^{l_i}\}, t_i^{l_i+1})\}_{i=1}^N = \{T_i, t_i^{l_i+1}\}_{i=1}^N$, where each $u_i \in \mathcal{U}$ and $t_i^{k=1,...,l} \in \mathcal{T}$. In the i-th sample in $\mathcal{T}$, the items interacted with the user u_i in history is represented as a sequence $T_i = \{t_i^1, t_i^2, ..., t_i^{l_i}\}$ and $t_i^{l_i+1}$ is the next item. Given $\{\mathcal{U}, \mathcal{I}, \mathcal{T}\}$, sequential recommendation aims at modeling the impact of the historical items on the next one, and accordingly, predicting the users' future preferences. To achieve this aim, we often learn a probabilistic model, denoted as $\mathcal{A}$, by solving the following optimization problem:

$$\max_{\mathcal{A}} \prod_{(T_i, t_i^{l_i+1}) \in \mathcal{T} \cup \mathcal{T}^-} \log \mathcal{A}(t_i^{l_i+1} | T_i)^{y_i} (1 - \mathcal{A}(t_i^{l_i+1} | T_i))^{(1-y_i)} \quad (1)$$

where $\mathcal{A}$ outputs the probability of interacting with $t_i^{l_i+1}$ given the history information T_i. $\mathcal{T}^- = \{(\{u_s, t_s^1, t_s^2, ..., t_s^{l_s}\}, t_s^{l_s+1})\}_{s=1}^M$ is the set of negative samples, and $t_s^{l_s+1}$ is randomly selected from the items not interacted by u_s. The label $y_i = 1$ if $(T_i, t_i^{l_i+1}) \in \mathcal{T}$, otherwise $y_i = 0$. In the past few years, people have explored different models like CNN [28], RNN [19] or even transformer [17] to implement $\mathcal{A}$, but little attention has been paid to the sparsity of the training data, which may pose great challenges for learning reliable item correlations and achieving better recommendation performance. To mitigate this problem, we establish a novel framework called CASR for training sequential recommendation models.

Our CASR framework is inspired by the human introspection behaviors, such as "what if I took another road?" and "what if I bought a lens after a camera?". Such behaviors are described by the concept of counterfactual thinking in the causal inference domain [1, 6], which helps to explore (or imagine) the potential results of the alternative previous actions. The key counterfactual question for sequential recommendation is "what would a user buy if her previously purchased items had been different?". As shown in Figure 1, beyond training a sequential recommendation model $\mathcal{A}$ (*i.e.*, the anchor model), our CASR framework introduces a sampler model, denoted as $\mathcal{S}$, to answer this counterfactual question by generating counterfactual sequences. In our framework, both $\mathcal{S}$ and $\mathcal{A}$ are pre-trained based on the original dataset in the beginning. Then, the counterfactual sequences produced from $\mathcal{S}$ are leveraged to re-optimize $\mathcal{A}$, which is finally leveraged to provide the recommendation list. Clearly, the key of our framework is the design of the sampler, which has a significant influence on the quality of the generated sequences. We implement the sampler based on both heuristic and learning-based methods, which will be introduced in the following contents.

2.2 Heuristic Sampler

The most straightforward way to generate counterfactual sequences is replacing the historical items with the other ones in a random manner. For a training sample $(\{u, t^1, t^2, ..., t^l\}, t^{l+1}) \in \mathcal{T}$, we first indicate an index d, and replace t^d with a random item t^a. Then,

[2]In practice, interact can be click, purchase, add to shopping cart and $etc.$.

we bring $\{u, t^1, ..., t^{d-1}, t^a, t^{d+1}, ..., t^l\}$ into $\mathcal{S}$ to derive the next item $\hat{t}^{l+1}$, that is,

$$\hat{t}^{l+1} = \arg\max_{t \in \mathcal{I}} \mathcal{S}(t|u, t^1, ..., t^{d-1}, t^a, t^{d+1}, ..., t^l) \quad (2)$$

At last, $(\{u, t^1, ..., t^{d-1}, t^a, t^{d+1}, ..., t^l\}, \hat{t}^{l+1})$ is the generated counterfactual sequence. This heuristic sampler model is easy to implement. However, it has too many degree of freedom, and thus, introduce too much randomness into the generated sequences. It has been studied in the previous work [6, 7, 29] that different samples are not equally important for model optimization. To guarantee the effectiveness of the generated sequences on the anchor model's optimization, we need to design some samplers with more reasonable mechanisms.

2.3 Learning-based Samplers

In order to make the generated sequences more effective, in this section, we improve the heuristic sampler by learning-based methods from the data and model perspectives, respectively.

Data-oriented counterfactual sequence learning. In a classification problem, the input feature space can be split into many subspaces according to different output labels. The borders between different input subspaces are referred as the decision boundaries. For the samples near the decision boundaries, the labels can be easily altered even with a small change on the input features. As demonstrated in the previous work [1, 9], these decision boundary samples are usually discriminative in revealing the underlying data patterns, and training based on them may lead to improved model performance. Our data-oriented sampler is motivated by this principle, and we generate the counterfactual sequence by "minimally" changing the user's historical items, such that her currently interacted item can be "exactly" altered. Formally, suppose $\boldsymbol{e}_t \in \mathbb{R}^D$ is the embedding of item t, we measure the change of the user behaviors based on the embedding space. For a given real sequence $(\{u, t^1, t^2, ..., t^l\}, t^{l+1}) \in \mathcal{T}$ and an index d, we optimize the following objective:

$$\begin{aligned} &\min_{t^a \in C} \|\boldsymbol{e}_{t^a} - \boldsymbol{e}_{t^d}\|_2^2 \\ &\text{s.t. } t^{l+1} \neq \arg\max_{t \in \mathcal{I}} \mathcal{S}(t|u, t^1, ..., t^{d-1}, t^a, t^{d+1}, ..., t^l) \end{aligned} \quad (3)$$

where $\boldsymbol{e}_{t^a}$ and $\boldsymbol{e}_{t^d}$ are the embeddings of the changed and original items. C indicates the item set for replacement, which can be specified as $\mathcal{I}$ or other set to involve prior knowledge. In this equation, the objective aims to minimize the distance between the item embeddings before and after the change (i.e., minimally changing the user's past behavior). The constraint ensures that the current item is no longer t^{l+1}. By combining them, we would like to change the past behaviors in a minimum manner, such that the current item can be exactly altered. Once we obtained t^a, the final counterfactual sequence is $(\{u, t^1, ..., t^{d-1}, t^a, t^{d+1}, ..., t^l\}, \hat{t}^{l+1})$, where $\hat{t}^{l+1}$ is derived according to equation (2). By such "minimum-exact" optimization, in the generated counterfactual sequence, some small change on t^a will make the distance $\|\boldsymbol{e}_{t^a} - \boldsymbol{e}_{t^d}\|_2^2$ not large enough to alter t^{l+1}, i.e., the current item $\hat{t}^{l+1}$ will return to the original item t^{l+1}, which implies that the counterfactual sequences are near the decision boundaries. An intuitive illustration of this method can be seen in Figure 2.

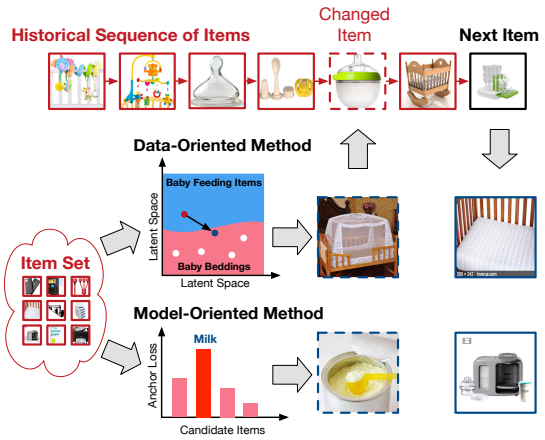

Figure 2: Illustration of the learning-based samplers. In the data-oriented method, the "feeding bottle" is replaced by a "baby bed", which exactly change the next item from baby feeding items to baby beddings. In the model-oriented method, the "milk" can induce a counterfactual sequence with the largest anchor model loss, which is selected as the replacement item.

In practice, the major challenge comes from the non-differentiable nature of objective (3). To solve this problem, we introduce a "virtual" item $t^{\widetilde{a}}$, and its embedding is $\boldsymbol{e}_{t^{\widetilde{a}}} = \boldsymbol{e}_{t^d} + \Delta$, where Δ is a continuous vector indicating the distance between $t^{\widetilde{a}}$ and t^d. Our general idea is firstly learning $\boldsymbol{e}_{t^{\widetilde{a}}}$ in a differentiable manner, and then projecting $\boldsymbol{e}_{t^{\widetilde{a}}}$ to the nearest real item. In order to learn $\boldsymbol{e}_{t^{\widetilde{a}}}$, we relax objective (3) to the following differentiable target:

$$\min_{\Delta \in \mathbb{R}^D} \|\Delta\|_2^2 + \alpha \mathcal{S}(\boldsymbol{e}_{t^{l+1}}|e_u, \boldsymbol{e}_{t^1}, .., \boldsymbol{e}_{t^{d-1}}, \boldsymbol{e}_{t^d} + \Delta, \boldsymbol{e}_{t^{d+1}}, .., \boldsymbol{e}_{t^l}) \quad (4)$$

where we replace the item IDs in equation (3) with their embeddings for clear presentation, and Δ is the only learnable parameter. The first term aims to minimize the distance between the virtual and real items, that is, minimizing the change of the user behaviors. The second term penalizes the larger probability of interacting with the true current item $\boldsymbol{e}_{t^{l+1}}$, in other words, $\boldsymbol{e}_{t^{l+1}}$ is not expected to be predicted by $\mathcal{S}$. Thus, this term plays a similar role as the constraint in (3). α is a tuning parameter balancing different objectives. Once we obtained $\boldsymbol{e}_{t^{\widetilde{a}}}$, the replacement item t^a is derived according to the following projection method:

$$t^a = \arg\min_{t^a \in C} \|\boldsymbol{e}_{t^{\widetilde{a}}} - \boldsymbol{e}_{t^a}\|_2^2, \quad (5)$$

Model-oriented counterfactual sequence learning. Besides the two samplers described above, the third potential implementation is a model-oriented method. This method is motivated by the work in [6, 7], which leverages the principle that the samples with larger loss can usually provide more knowledge to widen the model's experience and improve the model performance. In our framework, we learn the counterfactual sequences via maximizing the loss of the anchor model, and an intuitive illustration of this method is provided in Figure 2. Formally, given a training sequence $(\{u, t^1, t^2, ..., t^l\}, t^{l+1}) \in \mathcal{T}$ and an index d, we solve the following

optimization problem:

$$\begin{aligned} &\max_{t^a \in C} -\log \mathcal{A}(\hat{t}^{l+1}|u, t^1, ..., t^{d-1}, t^a, t^{d+1}, ..., t^l) \\ &\text{s.t. } \hat{t}^{l+1} = \arg\max_{t \in \mathcal{I}} \mathcal{S}(t|u, t^1, ..., t^{d-1}, t^a, t^{d+1}, ..., t^l) \\ &\|\boldsymbol{e}^{\mathcal{S}}_{t^a} - \boldsymbol{e}^{\mathcal{S}}_{t^d}\|_2^2 \leq \lambda \end{aligned} \quad (6)$$

where the objective is to maximize the loss of the anchor model[3], the first constraint states that the sequence is generated based on $\mathcal{S}$. The second constraint controls the distance between the revised and original item embeddings. Smaller λ means that $\boldsymbol{e}^{\mathcal{S}}_{t^a}$ should be selected near the original item embedding, while larger λ allows $\boldsymbol{e}^{\mathcal{S}}_{t^a}$ to be explored in a broader space.

To make this objective more tractable, we adopt a similar method as used in the data-oriented method. We introduce a "virtual" item $t^{\widetilde{a}}$, whose embedding is Δ-away from that of t^d. We denote $\boldsymbol{E}^{\mathcal{F}}_u = \{\boldsymbol{e}_u, \boldsymbol{e}^{\mathcal{F}}_{t^1}, ..., \boldsymbol{e}^{\mathcal{F}}_{t^{d-1}}, \boldsymbol{e}^{\mathcal{F}}_{t^d} + \Delta, \boldsymbol{e}^{\mathcal{F}}_{t^{d+1}}, ..., \boldsymbol{e}^{\mathcal{F}}_{t^l}\}$ as the collection of the user and revised item sequence embeddings, where $\mathcal{F}$ is either $\mathcal{A}$ or $\mathcal{S}$. We soften the selection process of $\hat{t}^{l+1}$ to make the objective fully differentiable, and the relaxed objective is:

$$\max_{\Delta \in \mathbb{R}^D} -\sum_{i=1}^{|\mathcal{I}|} \frac{\exp(\tau \mathcal{S}(\boldsymbol{e}^{\mathcal{S}}_i | \boldsymbol{E}^{\mathcal{S}}_u))}{\sum_{j=1}^{|\mathcal{I}|} \exp(\tau \mathcal{S}(\boldsymbol{e}^{\mathcal{S}}_j | \boldsymbol{E}^{\mathcal{S}}_u))} \log \mathcal{A}(\boldsymbol{e}^{\mathcal{A}}_i | \boldsymbol{E}^{\mathcal{A}}_u) - \beta ||\Delta||_2^2 \quad (7)$$

where β is a tunning parameter controlling the importance of the distance constraint. $-\log \mathcal{A}(\boldsymbol{e}^{\mathcal{A}}_i | \boldsymbol{E}^{\mathcal{A}}_u)$ is the loss on the sample when i is the current item. $\frac{\exp(\tau \mathcal{S}(\boldsymbol{e}^{\mathcal{S}}_i | \boldsymbol{E}^{\mathcal{S}}_u))}{\sum_{j=1}^{|\mathcal{I}|} \exp(\tau \mathcal{S}(\boldsymbol{e}^{\mathcal{S}}_j | \boldsymbol{E}^{\mathcal{S}}_u))}$ imposes a softmax layer upon $\mathcal{S}$, indicating the probability of interacting with item i. By multiplying them, the first term of equation (7) is actually computing the expectation of the anchor model loss supported by $\mathcal{I}$. τ is the temperature parameter, which tunes the softness of the item distribution. When $\tau \to 0$, all the items are equally considered in the expectation. When $\tau \to \infty$, the mass of the distribution is concentrated to a single item, and the expectation is reduced to the arg max operation as used in the original objective (6).

Remark. Basically, data-oriented method learns the counterfactual sequences based on the underlying patterns of the training data. This is a *"bottom-up"* method, which is model agnostic, i.e., do not rely on the specific anchor model. In contrast, model-oriented method learns the counterfactual sequences under the guidance of the anchor model, which is a *"top-down"* method, and highly depends on the model it serves for. These methods are designed following different principles, and may play different roles in practice.

2.4 Learning Algorithm

We summarize the complete learning algorithm of our framework in Algorithm 1. To begin with, the sampler model $\mathcal{S}$ and anchor model $\mathcal{A}$ are both optimized based on the observed dataset $\mathcal{T}$. After indicating the number of running times M and the index d, in each iteration, the sampler model is executed on the observed sequence to generate the counterfactual sequence. Once we have collected the whole counterfactual dataset $\mathcal{T}_c$, the anchor model is retrained based on $\mathcal{T} \cup \mathcal{T}_c$. There are three implementations of the sampler

[3]This objective is derived by bringing $y_i = 1$ into equation (1).

Algorithm 1: Learning Algorithm of CASR

```
Pre-train S and A based on the observed dataset T.
Initialize the counterfactual dataset T_c = ∅.
Indicate the number of running times M.
Indicate the index d.
for i in [0, M] do
    Select a training sample ({u, t^1, t^2, ..., t^l}, t^{l+1}).
    Heuristic Method:
    Randomly select an item t^a.
    Change the input sequence as
      {t^1, ..., t^{d-1}, t^a, t^{d+1}, ..., t^l}
    Derive t̂^{l+1} by equation (2).
    T_c ← T_c ∪ ({u, t^1, ..., t^{d-1}, t^a, t^{d+1}, ..., t^l}, t̂^{l+1}).
    Data-oriented Method:
    Learn the distance vector Δ by equation (4).
    Derive t^a by equation (5).
    Derive t̂^{l+1} by equation (2).
    if t̂^{l+1} ≠ t^{l+1} then
        T_c ← T_c ∪ ({u, t^1, ..., t^{d-1}, t^a, t^{d+1}, ..., t^l}, t̂^{l+1}).
    end
    Model-oriented Method:
    Learn the distance vector Δ by equation (7).
    Derive t̂^{l+1} by bring t^{ã} into equation (2).
    T_c ← T_c ∪ ({u, t^1, ..., t^{d-1}, t^{ã}, t^{d+1}, ..., t^l}, t̂^{l+1}).
end
Train A based on T ∪ T_c.
```

model to generate the counterfactual sequences. In the heuristic sampler, the replacement item t^a is determined in a random manner without any constraint. The next item, and accordingly, the whole counterfactual sequence are derived based on equation (2). In the learning-based samplers, the replacement items are selected by learning to achieve some reasonable targets. More specifically, in the data-oriented method, a virtual item embedding is firstly learned to achieve the decision boundary sample in a differentiable manner based on equation (4). Then the virtual item is projected to the nearest real replacement item based on equation (5). Similar to the heuristic sampler, the next item is derived based on equation (2), but to ensure that it is different from the original item, there is a further check on whether $\hat{t}^{l+1} \neq t^{l+1}$. In the model-oriented method, the virtual item embedding is learned based on equation (7), which is directly used as the replacement item without approximation, and its embedding is $\boldsymbol{e}^{\mathcal{A}}_{t^d} + \Delta$. Careful readers may find that, this method is basically playing an adversarial game. The critical sequences are generated to maximize the anchor model loss, which is later minimized to learn the parameters in $\mathcal{A}$. In this way, the generated sequences can well challenge the anchor model, which broadens the views of the anchor model and helps to achieve better performance.

2.5 Theoretical Analysis

In our framework, the current interacted item is estimated based on equation (2). However, the sampler model $\mathcal{S}$ may be not perfect, and its predicted results may contain noise. In this section, we

theoretically analyze our method within the PAC learning framework [25]. We would like to answer: "given the noise level of the sampler model, how many samples one need to achieve sufficiently well performance?". Suppose $g \in \{0, 1\}$ is the true result of whether a product is the real current item, and $\mathcal{S}$ can correctly estimate g with the probability of $1 - \eta$, where $\eta \in (0, 0.5)$ indicates the noise level of $\mathcal{S}$. If $\mathcal{S}$ can exactly recover g (i.e., $\eta = 0$), then the generated sequences are perfect without any noise. On the contrary, $\eta = 0.5$ means that $\mathcal{S}$ can only produce random results, and the generated sequences are fully noisy. Then we have the following theory:

Theorem 1. *Given a hypothesis class $\mathcal{H}$, for any $\epsilon, \delta \in (0, 1)$ and $\eta \in (0, 0.5)$, if $\mathcal{A} \in \mathcal{H}$ is the anchor model learned based on the empirical risk minimization (ERM), and the sample complexity (i.e., number of samples) is larger than $\frac{2\log(\frac{2|\mathcal{H}|}{\delta})}{\epsilon^2(1-2\eta)^2}$, then the error between the model estimated and true results is smaller than ϵ with probability larger than $1 - \delta$.*

Proof. Suppose the prediction error of $\mathcal{A}$ is s (i.e., $\sum \mathbb{I}(g_{\mathcal{A}} \neq g) = s$),[4] then the mis-matching probability between the observed and predicted results comes from two parts: (1) The observed result is true, but the prediction is wrong, that is, $s(1-\eta)$. (2) The observed result is wrong, but the prediction is right, that is $(1 - s)\eta$. Thus, the total mis-matching probability is $\eta + s(1 - 2\eta)$.

The following proof is based on the reduction to absurdity. We firstly propose an assumption, and then derive contradicts to invalidate the assumption.

• *Assumption.* Suppose the prediction error of $\mathcal{A}$ (i.e., s) is larger than ϵ, Then, at least one of the following statements hold: (1) The empirical mis-matching rate of $\mathcal{A}$ is smaller than $\eta + \frac{\epsilon(1-2\eta)}{2}$. (2) The empirical mis-matching rate of the optimal $h^* \in \mathcal{H}$ (i.e., the prediction error of h^* is 0) is larger than $\eta + \frac{\epsilon(1-2\eta)}{2}$. These statements are easy to understand, since if both of them do not hold, we can conclude that the empirical loss of $\mathcal{A}$ is larger than that of h^*, which does not agree with the ERM definition.

• *Contradicts.* To begin with, we review the uniform convergence properties [25] by the following lemma:

Lemma 2.1. *Let $\mathcal{H}$ be a hypothesis class, then for any $\epsilon \in (0, 1)$ and $h \in \mathcal{H}$, if the number of training samples is m, the following formula holds:*

$$\mathbb{P}(|R(h) - \hat{R}(h)| > \epsilon) < 2|\mathcal{H}| \exp(-2m\epsilon^2) \tag{8}$$

where R and $\hat{R}$ are the expectation and empirical losses, respectively.

For statement (1), since the prediction error of $\mathcal{A}$ is larger than ϵ, the expectation loss $R(\mathcal{A})$ is larger than $\eta+\epsilon(1-2\eta)$. If the empirical loss $\hat{R}(\mathcal{A})$ is smaller than $\eta + \frac{\epsilon(1-2\eta)}{2}$, then $|R(\mathcal{A}) - \hat{R}(\mathcal{A})|$ should be larger than $\frac{\epsilon(1-2\eta)}{2}$. At the same time, according to Lemma 2.1, when the sample number m is larger than $\frac{2\log(\frac{2|\mathcal{H}|}{\delta})}{\epsilon^2(1-2\eta)^2}$, we have $\mathbb{P}\left(|R(\mathcal{A}) - \hat{R}(\mathcal{A})| > \frac{\epsilon(1-2\eta)}{2}\right) < \delta$.

For statement (2), the expectation loss of h^* is η, i.e., $R(h^*) = \eta$. If the empirical loss $\hat{R}(h^*)$ is larger than $\eta + \frac{\epsilon(1-2\eta)}{2}$, then $|R(h^*) - \hat{R}(h^*)|$ should be larger than $\frac{\epsilon(1-2\eta)}{2}$. According to Lemma 2.1, when the sample number m is larger than $\frac{2\log(\frac{2|\mathcal{H}|}{\delta})}{\epsilon^2(1-2\eta)^2}$, we have $\mathbb{P}\left(|R(h^*) - \hat{R}(h^*)| > \frac{\epsilon(1-2\eta)}{2}\right) < \delta$.

As a result, both of the above statements hold with the probability smaller than δ, which implies that the prediction error of $\mathcal{A}$ is smaller than ϵ with the probability larger than $1 - \delta$. □

From this theory, we can see: in order to guarantee some performance with a given probability (i.e., ϵ and δ are fixed), one needs to generate more than $\frac{2\log(\frac{2|\mathcal{H}|}{\delta})}{\epsilon^2(1-2\eta)^2}$ sequences. If the noise level of $\mathcal{S}$ (i.e., η) is larger, than more sequences have to be generated. Extremely, if $\mathcal{S}$ can only produce noisy information (i.e., $\eta = 0.5$), then infinity number of samples are required, which is impossible in realities. This theory reveals the relation between the number of generated sequences and the potential noisy information contained in them, which helps to understand our framework in theory.

Tuning the noisy information. Inspired by this theory, we introduce a confidence parameter $\kappa \in [0, 1)$ to control the noisy information. In specific, the new sequences are generated according to equation (2) only when $\max_{t \in \mathcal{I}} \mathcal{S}(t|u, t^1, ..., t^{d-1}, t^a, t^{d+1}, ..., t^l) > \kappa$, that is, the sampler model has more confidence on the predicted results. In such a method, when κ is larger, there can be less noisy information. But at the same time, the number of counterfactual sequences is smaller, which may impact the effectiveness of our framework for improving the recommendation performance. If we select a smaller κ, more sequences will be generated for sufficiently training the anchor model, but the noise rate can also be increased. Thus, κ controls the trade-off between the number and reliability of the generated sequences. While such noise control method is simple, it can achieve promising results in the experiments, and we leave more advanced methods as the future work.

3 RELATED WORK

Relation with sequential recommendation. Sequential recommendation has recently attracted increasing attention from the research community. It basically aims to discover the underlying patterns of the user sequential behaviors. In this field, people have designed quite a lot of models based on Markov Chain [11, 24], RNN [14, 19, 21, 31], CNN [28], Memory Network [4, 15, 22] and Transformer [26, 30]. These studies mostly focus on designing advanced architectures to model the observed data. However, as mentioned before, the recorded user sequential behaviors in a recommendation dataset can be extremely insufficient. It is well known that a machine learning problem includes two inseparable aspects, i.e., "model" and "data". Our framework aims to improve sequential recommendation from the data perspective, which is orthogonal to the prevailing model-based research in this field.

Relation with counterfactual thinking. Counterfactual thinking is a concept describing the human introspection behaviors. The typical question one may ask is: "what would ... if ...?". In the machine learning community, many people leverage counterfactual thinking to design more explainable, robust and fair models [8, 9, 18, 32], which has achieved many promising results. Another important application of counterfactual thinking is augmenting the training samples for data-scarce tasks. Along this research line, many successes have been witnessed in the fields of neural

[4] $g_{\mathcal{A}}$ is the prediction from $\mathcal{A}$.

Table 1: Statistics of the datasets used in our experiments.

Dataset	# User	# Item	# Interaction	Density	Domain
MovieLens	6,040	3,629	836,478	3.82%	Movies
Lastfm	1,860	2,824	71,355	1.36%	Music
Book-Crossing	22,817	319,199	1,028,948	0.02%	Books
Foursquare	2,288	7,056	128,530	0.80%	Check-in
Jester	73,422	101	4,136,360	55.78%	Jokes
Diginetica	33,364	43,589	223,562	0.02%	E-commerce
Amazon-Video	5,131	1,686	37,126	0.43%	E-commerce
Amazon-Baby	19,446	7,051	160,792	0.12%	E-commerce
Amazon-Beauty	22,364	12,102	198,502	0.07%	E-commerce

language processing (NLP) [35] and computer vision (CV) [2, 3, 6]. Our work falls into this category, and we realize the idea of counterfactual thinking in the field of sequential recommendation. More importantly, we provide theoretical analysis on the relation between the number of generated sequences and the potential noisy information contained in them.

Relation with adversarial training. Adversarial training [5] is a promising learning paradigm, which has shed lights on a number of machine learning applications, such as image generation [16, 33], language generation [20] and robust recommender system [12]. The basic idea of adversarial training is to introduce an opponent into the model optimization process. The opponent aims to set "difficulties" for the original model, which is optimized by learning from these "difficulties". In this work, we borrow the idea of adversarial training to generate counterfactual sequences. The adversarial game is played between the sampler and anchor models. With the purpose of challenging the anchor model, the sequences generated from the sampler model are effective for optimizing the anchor model.

4 EXPERIMENTS

4.1 Experiment Setup

Datasets. In order to verify the effectiveness and generality of our framework. We base the experiments on nine public available datasets, which cover six various recommendation domains. In specific, *MovieLens 1M*[5] is a dataset about user behaviors on watching movies. *Last.fm*[6] records user habits when listening to the music. *Book-crossing*[7] is a dataset about user preferences on the books. *Foursquare*[8] contains user check-in behaviors in NYC and Tokyo. *Jester*[9] is a dataset including user ratings on the jokes. *Diginetica*[10], *Amazon-Instant-video*, *Amazon-Beauty* and *Amazon-Baby*[11] are e-commerce datasets, each of which is composed of user purchasing behaviors on the websites. The statistics of these datasets are summarized in Table 1. Based on these quite different recommendation domains, we would like to examine whether our framework is generally effective for different user behavior patterns.

Baselines. We compare our model with the following representative models[12]: **BPR** [23] is a well know recommendation algorithm for capturing user implicit feedback. **NAIS** [13] is a popular attention-based neural recommender model. **FPMC** [24] is an early sequential recommender model, where the behavior correlations are captured by Markov chain. **NARM** [19] is a sequential recommendation model based on attentive recurrent neural network. **STAMP** [21] is a neural sequential model by incorporating user short-term memories and preferences. **SASRec** [17] is a sequential recommendation model based on the self-attention mechanism.

Implementation details. In the experiments, each user's interacted items are chronologically organized into a session, which is separated into many training samples recurrently. For example, if the original session is $ABCD$, then the generated samples are $A \rightarrow B$, $AB \rightarrow C$ and $ABC \rightarrow D$. Following the common practice, for each user, the last and second last samples are used for model testing and validation, while the others are left for training. Our framework is executed on each sample in the training set. We evaluate different recommendation models based on the well-known metrics including F1 and NDCG, and we recommend 5 items to compare with the ground truth. The hyper-parameters are determined based on grid search. In specific, the learning rate and batch size are tuned in the ranges of $[10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}]$ and [64, 128, 256, 512, 1024], respectively. The regularization parameters α and β are both searched in the range of $\{10^i | \text{i is an integer in [-5, 5]}\}$. The influence of different confidence parameter κ's and index d's are discussed in the following experiments. For the baselines, we set the parameters as the optimal values reported in the original paper or tune them in the same ranges as our model's.

4.2 Overall Comparison

The overall comparison results are presented in Table 2, from which we can see: among the baselines, NAIS and FPMC outperform BPR in most cases. The winner between NAIS and FPMC varies across different datasets. NAIS is able to capture the potential non-linear feature correlations, while FPMC is better at modeling user sequential behavior patterns. We speculate that such different advantages make them suitable for different recommendation scenarios, which vary their performances on different datasets. By combining the merits of neural network and the idea of sequential behavior modeling, NARM, SASRec and STAMP exhibit better performance comparing with the other baselines, which is aligned with the previous studies [17, 19, 21].

It is encouraging to see that the best performance of our framework is better than all the baselines, and the improvement is consistent on all the datasets and evaluation metrics. Considering that the datasets span a large range of recommendation domains, this result demonstrates the generality of our framework. Among different data enrichment strategies, the heuristic method is the worst, for the Diginetica dataset, it even lowers the performance of SASRec. This observation manifests that while the random strategy is straightforward, it is usually suboptimal. The reason can be that the counterfactual sequence space is very large, the randomly selected sequences can be not critical for optimizing the anchor model, which lead to the unsatisfied performance. Between the data- and model-oriented methods, the latter can achieve better performance in most cases. This is actually not surprising, since the sequences generated from the model-oriented method are more targeted, which is tailored for improving the anchor model. However, it should be noted that the

[5] https://grouplens.org/datasets/movielens/
[6] http://millionsongdataset.com/lastfm/
[7] http://www2.informatik.uni-freiburg.de/ cziegler/BX/
[8] https://sites.google.com/site/yangdingqi/home/foursquare-dataset
[9] http://eigentaste.berkeley.edu/dataset/
[10] https://competitions.codalab.org/competitions/11161
[11] http://jmcauley.ucsd.edu/data/amazon/
[12] which is implemented in Bole1.0 [34] https://recbole.io/

Table 2: Overall comparison between the baselines and our models. In the second module (line 6-9), we present the performance of the original NARM and the results of applying our framework on NARM. The third and forth modules show the effectiveness of our framework for SASRec and STAMP, respectively. In each module, we use bold fonts to label the best performance.

Datasets	MovieLens		Diginetica		Book-Crossing		Video		Baby		Beauty		Foursquare		Lastfm		Jester	
Metric (@5)	F1	NDCG	F1	NDCG	F1	NDCG	F1	NDCG	F1	NDCG	F1	NDCG	F1	NDCG	F1	NDCG	F1	NDCG
BPR	1.47	2.73	3.33	7.11	0.35	0.66	4.33	9.62	0.60	1.17	1.17	2.33	2.13	4.37	0.35	0.67	7.98	22.14
NAIS	1.43	2.68	3.92	8.12	1.02	1.70	4.17	9.13	0.52	1.03	1.35	2.57	2.88	6.88	0.42	0.80	9.72	24.65
FPMC	2.67	4.67	6.67	13.35	0.43	0.93	3.83	7.59	0.63	1.20	0.98	1.77	2.40	2.40	2.05	4.19	8.77	17.09
NARM	4.93	9.45	7.15	14.45	0.50	1.12	4.50	9.53	0.90	1.69	1.33	2.57	3.55	7.27	2.75	5.53	7.25	14.69
H-NARM	5.07	9.46	7.72	15.35	1.07	2.44	5.48	10.84	0.95	1.79	1.67	3.19	**4.33**	8.52	4.97	9.65	11.68	27.01
D-NARM	5.18	9.59	**8.32**	16.71	1.13	2.68	**5.90**	11.85	1.12	2.07	1.73	3.29	**4.33**	8.62	5.05	9.82	11.87	27.27
M-NARM	**5.27**	**9.67**	**8.32**	**16.72**	**1.22**	**2.89**	5.88	**11.95**	**1.22**	**2.27**	**1.75**	**3.36**	**4.33**	**8.79**	**5.10**	**9.99**	**11.98**	**27.47**
SASRec	4.73	8.77	8.15	16.68	0.92	1.83	4.95	8.48	0.85	1.57	1.32	2.44	3.97	7.43	2.65	5.11	7.43	15.08
H-SASRec	4.75	8.77	6.65 ↓	13.64 ↓	1.03	1.93	5.52	9.75	0.98	1.89	1.58	3.04	4.55	9.04	4.75	9.18	11.13	25.82
D-SASRec	4.93	9.05	8.77	17.27	1.27	2.32	5.98	10.41	0.98	1.89	1.62	3.12	**4.75**	9.17	4.90	9.37	11.58	26.82
M-SASRec	**4.95**	**9.07**	**8.82**	**17.34**	**1.32**	**2.51**	**6.07**	**10.54**	**1.02**	**1.92**	**1.67**	**3.15**	4.63	**9.33**	**4.93**	**9.50**	**11.92**	**27.41**
STAMP	4.57	9.16	7.18	14.71	0.80	1.87	4.55	9.41	0.77	1.40	1.30	2.67	3.38	6.74	2.77	5.48	7.45	14.98
H-STAMP	4.75	9.35	7.83	15.87	1.15	2.61	5.47	10.15	0.83	1.65	1.72	3.28	3.85	8.09	4.30	9.12	10.47	25.09
D-STAMP	4.82	9.45	7.90	16.02	1.23	2.80	5.75	11.73	0.88	1.71	**1.75**	3.35	3.92	8.20	**4.35**	9.26	10.58	25.48
M-STAMP	**4.92**	**9.58**	**7.95**	**16.14**	**1.25**	**2.94**	**5.83**	**11.97**	**0.92**	**1.76**	1.70	**3.45**	**4.00**	**8.34**	**4.35**	**9.45**	**11.00**	**26.05**

[1] All the numbers are percentage values with "%" omitted, and we use ↓ to label the lowered performances after using our framework.
[2] We use "H-X", "D-X" and "M-X" to represent the heuristic, data- and model-oriented methods when the anchor model is "X".
[3] The prefix of Amazon-Video, Amazon-Baby and Amazon-Beauty are removed for simplicity.

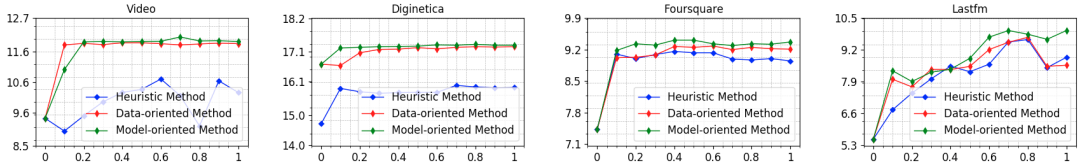

Figure 3: Influence of the number of generated sequences.

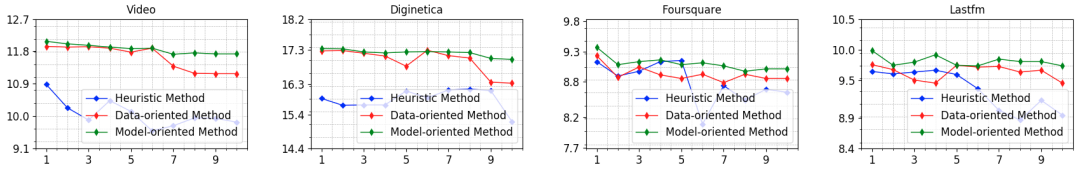

Figure 4: Influence of the iteration number.

better performance of the model-oriented method is achieved by using the information of the anchor model. If we do not know the anchor model in advance, this strategy can be unavailable. While the performance of the data-oriented method is slightly worse, it is model agnostic, and can be leveraged in a pre-training fashion, which can simultaneously serve for different anchor models.

4.3 The Number of Generated Sequences

In this section, we investigate the influence of the number of generated sequences. Suppose the size of the training set is N, we control the number of generated sequences by running our framework on r% of the complete training set. We tune r in [20, 40, 60, 80, 100], and the intervened sequences are randomly selected. The parameters are set as their optimal values tuned in the above experiments. Due to the space limitation, we present the performance of NDCG@5 on the datasets of *Video*, *Diginetica*, *Foursquare* and *Lastfm*, respectively. The results on the other datasets and F1@5 are similar and omitted. For each dataset, the sampler and anchor models are selected when their combinations can achieve the best performance according to Table 2 and 3.

From Figure 3, we can see: in general, the performances of different data enrichment strategies continue to rise until reaching some optimal values, and then the results remain stable or go down as r becomes larger. In order to achieve similar performance, heuristic method may need to involve more training sequences. This observation is interesting, we speculate that the heuristic method can be too arbitrary, which makes the sample utilization efficiency not high. In contrast, the data- and model-oriented methods can effectively discover the key information and generate critical sequences, which results in similar performance even with fewer training samples.

Careful readers may be curious about that if we further run our framework on the generated counterfactual sequences, then how would the produced data influence the final performance? Actually, such process can be iteratively proceeded[13], and the number of iteration times is defined as L. To answer the above question, we tune L in the range of [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], and the results are presented in Figure 4. We can see: in most cases, the performance

[13]It should be noted that the framework is only run on the most recently generated samples, since running on the former data can lead to exactly same samples which have already been generated.

Table 3: Influence of different sampler models. In the first column, we present the data enrichment strategies, where H, D and M correspond the heuristic, data- and model-oriented methods, respectively. The second column indicates the sampler model (abbreviated as S-Model). For each anchor model, we present the original performance in the first row for reference.

Datasets		MovieLens		Diginetica		Book-Crossing		Video		Baby		Beauty		Foursquare		Lastfm		Jester	
Method	S-Model	F1	NDCG	F1	NDCG	F1	NDCG	F1	NDCG	F1	NDCG	F1	NDCG	F1	NDCG	F1	NDCG	F1	NDCG
Original NARM		4.93	9.45	7.15	14.45	0.50	1.12	4.50	9.53	0.90	1.69	1.33	2.57	3.55	7.27	2.75	5.53	7.25	14.69
H	NARM	4.95	9.23	7.03	14.12	1.00	2.17	5.15	10.27	0.90	1.68	1.58	2.98	4.33	8.52	4.72	8.98	11.17	26.00
	SASRec	5.07	9.46	7.72	15.35	0.98	2.08	5.48	10.84	0.88	1.66	1.67	3.19	4.28	8.54	4.97	9.65	11.68	27.01
	STAMP	5.07	9.46	7.72	15.35	0.98	2.08	5.48	10.84	0.88	1.66	1.67	3.19	4.28	8.54	4.97	9.65	11.20	24.96
D	NARM	5.17	9.56	8.32	16.71	0.98	2.12	5.90	11.85	1.00	1.84	1.73	3.33	4.32	8.65	5.05	9.82	11.85	27.25
	SASRec	5.13	9.59	8.17	16.37	1.05	2.24	5.53	10.98	0.92	1.72	1.73	3.29	4.33	8.62	5.03	9.76	11.87	27.27
	STAMP	5.18	9.59	8.27	16.59	1.13	2.68	5.82	11.79	1.12	2.07	1.70	3.26	4.25	8.52	5.02	9.75	11.58	26.51
M	NARM	5.27	9.67	8.32	16.72	1.05	2.26	5.88	11.93	1.03	1.91	1.75	3.36	4.32	8.73	5.03	9.95	11.98	27.47
	SASRec	5.27	9.67	8.30	16.71	1.10	2.32	5.88	11.95	0.95	1.80	1.75	3.36	4.33	8.79	5.12	9.93	11.97	27.47
	STAMP	5.27	9.68	8.32	16.71	1.22	2.89	5.83	11.91	1.22	2.27	1.75	3.36	4.22	8.63	5.10	9.99	11.97	27.46
Origin SASRec		4.73	8.77	8.15	16.68	0.92	1.83	4.95	8.48	0.85	1.57	1.32	2.44	3.97	7.43	2.65	5.11	7.43	15.08
H	NARM	4.72	8.79	6.65	13.64	0.88	1.70	5.15	9.65	0.92	1.73	1.50	2.84	4.17	8.73	4.48	8.74	11.13	25.82
	SASRec	4.70	8.73	5.20	11.38	1.03	1.93	5.52	9.75	0.98	1.89	1.50	2.85	4.55	9.04	4.75	9.18	10.85	25.76
	STAMP	4.75	8.77	6.13	13.26	0.97	2.33	5.43	9.73	0.93	1.82	1.58	3.04	4.40	9.13	4.27	9.01	7.43	20.53
D	NARM	4.82	8.98	8.77	17.24	1.20	2.30	5.60	10.44	0.98	1.88	1.55	2.93	4.53	9.22	4.82	9.24	11.58	26.82
	SASRec	4.90	9.08	8.77	17.27	1.27	2.32	5.98	10.41	0.98	1.89	1.58	2.98	4.75	9.17	4.87	9.34	11.30	26.48
	STAMP	4.93	9.05	8.60	17.03	1.03	2.53	5.75	10.10	0.98	1.88	1.62	3.12	4.43	9.23	4.90	9.37	10.28	24.49
M	NARM	4.88	9.03	8.80	17.34	1.32	2.51	6.07	10.54	1.00	1.91	1.65	3.11	4.57	9.34	4.88	9.45	11.85	27.32
	SASRec	4.92	9.06	8.82	17.34	1.30	2.42	6.07	10.54	1.02	1.92	1.65	3.11	4.63	9.33	4.90	9.40	11.92	27.41
	STAMP	4.95	9.07	8.80	17.33	1.17	2.79	6.00	10.66	1.00	1.90	1.67	3.15	4.63	9.37	4.93	9.50	11.90	27.39
Origin STAMP		4.57	9.16	7.18	14.71	0.80	1.87	4.55	9.41	0.77	1.40	1.30	2.67	3.38	6.74	2.77	5.48	7.45	14.98
H	NARM	4.47	8.92	7.83	15.87	1.07	2.41	5.47	10.15	0.82	1.57	1.63	3.11	3.80	8.14	4.15	9.13	10.87	23.72
	SASRec	4.65	9.14	7.70	15.51	1.05	2.37	5.38	10.88	0.83	1.65	1.60	3.19	3.85	8.09	4.30	9.12	10.47	25.09
	STAMP	4.75	9.35	7.83	15.83	1.15	2.61	5.20	10.43	0.78	1.55	1.72	3.28	3.77	7.72	4.28	9.07	10.38	24.96
D	NARM	4.78	9.41	7.90	16.02	1.20	2.72	5.68	11.61	0.87	1.67	1.75	3.35	3.92	8.21	4.33	9.35	10.58	25.48
	SASRec	4.82	9.45	7.88	16.05	1.22	2.83	5.72	11.92	0.88	1.71	1.63	3.34	3.92	8.20	4.33	9.33	10.50	25.17
	STAMP	4.78	9.45	7.88	16.05	1.23	2.80	5.75	11.73	0.85	1.66	1.75	3.34	3.90	8.17	4.35	9.26	10.55	25.22
H	NARM	4.87	9.57	7.95	16.14	1.25	2.91	5.83	11.97	0.92	1.78	1.70	3.46	4.00	8.34	4.43	9.46	11.00	26.05
	SASRec	4.92	9.58	7.95	16.14	1.25	2.94	5.73	12.08	0.92	1.76	1.70	3.45	3.92	8.30	4.35	9.45	10.63	25.36
	STAMP	4.83	9.52	7.95	16.14	1.23	2.90	5.78	11.98	0.92	1.76	1.70	3.45	3.90	8.23	4.33	9.43	10.62	25.33

[1] We omit "%" on the numbers and "@5" on the evaluation metrics for simplicity.

declines as L becomes larger. The reason can be that in each iteration, the generated samples can be noisy, running our framework on them may accumulate the error to the next iteration. As the iteration number increases, the samples are severely biased which may lead to lowered performance.

4.4 Different Sampler Implementations

In this section, we investigate the influence of different sampler implementations for the final performance. For each of the heuristic and learning-based samplers, we use NARM, SASRec and STAMP as the specific implementations, respectively. The parameters are set as their optimal values tuned in the above experiments. From the results presented in Table 3, we can see: in most cases, the improvement of our framework is robust to different sampler implementations, and the results are consistent for different anchor models. More specifically, for the same sampler implementation, the performance ranking between the heuristic, data- and model-oriented methods are stable in most cases. This result suggests that the performance can be more relevant with the internal designs of different samplers, but not their specific implementations. It is interesting to see that for each anchor model, the best performance is achieved when the sampler implementation is different in more cases. We speculate that different model architectures may reveal the user sequential behavior patterns from different perspectives, which may introduce more complementary knowledge to enhance the final recommendation performance. Actually, this observation is aligned with the spirit of many machine learning paradigms such as co-teaching [10] and multi-view learning [27], which believe that models with different parameterizations may inspire each other to promote the final performance.

4.5 Hyper-parameter Analysis

In this section, we analyze the influence of the key hyper parameters in our framework. All the other parameters are set as their optimal values, and we report the results on the same datasets and evaluation metric as the above experiments.

Influence of the confidence parameter κ: κ sets the confidence of the sampler model on the generated sequences. We tune it in $[0.0, 10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]$. For the extreme cases, $\kappa = 0.0$ means that we do not control the noise, all the samples are allowed to participate the training of the anchor model. When κ is set as 1.0, no samples will be generated, the result is exactly the performance of the original anchor model. We present the results in Figure 5, from which we can see: the best performance is usually achieved when κ is moderate. This agrees with our opinion in section 2.5, i.e., too small κ may introduce too much noisy information into the training process, while too large κ may severely reduce the number of generated sequences. Both of

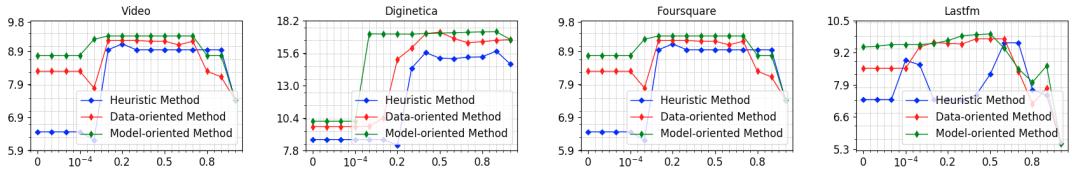

Figure 5: Influence of the confidence parameter κ.

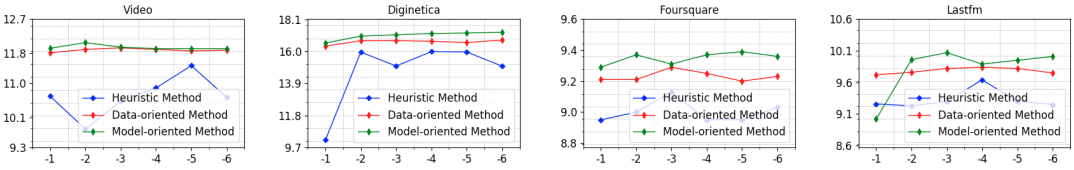

Figure 6: Influence of the index d.

these situations are suboptimal. By tunning κ in proper ranges, we are allowed to achieve better trade-offs to improve the recommendation performance. **Influence of the intervention index d:** In order to investigate the influence of the user recent behaviors, we tune d in the range of $[-1, -2, -3, -4, -5, -6]$, where the negative value means that the index is from back to front. The results are presented in Figure 6. We can see: for the heuristic method, the performance fluctuates a lot as d varies, but even the best result is still unsatisfied. In most cases, the performance of the learning-based methods are more robust to the index d, and the slightly better performances are usually achieved when d falls in the range of $[-2, -5]$, which may suggest that moderate d's can be better choices in practice.

4.6 Case Study

In order to provide an intuitive understanding of the generated sequences by our framework. We present a case study based on the MovieLens dataset in Figure 7, where the second last item is altered, and the replacement item is selected from the whole item set (i.e., $C = \mathcal{I}$). In the heuristic method (the second line), the item is altered in a random manner, and "House" is selected to replace the original movie–"Run Lola Run". However, the categories of these movies are quite different, which makes the generated sequence less reasonable, and may limit its effectiveness for promoting the anchor model performance. In the learning-based methods (the third and forth lines), the selected movies share the same categories with the original item, i.e., "Ever After" and "Run Lola Run" are both Romance movies, "Godfather" and "Run Lola Run" belong to the same action and crime categories. This manifests that learning-based methods can generate more reasonable sequences, which can also be justified by the superior performance of these methods according to above quantitative experiments.

5 CONCLUSION

In this paper, we propose to improve sequential recommendation by enriching the user behavior sequences based on the idea of counterfactual thinking. To achieve this goal, we design three implementations including both heuristic and learning-based methods. We also analyze the sample complexity of the designed framework, and propose a simple but effective method to control the noisy information. In the experiments, we evaluate our framework based

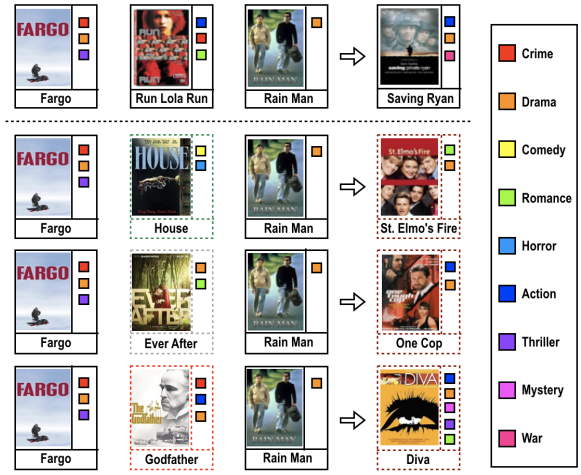

Figure 7: Case study. In the first line, we present the original user behavior sequence. The second line is the sequence generated from the heuristic method. The sequences produced from the learning-based methods are shown in the third (data-oriented method) and forth (model-oriented method) lines. The picture of each movie is downloaded from https://www.amazon.com/, and the categories of the movies are also presented for reference.

on nine real-world datasets to demonstrate its effectiveness and generality. This paper makes a first step on applying the idea of counterfactual thinking to the field of sequential recommendation. There is still much room for improvement. To begin with, one can study how to simultaneously replace multiple items, or investigate how to automatically learn the index of the item to be replaced. Then, it is also interesting to explore other methods for relaxing the intractable optimization targets, e.g., formulating the learning process as a reinforcement learning problem.

6 ACKNOWLEDGMENT

This work is supported in part by Beijing Outstanding Young Scientist Program NO. BJJWZYJH012019100020098 and National Natural Science Foundation of China (No. 61832017).

REFERENCES

[1] Ehsan Abbasnejad, Damien Teney, Amin Parvaneh, Javen Shi, and Anton van den Hengel. 2020. Counterfactual vision and language learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10044–10054.

[2] Oron Ashual and Lior Wolf. 2019. Specifying object attributes and relations in interactive scene generation. In *Proceedings of the IEEE International Conference on Computer Vision*. 4561–4569.

[3] 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 *Proceedings of the IEEE International Conference on Computer Vision*. 4613–4623.

[4] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 108–116.

[5] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. 2018. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine* 35, 1 (2018), 53–65.

[6] Tsu-Jui Fu, Xin Eric Wang, Matthew F Peterson, Scott T Grafton, Miguel P Eckstein, and William Yang Wang. 2020. Counterfactual Vision-and-Language Navigation via Adversarial Path Sampler. In *European Conference on Computer Vision*. Springer, 71–86.

[7] Hongchang Gao and Heng Huang. 2018. Self-paced network embedding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1406–1415.

[8] Sahaj Garg, Vincent Perot, Nicole Limtiaco, Ankur Taly, Ed H Chi, and Alex Beutel. 2019. Counterfactual fairness in text classification through robustness. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*. 219–226.

[9] Yash Goyal, Ziyan Wu, Jan Ernst, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Counterfactual visual explanations. *arXiv preprint arXiv:1904.07451* (2019).

[10] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *arXiv preprint arXiv:1804.06872* (2018).

[11] Ruining He and Julian McAuley. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 191–200.

[12] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial personalized ranking for recommendation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 355–364.

[13] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. Nais: Neural attentive item similarity model for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 30, 12 (2018), 2354–2366.

[14] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[15] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 505–514.

[16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1125–1134.

[17] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.

[18] Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. 2017. Counterfactual fairness. In *Advances in neural information processing systems*. 4066–4076.

[19] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1419–1428.

[20] Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial ranking for language generation. *Advances in neural information processing systems* 30 (2017), 3155–3165.

[21] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1831–1839.

[22] Chen Ma, Liheng Ma, Yingxue Zhang, Jianing Sun, Xue Liu, and Mark Coates. 2020. Memory Augmented Graph Neural Networks for Sequential Recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 5045–5052.

[23] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

[24] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. 811–820.

[25] Shai Shalev-Shwartz and Shai Ben-David. 2014. *Understanding machine learning: From theory to algorithms*. Cambridge university press.

[26] 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 *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1441–1450.

[27] Shiliang Sun. 2013. A survey of multi-view machine learning. *Neural computing and applications* 23, 7 (2013), 2031–2038.

[28] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 565–573.

[29] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. 515–524.

[30] Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James Sharpnack. 2020. SSE-PT: Sequential recommendation via personalized transformer. In *Fourteenth ACM Conference on Recommender Systems*. 328–337.

[31] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Jiajie Xu, Victor S Sheng S. Sheng, Zhiming Cui, Xiaofang Zhou, and Hui Xiong. 2019. Recurrent convolutional neural network for sequential recommendation. In *The World Wide Web Conference*. 3398–3404.

[32] Shuyuan Xu, Yunqi Li, Shuchang Liu, Zuohui Fu, and Yongfeng Zhang. 2020. Learning Post-Hoc Causal Explanations for Recommendation. *arXiv preprint arXiv:2006.16977* (2020).

[33] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. 2018. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1316–1324.

[34] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Kaiyuan Li, Yushuo Chen, Yujie Lu, Hui Wang, Changxin Tian, Xingyu Pan, Yingqian Min, Zhichao Feng, Xinyan Fan, Xu Chen, Pengfei Wang, Wendi Ji, Yaliang Li, Xiaoling Wang, and Ji-Rong Wen. 2020. RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms. *arXiv preprint arXiv:2011.01731* (2020).

[35] Ran Zmigrod, Sabrina J Mielke, Hanna Wallach, and Ryan Cotterell. 2019. Counterfactual data augmentation for mitigating gender stereotypes in languages with rich morphology. *arXiv preprint arXiv:1906.04571* (2019).