Learning to Counterfactually Explain Recommendations

Yuanshun Yao, Chong Wang*, Hang Li
ByteDance
{kevin.yao,chong.wang,lihang.lh}@bytedance.com

Abstract

Recommender system practitioners are facing increasing pressure to explain recommendations. We explore how to explain recommendations using counterfactual logic, *i.e.* "Had you not interacted with the following items before, it is likely we would not recommend this item." Compared to traditional explanation logic, counterfactual explanations are easier to understand and more technically verifiable. The major challenge of generating such explanations is the computational cost because it requires repeatedly retraining the models to obtain the effect on a recommendation caused by removing user (interaction) history. We propose a learning-based framework to generate counterfactual explanations. The key idea is to train a *surrogate model* to learn *the effect of removing a subset of user history on the recommendation*. To this end, we first artificially simulate the counterfactual outcomes on the recommendation after deleting subsets of history. Then we train surrogate models to learn the mapping between a history deletion and the change in the recommendation caused by the deletion. Finally, to generate an explanation, we find the history subset predicted by the surrogate model that is most likely to remove the recommendation. Through offline experiments and online user studies, we show our method, compared to baselines, can generate explanations that are more counterfactually valid and more satisfactory considered by users.

1 Introduction

Today's recommender systems need to fulfill a wide range of needs from users. One increasingly important demand is to explain the recommendation results that users receive. In the literature on explainable recommender system, one of the most popular approaches is *item-based collaborative filtering*, which explains the recommended item by showing a set of items relevant to it. The logic is simple and easy to deploy.

However, there are two limitations in the item-based collaborative filtering logic. Consider the conventional explanation "You receive the recommendation of A because you visited item B, C, and D." *First*, the causal link is missing, *i.e.* how did visiting B, C, and D lead to the recommendation of A? Without a more explicitly stated logic, users might not be satisfied. *Second*, the technical claim is more or less questionable. Did the model truly make the decision *solely* based on the item relevance? This is unlikely to be true in today's complex recommender system. Therefore the explanations might lead to doubts from the users or even potential legal accountability.

One solution is to explain by *counterfactual logic*, as illustrated in Figure 1. Using the same example, the corresponding counterfactual explanation is "Had you not visited item B, C, and D, it is likely we would not recommend A." Compared to item-based collaborative filtering logic, it

*Work done while at ByteDance.

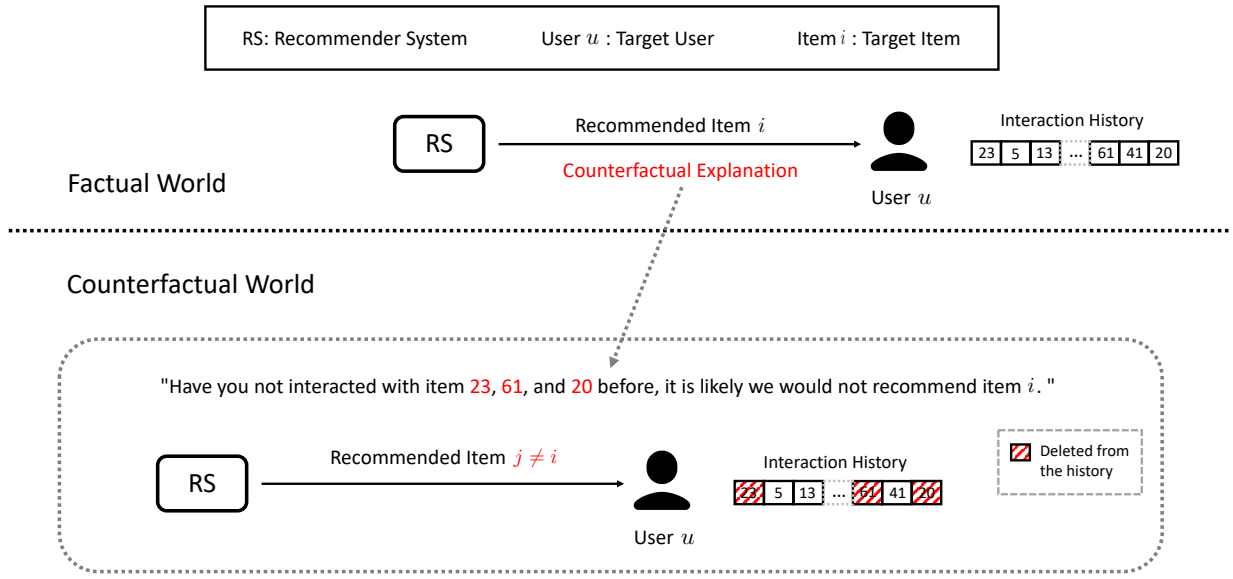


Figure 1: Definition of a counterfactual explanation. When the recommender system recommends an item i to a user u, the explanation is "Have you not interacted with the following items before, it is likely we would not recommend this item." In other words, we show the set of items in user u's interaction history that, if were removed, would change the current recommendation to a different item.

provides a more explicitly stated causal connection via "What-if" reasoning, and therefore more convincing, as shown in psychology [1, 2, 3, 4]. In addition, the claim is more technically verifiable because we can empirically test its truthfulness (by removing the claimed explanatory items and observing if the resulting recommendation actually changes or not).

However, generating such counterfactual explanations is computationally costly because it requires retraining the recommender model after we delete training subsets of user interaction history to observe the resulting change in the recommendation. So far, only a few works study how to generate counterfactual explanations in recommender systems. One approach uses Influence Function [5] to estimate how the predicted score on the recommendation might change if some training items were removed [6]. However, Influence Function is shown to be relatively inaccurate on deep models [7], and therefore it is unclear how counterfactually valid the generated explanation can be in practice. We compare with influence function in our experiments.

We propose a learning-based framework to generate counterfactual explanations. Our key insight is simple: we train a surrogate model to learn the mapping from a change in the recommendation model's training inputs (*i.e.* user history) to its causal effect on the recommendation. To this end, we first empirically simulate the counterfactual events repeatedly by removing subsets of user history, retraining the model, and observing the change in recommendation. We then train a surrogate model to predict how a recommendation would change given a deleted history. Once we have the surrogate model, we can replace every expensive actual retraining with a surrogate model inference that estimates the effect of deletion on the recommendation. Finally, we generate explanations by searching for the deleted history that is predicted by the surrogate model to have the maximum chance to remove the current recommended item from the recommendation list.

Our approach shifts the computational bottleneck from generating explanations on-the-fly to simulating counterfactual events, which is parallelizable and can be performed offline. In addition, our method reduces the number of retraining needed by relying on the surrogate model's generalizability. We only need to simulate a fraction of all possible deletions that are enough to train the surrogate model, and then rely on the surrogate model to generalize to unseen deletions. Furthermore, our method can be applied to any recommendation model. Through extensive experiments and user studies, we show that our method can generate more counterfactually valid explanations that are considered more satisfactory by users than baselines.

2 Background: Explainable Recommender System

We briefly summarize the major categories of existing approaches in the area of explainable recommender system. For more details, refer to [8] for a general survey.

The first and perhaps the most popular explanation approach is *collaborative filtering based logic*, which uses relevant users or items to explain recommendations. In *user-based collaborative filtering* [9, 10] logic, the recommender system finds a group of users similar to the current user, and explains that the system recommends the current item because some neighborhood users had some positive feedback on it. On the other hand, *item-based collaborative filtering* [11] logic explains the recommendation by finding a group of items from the current user's interaction history and informing users the reason for receiving such recommendation is that the user has interacted with those similar items in the past. For both logic, the relevance is usually obtained from user-item interactions. Broadly speaking, it can also be defined by features of items or users, *e.g.* user demographic information [12, 13, 14] or item features [15, 16, 17, 18].

The second category is *user opinion based logic*, which explains the recommendation through user opinions, mostly from user reviews. Most of the work is based on natural language processing techniques. For example, some prior work uses topic modeling to generate word clouds as explanations [19, 20]. In addition, using natural language generation, Costa *et al.* [21] train language models on user review corpus, and generate explanations in the form of natural language paragraphs based on the learned language models. Chang *et al.* [22] generate personalized natural language explanations based on explanations collected from crowdsourcing. Furthermore, Wang *et al.* [23] propose a multi-task learning modeling framework that predicts how the user would appreciate a particular item.

The last category is *social interaction based logic*, which generates explanations based on the users' social information. Sharma and Cosley show users a number of friends who also liked the item [24]. Chaney *et al.* [25] incorporate social influence information from social networks into the model, and provide explanations by indicating friends as the source of influence. Wang *et al.* [26] find a group of users who are the most persuasive in terms of explaining, and provide explanations in the text form "A and B also like the item."

Note that different explaining logic would require different explaining user interfaces. For example, *collaborative filtering* logic displays the relevant users or items, *user opinion* logic shows a text block, and *social interaction* logic has different user interface designs, ranging from a text sentence/paragraph to links to source influence. Because of their difference in displaying forms, it is difficult to compare explanations generated based on different logic in a controlled experimental setting. In this work, we focus on *item-based collaborative filtering* logic.

3 Problem Formulation

In a recommender system, let $\mathcal{U}$ denote the set of all users (*i.e.* all user IDs), $\mathcal{I}$ denote the set of all items (*i.e.* all item IDs), $\mathcal{R}$ denote the set of all ratings (*e.g.* 1-5 stars), and $\mathcal{S} \subseteq \mathcal{U} \times \mathcal{I} \times \mathcal{R}$ denote the history of all user-item interactions with the ratings used as the training set of the recommendation model $\hat{\theta}$.

Definition of Counterfactual Explanation. For a user $u \in \mathcal{U}$ who interacted with a set of items $I_u := \{i \in \mathcal{I} : (u, i, .) \in \mathcal{S}\}$ in the training set (*i.e.* u's visiting history), if the recommendation model $\hat{\theta}$ recommends an item i to user u, we want to find a subset of u's visiting history $E_{u,i} \subseteq I_u$ such that if $E_{u,i}$ were removed from the training set $\mathcal{S}$, then the model $\hat{\theta}$ would not recommend item i to user u.

Formally, if we remove an arbitrary subset of user u's history $D_u \subseteq I_u$ and retrain the model on the training set with D_u deleted, the resulting counterfactual recommendation model is[1]:

$$\hat{\theta}(-D_u) := \arg\min_{\theta \in \Theta} \sum_{(\tilde{u}, \tilde{i}, \tilde{y}) \in S} \Big[1 - \mathbb{I}(\tilde{u} = u \text{ and } \tilde{i} \in D_u)\Big] \cdot \ell(\tilde{y}, f(\tilde{u}, \tilde{i}; \theta)) \quad (1)$$

where $\mathbb{I}(.)$ is binary indicator function, $\tilde{y}$ is rating from user $\tilde{u}$ on item $\tilde{i}$ in the training set $\mathcal{S}$, $f(\tilde{u}, \tilde{i}; \theta)$ is the predicted score on item $\tilde{i}$ for user $\tilde{u}$ by the model θ , and $\ell(.)$ is recommendation model's training loss.

Suppose the recommender system shows top-K items as the recommendation list to all users, let $\pi_{u,\hat{\theta}}$ denote the ranking list of the items for user u produced by the original recommendation model $\hat{\theta}$ and let $\pi_{u,\hat{\theta}}(i)$ denote the rank of item i in $\pi_{u,\hat{\theta}}$, then an explanation $E_{u,i}$ is defined to be counterfactual if:

$$\pi_{u,\hat{\theta}(-E_{u,i})}(i) > K \quad (2)$$

In other words, the condition is the counterfactual model ranks the recommended item i at a position lower than K, which is the minimal rank decided by the recommender system for an item to appear in u's recommendation list.

Naive Approach. The naive way to generate $E_{u,i}$ is to exhaustively search on all possible subsets of I_u, *i.e.* the power set 2^{I_u}, as shown in Algorithm 1. It is, of course, forbiddingly expensive. To generate an explanation for one user and one received recommended item, the worst case is to retrain the model $O(2^{|I_u|})$ times. If we want to generate explanations for every user's top-K recommendation list, we would need to retrain $O(2^{\overline{|I_u|}} \times |\mathcal{U}| \times K)$ times where $\overline{|I_u|}$ is the average number of items users interacted in the training set.

Model Reuse Trick. One simple way to reduce the number of retraining times required is to reuse the counterfactually retrained models. Because for a user u and a removed subset D_u, the counterfactual model $\hat{\theta}(-D_u)$ is the same for any item $i \in \mathcal{I}$, we can use the same $\hat{\theta}(-D_u)$ to explain all items for the user u if D_u were removed. This reduces the number of retraining times to $O(2^{\overline{|I_u|}} \times |\mathcal{U}|)$. We use this trick in our method.

4 Insights and Methodology Overview

In this section, we explain the potential challenges of generating counterfactual explanations, key insights into our approach, and an overview of our method.

[1]Technically $\hat{\theta}(.)$ should have two inputs u and D_u to represent the counterfactual model, we fold u into D_u to simplify the notation.

Algorithm 1 Naive approach to generate counterfactual explanations.

Input: Target user: u, Target item: i, Rank Threshold: K.
Output: A counterfactual explanation $E_{u,i}$.
for $D_u \in 2^{I_u}$ **do**
    Retrain to get counterfactual model $\hat{\theta}(-D_u)$ with eq (1).
    if $\pi_{u,\hat{\theta}(-D_u)}(i) > K$ **then**
        Return D_u.
    end if
end for
Return $\emptyset$.

4.1 Key Insights

Challenges. There are three main challenges to generating counterfactual explanations. *First*, how to scale? As illustrated by the naive approach, finding such explanations can be computationally costly. *Second*, how to make the method be applicable to any type of recommendation model? Ideally we do not want to change the existing recommendation module so that the explanation module can be independently deployed and maintained. *Third*, how to describe the counterfactual events? Because machine learning lacks the proper language to model counterfactual events since the majority of ML training algorithms only concern what has happened empirically and factually rather than the "what-if" scenarios. One popular approach is causal graphs [27]. However, there are numerous variables and confounders in today's complex recommender system that makes such causal graph design non-trivial.

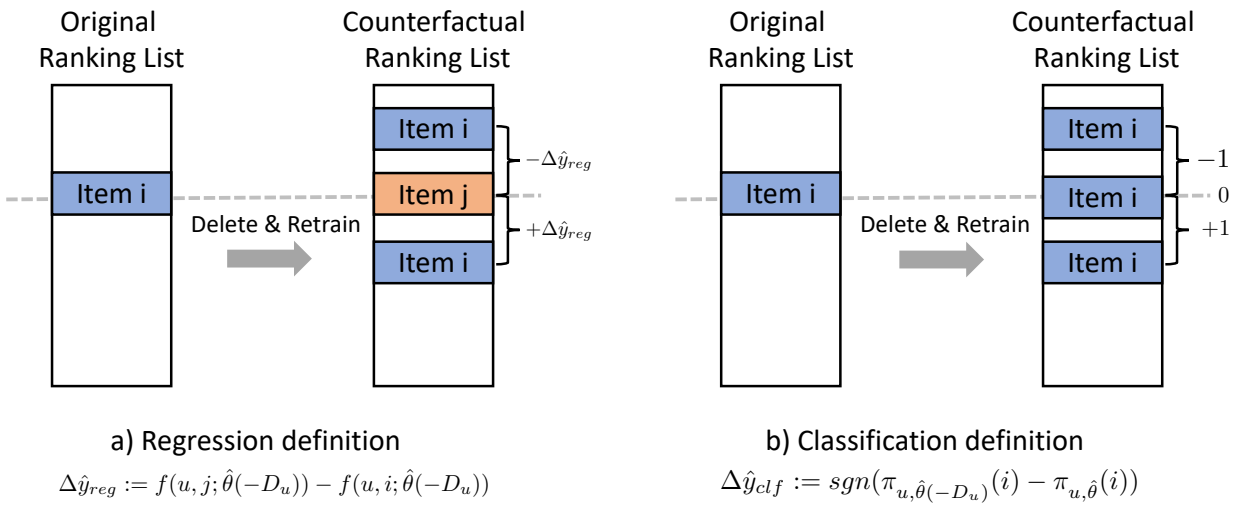
Insights. The key insight is to turn the problem into a learning problem. To generate counterfactual explanations, all we need to know is *how model outputs would change after removing a subset of user history in the training set.* Specifically, given a training subset is removed, we can directly predict its causal effect on the model outputs. If we can build a *surrogate model* to predict such causal effects, then by querying the surrogate model, we can estimate the change of model outputs on the recommended item for a particular user when removing an arbitrary subset of his or her history. Hence our search would be much faster by replacing every expensive model retraining with cheap surrogate model inference. In addition, the whole process speeds up by not having to retrain every time to explain to every user every possible recommended item. Instead, we can only retrain on a subset of all possible deletions to generate enough surrogate model's training data, and then rely on the surrogate model's generalizability to unseen deletions.

4.2 Methodology Overview

Our method has three main steps as the following:

1. **Simulating Counterfactual Outcomes**: We empirically obtain *ground-truth* of counterfactual outcomes. We sample a group of training subsets, remove them from the training set, and retrain the recommendation model to get the counterfactual models. We then record counterfactual models' prediction outputs. This step generates the training data for the surrogate model.

2. **Training Surrogate Models**: Given the empirically simulated data that describes the mapping between which part of the user history is removed and how the model's predicted output would change consequently, we design and train surrogate models to learn the mapping.



a) Regression definition

$\Delta\hat{y}_{reg} := f(u, j; \hat{\theta}(-D_u)) - f(u, i; \hat{\theta}(-D_u))$

b) Classification definition

$\Delta\hat{y}_{clf} := sgn(\pi_{u,\hat{\theta}(-D_u)}(i) - \pi_{u,\hat{\theta}}(i))$

Figure 2: Two definitions of model output change on item i caused by history deletion, *i.e.* the causal effect that the surrogate models predict. (a): Regression definition: predicted score difference between item i and the item ranked at item i's original position in the counterfactual ranking list (item j in the figure). (b): Classification definition: Direction of rank change (up, down, or unchanged) on item i.

3. **Generating Counterfactual Explanations**: Given a trained surrogate model, we generate the counterfactual explanations by searching for the subset of user history predicted by the surrogate model that, if were deleted, would have the maximal chance of removing the recommended item from the user's recommendation list.

Advantages. *First*, compared to the naive approach, our method requires less retraining by relying on the surrogate model's generalizability. We only need to repeat retraining a certain number of times that are enough to generate sufficient training data for the surrogate model. Then we can rely on surrogate models to generalize to unseen deletions (including unseen users and items). Note that the most computationally expensive step is the first step that simulates counterfactual outcomes, because it still needs to repeatedly retrain the model although the number of retraining times required is much smaller. However this step is perfectly parallelizable and can be performed offline. In other words, we can shift the major computational burden from online to offline. *Second*, our method is applicable to any recommendation models because we only need to know how the recommender model's inputs and outputs change regardless of what the model does within itself, *i.e.* treating the model as a black-box.

5 Methodology: Details

In this section, we explain the three steps to generate counterfactual explanations in detail.

5.1 Step 1: Simulating Counterfactual Outcomes

We first define the change in model outputs, denoted as $\Delta\hat{y}$. It is what we measure on counterfactual models after retraining to build the dataset that describes the counterfactual outcomes. It is also

later what surrogate models predict. Specifically, $\Delta\hat{y}$ describes the following: for a given user u, if we were to remove an arbitrary subset D_u from his or her history, how would the model prediction on item i change consequently?

We propose two definitions of model output change on item i. The first definition focuses on the difference between predicted scores, and therefore $\Delta\hat{y}$ is a numerical value, and the resulting surrogate model is a regression model. The second definition focuses on the direction of rank change (up, down, or unchanged) which is a discrete class, and therefore the corresponding surrogate model is a classification model.

Regression Definition of $\Delta\hat{y}$. One straightforward way to define counterfactual change on the target item i's predicted score is to compare the item i's counterfactual score with its original score. However the comparison is unfair because two different ranking lists have different score ranges. Instead we find an item as the benchmark that represents item i's original position in the counterfactual list, *i.e.* the item that takes item i's original position in the counterfactual list, illustrated in Figure 2 (a). If item i is ranked at the k-th position in the original ranking list, then we look at the item ranked at the k-th position in the counterfactual ranking list as the benchmark. And we compute the score difference in the counterfactual ranking list between item i and the benchmark item. If item i's score is lower than the benchmark item's, then it means it is now ranked lower by the counterfactual model than the original model, and therefore item i is more likely to be removed from the recommendation in the counterfactual world.

Formally, recall that $\pi_{u,\hat{\theta}}$ is the original ranking list of the items for user u by the model $\hat{\theta}$; $\pi_{u,\hat{\theta}}(i)$ is the rank of item i in the original ranking list $\pi_{u,\hat{\theta}}$, $\pi_{u,\hat{\theta}(-D_u)}$ is the counterfactual ranking list generated by the counterfactual model $\hat{\theta}(-D_u)$. Further define $\pi^{-1}_{u,\hat{\theta}(-D_u)}(k)$ to be the item at k-th rank in the counterfactual ranking list $\pi_{u,\hat{\theta}(-D_u)}$, the regression definition of model output change on item i for user u after removing D_u is:

$$\begin{aligned} &\Delta\hat{y}_{reg}(u, D_u, i) := f\Big(u, j; \hat{\theta}(-D_u)\Big) - f\Big(u, i; \hat{\theta}(-D_u)\Big), \\ &\text{where } j := \pi^{-1}_{u,\hat{\theta}(-D_u)}(\pi_{u,\hat{\theta}}(i)) \end{aligned} \tag{3}$$

If the predicted score is normalized into $[0, 1]$, then the range of $\Delta\hat{y}_{reg}(.)$ is $[-1, 1]$. In addition, if $\Delta\hat{y}_{reg}$ is positive, it means item i is now ranked lower than its original position after removing D_u; if it is negative, it means higher; if zero, it means the rank does not change. Therefore, a larger $\Delta\hat{y}_{reg}$ means, in the counterfactual case, the model is less likely to recommend i for u, and therefore item i is more likely to disappear from u's recommendation list after removing D_u. Hence we want to maximize $\Delta\hat{y}_{reg}$ when generating counterfactual explanations.

Classification Definition of $\Delta\hat{y}$. An alternative definition is to look at the direction of rank change (up, down, or unchanged) in item i. In this case, we only compare item i's position with itself between the original and counterfactual ranking list, shown in Figure 2(b). The resulting change is a qualitative class rather than a quantitative score:

$$\Delta\hat{y}_{clf}(u, D_u, i) := sgn\big(\pi_{u,\hat{\theta}(-D_u)}(i) - \pi_{u,\hat{\theta}}(i)\big) \tag{4}$$

Note that there might be some alternative ways to define model output change, see Section 6.3 for ablation studies.

Generating Surrogate Model's Training Set. We describe how to generate surrogate model's training set, summarized in Algorithm 2. We first sample a subset from all training users. Then for each user u, we sample a group of subsets from user u's history. For each history subset, we remove it from the full training dataset and retrain the recommendation model to get the counterfactual

Algorithm 2 Counterfactual Data Generation.

Input: Recommendation model $\hat{\theta}$ and its training dataset $\mathcal{S}$, rank threshold N that determines recommendation.
Output: Counterfactual training dataset $\mathcal{D}_{cf}$.
$\mathcal{D}_{cf} \leftarrow \emptyset$.
Sample a subset of users $\mathcal{U}^{tr}$ from $\mathcal{U}$ in $\mathcal{S}$.
for $u \in \mathcal{U}^{tr}$ **do**
    Sample a group of history subsets $\mathcal{D}_u^{tr}$ from u's history I_u.
    for $D_u \in \mathcal{D}_u^{tr}$ **do**
        Retrain to obtain $\hat{\theta}(-D_u)$ with eq(1).
        for $i \in \pi_{u,\hat{\theta}}[:N]$ **do** // Top-N items by $\hat{\theta}$.
            $x_{cf} \leftarrow (u, D_u, i)$.
            $y_{cf} \leftarrow \Delta\hat{y}(u, D_u, i)$. // By eq(3) or (4) with $\hat{\theta}(-D_u)$.
            $\mathcal{D}_{cf} \leftarrow \mathcal{D}_{cf} \cup \{(x_{cf}, y_{cf})\}$.
        end for
    end for
end for
Return $\mathcal{D}_{cf}$.

	# of Users	# of Items	# of Interactions
MovieLens-100K [28]	943	1682	100,000
Netflix (Small) [29]	10,000	5,000	607,803
Amazon Music [29]	5,541	3,568	64,706

Table 1: Datasets used in our experiments.

model. After it, we generate the counterfactual ranking list for user u based on the counterfactual model. Given the counterfactual list, we need to decide which item's change $\Delta\hat{y}$ to include in the surrogate model's training set. Technically, we can include all items in the system (recall the model reusing trick, for the same user, we do not need to retrain a different model to get the score change on a new item) but it is unnecessary because eventually we only need to explain items recommended to the user. Therefore we only need to include items likely to be recommended, *i.e.* top items. Assume top-N items are shown to the user, then we only need to compute the score change on them as the training set.

5.2 Step 2: Training Surrogate Models

$\Delta\hat{y}$ is a complex function that involves *training and evaluating the recommendation model on different subsets*, and therefore cannot be directly formulated and obtained. The goal of the surrogate model is to empirically approximate $\Delta\hat{y}$ in an end-to-end manner. The input space of the surrogate model is large if we consider learning the effect on every item caused by every user's every possible deletion of his or her interactions. Recall $\mathcal{S}$ is the interactions in the training set, the entire input space of the surrogate model would be $2^{|\mathcal{S}|} \times |\mathcal{I}| \times |\mathcal{U}|$. And therefore it would be difficult to generate enough training instances to learn the mapping well.

User Independence Assumption. Recall that when we define the counterfactual explanation (Figure 1), we implicitly assume that the user u's recommendation is only impacted by u's own history rather than other users' history. This is also known as *Stable Unit Treatment Value Assumption* in the causal inference literature [30]. This is a design choice because it is easier for users to understand. Consider the following explanation that might more faithfully reflect the model's reasoning: "Had some other strangers that you do not know interacted with those items, then we would not recommend it." This kind of explanation would confuse users. By assuming u's recommendation is only dependent on u's own interaction history rather than all users, we can reduce the sampling space of deletion from $2^{|\mathcal{S}|}$ to $2^{\overline{|I_u|}}$ where $\overline{|I_u|}$ is the average number of items users interacted in the training set. Therefore the form of regression surrogate model is:

$$\alpha_{reg} : \mathcal{U} \times 2^{\overline{|I_u|}} \times \mathcal{I} \to \mathbb{R} \tag{5}$$

And the form of classification surrogate model is:

$$\alpha_{clf} : \mathcal{U} \times 2^{\overline{|I_u|}} \times \mathcal{I} \to \{-1, 0, 1\} \tag{6}$$

We formalize the training process of the surrogate model that empirically approximates $\Delta\hat{y}$. Let $\mathcal{D}_{cf}$ be the training data collected in Algorithm 2 from the previous step, *i.e.* $\mathcal{X}_{cf} := \{x_{tr} : (x_{tr}, \cdot) \in \mathcal{D}_{cf}\}$ be the set of training inputs (collected in Line 10 Algorithm 2), $\mathcal{U}^{tr} := \{u \in \mathcal{U} : (u, \cdot, \cdot) \in \mathcal{X}_{cf}\}$ be the training users collected, $\mathcal{D}_u^{tr} := \{D_u \in 2^{I_u} : (u, D_u, \cdot) \in \mathcal{X}_{cf}\}$ be the sampled deleted subsets of user u's history and $\mathcal{I}_{u,D_u}^{tr} := \{i \in \mathcal{I} : (u, D_u, i) \in \mathcal{X}_{cf}\}$ be top-N items whose $\Delta\hat{y}$ we collect for user u's deletion D_u. Then the surrogate model can be trained with the following Empirical Risk Minimization:

$$\hat{\alpha} := \arg\min_{\alpha \in \mathrm{A}} \sum_{u \in \mathcal{U}^{tr}} \sum_{D_u \in \mathcal{D}_u^{tr}} \sum_{i \in \mathcal{I}_{u,D_u}^{tr}} \ell_\alpha\big(g(u, D_u, i; \alpha), \Delta\hat{y}(u, D_u, i)\big) \tag{7}$$

where $g(.;\alpha)$ is the surrogate model α's prediction; $\ell_\alpha(.)$ is the loss of the surrogate model, which is MSE for the regression and cross-entropy for the classification; $\Delta\hat{y}(u, D_u, i)$ is the corresponding ground-truth value of $\Delta\hat{y}$ collected.

Input Representation. An important design choice is how to represent the surrogate model's three inputs u (target user), D_u (u's deleted history), and i (target item). To represent u and i, we simply use the recommendation model's generated embeddings (or any user/item embeddings that exist in the recommender system). It is a common practice in recommender system because the learned embeddings usually can represent user and item information well. In terms of deleted items (D_u) which is an item set, we use a simple heuristic: the sum of the embedding of the deleted items. Formally, let $\phi_u(u) \in \mathbb{R}^p$ be the user embedding of user u, and $\phi_i(i) \in \mathbb{R}^q$ be the item embedding of item i, we simply concatenate those three features into a single feature vector as the surrogate model's training input as the following:

$$\big[\phi_u(u) \quad \sum_{\tilde{i} \in D_u} \phi_i(\tilde{i}) \quad \phi_i(i)\big] \tag{8}$$

We experiment with different representations of D_u, for example DeepSets [31] (one can view this sum of embeddings as DeepSets with identity mapping), and find this simple method works well. See Section 6.3 for the related ablation studies.

Surrogate Model Choice. Technically speaking, the surrogate model can be any regression or classification model. We choose LASSO [32] and a simple 3-layer MLP for regression and logistic regression and the same MLP architecture for classification. We empirically find that those simple models outperform deep and complex models (in terms of the generated explanation's ability to satisfy the counterfactual definition). See Section 6.3 for related ablation studies.

	KNN	Influence Function	Linear Surrogate (reg)	MLP Surrogate (reg)
Time (sec)	$1.25 \times 1e^{-3}$	6.47	1.75	2.60

Table 2: Average generation time (seconds) per explanation on *MovieLens* with Matrix Factorization model.

	Matrix Factorization									Neural Collaborative Filtering								
	MovieLens			Netflix			Amazon			MovieLens			Netflix			Amazon		
	K = 1	K = 3	K = 5	K = 1	K = 3	K = 5	K = 1	K = 3	K = 5	K = 1	K = 3	K = 5	K = 1	K = 3	K = 5	K = 1	K = 3	K = 5
KNN	17%	10%	3%	30%	18%	12%	18.5%	8%	5.5%	42%	27%	21.5%	38.5%	26%	24%	40%	28%	21%
Influence Function	35.5%	15.5%	9.5%	37%	18%	10.5%	37.5%	19.5%	11.5%	44%	29.5%	23.5%	40.5%	28.5%	24%	41%	29%	23%
Linear Surrogate (reg)	**82%**	**68%**	**54.5%**	**89%**	**73.5%**	**62%**	**81%**	**57%**	**44%**	**86%**	**60%**	**44%**	**68.5%**	**41.5%**	**32%**	**74.5%**	**54%**	**40.5%**
MLP Surrogate (reg)	76.5%	52.5%	42.5%	80.5%	50.5%	40.5%	77%	54%	**44%**	83%	57.5%	42%	**68.5%**	39.5%	30%	71%	52%	38.5%
Linear Surrogate (clf)	79%	61%	51%	85%	68.5%	57.5%	71%	50.5%	40.5%	81%	53.5%	38.5%	61.5%	36.5%	32%	72%	49.5%	38.5%
MLP Surrogate (clf)	72.5%	48%	41%	76%	61%	51%	68%	48.5%	38.5%	75.5%	49.5%	36.5%	63.5%	35%	27.5%	68.5%	48.5%	36%

Table 3: Percentage of recommended items that are removed from the top-K (K=1,3,5) recommendation list if the generated explanations were actually deleted and the model were retrained.

5.3 Step 3: Generating Counterfactual Explanations

Fixed Explanation Size. Conventionally in the counterfactual explanation literature, the explanation size should be as small as possible. However, we argue that, in the context of recommender system, this is not only unnecessary but might even be harmful. Instead, it would be better to have fixed-size explanations. It is because displaying an uncontrollable number of items when explaining to users would mess up the user interface and make users feel overwhelmed and incomprehensible if showing too many items. In fact, a fixed explanation size is the common design choice of the majority of *item-based collaborative filtering* explaining logic [9, 10].

Given a fixed explanation size e, the trained surrogate model $\hat{\alpha}$, target user u and target item i, to generate an explanation we first randomly sample a fraction of all possible e-size subsets in u's history I_u, and then search for the history subset with maximal [2] predicted $\Delta\hat{y}$ as the generated explanation, *i.e.*

$$E^*_{u,i} = \underset{E \sim 2^{I_u}; |E|=e}{\arg\max} \; g(u, E, i; \hat{\alpha}) \tag{9}$$

Ideally one can improve from this simple subset searching by optimization and approximation techniques. In practice, we find this step is fast enough (See Section 6.2) because our surrogate model is simple and therefore fast in making inferences. In addition, thanks to the fixed explanation size design, the sampling space is much smaller compared to finding the minimal size . We experiment with some approximation techniques, and find they not only do not significantly speed up the process but also lead to worse performance on the generated explanations. In practice, if the sampling size is too large, we can cap it in the experiments. See Section 6.1 for details.

6 Experiments

In this section, we show the experimental results that evaluate the proposed method.

[2]Recall that in the regression definition, larger predicted $\Delta\hat{y}$ means target item i is more likely to be ranked lower than the item ranked at its original position, and therefore more likely to satisfy the counterfactual definition. In the classification definition, larger predicted $\Delta\hat{y}$ (which is the classification surrogate model's logit) means target item i's rank is more likely to be larger than its original rank, and therefore is more likely to disappear from the recommendation.

6.1 Experimental Setup

Dataset and Recommendation Model. We use three datasets summarized in Table 1. For each dataset, we experiment with two recommendation models: Matrix Factorization (SVD) and Neural Collaborative Filtering [33]. We randomly split all datasets into 70% training and 30% test set.

Explanation Generation. In *Simulating Counterfactual Outcomes* step, we randomly sample 200, 100, and 80 users from *MovieLens*, *Netflix*, and *Amazon* respectively as surrogate model's training users. For each training user u, we randomly remove $1,000$ 3-item subsets from u's history (all deleted history subsets have the same size) and retrain the model to obtain counterfactual models. The resulting number of retrained counterfactual models is $200K$, $100K$, and $80K$ for *MovieLens*, *Netflix*, and *Amazon* respectively. For each counterfactual model, we generate $\Delta\hat{y}$ on the top-20 (*i.e.* $N = 20$ in Algorithm 2) recommended items for the target user. In *Training Surrogate Model* step, when we train LASSO for the regression model and logistic regression for the classification model, we use k-folder cross validation to tune the hyper-parameters. In *Generating Counterfactual Explanations* step, we choose the explanation size to be the same as the size of each deleted history subset in *Simulating Counterfactual Outcomes* stage, which is 3 items (*i.e.* $e = 3$ in eq (9), see Section 6.3 for related ablation study), and we randomly sample at most $100K$ explanation candidates to search for the one with maximum surrogate model prediction.

Evaluation Metric. As mentioned in Section 5.3, we argue fixed-size explanations might be more beneficial from the perspective of implementation and user comprehension. Therefore aiming for a small explanation size is out of our evaluation scope, and we focus on the counterfactual validity, which means if the explanations were indeed removed from the training set, the user actually would not receive the recommendation. Therefore we judge a counterfactual explanation is valid if it removes the recommended item from the user's recommendation list, *i.e.* eq (2). How many items are shown to users in a recommendation list is decided by the design of the recommender system. For example, if a recommender system shows the top-K items to users, then an explanation is valid if the current item would fall out of the top-K ranking list. We evaluate different choices of K.

In all experiments, we sample 200 test users disjointly from the training users included in the surrogate model's training set. For each user u, we choose the target item that we explain as u's top-1 item. Then we remove the generated explanation w.r.t to the target item from the user's history I_u, and retrain the model on the deleted training set to obtain the ground-truth counterfactual model for evaluation. After it, we look at the ground-truth counterfactual model's top-K recommendation on the target user to see if it still includes the target item or not. All the explanations generated by our method or baselines have size 3 items.

Baseline. We consider two baselines. The first is *k-nearest neighbors* (KNN) which searches for the items in the history that are the nearest neighbors of the target item in the item embedding space. The second is *Influence Function*, which is a generic explaining approach that can be used to estimate the impact of removing a training point on a test point without fully retraining the model. The influence function in [5] is defined on the loss of the test point, which is, in our case, the target item i. However its ground-truth rating is unavailable, and therefore we cannot compute the model's loss on it. Similar to [34, 6], we adapt the influence function's definition for the recommendation by defining it on the test point's predicted score. For the target user u and the target item i, the impact of removing any training item $\tilde{i}$ is:

$$Infl(u, i, \tilde{i}) := \frac{1}{n} \nabla_\theta f(u, i; \hat{\theta})^\intercal H_{\hat{\theta}}^{-1} \nabla_\theta L(u, \tilde{i}; \hat{\theta}) \tag{10}$$

where n is the number of training samples, $H_{\hat{\theta}}$ is Hessian of the original model $\hat{\theta}$ and $L(u, \tilde{i}; \hat{\theta})$

is the loss on the training item $\tilde{i}$. We compute the influence function on all $\tilde{i} \in I_u$ and select items with maximal influence w.r.t the target item i (*i.e.* removing them would decrease the predicted score on i the most, as predicted by the Influence Function) as the generated explanation.

6.2 Main Results

Generation Time. We report the generation time of explanations in Table 2. Our method generates explanations faster than Influence Function (which is known to be slow in recommendation [34]) but slower than the simple KNN. In addition, using MLP as the surrogate model does not significantly slow down the generation because our MLP model architecture is simple (only 3 layers).

Results. Table 3 summarizes the performance of our generated explanations (both regression and classification surrogate model) along with baselines. The first observation is our surrogate methods, either regression or classification or linear model or MLP model, outperform the baselines (see next paragraph for more analysis). The second observation is the best performing method is the regression surrogate model. It has two implications. *First*, regression models outperform classification models. This is likely because $\Delta\hat{y}_{reg}$ is a numerical score that can preserve more fine-grained information about how the target item is impacted than the categorical $\Delta\hat{y}_{clf}$. *Second*, linear models outperform MLP models, in both regression and classification, which is consistent with the findings in [35] that also point out linear surrogate models can predict a complex model's predictions surprisingly well. See Section 6.3 for an ablation study on linear surrogate models.

Analysis of Baselines. We explain why the baselines do not perform well. What all methods aim to approximate ultimately is the decrease of target item i's score if the explanation were removed. An ideal distance metric should be positively correlated with the i's score decrease in order to serve as a valid counterfactual explanation. To validate to how much degree this is the case in KNN and influence function, we randomly sample 500 item sets from a user's history (on MovieLens dataset with Matrix Factorization model) as explanation candidates, and then measure the average item embedding distance and influence function distance between explanations and the recommended item, and compare them to the ground-truth score decrease of the recommended item after actually removing explanations and retraining the model. In Figure 3 (Appendix B), we can see neither metric is highly correlated with the ground-truth score decrease (with Pearson correlation 0.175 and 0.356 only), meaning they cannot serve as good proxies of measuring counterfactual impact on the recommended item, and therefore do not lead to valid counterfactual explanations. On the other hand, since our surrogate model is trained on the *ground-truth* score change obtained by *actual* deletions and retraining, it serves as a more accurate counterfactual measure. This shows obtaining exact ground-truth change on model outputs through retraining is vital to counterfactual validity, even though it is costly.

6.3 Ablation Studies

To better understand some of our design choices, we show a series of ablation studies. All the following experiments are conducted on MovieLens dataset with Matrix Factorization model.

Definition of $\Delta\hat{y}$. There could be some alternative ways to define the model output change other than the definition used in eq(3) and (4). One alternative is the rank difference on the target item i between the original and the counterfactual list (which leads to a regression model that predicts an integer value). See Appendix A for the full definition. We find this definition leads to a slightly lower performance than the regression linear surrogate model-the performance when K=1 would

	KNN	Influence Function	Linear Surrogate (reg)	MLP Surrogate (reg)
Avg. Rating	3.586 ± 0.74	3.658 ± 0.67	**3.803 ± 0.87**	3.789 ± 0.89

Table 4: Average rating on explanations from the user study. Higher ratings mean more satisfaction. One-sided paired t-test shows explanations generated by the linear surrogate method have higher ratings than KNN, Influence Function, and the MLP surrogate with p-value 0.011, 0.034, and 0.393 respectively.

decrease by 1.5%. This is because the rank difference is an integer, which is less fine-grained than the float score in $\Delta\hat{y}_{reg}$.

Another alternative definition is the score difference of item i between the original and the counterfactual ranking list. Intuitively, if the score becomes lower in the counterfactual list, then one might think it signifies the satisfaction of the counterfactual definition. However, the difficulty of this definition lies in the fact that the original and the counterfactual ranking list have a different score range, and therefore the score difference might not represent the positional difference. One solution is to normalize the score of item i in two lists by the score of the top-1 item in two lists respectively (See Appendix A for the complete definition). Our ablation study shows this definition leads to a noticeably worse performance: 7.4% lower than the regression linear surrogate model's performance in *MovieLens* (K = 1). This is because merely normalizing scores does not compensate enough for the score range discrepancy between different ranking lists.

Deletion Size. In *Simulating Counterfactual Outcomes* step, we need to decide how many items in a history subset to delete when empirically simulating the effect of removal. We choose the removal size to be the same as the fixed explanation size in the final generation step. We experiment with the following alternative: if we remove 40% of a user's history instead of a fixed 3 items, the generated 3-item explanations show a slightly lower performance: 4.5% decrease on the regression linear surrogate model *MovieLens* (K = 1). The number of items deleted in the generation stage has a certain impact on the performance though the impact is not significant.

Representation of D_u. We experimented with more complex set represents of D_u, *i.e.* DeepSets [31], and find it leads to 11.5% decrease in the final performance in the regression and linear surrogate model (K = 1). This might suggest that the simple design of input representation is better probably because the surrogate model design is simple.

Analysis of Linear Surrogate Models. In addition to linear and MLP surrogate models, we experimented with deeper models and more complex input representation, *e.g.* transformer, attention, mask representation of deleted items *etc.*But eventually we find the simple surrogate models work the best. One observation from Table 3 is the linear models outperform MLP models by a larger margin in Matrix Factorization (5.5–11% in regression when $K = 1$) than in Neural Collaborative Filtering (0%–3%). One hypothesis is Matrix Factorization is a (bi)linear model, and therefore its input-output relationship might be more linear, and therefore easier for the linear surrogate model to learn. To verify this hypothesis, we visualize the surrogate model's training data (on MovieLens with Matrix Factorization using $\Delta\hat{y}_{reg}$ definition) via t-SNE [36] (Figure 4 in Appendix C). However we do not observe a clear linear separability. It remains unclear why linear models can predict a complex model's predictions surprisingly well, a phenomenon also pointed out by [35] as an open problem.

7 User Study

To examine the quality of our explanations perceived by the users, we carry out a survey-based user study.

Survey Design. Our survey contains 5 questions. In each question, we first show participants a list of movies that a user from MovieLens dataset had watched in the past, and then ask participants to imagine being that user. Next we present the top-1 recommended movie by the Matrix Factorization model. Afterward, we show, in random order, four different explanations: KNN, Influence Function, Linear Surrogate (regression), and MLP surrogate (regression). For KNN and Influence Function, the explanation text is "We recommend this movie because you watched the following movies." For linear and MLP surrogate explanations, the explanation text is "Have you not watched the following movies, it is likely we would not recommend this movie." Finally we ask participants to rate their satisfaction on each explanation ("Given a scale from 1 to 5, how much are you satisfied with this explanation?").

Results. We collect survey results from 100 users on *Amazon Mechanical Turk*. See Appendix D for details about our quality control and result collecting. Table 4 shows the average rating on four types of explanations over all questions. *First*, explanations generated by the linear surrogate (regression) model have the highest ratings. *Second*, one-sided paired t-tests show explanation ratings on the linear surrogate method are higher than KNN and Influence Function with statistical significance (with p-value 0.011 and 0.034 respectively), but no significance when compared to the MLP surrogate's explanations (with p-value 0.393).

8 Related Work

Counterfactual Explanation. In classification task, Wachter *et al.* [37] use white-box optimization to find the smallest change on an input's features that can alter the model prediction. Mothilal *et al.* [38] add a diversity loss into the optimization that maximizes the diversity between explanations. Relatively fewer works consider counterfactual explanations in recommender system. Ghazimatin *et al.* [39] consider graph recommender models that encode user-item interactions. They propose a Personal PageRank based approach that searches for the minimal change on the graph. However, this approach is only applicable to PageRank-based recommendation models. In addition, Tran *et al.* [6] propose an influence function based approach that estimates the influence of a training sample on the current model prediction. Furthermore, Kaffes *et al.* [40] propose a black-box solution that performs Breadth First Search with heuristics that combine search length and drop of item rank.

Predicting Model Predictions. The essential goal of the surrogate model is to predict the recommendation model's predictions. There are a few works that start to study predicting model predictions recently. Ilya *et al.* [35] use a linear surrogate model to predict a neural network's predictions when a subset of the neural network's training samples are removed. They also find the simple linear model works well and the reason is unclear. Saunshi *et al.* [41] suggest a linear model works might be because it can approximate functions related to test error well. In addition, our work differs fundamentally from instance-level explaining techniques using local surrogate models like LIME [42] and SHAP [43]. They focus on a fixed trained model without considering the counterfactual impact of changing the training samples.

Machine Unlearning. Machine unlearning studies how to delete training samples from a model without retraining the model from scratch, which is related to how to quickly obtain the counterfactual model. Bourtoule *et al.* [44] propose an exact unlearning framework, which replaces

the original model with an ensemble model with each sub-model is trained on a data subset. Then removing a training sample is done by discarding its sub-model. In addition, Golatkar *et al.* [45] use information theory methods to remove information from the trained model weights. Furthermore, Golatkar *et al.* [46] assume there is a core data subset that is not removed, and train a backbone model on it. Then the model for the non-core data is a linear approximation model which is fast to unlearn. Researchers also propose to use caching information during training [47], or design unlearning methods for a particular class of models like tree-based [48].

9 Conclusion and Future Work

We propose a learning-based framework to generate counterfactual explanations in a recommender system. Our key insight is simple: by training a surrogate model to predict the effect on the recommendation caused by deleting a history subset, we can estimate the counterfactual impact that a deletion would have on a recommendation. And then we can generate the counterfactual explanations by searching for the deletion that is most likely to remove the recommended item from the recommendation list. We hope our work can inspire more research on counterfactual explanations, which might be appealing as an alternative way to explain recommendations because it provides an explicitly stated and verifiable logic.

We point out two limitations and future work. *First*, in *Simulating Counterfactual Outcomes* step, although we can parallelize retraining, we still need to retrain a large number of times to generate enough ground-truth data to train surrogate models. One future work is to accelerate this step by inexactly approximating retraining. *Second*, we do not have a clear understanding of why linear surrogate models can generalize well when predicting model predictions, which is an open problem also pointed out by [35]. One future work is to test surrogate models that are middle grounds between linear and MLP model, *e.g.* kernel ridge regression.

References

[1] Daniel Kahneman and Amos Tversky. The simulation heuristic. Technical report, Stanford Univ CA Dept of Psychology, 1981.

[2] David R Mandel, Denis J Hilton, and Patrizia Ed Catellani. *The psychology of counterfactual thinking.* Routledge, 2005.

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

[4] Barbara A Spellman and David R Mandel. When possibility informs reality: Counterfactual thinking as a cue to causality. *Current Directions in Psychological Science*, 8(4):120–123, 1999.

[5] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proc. of ICML*, 2017.

[6] Khanh Hiep Tran, Azin Ghazimatin, and Rishiraj Saha Roy. Counterfactual explanations for neural recommenders. In *Proc. of SIGIR*, 2021.

[7] Samyadeep Basu, Philip Pope, and Soheil Feizi. Influence functions in deep learning are fragile. In *Proc. of ICLR*, 2021.

[8] Yongfeng Zhang and Xu Chen. Explainable recommendation: A survey and new perspectives. *arXiv preprint arXiv:1804.11192*, 2018.

[9] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proc. of CSCW*, 1994.

[10] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *Proc. of CSCW*, 2000.

[11] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proc. of TheWebConf*, 2001.

[12] Michael J Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial intelligence review*, 13(5):393–408, 1999.

[13] Xin Wayne Zhao, Yanwei Guo, Yulan He, Han Jiang, Yuexin Wu, and Xiaoming Li. We know what you want to buy: a demographic-based system for product recommendation on microblogs. In *Proc. of KDD*, 2014.

[14] Wayne Xin Zhao, Sui Li, Yulan He, Liwei Wang, Ji-Rong Wen, and Xiaoming Li. Exploring demographic information in social media for product recommendation. *Knowledge and Information Systems*, 49(1):61–89, 2016.

[15] Jesse Vig, Shilad Sen, and John Riedl. Tagsplanations: explaining recommendations using tags. In *Proc. of IUI*, 2009.

[16] Henriette Cramer, Vanessa Evers, Satyan Ramlal, Maarten Van Someren, Lloyd Rutledge, Natalia Stash, Lora Aroyo, and Bob Wielinga. The effects of transparency on trust in and acceptance of a content-based art recommender. *User Modeling and User-adapted interaction*, 18(5):455, 2008.

[17] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[18] Yunfeng Hou, Ning Yang, Yi Wu, and S Yu Philip. Explainable recommendation with fusion of aspect information. *World Wide Web*, 22(1):221–240, 2019.

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

[20] Yao Wu and Martin Ester. Flame: A probabilistic model combining aspect based opinion mining and collaborative filtering. In *Proc. of WSDM*, 2015.

[21] Felipe Costa, Sixun Ouyang, Peter Dolog, and Aonghus Lawlor. Automatic generation of natural language explanations. In *Proc. of IUI*, 2018.

[22] Shuo Chang, F Maxwell Harper, and Loren Gilbert Terveen. Crowd-based personalized natural language explanations for recommendations. In *Proc. of RecSys*, 2016.

[23] Nan Wang, Hongning Wang, Yiling Jia, and Yue Yin. Explainable recommendation via multi-task learning in opinionated text data. In *Proc. of SIGIR*, 2018.

[24] Amit Sharma and Dan Cosley. Do social explanations work? studying and modeling the effects of social explanations in recommender systems. In *Proc. of TheWebConf*, 2013.

[25] Allison JB Chaney, David M Blei, and Tina Eliassi-Rad. A probabilistic model for using social networks in personalized item recommendation. In *Proc. of RecSys*, 2015.

[26] Beidou Wang, Martin Ester, Jiajun Bu, and Deng Cai. Who also likes it? generating the most persuasive social explanations in recommender systems. In *Proc. of AAAI*, 2014.

[27] Judea Pearl. *Causality*. Cambridge university press, 2009.

[28] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems*, 5(4):1–19, 2015.

[29] Cornac datasets. `https://github.com/PreferredAI/cornac/tree/master/cornac/datasets`, Accessed by 2022.

[30] Joshua D Angrist, Guido W Imbens, and Donald B Rubin. Identification of causal effects using instrumental variables. *Journal of the American statistical Association*, 91(434):444–455, 1996.

[31] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Proc. of NeurIPS*, 2017.

[32] Robert Tibshirani. Regression shrinkage and selection via the lasso: a retrospective. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(3):273–282, 2011.

[33] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proc. of TheWebConf*, 2017.

[34] Weiyu Cheng, Yanyan Shen, Yanmin Zhu, and Linpeng Huang. Explaining latent factor models for recommendation with influence functions. In *Proc. of KDD*, 2019.

[35] Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. In *Proc. of ICML*, 2022.

[36] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[37] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.

[38] Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proc. of FAccT*, 2020.

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

[40] Vassilis Kaffes, Dimitris Sacharidis, and Giorgos Giannopoulos. Model-agnostic counterfactual explanations of recommendations. In *Proc. of UMAP*, 2021.

[41] Nikunj Saunshi, Arushi Gupta, Mark Braverman, and Sanjeev Arora. Understanding influence functions and datamodels via harmonic analysis. *arXiv preprint arXiv:2210.01072*, 2022.

[42] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you? explaining the predictions of any classifier. In *Proc. of KDD*, 2016.

[43] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proc. of NeurIPS*, 2017.

[44] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *Proc. of IEEE S & P*, 2021.

[45] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proc. of CVPR*, 2020.

[46] Aditya Golatkar, Alessandro Achille, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. Mixed-privacy forgetting in deep networks. In *Proc. of CVPR*, 2021.

[47] Yinjun Wu, Edgar Dobriban, and Susan Davidson. Deltagrad: Rapid retraining of machine learning models. In *Proc. of ICML*, 2020.

[48] Jonathan Brophy and Daniel Lowd. Machine unlearning for random forests. In *Proc. of ICML*, 2021.

A Ablation Study: Definition of $\Delta\hat{y}$

Rank Difference Definition. Define $\pi_{u,\hat{\theta}}(i)$ to be the rank of the target item i in the ranking list produced by the original model $\hat{\theta}$ and $\pi_{u,\hat{\theta}(-D_u)}(i)$ to be the item i's rank in the counterfactual ranking list, then the rank difference definition of $\Delta\hat{y}$ is:

$$\Delta\hat{y}_{rank}(u, D_u, i) := \pi_{u,\hat{\theta}(-D_u)}(i) - \pi_{u,\hat{\theta}}(i) \tag{11}$$

If it is positive, it means item i is ranked lower counterfactually than the original.

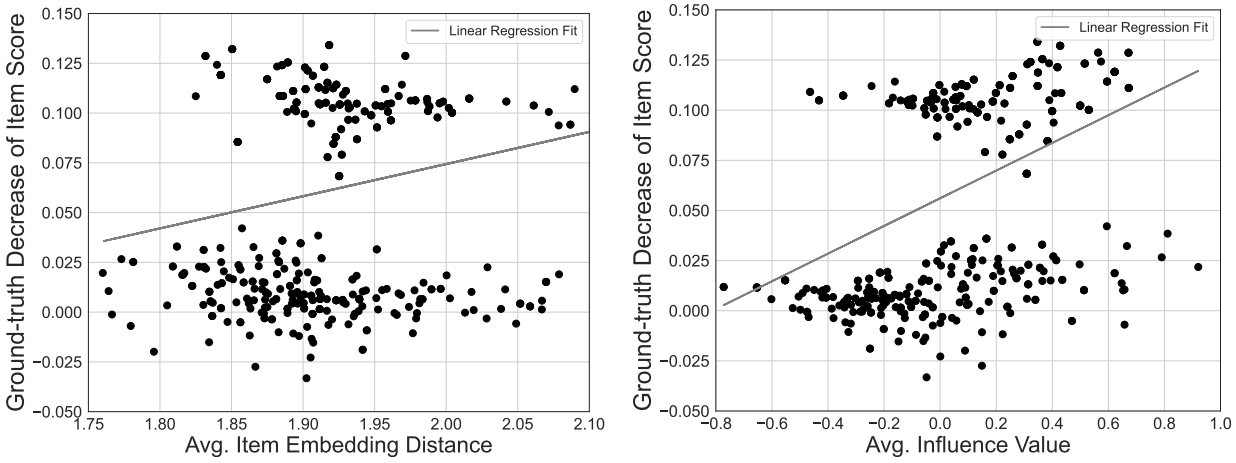
Cross Ranking List Definition. Recall that $f(u, i; \hat{\theta})$ is the predicted score on the target item i by the original model and $\pi^{-1}_{u,\hat{\theta}}(1)$ is the top-1 item in the original ranking list; $f(u, i; \hat{\theta}(-D_u))$ is the predicted score on the target item i by the counterfactual model and $\pi^{-1}_{u,\hat{\theta}(-D_u)}(1)$ is the top-1 item in the counterfactual ranking list, the cross ranking list definition of $\Delta\hat{y}$ is:

$$\Delta\hat{y}_{cross}(u, D_u, i) := \frac{f(u,i;\hat{\theta})}{f(u,\pi^{-1}_{u,\hat{\theta}}(1);\hat{\theta})} - \frac{f(u,i;\hat{\theta}(-D_u))}{f(u,\pi^{-1}_{u,\hat{\theta}(-D_u)}(1);\hat{\theta}(-D_u))} \tag{12}$$

If it is positive, it means the normalized score in the counterfactual list of item i is smaller than the original, and therefore it is likely that the position will be lower in the counterfactual case.

B Performance Analysis of Baselines

In Figure 3, we show the relationship between the distance measured by KNN and Influence Function on explanations and the ground-truth score decrease of the recommendation if we actually delete explanations from the training set and retrain the model. As we can see, neither KNN nor Influence Function is highly correlated with the ground-truth score decrease with Pearson correlation 0.175 and 0.356 only.



(a) Item embedding distance.

(b) Influence function distance.

Figure 3: (a): Average item embedding distance between an explanation and the recommended item vs. the ground-truth decrease of the recommended item's score if the explanation were actually removed and the model were retrained. (b): Average influence function value of an explanation vs. the ground-truth score decrease. Overall, neither distance measure is highly correlated with the ground-truth score decrease with Pearson correlation 0.175 and 0.356 only for item embedding distance and influence function respectively.

C Ablation Study: Visualization of Surrogate Model's Training Inputs

In Figure 4, we show t-SNE visualization of the training data of the surrogate model (on MovieLens data with Matrix Factorization model, with $\Delta\hat{y}_{reg}$ definition) in 2-D space. One hypothesis is because the recommendation model is Matrix Factorization, a (bi)linear model, its input-output relationship is linear as well, and therefore easier for a linear model to learn. However from the visualization, we do not observe a clear linear separability of the surrogate model's training data.

D User Study: Details

We include more details of our user study regarding quality control and results collection.

Quality Control. We implement a series of mechanisms to ensure response quality. *First*, we only choose MTurk workers with at least 80% approval rate in participating history (HIT rate) and with at least 50 tasks approved in the past. *Second*, we declare the requirement of movie familiarity to the participants in the advertisement. *Third*, we insert a trivial task ("Please choose both A and D.") in the middle of the survey, and filter out any responses that fail to answer correctly. *Fourth*, we participants how many movies they usually watch in a month, and remove any participants who answer less than one movie per month. *Lastly*, we ask participants, at the end of the survey, to disclose their familiarity with the movies that they have seen in the survey ("What percentage of movies shown in this survey are familiar to you?"), and exclude participants who claim more than 40% of movies they see are unfamiliar.

Collection. For 100 responses we receive, we exclude 8 users who fail to pass the quality control tests. Of the remaining 92 participants, 60.9% indicated male and 39.1% indicated female. The

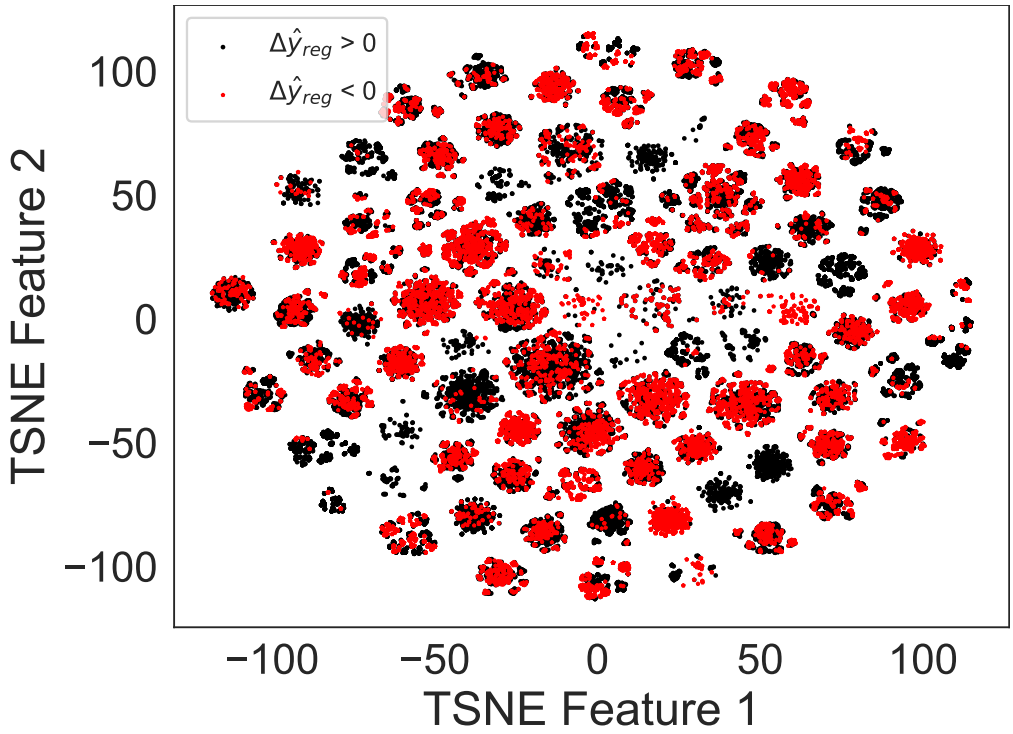


Figure 4: t-SNE visualization of surrogate model's training data (MovieLens data and Matrix Factorization model, with $\Delta\hat{y}_{reg}$ definition, on 100 training users) in 2-D space.

majority of participant ages fall into the ranges of 21 to 30 (38.0%) or 31 to 40 (33.7%), with 15.2% falling between 41 and 50, and the rest being older than 50.