Counterfactual Review-based Recommendation

Kun Xiong[4,#], Wenwen Ye[4,#], Xu Chen[1,2,*], Yongfeng Zhang[3], Wayne Xin Zhao[1,2]
Binbin Hu[5], Zhiqiang Zhang[5], Jun Zhou[5]
[1]Beijing Key Laboratory of Big Data Management and Analysis Methods, Beijing, China
[2]Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China
[3]Department of Computer Science, Rutgers University, [4]School of Software, Tsinghua University, [5]Ant Group

ABSTRACT

Incorporating review information into the recommender system has been demonstrated to be an effective method for boosting the recommendation performance. Previous research mainly focus on designing advanced architectures to better profile the users and items. However, the review information in realities can be highly sparse and imbalanced, which poses great challenges for effective user/item representations and satisfied performance enhancement. To alleviate this problem, in this paper, we propose to improve review-based recommendation by counterfactually augmenting the training samples. We focus on a common setting — feature-aware recommendation, and the main building block of our idea lies in the counterfactual question: "what would be the user's decision if her feature-level preference had been different?". When augmenting the training samples, we actively change the user preference (also called intervention), and predict the user feedback on the items based on pre-trained recommender models. Instead of changing the user preference in a random manner, we design a learning-based method to discover the samples which are more effective for model optimization. In order to improve the sample qualities, we propose two strategies — constrained feature perturbation and frequency-based sampling — to equip our model. Since the sample generation model can be not perfect, we theoretically analyze the relation between the model prediction error and the number of generated samples. In addition, our framework can explain user pair-wise preferences, which is complementary to the traditional point-wise explanations. We conduct extensive experiments to demonstrate the effectiveness of our model.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Recommendation, User reviews, Counterfactual data augmentation

ACM Reference Format:
Kun Xiong[4,#], Wenwen Ye[4,#], Xu Chen[1,2,*], Yongfeng Zhang[3], Wayne Xin Zhao[1,2] and Binbin Hu[5], Zhiqiang Zhang[5], Jun Zhou[5]. 2021. Counterfactual Review-based Recommendation. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3482244

∗ Corresponding author, # Co-first author.

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482244

1 INTRODUCTION

Recommender system, as an effective remedy for information overloading, has been successfully applied to a number of real-world applications. The key of a successful recommender system lies in the accurate understanding of the user preference. To achieve this goal, recent years have witnessed an emerging trend of incorporating user review information into the recommender system. Comparing with the rating or implicit feedback, user reviews are much more informative, pooling an extensive wealth of knowledge about user opinions and sentiments, which helps to understand the user preference in a more comprehensive manner.

Previous review-based recommender models can be classified into two categories. On the one hand, many models process the review information on the document level [4, 5, 16, 17, 21, 21, 23, 23, 24, 30]. All the review contents are squeezed into an embedding vector to improve the user or item representation. Despite straightforward, these methods inevitably introduce too much user/item irrelevant information into the learning process, which brings difficulties for identifying the real user preference and enhancing the recommendation performance. On the other hand, many models utilize the review information by extracting user feature-level preferences (a.k.a. feature-aware recommendation). In specific, each user review is converted into many "(user, item, feature, sentiment)" tuples, which indicate the users' sentiments towards the items' features in a structured manner [7, 8, 26, 28]. As exampled in Figure 1(a), in the review of "I like the collar of this shirt, but the sleeve is not satisfied, since it is too tight for me.", the features are "collar" and "sleeve", and the user expresses positive and negative sentiments on them. The final extracted tuples are "(user, item, collar, positive)" and "(user, item, sleeve, negative)", respectively. Based on such user feature-level preference, people have devoted much effort to designing models based on matrix factorization [28], tensor [8, 26] factorization and deep neural network [7]. These models have shown great potential for improving the recommendation performance, but a fundamental problem has been largely ignored, that is, the review information can be not as ideal as expected. In real-world scenarios, different people may have various reviewing habits. Figure 1(b) and 1(c) present some statistics on the real-world Amazon[1] dataset, we can see: only a small amount of people frequently write reviews on their purchased products, while more inactive users only comment on very few items. In each review, the users may not write down all her preferences, and only a small

[1]http://jmcauley.ucsd.edu/data/amazon/

number of features are mentioned. These review- and feature-level imbalanced and sparse characters pose great challenges for better incorporating the review information into the recommender systems.

Counterfactual thinking is a recently emerged technique for enhancing the model performance and robustness [3, 6, 11, 25]. It explores the use of alternative actions that are not taken by the agent, which may allow the model to operate better in data-scarce scenarios. In this paper, we borrow the idea of counterfactual thinking to build review-based recommender models, which enables us to generate more training samples for alleviating the data insufficiency problem. In our method, the user-item similarity is predicted by matching the users' feature-level preference and the items' qualities on these features. We generate new samples by intervening on the users' feature-level preference, which simulates the counterfactuals of "what would be the user's behavior if her feature-level preferences had been different?". We focus on the users' pair-wise ranking behavior. For generating more effective training samples, we learn the "minimum" change of the user feature-level preference, which can "exactly" reverse the preference ranking of the user on a given item pair. To improve the sample qualities, we design the strategies of constrained feature perturbation and frequency-based sampling. Considering that the sample generation model can be not perfect, we theoretically analyze the relation between the number of generated samples and the model prediction error. Inspired by this theory, we propose a simple but effective method to control the potential noisy information contained in the generated samples. As a byproduct, our model can provide pair-wise recommendation explanations, which is able to explain why a user prefers an item to another one. We conduct extensive experiments based on real-world datasets to demonstrate our model's superiorities. In a summary, the main contributions of this paper can be concluded as follows:

- We propose to improve review-based recommendation by augmenting the training samples based on the idea of counterfactual thinking.
- We design a learning-based intervention method to discover critical samples for better model optimization. Our proposed method can also provide recommendation explanations for user pair-wise preference.
- We theoretically analyze the relation between the number of generated training samples and the model prediction error, and design a simple but effective method to enhance the quality of the generated samples.
- We conduct extensive experiments to evaluate our model's effectiveness and also present intuitive examples to illustrate the recommendation explanations provided by our model.

2 COUNTERFACTUAL FEATURE-AWARE COLLABORATIVE FILTERING

In this section, we introduce the proposed model — counterfactual feature-aware collaborative filtering (**CF**^{2}). Before describing the model details, we formally define the studied problem at first. And then, we illustrate our counterfactual data generation idea as well as the strategies of constrained feature perturbation and frequency-based sampling. In the next, we discuss our model on the learning algorithm, computational cost and explanation generation method. At last, we theoretically analyze the relation between the number of generated samples and the potential model prediction error.

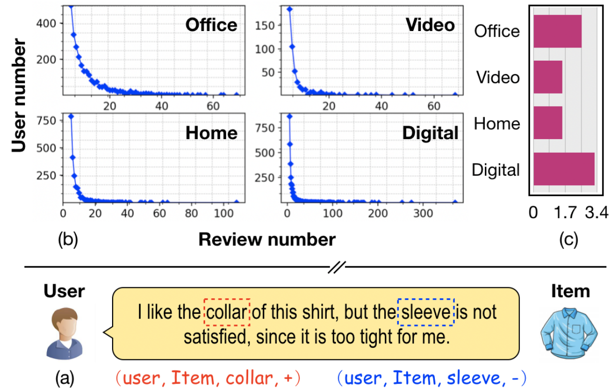

Figure 1: (a) An example of converting the review information into structured user feature-level preferences. (b) Statistics of the Amazon (we select four representative categories for presentation) dataset, where we plot the relation between the numbers of reviews and users. We can see only a small amount of people comment on large numbers of items. (c) Average numbers of features mentioned in each review.

2.1 Problem Definition

Suppose we have a user set $\mathcal{U}$ and an item set $\mathcal{I}$. Their interaction set is defined as $\mathcal{T} = \{(u,i)|u \in \mathcal{U}, i \in \mathcal{I}, u \text{ has interacted with } i\}$. The raw review information is converted into a set of quadruples $\mathcal{W} = \{(u_l, i_l, f_l, s_l)\}_{l=1}^{N}$ based on an open sourced toolkit called "Sentires"[2], where each element (u_l, i_l, f_l, s_l) means user $u_l (\in \mathcal{U})$ mentioned feature f_l in her review on item $i_l (\in \mathcal{I})$ with sentiment $s_l (\in \{-1,+1\})$. Obviously, if a user review contains more than one features, then it corresponds to multiple elements in $\mathcal{W}$. We denote the set of all features as $\mathcal{F}$, then based on $\mathcal{W}$, we follow the previous work [7, 28] to build a user-feature attentions matrix $\boldsymbol{A} = [\boldsymbol{A}_u] \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{F}|}$ and an item-feature qualities matrix $\boldsymbol{B} = [\boldsymbol{B}_i] \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{F}|}$, where $\boldsymbol{A}_{uf}$ and $\boldsymbol{B}_{if}$ represent the attention of user u and quality of item i on feature f, respectively. Given $\{\mathcal{U}, \mathcal{I}, \boldsymbol{A}, \boldsymbol{B}, \mathcal{T}\}$, our task is to learn a predictive function g, such that for each user, it can accurately rank all the items, and the rankings can be well explained based on the item features.

2.2 The Model Details

Given the user-feature attention matrix $\boldsymbol{A}$ and item-feature quality matrix $\boldsymbol{B}$, we define a target model g, which predicts the user-item affinity score via the feature information by:

$$r_{ui} = g(\boldsymbol{A}_u, \boldsymbol{B}_i) \tag{1}$$

where $\boldsymbol{A}_u \in \mathbb{R}^{1\times|\mathcal{F}|}$ and $\boldsymbol{B}_i \in \mathbb{R}^{1\times|\mathcal{F}|}$ are the uth and ith row of $\boldsymbol{A}$ and $\boldsymbol{B}$, respectively, representing the attentions of u and qualities of i on all the features. The implementation of g will be detailed later.

[2] https://github.com/evison/Sentires

Recommender system aims to predict the users' most favorite items, which is basically a ranking problem. In this paper, we focus on the user's pair-wise preference, which is usually modeled by the following BPR loss [19]:

$$L_{\text{BPR}} = -\sum_{(u,i,j)\in \boldsymbol{O}} \log\left[\sigma(r_{ui} - r_{uj})\right] + \lambda||g||^2 \tag{2}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function. $\lambda||g||^2$ is the regularization term. $\boldsymbol{O}$ denotes the set of all training samples. Each element (u, i, j) means user u prefers item i to item j.

Counterfactual data generation. As mentioned before, the user review information can be quite insufficient in realities. In order to more comprehensively optimize the model, a nature idea is to generate more training samples. Intuitively, the users' pair-wise preferences are determined by their feature-level attentions. As exampled in Figure 2(a), for two candidate mobile phones, if a user casts more attention on the brand, then iPhone can be her better choice. While if she cares more on the price, then Xiaomi can be more attractive for her. Different user-feature attentions may lead to different rankings for the same item pair, which inspires us to generate new samples by asking "what would be the user's propensity on a given item pair if her feature-level attentions had been different?".

Straightforwardly, one can change the user-feature attentions in a random manner, and predict the item rankings by g to construct new samples. However, this method can be suboptimal, since different training samples are not equally important in terms of model optimization [12]. There is no mechanism in the random method to ensure superiorities of the generated samples. In order to solve this problem, we develop a learning-based method to discover more effective samples.

In a typical classification problem, the decision boundaries refer to the samples separating the feature spaces which can induce different output labels. The unique character of these samples is that: the output label can be altered even with a small alteration on the input features. Previous work [1, 12] have demonstrated that, these boundary samples are discriminative in revealing the underlying data patterns, and training based on them may lead to improved model performance. Our method is inspired by this principle. Intuitively, in our problem, the boundary sample is the user preference which exactly discriminates the ranking directions of a given item pair (as illustrated in Figure 2(b)). We learn such sample by "minimally" changing the observed user-feature attentions (i.e., $\boldsymbol{A}_u$), such that the preference ranking for a given item pair can be "exactly" reversed. Formally, we define a perturbation variable $\boldsymbol{\tau} \in \mathbb{R}^{|\mathcal{F}|}$ with each element representing the attention change applied to the corresponding item feature. We learn $\boldsymbol{\tau}$ for each triplet $(u, i, j) \in \boldsymbol{O}$ independently by the following objective[3]:

$$\min_{\boldsymbol{\tau}} ||\boldsymbol{\tau}||_2^2 + \alpha \log\left[\sigma(r^*_{ui} - r^*_{uj})\right] \tag{3}$$

where $r^*_{ui} = g(\boldsymbol{A}_u + \boldsymbol{\tau}, \boldsymbol{B}_i)$ is the estimated score after changing the user preference. α is a tunning parameter balancing different terms. The parameters of g is fixed in the optimization process.

[3]Since all the following description is focused on one sample, we omit the index of the user and item pair on $\boldsymbol{\tau}$ for simplicity.

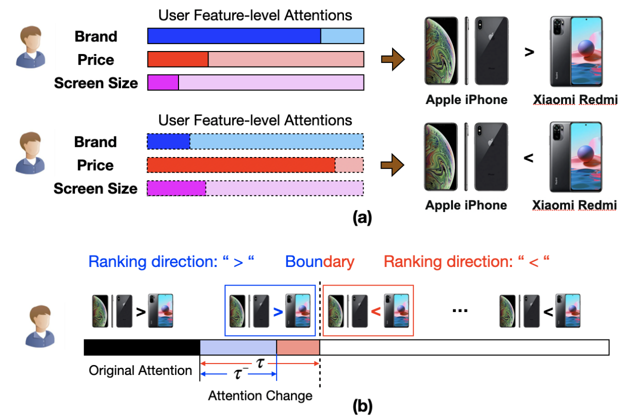

Figure 2: (a) An example on the effects of user feature-level preference on item ranking. (b) An illustration on the decision boundary, where we simplize the problem to include just one feature, and the counterfactual sample we would like to generate is near the boundary with the original order reversed. The red strap indicates the minimum user attention change τ in order to reverse the item ranking. The blue strap illustrates that if the attention change is not large enough (e.g., τ^-), then the item ranking remains unchanged.

In this objective, the first term aims to minimize the change of the user feature-level preferences. The second term tries to reverse the preference ranking between item i and j. By jointly optimizing them, we would like to change the user preference in a minimum manner, such that the item ranking can be exactly altered.

Once $\boldsymbol{\tau}$ is learned, the new sample is generated by:

$$\begin{cases} \text{Generate } (u^*, j, i). & \text{if } r^*_{ui} \leq r^*_{uj} \\ \quad \text{No generation.} & \text{otherwise} \end{cases} \tag{4}$$

where the feature attentions of u^* is $\boldsymbol{A}_u + \boldsymbol{\tau}$. Since we minimize $\boldsymbol{\tau}$ in equation (3), for each generated sample, a small alteration on the user-feature attentions (e.g., the blue strap in Figure 2(b)) will make the perturbation variable $\boldsymbol{\tau}$ not large enough to reverse the preference ranking, which implies that the sample is near the decision boundary.

Constrained feature perturbation. In practice, there can be lots of item features in the system, but usually, people may only consider a small part of them in the decision process [28]. To incorporate such character into our method, we design both hard and soft methods to impose additional constraints on the perturbed features. In the hard method, we restrict the perturbation to the users' mostly cared features. In specific, we select K largest elements in $\boldsymbol{A}_u$, and denote the set of their indexes by z_u. The user attentions are only allowed to change on the features in z_u, which leads to the following objective:

$$\min_{\boldsymbol{\tau}} ||\boldsymbol{k}^u \odot \boldsymbol{\tau}||_2^2 + \alpha \log\left[\sigma(\bar{r}_{ui} - \bar{r}_{uj})\right] \tag{5}$$

where $\odot$ is the element-wise product (a.k.a. Hadamard product). $\bar{r}_{ui} = g(\boldsymbol{A}_u + \boldsymbol{k}^u \odot \boldsymbol{\tau}, \boldsymbol{B}_i)$. $\boldsymbol{k}^u \in \mathbb{R}^{|\mathcal{F}|}$ is a mask vector, and $\boldsymbol{k}^u_i = 1$ if $i \in z_u$, otherwise $\boldsymbol{k}^u_i = 0$. This formula is a generalized case of (3), and will immediately reduce to (3) when $K = |\mathcal{F}|$.

Despite straightforward, the optimal K may vary on different samples, and it is too time consuming to tune K for each sample separately. For solving this problem, we introduce L_1-norm to encourage the sparse structure of $\boldsymbol{\tau}$, which automatically selects the important features in a soft manner. The corresponding objective is:

$$\min_{\boldsymbol{\tau}} ||\boldsymbol{\tau}||_2^2 + ||\boldsymbol{\tau}||_1 + \alpha \log [\sigma(r_{ui}^* - r_{uj}^*)] \tag{6}$$

Both of the hard and soft methods have their own advantages and shortcomings. The hard method costs more effort to determine the hyper-parameter K, but it can incorporate intuitive prior knowledge (e.g., perturbing only on the users' most cared features) for better performance. The soft method needs not to tune K, but the model can be too flexible to efficiently converge to the optimal results.

Frequency-based sampling. In order to fairly train different users, we balance the number of generated samples according to the users' reviewing frequency. In specific, suppose there are n_u reviews for user u, then we generate new samples for her with the probability of $\frac{\frac{1}{n_u}}{\sum_{i=1}^{|\mathcal{U}|}\frac{1}{n_i}}$. In this way, more samples will be generated for the users with less reviews in the original data, which may help to train these users more sufficiently, while the users with more reviews are suppressed to have less generated samples. By this strategy, our model can be equally optimized for different users, and the learned parameters will not over-represent only a small amount of users.

Implementation of g. Actually, the above counterfactual idea is a framework, and we explore different implementations of g to demonstrate its effectiveness. The general architecture of g is a multi-layer neural network, that is:

$$r_{ui} = \boldsymbol{W}_T \sigma_T(\boldsymbol{W}_{T-1}\sigma_{T-1}(...(\boldsymbol{W}_1\sigma_1(m(\boldsymbol{A}_u, \boldsymbol{B}_i)) + \boldsymbol{b}_1) + ...) + \boldsymbol{b}_{T-1}) + b_T \tag{7}$$

where, for the tth layer ($t \in [1, l]$), σ_t is a non-linear activation function, and we specify it as ReLU for all the layers. $\boldsymbol{W}_t \in \mathbb{R}^{d_t \times d_{t-1}}$ and $\boldsymbol{b}_t \in \mathbb{R}^{d_t}$ are weighting parameters with $d_T = 1$. $m(\cdot)$ is an operator merging the user and item feature-level properties, and we explore it within the following functions:

- **Element-wise product:**

$$m(\boldsymbol{A}_u, \boldsymbol{B}_i) = \boldsymbol{W}_U^p \boldsymbol{A}_u^T \odot \boldsymbol{W}_I^p \boldsymbol{B}_i^T \tag{8}$$

where $\boldsymbol{W}_U^p \in \mathbb{R}^{d_0 \times |\mathcal{F}|}$ and $\boldsymbol{W}_I^p \in \mathbb{R}^{d_0 \times |\mathcal{F}|}$ are trainable parameters.

- **Element-wise add:**

$$m(\boldsymbol{A}_u, \boldsymbol{B}_i) = \boldsymbol{W}_U^a \boldsymbol{A}_u^T + \boldsymbol{W}_I^a \boldsymbol{B}_i^T \tag{9}$$

where $\boldsymbol{W}_U^a \in \mathbb{R}^{d_0 \times |\mathcal{F}|}$ and $\boldsymbol{W}_I^a \in \mathbb{R}^{d_0 \times |\mathcal{F}|}$ are trainable parameters.

- **Hybrid method:**

$$m(\boldsymbol{A}_u, \boldsymbol{B}_i) = [\boldsymbol{W}_U^{h1} \boldsymbol{A}_u^T \odot \boldsymbol{W}_I^{h1} \boldsymbol{B}_i^T, \boldsymbol{W}_U^{h2} \boldsymbol{A}_u^T + \boldsymbol{W}_I^{h2} \boldsymbol{B}_i^T] \tag{10}$$

where $\boldsymbol{W}_U^{h1}, \boldsymbol{W}_U^{h2} \in \mathbb{R}^{d_0 \times |\mathcal{F}|}$ and $\boldsymbol{W}_I^{h1}, \boldsymbol{W}_I^{h2} \in \mathbb{R}^{d_0 \times |\mathcal{F}|}$ are trainable parameters.

- **Attention-based method:**

$$m(\boldsymbol{A}_u, \boldsymbol{B}_i) = \boldsymbol{W}^{att}[\boldsymbol{\alpha}_{ui} \odot (\boldsymbol{A}_u^T \odot \boldsymbol{B}_i^T)] \tag{11}$$

where $\boldsymbol{\alpha}_{ui} = [\alpha_{ui,j}]_{j=1}^{|\mathcal{F}|}$ are the attention weights, and $\alpha_{ui,j}$ is computed as $\frac{\exp(w_1 A_{u,j} + w_2 B_{i,j})}{\sum_{k=1}^{|\mathcal{F}|}\exp(w_1 A_{u,k} + w_2 B_{i,k})}$. $w_1 \in \mathbb{R}$, $w_2 \in \mathbb{R}$ and $\boldsymbol{W}^{att} \in \mathbb{R}^{d_0 \times |\mathcal{F}|}$ are trainable parameters.

2.3 Further Discussion

In the above section, we have introduced our main idea. Here, we make more discussions on the proposed framework to it more complete and insightful.

On the complete learning process. We present the complete training process of our model in Algorithm 1. To begin with, the target model g is trained based on the original dataset by equation (2) (*line 1*). Then, we generate M counterfactual samples according to the user reviewing frequency based on formula (5) or (6) (*line 4-7*). At last, the target model is further learned by combining the original and generated data (*line 8-10*).

On the computational cost. During the optimization process, $\boldsymbol{\tau}$ is updated according to the following rule:

$$\boldsymbol{\tau} = \boldsymbol{\tau} - \beta[\frac{\alpha\partial \log [\sigma(r_{ui}^* - r_{uj}^*)]}{\partial \boldsymbol{\tau}} + \frac{\partial C(\boldsymbol{\tau})}{\partial \boldsymbol{\tau}}] \tag{12}$$

where β is the learning rate, $C(\boldsymbol{\tau})$ is $||\boldsymbol{k}^u \odot \boldsymbol{\tau}||_2^2$ for objective (5) and $||\boldsymbol{\tau}||_2^2 + ||\boldsymbol{\tau}||_1$ for (6). Since $\frac{\partial C(\boldsymbol{\tau})}{\partial \boldsymbol{\tau}}$ can be computed with constant cost, we focus our analysis on $\frac{\partial \log[\sigma(r_{ui}^* - r_{uj}^*)]}{\partial \boldsymbol{\tau}}$. Suppose we denote $x = r_{ui}^* - r_{uj}^*$, then we have

$$\frac{\partial \log[\sigma(x)]}{\partial \boldsymbol{\tau}} = \frac{\partial \log[\sigma(x)]}{\partial x} \cdot \frac{\partial x}{\partial \boldsymbol{\tau}} = \underbrace{[1 - \sigma(x)]}_{A} \cdot \underbrace{\frac{\partial x}{\partial \boldsymbol{\tau}}}_{B} \tag{13}$$

The computational cost of part A is in proportion to that of r_{ui}. Suppose the cost of operator $m(\cdot)$ is M, then part A costs $O(M + \sum_{t=1}^{T} d_{t-1}d_t)$. For analyzing part B, we rewrite equation (7) as:

$$\begin{aligned} r_{ui} &= g_T(g_{T-1}(...g_1(m(\boldsymbol{A}_u, \boldsymbol{B}_i))...)) \\ g_t(s) &= \boldsymbol{W}_t\sigma_t(s) + \boldsymbol{b}_t \quad t \in [1, T] \end{aligned} \tag{14}$$

Obviously, the computational cost of $\frac{\partial x}{\partial \boldsymbol{\tau}}$ is mainly determined by $\frac{\partial r_{ui}}{\partial \boldsymbol{\tau}}$. Suppose we denote $L_t = g_t(g_{t-1}(...g_1(m(\boldsymbol{A}_u, \boldsymbol{B}_i))...))$, then $L_t = \boldsymbol{W}_t\sigma_t(L_{t-1}) + \boldsymbol{b}_t$, and we have:

$$\frac{\partial r_{ui}}{\partial \boldsymbol{\tau}} = \frac{\partial L_T}{\partial L_{T-1}} \cdot \frac{\partial L_{T-1}}{\partial \boldsymbol{\tau}} = \boldsymbol{W}_T \cdot \frac{\partial L_{T-1}}{\partial \boldsymbol{\tau}} = \boldsymbol{W}_T \cdot \boldsymbol{W}_{T-1} \cdot ...\boldsymbol{W}_1 \cdot \frac{\partial m}{\partial \boldsymbol{\tau}} \tag{15}$$

where the second equation holds because ReLU is used as the activation function. Suppose the cost of $\frac{\partial m}{\partial \boldsymbol{\tau}}$ is M', then part B costs $O(M' + \sum_{t=1}^{T} d_{t-1}d_t)$. As a result, the total computational cost of equation (12) is $O(M' + M + \sum_{t=1}^{T} d_{t-1}d_t)$.

On the explainability. For generating recommendation explanations, we learn $\boldsymbol{\tau}$ for each feature separately, where we set $\boldsymbol{k}^u$ in equation (5) as a one-hot vector with k_s^u ($s \in [1, |\mathcal{F}|]$) as 1, then we have the following objective:

$$\min_{\tau_s} ||\tau_s||_2^2 + \alpha \log[\sigma(\bar{r}_{ui} - \bar{r}_{uj})] \tag{16}$$

where τ_s is the sth element of $\boldsymbol{\tau}$, representing the attention change on the sth feature.

$$\bar{r}_{ui} = g(\boldsymbol{A}_u + [\underbrace{0, 0, ..., 0}_{(s-1)\text{ zeros}}, \tau_s, \underbrace{0, 0, ..., 0}_{(|\mathcal{F}|-s)\text{ zeros}}], \boldsymbol{B}_i) \tag{17}$$

Intuitively, if we can alter the item ranking with smaller changes on the user-feature attentions, then the corresponding features should be more important, which are selected as the explanation features. The template for generating explanations can be: "We

Algorithm 1: Learning algorithm of CF^2

Train the target model g with the original dataset $\mathcal{O}$.
Initialize the counterfactual sample set $\mathcal{O}_c = \emptyset$.
for *i in [1, M]* **do**
    Sample a user u with the probability of $\frac{\frac{1}{n_u}}{\sum_{i=1}^{|\mathcal{U}|}\frac{1}{n_i}}$.
    Sample a triplet (u, i, j) in $\mathcal{O}$.
    Optimize formula (5) or (6) to get $\boldsymbol{\tau}$.
    Compute r^*_{ui} and r^*_{uj} based on $\boldsymbol{\tau}$.
    if $r^*_{ui} \le r^*_{uj}$ **then**
        $\mathcal{O}_c \leftarrow \mathcal{O}_c \cup (u^*, j, i)$
    end
end
Train the target model g based on $\mathcal{O} \cup \mathcal{O}_c$.

recommend you with item i instead of item j because of your cared feature s.".

While explainable recommendation has been widely studied before [5, 15, 18, 26, 28], existing methods mostly explain the items independently. However, in real-world scenarios, people always making decisions via comparisons, e.g., "which item is more expensive?", "which movie is more popular?". Our model can satisfy such human nature by explaining the ranking of an item pair, where the explanation for an item is not static, but influenced by the compared item. If the previous models can be concluded as providing point-wise explanations, our framework can be seen as a kind of pair-wise explainable recommender model.

2.4 Theoretical Analysis

Careful readers may find that the sample generation process highly depends on model g. If g is not accurate, then the augmented data can be noisy. In this section, we theoretically analyze the relation between the number of generated samples and the prediction error of g, if one wants to achieve sufficiently well performance. We base our analysis within the PAC learning framework. To begin with, we assume that equation (4) can recover the true ranking of the item pairs based on the noisy parameter $\eta \in (0, 0.5)$, i.e., suppose the true triplet is (u, i, j), then equation (4) generates the true (i.e., (u, i, j)) and wrong (i.e., (u, j, i)) samples with the probabilities of $1 - \eta$ and η, respectively. We have the following theory:

Theorem 1. *Suppose $h \in \mathcal{H}$ is an item ranking predictor*[4]*, where $\mathcal{H}$ is the hypothesis class. For any $\epsilon, \delta \in (0, 1)$ and $\eta \in (0, 0.5)$, if h is learned based on empirical risk minimization (ERM), and sample number is larger than $\frac{2\log(\frac{2|\mathcal{H}|}{\delta})}{\epsilon^2(1-2\eta)^2}$, then the error between the estimated result of h and true value is smaller than ϵ with probability larger than $1 - \delta$.*

The proof of this theory is similar to [27]. Suppose the prediction error of a hypothesis in $\mathcal{H}$ is s, then the total error is $\eta + s(1 - 2\eta)$, considering that the generated data is noisy. If the prediction error of h (i.e., s) is larger than ϵ, Then, we have the empirical mismatching rate of h is smaller than $\eta + \frac{\epsilon(1-2\eta)}{2}$. or the empirical mismatching rate of the optimal h^* is larger than $\eta + \frac{\epsilon(1-2\eta)}{2}$. Similar to [27], the probability of making both of the above statements hold is smaller than δ, which implies that the prediction error of h is smaller than ϵ with the probability larger than $1 - \delta$.

This theory provides insights on the relation between the number of generated samples and the noisy parameter. From the sample complexity $\frac{2\log(\frac{2|\mathcal{H}|}{\delta})}{\epsilon^2(1-2\eta)^2}$, we can see, as the noisy parameter η becoming larger, more samples are needed to achieve sufficiently well performance.

Controlling the noisy information. Inspired by this theory, we design a heuristic method to control the noisy information. In general, we only remain the samples which are more reliable. More specifically, we improve equation (4) by introducing a threshold $\kappa \in \mathbb{R}_{-}$, that is:

$$\begin{cases} \text{Generate } (u^*, j, i). & r^*_{ui} - r^*_{uj} \le \kappa \\ \text{No generation.} & r^*_{ui} - r^*_{uj} > \kappa \end{cases} \tag{18}$$

In this equation, if we use a smaller κ, the model has more confidence on the generated samples, and the noisy information is reduced. But at the same time, the number of new samples will be less, which may impact the model performance. If we select a larger κ, more samples will be generated for sufficient training, but the noise rate can also be increased. Thus, κ controls the trade-off between the number and reliability of the generated samples. While such noise control method is simple, it can achieve promising results in our experiments, and we left more advanced methods as the future work.

Table 1: Statistics of the datasets.

Dataset	#User	#Item	#Interaction	Density
Office Products	4905	2420	53258	0.45%
Digital Music	5541	3568	64706	0.33%
Tools & Home	16638	10217	134476	0.08%
Home & Kitchen	66519	28237	551682	0.03%
Yelp	4777	11774	187615	0.33%

3 EXPERIMENTS

In this section, we conduct experiments to verify our model's effectiveness, focusing on the following research questions:

RQ1: What is the overall performance of our model comparing with the baselines?

RQ2: How different components in our model contribute the final performance?

RQ3: How different hyper-parameters influence the model performance?

RQ4: Whether the explanations provided from our framework are reasonable?

In the following, we begin by introducing the experiment setup, and then present and analyze the results to answer the above questions.

[4]If h can accurately predict the ranking of any item pairs, then it can provide high reliable recommendation results.

Table 2: **Performance comparison between the baselines and our model. For each metric on different datasets, we use bold fonts to label the best performance.**

Dataset	Office Products			Digital			Tools & Home			Home & Kitchen			Yelp		
Metric (@5)	F_1	NDCG	HR	F_1	NDCG	HR	F_1	NDCG	HR	F_1	NDCG	HR	F_1	NDCG	HR
BPR	0.088	0.110	0.420	0.086	0.147	0.429	0.050	0.071	0.263	0.081	0.122	0.409	0.190	0.290	0.755
NCF	0.102	0.127	0.464	0.081	0.115	0.352	0.058	0.080	0.303	0.082	0.128	0.429	0.178	0.236	0.783
MPCN	0.109	0.131	0.477	0.089	0.125	0.371	0.061	0.086	0.323	0.110	0.192	0.566	0.181	0.255	0.791
EFM	0.108	0.135	0.469	0.091	0.149	0.453	0.079	0.134	0.391	0.130	0.229	0.580	0.193	0.289	0.801
A2CF	0.113	0.171	0.550	0.092	0.155	0.461	0.080	0.138	0.413	0.133	0.238	0.590	0.197	0.292	0.805
CF^2_{base}-P	0.117	0.176	0.543	0.091	0.158	0.455	0.078	0.137	0.420	0.135	0.224	0.581	0.194	0.284	0.798
CF^2_{rand}-P	0.112	0.162	0.523	0.089	0.146	0.448	0.075	0.123	0.401	0.141	0.227	0.589	0.191	0.274	0.793
CF^2_{hard}-P	0.127	0.179	0.571	0.099	0.164	0.479	0.084	0.154	0.434	0.138	0.232	0.587	0.207	0.298	0.812
CF^2_{soft}-P	0.126	0.184	0.570	0.099	0.154	0.491	0.085	0.135	0.440	0.140	0.228	0.592	0.197	0.281	0.811
CF^2_{base}-A	0.114	0.165	0.534	0.088	0.135	0.445	0.080	0.125	0.430	0.140	0.233	0.584	0.204	0.284	0.810
CF^2_{rand}-A	0.110	0.160	0.523	0.090	0.143	0.447	0.080	0.135	0.421	0.139	0.232	0.581	0.204	0.283	0.803
CF^2_{hard}-A	0.120	0.170	0.547	0.100	0.166	0.482	0.084	0.138	0.431	0.144	0.238	**0.601**	0.207	0.286	0.812
CF^2_{soft}-A	0.123	0.174	0.564	0.094	0.141	0.458	0.083	0.138	0.428	0.143	0.234	0.592	0.208	0.288	0.812
CF^2_{base}-H	0.119	0.184	0.557	0.089	0.152	0.443	0.082	0.151	0.432	0.138	0.233	0.575	0.201	0.277	0.804
CF^2_{rand}-H	0.114	0.180	0.552	0.088	0.150	0.436	0.080	0.130	0.430	0.137	0.230	0.577	0.202	0.280	0.813
CF^2_{hard}-H	**0.127**	**0.193**	**0.575**	0.097	0.161	0.472	0.087	0.159	0.436	0.143	0.239	0.596	0.210	0.289	0.817
CF^2_{soft}-H	0.126	0.188	0.571	0.096	0.143	0.467	0.084	0.156	0.433	0.142	0.239	0.594	0.209	0.287	0.816
CF^2_{base}-AT	0.118	0.164	0.540	0.098	0.173	0.482	0.085	0.145	0.435	0.139	0.234	0.587	0.209	0.281	0.813
CF^2_{rand}-AT	0.113	0.165	0.530	0.100	0.176	0.493	0.087	0.147	0.444	0.138	0.226	0.586	0.205	0.284	0.809
CF^2_{hard}-AT	0.124	0.169	0.552	**0.106**	**0.183**	**0.504**	**0.093**	**0.158**	**0.474**	**0.148**	**0.246**	0.599	**0.216**	**0.301**	**0.827**
CF^2_{soft}-AT	0.121	0.181	0.557	0.104	0.176	0.486	0.089	0.154	0.454	0.143	0.241	0.592	0.213	0.291	0.819

3.1 Experiment Setup

Datasets. We base our experiments on the Amazon and Yelp[5] datasets. Amazon is an e-commerce dataset, containing user review information on the products from 25 categories. We select four representative categories including *Office Products*, *Digital Music*, *Tools & Home* and *Home & Kitchen*. These datasets cover different characters, verying on the scale and density, e.g., *Office Products* is a small and dense dataset, while *Tool and Home Improvement* is much larger but sparser. Yelp is a reviewing dataset, which contains user comments on the Restaurants, Bars, Dentists and etc. The statistics of these datasets are presented in Table 1.

Baselines. We compare our model with the following representative baselines and most of these baselines can be used directly in the Bole project [29]:

BPR [19] is a well known recommender model for capturing user implicit feedback.

NCF [13] is a famous neural recommender model, which is able to capture the non-linear relationships between the user preferences and item properties.

MPCN [24] is a state-of-the-art review-based recommender model, which processes the review information on the document-level.

EFM [28] is a well known feature-aware recommender model based on matrix factorization.

A2CF [7] is a state-of-the-art feature-aware recommender model, where the user-item-feature correlations are captured by the attentive neural network.

$\mathbf{CF^2_{base}}$ is the model implemented by equation (7), and we do not augment the training data in this method.

$\mathbf{CF^2_{rand}}$ is a simple data augmentation model, where the user feature-level preference is randomly changed without learning the boundary samples.

We denote our framework based on equation (5) and (6) as $\mathbf{CF^2_{hard}}$ and $\mathbf{CF^2_{soft}}$, respectively. There are four options to implement $m(\cdot)$, and we call them as "**-P**" (element-wise product), "**-A**" (element-wise add), "**-H**" (hybrid method) and "**-AT**" (attention-based method), respectively.

Implementation details. In the experiments, each user's last and second last interactions are used for model testing and validation, while the others are left for training. The commonly used metrics including F_1, NDCG and Hit Ratio are leveraged for comparing different models. For each user, we recommend 5 items, which are compared with the ground truth for computing different metrics. In our model, the parameters are learned based on stochastic gradient decent (SGD). The hyper-parameters are determined by grid search.

[5]https://www.yelp.com/dataset/download

More specifically, the learning rate, batch size, K and threshold κ are tuned in the ranges of $[0.0001, 0.001, 0.01, 0.1, 0.3, 0.5, 0.7, 0.9, 1.5]$, $[32, 64, 128, 256, 512]$, $[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]$ and $[0.0, -0.1, -0.2, -0.3, -0.4, -0.5]$, respectively. For the baseline models, we set the parameters as the values reported in the original papers or tune them in the same ranges as our model's.

3.2 Overall Comparison

The overall comparison results are presented in Table 2, we can see: in most cases, the models without review information (i.e., BPR and NCF) perform worse than the other baselines, which verifies the effectiveness of user reviews in boosting the recommendation performance. Among the review-based models, EFM usually exhibits better performance than MPCN. We speculate that some review contents can be not related with the user or item properties. Blindly incorporating all the review information (like MPCN) may bias the model learning process and lower the final performance. By capturing the non-linear relationships between different features, A2CF outperforms EFM in most cases.

Encouragingly, our framework can consistently achieve the best performance on all the evaluation metrics across different datasets. For the same implementation of $m(\cdot)$, we can always observe improved performances of $\mathrm{CF}^2_{\mathrm{hard}}-X$ and $\mathrm{CF}^2_{\mathrm{soft}}-X$ against $\mathrm{CF}^2_{\mathrm{base}}-X$, where X belongs to {"**P**", "**A**", "**H**", "**AT**"}. This result demonstrates the effectiveness of our counterfactual data augmentation idea. However, if we take a closer comparison between $\mathrm{CF}^2_{\mathrm{rand}}-X$ and our model, we can conclude that while data augmentation is potentially useful, randomly changing the user-feature attentions is not a good strategy. In order to generate more informative data, we learn to discover the decision boundary samples, which is shown to be more effective in promoting the target model performance.

$\mathrm{CF}^2_{\mathrm{hard}}-X$ and $\mathrm{CF}^2_{\mathrm{soft}}-X$ alternatively obtain the best performance on different datasets. Notably, the better results of $\mathrm{CF}^2_{\mathrm{hard}}-X$ is achieved by exploring different K's, which can be computational inefficient. In order to make a selection between $\mathrm{CF}^2_{\mathrm{hard}}-X$ and $\mathrm{CF}^2_{\mathrm{soft}}-X$, one should balance the trade-off between the accuracy and computational cost. For different implementations of $m(\cdot)$, hybrid or attention-based methods can achieve better performance in most cases. We speculate that hybrid method can incorporate different feature aggregation strategies, while attention-based method can distinguish the importances of different features, thus both of them can obtain superior performances.

3.3 Ablation Studies

The above section evaluates our framework as a whole. Readers may also be interested in how different model components contribute the final performance. There are three important modules in our framework, that is, constrained feature perturbation, frequency-based sampling and noisy information control. In this section, we conduct ablation studies by asking the following questions:

- Whether the strategy of constrained feature perturbation is effective?
- Whether frequency-based sampling is useful in boosting the performance?
- Whether the noise control method can benefit the recommendation performance?

To answer these questions, we compare our model with its five variants: $\underline{\mathrm{CF}^2_{-\mathrm{cst}}}$-X does not impose any constraints on the perturbed features. $\underline{\mathrm{CF}^2_{\mathrm{hard},-\mathrm{samp}}}$-X and $\underline{\mathrm{CF}^2_{\mathrm{soft},-\mathrm{samp}}}$-X remove the strategy of frequency-based sampling, and we constraint the features in both hard and soft manners. In $\underline{\mathrm{CF}^2_{\mathrm{hard},-\kappa}}$-X and $\underline{\mathrm{CF}^2_{\mathrm{soft},-\kappa}}$-X, we do not filter the noisy information, and equation (4) is leveraged to generate new samples. Similarly, we regularize the features with both hard and soft methods. In the experiment, the model parameters are set as their optimal values, and we implement $m(\cdot)$ based on the hybrid (**H**) and attention-based (**AT**) methods (i.e., X is either **H** or **AT**), which have obtained better performance in the above experiments. We present the results on the Amazon datasets in Table 3, and the conclusions on Yelp are similar and omitted.

We can see: if we do not impose constraints on the features (i.e., $\mathrm{CF}^2_{-\mathrm{cst}}$-X), the performance of our framework is lowered on all the datasets and metrics. This result agrees with the observations in the previous work [28], and manifests that involving too many features may indeed introduce too much flexibility for accurate user modeling. Appropriately constraining the perturbed features is an effective strategy for generating unambiguous samples to boost the recommendation performance. Comparing with $\mathrm{CF}^2_{\mathrm{hard},-\mathrm{samp}}$-X and $\mathrm{CF}^2_{\mathrm{soft},-\mathrm{samp}}$-X, our final model can consistently achieve better performance, which confirms the effectiveness of the frequency-based sampling strategy. At last, we can observe lowered performance of $\mathrm{CF}^2_{\mathrm{hard},-\kappa}$-X and $\mathrm{CF}^2_{\mathrm{soft},-\kappa}$-X comparing with CF^2-X. This manifests that the noisy control strategy is important for the final result. While the designed thresholding method is simple, it brings quite promising performance gains. For the hybrid method, the performance can be enhanced by about 10.8%, 13.7%, 7.16% on F_1, NDCG, HR, respectively. For the attention-based method, the improvements on the same metrics are 18.4%, 20.1% and 12.1%.

3.4 Influence of the Hyper-parameters

In this section, we study the influence of different hyper-parameters. We present the results on the attention-based method (i.e., "**AT**"), and the results on the other implementations of $m(\cdot)$ are similar and omitted. When tunning one parameter, we fix the other ones as their optimal values determined in the above experiments.

Influence of K. In the hard feature perturbation method, K is an important parameter, determining how many features should be involved for altering the user preference. We tune K in the range of [10, 100], and the results are presented in the first line of Figure 3. We can see: the best performance is usually achieved when K is moderate. The reason can be that, too little features can be not enough to capture the users' potentially complex preferences. While if we involve too many features, the model may introduce too much uncertainties, which makes it hard to achieve satisfied performance.

Influence of κ. As mentioned above, κ controls the confidence of the generated data. Smaller κ means higher confidence. To observe its influence, we tune it from -0.5 to -0.1, and the results are presented in the second line of Figure 3. It is interesting to see that too small κ (high confidence) does not lead to better performance. The reason can be that if we lower κ, the sample generation conditions become more rigorous, which reduces the number of

Table 3: Comparison between our model and its variants. We use bold fonts to label the best performance.

Dataset	Office Products			Digital			Tools & Home			Home & Kitchen		
Metric (@5)	F_1	NDCG	HR	F_1	NDCG	HR	F_1	NDCG	HR	F_1	NDCG	HR
CF$^2_{-\text{cst}}$-H	0.124	0.180	0.562	0.093	0.140	0.461	0.082	0.147	0.431	0.142	0.238	0.593
CF$^2_{\text{hard},-\text{samp}}$-H	0.122	0.172	0.566	0.095	0.149	0.468	0.083	0.138	0.438	0.141	0.236	0.587
CF$^2_{\text{soft},-\text{samp}}$-H	0.121	0.169	0.555	0.094	0.144	0.463	0.078	0.124	0.428	0.131	0.205	0.576
CF$^2_{\text{hard},-\kappa}$-H	0.112	0.168	0.535	0.087	0.142	0.425	0.074	0.117	0.399	0.137	0.229	0.577
CF$^2_{\text{soft},-\kappa}$-H	0.117	0.176	0.562	0.087	0.142	0.425	0.081	0.148	0.424	0.127	0.215	0.542
CF^2-H	**0.127**	**0.193**	**0.575**	**0.097**	**0.161**	**0.472**	**0.087**	**0.159**	**0.436**	**0.143**	**0.239**	**0.596**
CF$^2_{-\text{cst}}$-AT	0.119	0.167	0.538	0.102	0.175	0.496	0.087	0.141	0.461	0.145	0.245	0.595
CF$^2_{\text{hard},-\text{samp}}$-AT	0.117	0.166	0.537	0.101	0.174	0.492	0.089	0.153	0.447	0.143	0.239	0.596
CF$^2_{\text{soft},-\text{samp}}$-AT	0.119	0.177	0.543	0.102	0.175	0.496	0.087	0.153	0.451	0.141	0.239	0.589
CF$^2_{\text{hard},-\kappa}$-AT	0.078	0.109	0.399	0.097	0.170	0.479	0.067	0.112	0.345	0.135	0.227	0.583
CF$^2_{\text{soft},-\kappa}$-AT	0.119	0.178	0.564	0.101	0.173	0.494	0.079	0.124	0.431	0.142	0.240	0.590
CF^2-AT	**0.124**	**0.181**	**0.557**	**0.106**	**0.183**	**0.504**	**0.093**	**0.158**	**0.474**	**0.148**	**0.246**	**0.599**

* In the last line of each block, we present the best performance of our framework for reference.

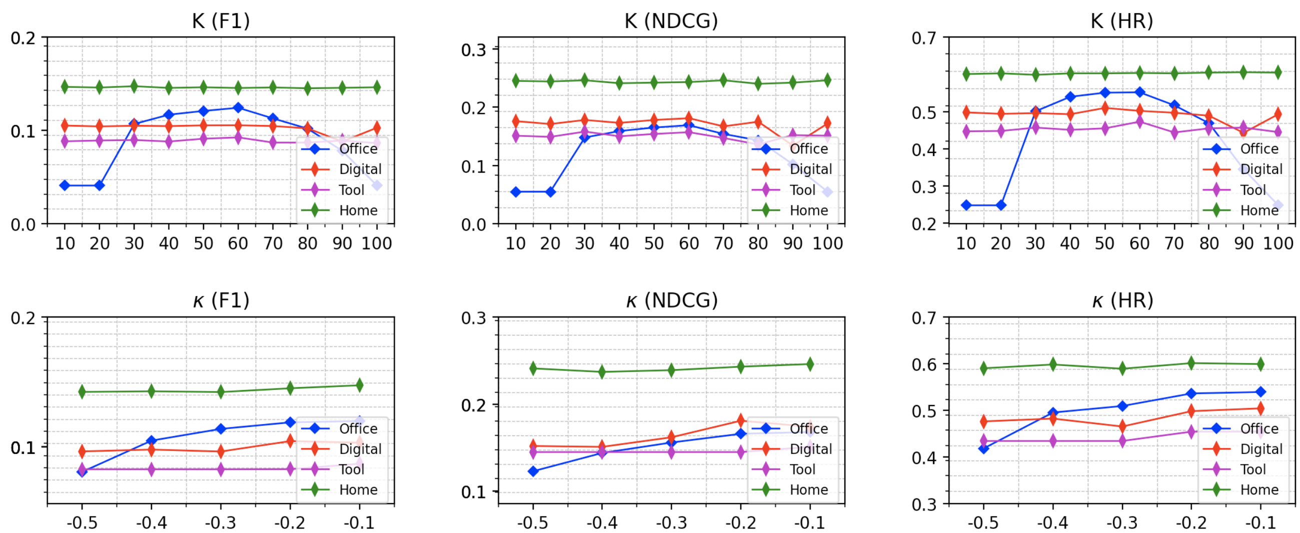

Figure 3: Influence of K and κ on the recommendation performance.

produced samples. This may lead to insufficient model optimization, and thus limit the recommendation performance. However, when κ reaches a relative large value, the performance tends to be stable. We speculate that while there can be more samples joining into the optimization process, they can be noisy, which is detrimental for the final performance.

3.5 Pair-wise Recommendation Explanations

In this section, we evaluate the explainability of our framework from both qualitative and quantitative perspectives.

3.5.1 Qualitative analysis. In order to provide intuitive understandings on our framework, in this section, we present many case studies to illustrate the generated pair-wise explanations in a qualitative manner. As mentioned in section 2.3, the features with smaller τ_s are more important for the current item ranking. We select five most important features for each case, and present the results in Figure 4(a). We can see: in the first case, the user is satisfied with the weight of the positive item, but complains on weight of the negative item. The weight can be an important feature influencing the user decisions, which is successfully learned from our model. In the second case, according to the user reviews, the item price can be an important feature in the user's mind when comparing the two items, which is accurately predicted by our model. These cases show the capability of our model in predicting the decisive features for the item rankings, which builds the basis for pair-wise explainable recommendation.

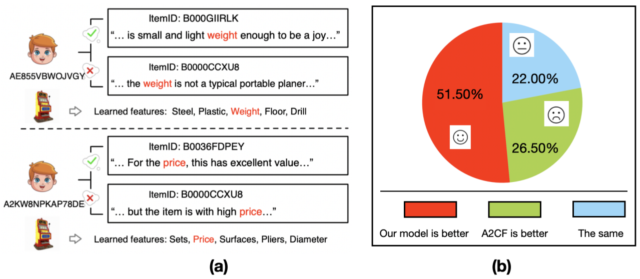

Figure 4: (a) Qualitative analysis. In each case, there is a user and an item pair, and we also present the real review information for reference. The bottom line shows the features learned from our model. (b) Results of the quantitative analysis.

3.5.2 Quantitative analysis. In addition to the above qualitative analysis, we also conduct quantitative studies on the generated recommendation explanations. More specifically, we compare our model with A2CF, which, as far as we know, is the only method for pair-wise explanations. We implement $m(\cdot)$ with the attention-based method, and the parameters are set as their optimal values. In the experiment, we randomly select 200 (user, positive item, negative item) triplets from the testing set of Tools & Home, where each user has at least 10 interactions in the training set. We ask the workers to read the reviews of each user in the training set. Since the user has sufficient interactions (>10), the workers can more accurately understand her preference. For both of our model and A2CF, we generate the most important features, and ask the workers to label: whether our model is better than, on par with or worse than A2CF. From the results shown in Figure 4(b), we can see, our model can indeed lead to more reasonable explanations, which demonstrates the effectiveness of leveraging the "sensitivity" of each feature to explain the recommendation results.

4 RELATED WORK

4.1 Feature-aware Recommendation

Feature-aware recommendation has attracted increasing attention in the past few years. It differentiates itself from the other review-based recommendation by extracting user feature-level preference from the raw user reviews. Among existing models, explicit factor model (EFM) [28] is a most famous algorithm, which captures the correlations between the users, items and features based on coupled matrix factorization. In this model, the user preferences and item qualities are connected based on the feature information. Once the model learned, one can explain the recommendation results with the item features. In the past few years, EFM has inspired many following research. MTER [26] and LRPPM [9] extend EFM by tensor factorization to emphasize the user personalized preference on the item features. A2CF [7] designs a neural network to capture the non-linear correlations among the users, items and features, and leverages attention mechanism to distinguish different feature importances. This method can also explain user pair-wise preferences, but it does not use the counterfactual idea to learn the minimum attention changes on the features. Different from these methods, which mostly focus on designing model architectures, we focus on a orthogonal direction, i.e., alleviating the training data insufficient problem, where we counterfactually augment the user review information to assist more comprehensive model optimization.

4.2 Counterfactual Thinking

Counterfactual thinking belongs to the human introspection behaviors, such as "what if I took another road?" and "what if I did not eat that apple?". Recently, the concept of counterfactual thinking has been introduced into the machine learning community to augment the training data by exploring the potential samples when the original conditions are revised [2, 10, 14, 20]. In the field of neural language processing (NLP), [31] leverages counterfactual data augmentation to mitigate gender stereotypes in the observed data. In the field of computer vision (CV), [11] generates additional trajectory data to enhance the vision-and-language navigation task in an adversarial manner. [6] incorporates the counterfactual idea into a multi-agent training process for scene graph generation. In this paper, we leverage the idea of counterfactual thinking to build review-based recommender models, which, to the best of our knowledge, is the first time in this field. In addition, we theoretically analyze that, if the generated samples are noisy, how many data one needs to generate in order to achieve sufficiently well performance within the PAC learning framework [22].

5 CONCLUSION

In this paper, we propose to enhance review-based recommendation based on the idea of counterfactual data augmentation. The key question for generating new samples is: "what would be the user's propensity on an item pair if her feature-attentions had been different?". Instead of randomly revising the users' feature-level preference, we learn to discover the decision boundary samples, which can be more effective in terms of model optimization. We also propose a method for providing pair-wise recommendation explanations, and theoretically analyze our framework when the generated samples are noisy. Extensive experiments are conducted to demonstrate our model's effectiveness.

This paper actually opens the door of incorporating causal inference into the field of review-based recommendation. There is still much room left for the following work. For example, one can introduce exogenous variables to model the users' previous status for more accurate sample generation. Since our model is a framework, one can easily extend it to other recommendation settings when the user and item can be represented by some types of "contents".

6 ACKNOWLEDGMENT

This work is supported in part by National Natural Science Foundation of China (No. 62102420 and No. 61832017), Beijing Outstanding Young Scientist Program NO. BJJWZYJH012019100020098, CCF-Ant Group Research Fund, Intelligent Social Governance Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China, and Public Computing Cloud, Renmin University of China.

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] Rupam Acharyya, Shouman Das, Ankani Chattoraj, and Md Iftekhar Tanveer. 2020. FairyTED: A Fair Rating Predictor for TED Talk Data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 338–345.

[3] 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.

[4] Rose Catherine and William Cohen. 2017. TransNets: Learning to Transform for Recommendation. *arXiv preprint arXiv:1704.02298* (2017).

[5] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural attentional rating regression with review-level explanations. In *WWW*. 1583–1592.

[6] 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.

[7] Tong Chen, Hongzhi Yin, Guanhua Ye, Zi Huang, Yang Wang, and Meng Wang. 2020. Try This Instead: Personalized and Interpretable Substitute Recommendation. (2020).

[8] Xu Chen, Zheng Qin, Yongfeng Zhang, and Tao Xu. 2016. Learning to Rank Features for Recommendation over Multiple Categories. In *SIGIR*.

[9] Xu Chen, Zheng Qin, Yongfeng Zhang, and Tao Xu. 2016. Learning to rank features for recommendation over multiple categories. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 305–314.

[10] Silvia Chiappa. 2019. Path-specific counterfactual fairness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 7801–7808.

[11] 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.

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

[13] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 173–182.

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

[15] Trung-Hoang Le and Hady W Lauw. 2021. Explainable Recommendation with Comparative Constraints on Product Aspects. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 967–975.

[16] Donghua Liu, Jing Li, Bo Du, Jun Chang, and Rong Gao. 2019. DAML: Dual Attention Mutual Learning between Ratings and Reviews for Item Recommendation. In *SIGKDD*. 344–352.

[17] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Recsys*.

[18] Deng Pan, Xiangrui Li, Xin Li, and Dongxiao Zhu. 2020. Explainable recommendation via interpretable feature mapping and evaluation of explainability. *arXiv preprint arXiv:2007.06133* (2020).

[19] 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.

[20] Chris Russell, Matt J Kusner, Joshua Loftus, and Ricardo Silva. 2017. When worlds collide: integrating different counterfactual assumptions in fairness. In *Advances in neural information processing systems*. 6414–6423.

[21] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. 2017. Interpretable Convolutional Neural Networks with Dual Local and Global Attention for Review Rating Prediction. In *Recsys*.

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

[23] Yunzhi Tan, Min Zhang, Yiqun Liu, and Shaoping Ma. 2016. Rating-Boosted Latent Topics: Understanding Users and Items with Ratings and Reviews.. In *IJCAI*.

[24] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-Pointer Co-Attention Networks for Recommendation. (2018).

[25] Khanh Hiep Tran, Azin Ghazimatin, and Rishiraj Saha Roy. 2021. Counterfactual Explanations for Neural Recommenders. *arXiv preprint arXiv:2105.05008* (2021).

[26] Nan Wang, Hongning Wang, Yiling Jia, and Yue Yin. 2018. Explainable Recommendation via Multi-Task Learning in Opinionated Text Data. *SIGIR* (2018).

[27] Zhenlei Wang, Jingsen Zhang, Hongteng Xu, Xu Chen, Yongfeng Zhang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Counterfactual Data-Augmented Sequential Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 347–356.

[28] 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*.

[29] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Kaiyuan Li, Yushuo Chen, Yujie Lu, Hui Wang, Changxin Tian, Xingyu Pan, et al. 2020. RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms. *arXiv preprint arXiv:2011.01731* (2020).

[30] Lei Zheng, Vahid Noroozi, and Philip S Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *WSDM*.

[31] 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).