The Dark Side of Explanations: Poisoning Recommender Systems with Counterfactual Examples

Ziheng Chen
ziheng.chen@stonybrook.edu
Stony Brook University, NY, USA

Fabrizio Silvestri
fsilvestri@diag.uniroma1.it
Sapienza University of Rome, Italy

Jia Wang
jia.wang02@xjtlu.edu.cn
The Xi'an Jiaotong-Liverpool University, Suzhou, China

Yongfeng Zhang
yongfeng.zhang@rutgers.edu
Rutgers University, NJ, USA

Gabriele Tolomei
tolomei@di.uniroma1.it
Sapienza University of Rome, Italy

ABSTRACT

Deep learning-based recommender systems have become an integral part of several online platforms. However, their black-box nature emphasizes the need for explainable artificial intelligence (XAI) approaches to provide human-understandable reasons why a specific item gets recommended to a given user. One such method is *counterfactual explanation* (CF). While CFs can be highly beneficial for users and system designers, malicious actors may also exploit these explanations to undermine the system's security.

In this work, we propose H-CARS, a novel strategy to poison recommender systems via CFs. Specifically, we first train a logical-reasoning-based surrogate model on training data derived from counterfactual explanations. By reversing the learning process of the recommendation model, we thus develop a proficient greedy algorithm to generate fabricated user profiles and their associated interaction records for the aforementioned surrogate model. Our experiments, which employ a well-known CF generation method and are conducted on two distinct datasets, show that H-CARS yields significant and successful attack performance.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → **Neural networks**.

KEYWORDS

Explainable recommender systems, Counterfactual explanations, Model poisoning attacks

ACM Reference Format:
Ziheng Chen, Fabrizio Silvestri, Jia Wang, Yongfeng Zhang, and Gabriele Tolomei. 2023. The Dark Side of Explanations: Poisoning Recommender Systems with Counterfactual Examples. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3539618.3592070

SIGIR '23, July 23–27, 2023, Taipei, Taiwan

ACM ISBN 978-1-4503-9408-6/23/07…$15.00
https://doi.org/10.1145/3539618.3592070

1 INTRODUCTION

The past few decades have witnessed the tremendous success of deep learning (DL) techniques in personalized recommender systems. Indeed, DL-based recommender systems overcome the obstacles of conventional models and achieve state-of-the-art performance [15, 27]. Despite that, they suffer from a lack of transparency and explainability. This issue may limit their deployment, especially in some critical domains, considering the increasing demand for explainable artificial intelligence (XAI) worldwide [4, 10, 12, 18, 19, 22, 24, 25].

Recently, *counterfactual explanation* (CF) has emerged as a key tool to attach motivation behind items recommended to users. Concretely, CFs generate a minimal set of meaningful interactions, without which the recommended items will not end up in the list of suggestions for specific users. While much prior research shows how counterfactual explanations can enhance recommender systems' transparency, few studies investigate their potential security and privacy threats. Instead, several studies have shown possible hazards of CFs for the classification task [1, 5, 14, 21, 32]. For example, Aïvodji et al. [1] demonstrate that a high-fidelity model could be extracted by detecting the information of decision boundaries embedded in CFs. This finding inspired DualCF [28], which leverages CFs and their counterfactual explanations to overcome decision boundary shift issues. Additionally, Pawelczyk et al. [21] conduct the membership inference attacks by using the distance between the data instances and their corresponding CFs.

This work investigates possible risks induced by CFs for the recommendation task. Specifically, we demonstrate how an adversary can use CFs to conduct poisoning attacks on black-box recommender systems. Technically, we first train a logical reasoning model as a surrogate by exploiting CFs. Then, a limited number of controlled users with fake crafted interactions are designed by matching the optimal representation of a target recommended item via an optimization framework. We refer to our attack strategy as H-CARS (Horn-Clause Attacks to Recommender Systems).

Overall, the main contributions of our work are as follows:

- We unveil security issues of applying counterfactual explanations in recommender systems and provide the first study of poisoning attacks for recommendation via CFs.
- We jointly model logical and counterfactual reasoning by leveraging CFs and partial factual interactions to enhance the surrogate model's performance. A counterfactual loss $\mathcal{L}_{cf}$ is

proposed to highlight the necessary items while mitigating spurious correlations.
- We propose a novel poisoning attack for neural logical-based recommender systems. Specifically, inspired by [30], we reverse the traditional attacking optimization procedure where, instead of re-training the recommendation model, we start from computing the optimal item embeddings, which we leverage to find the fake user-item interactions.
- We conduct experiments on two real datasets to analyze the attacking performance.

The remainder of this paper is organized as follows. We provide background and preliminaries in Section 2. In Section 3, we present the attack model and describe our proposed method H-CARS, in Section 4. We validate H-CARS in Section 5. Finally, we conclude our work in Section 6.

2 BACKGROUND AND PRELIMINARIES

We consider the standard collaborative filtering recommendation task. Let $\mathcal{U} = \{u_1, \ldots, u_m\}$ be a set of m users, and $\mathcal{I} = \{i_1, \ldots, i_n\}$ be a set of n items. We represent the *factual* interactions between users and items with a binary user-item matrix $\mathcal{Y} \in \{0,1\}^{m \times n}$, where $Y_{u,i} = y_{u,i} = 1$ indicates that user u has interacted with item i, or 0 otherwise. Specifically, we denote the interaction history of a user u as $\mathcal{I}_u = \{i \in \mathcal{I} \mid y_{u,i} = 1\}$.

We assume there exists a recommendation model f that can estimate the value $\hat{y}_{u,i}$ for each $u \in \mathcal{U}$ and each $i \in \mathcal{I}$, such that $i \notin \mathcal{I}_u$, as $\hat{y}_{u,i} = f(\boldsymbol{h}_u, \boldsymbol{h}_i)$, where $\boldsymbol{h}_u, \boldsymbol{h}_i \in \mathbb{R}^d$ are suitable user and item representations, respectively, and $f : \mathbb{R}^d \times \mathbb{R}^d \mapsto [0,1]$ is a function that measures the preference score for user u on item i.[1] The score computed with f is used to rank items to populate a list $\hat{\mathcal{I}}_u^k \subseteq \mathcal{I} \setminus \mathcal{I}_u$ of the top-k recommended items for user u, i.e., the list of k unrated items most likely relevant to user u according to f.

Thus, given a target item t recommended to a user u by f (i.e., $t \in \hat{\mathcal{I}}_u^k$), its counterfactual explanations are the minimal subset of u's historical item interactions $\mathcal{I}_u$ whose removal results in evicting t from the top-k list of recommendations $\hat{\mathcal{I}}_u^k$, namely $\mathcal{I}_{u,t}^{CF} \subset \mathcal{I}_u$.

3 ATTACK MODEL

Our strategy involves poisoning attacks that aim to promote target items to legitimate users by manipulating interactions of controlled users. Formally, let $\mathcal{T} \subseteq \mathcal{I}$ denote the set of target items, $\mathcal{U}$ and $\mathcal{U}'$ denote the set of legitimate users and users controlled by the attacker, respectively. Accordingly, $\mathcal{Y}$ and $\mathcal{Y}'$ denote the factual and fake interactions. In practice, online recommender systems limit access to the full user-item interaction data. Our attack model assumes that the factual interaction matrix $\mathcal{Y}$ can only be partially observed. Recommender systems are also typically considered "black-boxes," with the model's internals being inaccessible to attackers. However, recommender systems often have a public API that can be queried to generate a top-k list of suggestions for a user u, which an attacker can use. Additionally, an explanation API for the recommendation model can be exploited to produce CFs for any user and target item pair (u, t).

[1] A similar reasoning would apply if we instead considered explicit ratings, i.e., $f : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$.

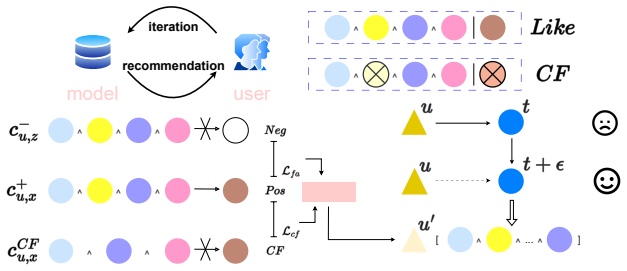

Figure 1: Our proposed H-CARS poisoning attack framework.

To quantify the power of the attack [6], we define its success rate $HR(t)$ – the hit ratio of item $t \in \mathcal{T}$ – as the probability that t appears in the top-k list of recommended items for an actual user, i.e., $\mathbb{P}(\{t \in \hat{\mathcal{I}}_u^k \mid u \in \mathcal{U}\})$.

The attacker aims to create fake interactions $\mathcal{Y}'$ for $\mathcal{U}'$, with the objective of maximizing the hit ratio of the target items. To formalize this problem, we follow prior work [30][8][17] and present a bi-level optimization formulation:

$$\begin{aligned} &\max \sum_{t \in \mathcal{T}} HR(t) \\ &\text{subject to: } \boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \Big\{ \mathcal{L}_{train}(\mathcal{Y}, \hat{\mathcal{Y}}_{\boldsymbol{\theta}}) + \mathcal{L}_{train}(\mathcal{Y}', \hat{\mathcal{Y}}'_{\boldsymbol{\theta}}) \Big\}, \end{aligned} \tag{1}$$

where $\boldsymbol{\theta}$ denotes the model parameters, and $\mathcal{L}_{train}$ denotes the training loss of the target recommendation model f. $\mathcal{Y}_{\boldsymbol{\theta}}$ and $\mathcal{Y}'_{\boldsymbol{\theta}}$ are predictions from the model trained on $\mathcal{Y} \cup \mathcal{Y}'$.

4 PROPOSED METHOD: H-CARS

This section describes H-CARS (Horn-Clause Attacks to Recommender Systems), our poisoning method for recommender systems, via counterfactual explanations, depicted in Fig. 1. The attacker generally cannot access the target model's internals, so it cannot optimize (1) directly. A common practice in black-box scenarios is first training a surrogate model $\hat{f} \approx f$ to generate fake user-item interactions that can be used to poison the target system f.

4.1 Extracting Logical Reasoning Models with Counterfactual Explanations

To build a surrogate model that effectively utilizes the API of CF reasoning, it is crucial to capture the inherent logic of the targeted black-box recommender system. To this end, we adopt the Neural Collaborative Reasoning (NCR) framework as the surrogate recommendation model in this study. NCR encapsulates logical reasoning into a dynamic neural architecture in Horn clause form (hence the name of our method). More specifically, NCR encodes a user-item interaction (u, j) into an *event vector*:

$$\boldsymbol{e}_{u,j} = \boldsymbol{W}_2 \phi(\boldsymbol{W}_1(u, j) + b_1) + b_2, \tag{2}$$

where $j \in \mathcal{I}_u$ and, with a slight abuse of notation, we refer to u and j as their corresponding embedding vectors ($\boldsymbol{h}_u$ and $\boldsymbol{h}_j$, respectively). Moreover, $\boldsymbol{W}_1$, $\boldsymbol{W}_2$, b_1 and b_2 are event parameters that need to be learned, whereas ϕ is the well-known ReLU activation function.

To transform the user-item interactions into a logical expression, basic neural logical modules, AND($\wedge$), OR($\vee$), NOT($\neg$), are introduced. Ideally, predicting if an item x is to be recommended to user

u based on the interaction history is equal to deciding whether the following Horn clause is `true` or `false`:

$$\boldsymbol{e}_{u,1} \wedge \boldsymbol{e}_{u,2} \wedge \cdots \wedge \boldsymbol{e}_{u,j} \Rightarrow \boldsymbol{e}_{u,x}, \tag{3}$$

where $i \in \mathcal{I}_u \; \forall i \in \{1, 2, \ldots, j\}$. Based on the definition of material implication, the above statement is equivalent to the following logic expression vector:

$$\boldsymbol{c}^{+}_{u,x} = (\neg \boldsymbol{e}_{u,1} \vee \neg \boldsymbol{e}_{u,2} \vee \cdots \vee \neg \boldsymbol{e}_{u,j}) \vee \boldsymbol{e}_{u,x}. \tag{4}$$

The recommendation score of x is calculated based on the similarity between the logic expression vector $\boldsymbol{c}^{+}_{u,x}$ and the constant `true` vector denoted by $\mathbf{1}$. We adopt the pair-wise learning algorithm for the factual interaction for model training. Specifically, we construct negative expression:

$$\boldsymbol{c}^{-}_{u,z} = (\neg \boldsymbol{e}_{u,1} \vee \neg \boldsymbol{e}_{u,2} \vee \cdots \vee \neg \boldsymbol{e}_{u,j}) \vee \boldsymbol{e}_{u,z}, \tag{5}$$

by sampling an item $z \in \mathcal{I} \setminus \mathcal{I}_u$ that u did not interact with. Then, we opt for the Bayesian Personalized Ranking (BPR) loss:

$$\mathcal{L}_{fa} = -\sum_{u \in \mathcal{U}} \sum_{x \in \mathcal{I}^{+}_u} \sum_{z \in \mathcal{I}^{-}_u} \log(sim(\boldsymbol{c}^{+}_{u,x}, \mathbf{1}) - sim(\boldsymbol{c}^{-}_{u,z}, \mathbf{1})), \tag{6}$$

where $\mathcal{I}^{+}_u$ and $\mathcal{I}^{-}_u = \mathcal{I} \setminus \mathcal{I}_u$ represents the positive and negative item sets for user u, respectively, and sim measures the similarity between two vectors (e.g., cosine similarity).

Since our partial factual interaction data is limited, CFs generated by the API can be used for data augmentation while alleviating spurious correlations [16][20][11]. For an item x, after subtracting $\mathcal{I}^{CF}_{u,x}$ from $\mathcal{I}_u$, the remaining set $\mathcal{I}_u \setminus \mathcal{I}^{CF}_{u,x}$ leads to a removal of item x from the top-k list of suggestions for user u, i.e., $x \notin \hat{\mathcal{I}}^{k}_u$ anymore. Logically, the counterfactual expression below tends to be `false`:

$$\boldsymbol{c}^{CF}_{u,x} = \left(\bigvee_{j \in \mathcal{I}_u \setminus \mathcal{I}^{CF}_{u,x}} \neg \boldsymbol{e}_{u,j} \right) \vee \boldsymbol{e}_{u,x} \tag{7}$$

For the generated counterfactual interaction, its expression vector $\boldsymbol{c}^{CF}_{u,x}$ is expected to be very distinct from $\boldsymbol{c}^{+}_{u,x}$, to weaken the spurious correlations by contrasting the difference. Hence, the counterfactual loss could be written as:

$$\mathcal{L}_{cf} = -\sum_{u \in \mathcal{U}U} \sum_{x \in \mathcal{I}^{+}_u} \log(sim(\boldsymbol{c}^{+}_{u,x}, \boldsymbol{c}^{CF}_{u,x})). \tag{8}$$

We integrate all the losses together to achieve the final objective:

$$\mathcal{L} = \mathcal{L}_{fa} + \lambda_1 \mathcal{L}_{cf} + \lambda_2 \mathcal{L}_{reg}, \tag{9}$$

where $\mathcal{L}_{reg}$ is a regularization term that ensures the logical module satisfies the logical laws [2][9].

4.2 Poisoning Logical Reasoning Models

Instead of constructing $\mathcal{Y}'$ and $\mathcal{U}'$ directly by solving (1), we start from the goal and backtrack the optimization process [31][30]. Consider a simple case where the attacker wants to promote the target item t to a user u with interaction history $\mathcal{I}_u$. To achieve this, we want the result of the following logic chain to be `true`:

$$\boldsymbol{c}^{+}_{u,t} = (\neg \boldsymbol{e}_{u,1} \vee \neg \boldsymbol{e}_{u,2} \vee \cdots \vee \neg \boldsymbol{e}_{u,j}) \vee \boldsymbol{e}_{u,t}. \tag{10}$$

Since the attacker cannot modify observed interactions of legitimate users in the training set, we focus on leveraging the controlled users to manipulate the embedding of t. Formally, we first determine the optimal item embedding $t + \epsilon$ by maximizing:

$$\varepsilon^{*}_{t} = \text{argmax}_{\varepsilon} sim\Big((\neg \boldsymbol{e}_{u,1} \vee \neg \boldsymbol{e}_{u,2} \vee \cdots \vee \neg \boldsymbol{e}_{u,j}) \vee \boldsymbol{e}_{u,t+\varepsilon}, \mathbf{1}\Big) \tag{11}$$

Then, we transform the problem of promoting target item t into the problem of shifting embedding t to $t + \varepsilon$. In other words, we need to ensure the sum of terms involving item t in the training loss decrease after the shifting.

$$-\sum_{u \in mathcalU \cup \mathcal{U}'} sim(\boldsymbol{c}^{+}_{u,t+\varepsilon}, \mathbf{1}) \leq -\sum_{u \in \mathcal{U} \cup \mathcal{U}'} sim(\boldsymbol{c}^{+}_{u,t}, \mathbf{1}). \tag{12}$$

Here, we utilize the training loss of the surrogate to emulate the optimization process of the original recommendation model. Since the attacker can only control users in $\mathcal{U}'$, their goal is to generate the optimal controlled user $m^* \in \mathcal{U}'$ and its interaction history $\mathcal{I}_{m*}$ that minimize the following loss function in each iteration,

$$m^*, \mathcal{I}_{m*} = \text{argmax}_{m, \mathcal{I}_m} sim(\boldsymbol{c}^{+}_{m,t+\varepsilon}, \mathbf{1}). \tag{13}$$

Practically, we iteratively generate the controlled users. In each iteration, we find the optimal direction of item perturbation by jointly considering the target items $t \in \mathcal{T}$. Then, according to (13), the new m^* and $\mathcal{I}_{m*}$ are obtained and included into $\mathcal{U}'$ and $\mathcal{Y}$.

5 EXPERIMENTS

5.1 Experimental Setup

Datasets. In this paper, we conduct experiments following [3] and [26] over two datasets: *Yelp* and *MovieLens100K*.[2] *Yelp* contains binary ratings from 31,668 users on 38,048 items, while *MovieLens* contains 943 users' rating scores (in the range $[1, 5]$) on 1,682 movies. According to [26], we binarize ratings in *MovieLens* as follows: any score greater than 4 is mapped to 1, and 0 otherwise.

Target Recommendation Models. We experiment with the following target recommendation models:

- ***Neural Collaborative Filtering (NCF)*** [7] replaces the user-item inner product with a neural architecture to capture the complex structure of the user interaction data.
- ***Relational Collaborative Filtering (RCF)*** [29] is developed to exploit multiple-item relations in recommender systems via a two-level hierarchical attention mechanism.

Counterfactual Explanation Method. We consider the following CF generation method:

- ***ACCENT*** [26] generates CFs for neural recommenders by using influence functions to find items most relevant to a recommendation.

Baseline Attack Methods. We compare our H-CARS attack strategy against the following baselines:

- ***DL-Attack*** [8] formulates the attack as an optimization problem such that the injected data would maximize the number of normal users to whom the target items are recommended.
- ***RAPU-R*** [30] starts from the attack goal and reverses the optimization process to obtain the crafted interactions.
- ***Bandwagon Attack*** [30] randomly selects popular items for crafted users who also interact with the target item.

[2]Hereinafter, we refer to it simply as *MovieLens*.

Table 1: $HR@10 \times 100$ for different attacks with 80% training data (left) and 30% training data (right) on two datasets.

Dataset	Recommendation Model	Attack Method	Percentage of Cotrolled Users 0.5%	1%	3%	5%
MovieLens	NCF	DL-Attack	0.34	0.39	0.72	0.82
		RAPU-R	0.32	0.38	0.79	0.86
		H-CARS	**0.37**	**0.42**	**0.82**	**0.91**
		Bandwagon	0.09	0.11	0.16	0.26
	RCF	DL-Attack	**0.25**	**0.29**	0.63	0.71
		RAPU-R	0.23	0.28	0.66	0.73
		H-CARS	0.24	**0.29**	**0.68**	**0.76**
		Bandwagon	0.06	0.10	0.13	0.14
Yelp	NCF	DL-Attack	**0.27**	0.30	0.63	0.73
		RAPU-R	0.25	0.31	0.65	0.74
		H-CARS	0.26	**0.33**	**0.66**	**0.76**
		Bandwagon	0.05	0.08	0.16	0.21
	RCF	DL-Attack	**0.24**	0.25	0.56	0.63
		RAPU-R	0.22	0.24	0.59	0.63
		H-CARS	0.23	**0.26**	**0.61**	**0.66**
		Bandwagon	0.03	0.04	0.12	0.18

Dataset	Recommendation Model	Attack Method	Percentage of Cotrolled Users 0.5%	1%	3%	5%
MovieLens	NCF	DL-Attack	0.12	0.14	0.21	0.23
		RAPU-R	0.10	0.14	0.23	0.29
		H-CARS	**0.13**	**0.18**	**0.29**	**0.35**
		Bandwagon	0.02	0.03	0.05	0.10
	RCF	DL-Attack	**0.06**	0.09	0.17	0.22
		RAPU-R	0.04	0.08	0.17	0.21
		H-CARS	**0.06**	**0.11**	**0.20**	**0.26**
		Bandwagon	0.03	0.03	0.05	0.08
Yelp	NCF	DL-Attack	0.05	0.06	0.09	0.11
		RAPU-R	0.05	0.05	0.09	0.12
		H-CARS	**0.06**	**0.08**	**0.12**	**0.16**
		Bandwagon	0.01	0.02	0.06	0.08
	RCF	DL-Attack	0.06	0.08	0.13	0.15
		RAPU-R	0.06	0.09	0.13	0.16
		H-CARS	**0.07**	**0.10**	**0.15**	**0.20**
		Bandwagon	0.02	0.03	0.05	0.10

5.2 Attack Setting

Adapting [23], we sample 5 target items as $\mathcal{T}$ and limit controlled user selections to 100 and 15 for *MovieLens* and *Yelp* respectively, with a learning rate of 0.001. Model extraction parameters are set: $\lambda_1 = 0.76$, $\lambda_2 = 0.0001$ for *MovieLens*; and $\lambda_1 = 0.68$, $\lambda_2 = 0.00001$ for *Yelp*. In the extraction stage, 60% interactions generate CFs per [26], also used as negative samples for DL-Attack, RAPU-R, and Bandwagon attack. Attackers' performance is assessed using Hit Ratio at 10 ($HR@10$), measuring the proportion of users with at least one $\mathcal{T}$ item in their top-10 recommendations [13].

5.3 Analysis of Results

Our H-CARS attack consistently outperforms or performs similarly to other methods in all target recommendation model and dataset combinations, as shown in Table 1. For instance, using the *MovieLens* dataset, our method improves over RAPU-R and DL-Attack by 5% and 8%, respectively. Notably, our method performs well with limited training data, outperforming RAPU-R by 0.6% with only 30% of the *MovieLens* training data. We observe that our method achieves good performance in the scenario with a limited number of training data. In particular, with only 30% *MovieLens* training data, our method outperforms RAPU-R by 0.6%. Similar conclusions can be drawn for other combinations with limited data. Moreover, as the number of controlled user increases, the performance gap between RAPU-R get enlarged. For instance, compared with RAPU-R, our H-CARS attack achieves 0.1% and 0.6% performance increases on the H-CARS dataset when the percentage of controlled users rises from 0.5% to 5%. Overall, our method achieves the best performance compared to other baselines.

5.4 Ablation Study

In our study, we evaluate the performance of our model extraction method using the MovieLens dataset and NCF as the original recommender. We apply the $P@10$ precision metric and compare three surrogate models: H-CARS-CF, H-CARS-wo-CF, and WRMF. Results in Figure 2 indicate that H-CARS-CF surpasses the others, validating the effectiveness of $\mathcal{L}_{cf}$-based CFs. Additionally, the performance gap between H-CARS-CF and H-CARS-wo-CF widens with increased training data, highlighting the synergy of CFs and factual data in data augmentation.

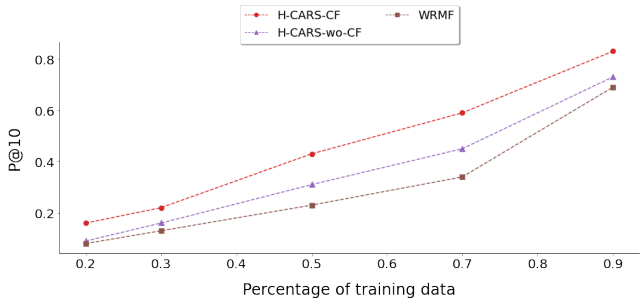

Figure 2: The impact of CFs on surrogate models.

6 CONCLUSION AND FUTURE WORK

We presented a novel approach, H-CARS, that exploits the vulnerabilities induced by counterfactual explanations to launch a poisoning attack on recommender systems. To the best of our knowledge, this is the first such attack proposed in the literature. Our experiments demonstrate that H-CARS is effective, highlighting the importance of considering the security implications of using explainability methods in recommender systems. Future research should explore the potential impact of such attacks on the integrity of the recommender system, as well as develop stronger defenses to mitigate risks in explainable recommender systems.

ACKNOWLEDGMENTS

This work was partially supported by projects FAIR (PE0000013) and SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU, the XJTLU Research Development Fund under RDF-21-01-053, TDF21/22-R23-160, Ningbo 2025 Key Scientific Research Programs, Grant/Award Number: 2019B10128, S10120220021, and National Science Foundation 2127918, 2046457 and 2124155.

REFERENCES

[1] Ulrich Aïvodji, Alexandre Bolot, and Sébastien Gambs. 2020. Model Extraction from Counterfactual Explanations. *arXiv preprint arXiv:2009.01884* (2020).

[2] Hanxiong Chen, Shaoyun Shi, Yunqi Li, and Yongfeng Zhang. 2021. Neural Collaborative Reasoning. In *Proc. of TheWebConf '21*. 1516–1527.

[3] Ziheng Chen, Fabrizio Silvestri, Jia Wang, Yongfeng Zhang, Zhenhua Huang, Hongshik Ahn, and Gabriele Tolomei. 2022. GREASE: Generate Factual and Counterfactual Explanations for GNN-based Recommendations. *arXiv preprint arXiv:2208.04222* (2022).

[4] Ziheng Chen, Fabrizio Silvestri, Jia Wang, He Zhu, Hongshik Ahn, and Gabriele Tolomei. 2022. ReLAX: Reinforcement Learning Agent Explainer for Arbitrary Predictive Models. In *Proc. of CIKM '22*. ACM, 252–261.

[5] Vasisht Duddu and Antoine Boutet. 2022. Inferring Sensitive Attributes from Model Explanations. In *Proc. of CIKM '22*. ACM, 416–425.

[6] Songyang Han, Sanbao Su, Sihong He, Shuo Han, Haizhao Yang, and Fei Miao. 2022. What is the Solution for State Adversarial Multi-Agent Reinforcement Learning? *arXiv preprint arXiv:2212.02705* (2022).

[7] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proc. of WWW '17*. 173–182.

[8] Hai Huang, Jiaming Mu, Neil Zhenqiang Gong, Qi Li, Bin Liu, and Mingwei Xu. 2021. Data Poisoning Attacks to Deep Learning Based Recommender Systems. *arXiv preprint arXiv:2101.02644* (2021).

[9] Ruoming Jin, Dong Li, Jing Gao, Zhi Liu, Li Chen, and Yang Zhou. 2021. Towards a Better Understanding of Linear Models for Recommendation. In *Proc. of KDD '21*. ACM, 776–785.

[10] Amir-Hossein Karimi, Gilles Barthe, Borja Balle, and Isabel Valera. 2020. Model-Agnostic Counterfactual Explanations for Consequential Decisions. In *Proc. of AISTATS '20*, Vol. 108. PMLR, 895–905.

[11] Ruoyan Kong, Haiyi Zhu, and Joseph A Konstan. 2021. Learning to Ignore: A Case Study of Organization-Wide Bulk Email Effectiveness. *Proc. of the ACM on Human-Computer Interaction* 5, CSCW1 (2021), 1–23.

[12] Thai Le, Suhang Wang, and Dongwon Lee. 2020. GRACE: Generating Concise and Informative Contrastive Sample to Explain Neural Network Model's Prediction. In *Proc. of KDD '20*. ACM, 238–248.

[13] Dong Li, Ruoming Jin, Jing Gao, and Zhi Liu. 2020. On Sampling Top-k Recommendation Evaluation. In *Proc. of KDD '20*. ACM, 2114–2124.

[14] Wei Li, Li Fan, Zhenyu Wang, Chao Ma, and Xiaohui Cui. 2021. Tackling Mode Collapse in Multi-Generator GANs with Orthogonal Vectors. *Pattern Recognition* 110 (2021), 107646.

[15] Wei Li, Zhixuan Liang, Ping Ma, Ruobei Wang, Xiaohui Cui, and Ping Chen. 2021. Hausdorff GAN: Improving GAN Generation Quality with Hausdorff Metric. *IEEE Transactions on Cybernetics* (2021).

[16] Xiaohan Li, Zheng Liu, Luyi Ma, Kaushiki Nag, Stephen Guo, S Yu Philip, and Kannan Achan. 2022. Mitigating Frequency Bias in Next-Basket Recommendation via Deconfounders. In *Proc. of BigData '22*. IEEE, 616–625.

[17] Xiaohan Li, Mengqi Zhang, Shu Wu, Zheng Liu, Liang Wang, and S Yu Philip. 2020. Dynamic Graph Collaborative Filtering. In *Proc. of ICDM '20*. IEEE, 322–331.

[18] Ana Lucic, Harrie Oosterhuis, Hinda Haned, and Maarten de Rijke. 2022. FOCUS: Flexible Optimizable Counterfactual Explanations for Tree Ensembles. In *Proc. of AAAI '22*. AAAI Press, 5313–5322.

[19] Ramaravind Kommiya Mothilal, Amit Sharma, and Chenhao Tan. 2020. Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations. In *Proc. of FAT* '20*. ACM, 607–617.

[20] Shanlei Mu, Yaliang Li, Wayne Xin Zhao, Jingyuan Wang, Bolin Ding, and Ji-Rong Wen. 2022. Alleviating Spurious Correlations in Knowledge-Aware Recommendations through Counterfactual Generator. In *Proc. of SIGIR '22*. ACM, 1401–1411.

[21] Martin Pawelczyk, Himabindu Lakkaraju, and Seth Neel. 2022. On the Privacy Risks of Algorithmic Recourse. *arXiv preprint arXiv:2211.05427* (2022).

[22] Federico Siciliano, Maria Sofia Bucarelli, Gabriele Tolomei, and Fabrizio Silvestri. 2022. NEWRON: A New Generalization of the Artificial Neuron to Enhance the Interpretability of Neural Networks. In *Proc. of IJCNN '22*. IEEE, 1–17.

[23] Jiaxi Tang, Hongyi Wen, and Ke Wang. 2020. Revisiting Adversarially Learned Injection Attacks against Recommender Systems. In *Proc. of RecSys '20*. ACM, 318–327.

[24] Gabriele Tolomei and Fabrizio Silvestri. 2021. Generating Actionable Interpretations from Ensembles of Decision Trees. *IEEE TKDE* 33, 4 (2021), 1540–1553.

[25] Gabriele Tolomei, Fabrizio Silvestri, Andrew Haines, and Mounia Lalmas. 2017. Interpretable Predictions of Tree-based Ensembles via Actionable Feature Tweaking. In *Proc. of KDD '17*. ACM, 465–474.

[26] Khanh Hiep Tran, Azin Ghazimatin, and Rishiraj Saha Roy. 2021. Counterfactual Explanations for Neural Recommenders. In *Proc. of SIGIR '21*. ACM, 1627–1631.

[27] Xinghua Wang, Zhaohui Peng, Senzhang Wang, Philip S Yu, Wenjing Fu, Xiaokang Xu, and Xiaoguang Hong. 2020. CDLFM: Cross-Domain Recommendation for Cold-Start Users via Latent Feature Mapping. *Knowledge and Information Systems* 62 (2020), 1723–1750.

[28] Yongjie Wang, Hangwei Qian, and Chunyan Miao. 2022. DualCF: Efficient Model Extraction Attack from Counterfactual Explanations. In *Proc. of FAccT '22*. ACM, 1318–1329.

[29] Xin Xin, Xiangnan He, Yongfeng Zhang, Yongdong Zhang, and Joemon Jose. 2019. Relational Collaborative Filtering: Modeling Multiple Item Relations for Recommendation. In *Proc. of SIGIR '19*. ACM, 125–134.

[30] Hengtong Zhang, Changxin Tian, Yaliang Li, Lu Su, Nan Yang, Wayne Xin Zhao, and Jing Gao. 2021. Data Poisoning Attack against Recommender System Using Incomplete and Perturbed Data. In *Proc. of KDD '21*. ACM, 2154–2164.

[31] Hongke Zhao, Qi Liu, Yong Ge, Ruoyan Kong, and Enhong Chen. 2016. Group Preference Aggregation: A Nash Equilibrium Approach. In *Proc. of ICDM '16*. IEEE, 679–688.

[32] Xuejun Zhao, Wencan Zhang, Xiaokui Xiao, and Brian Lim. 2021. Exploiting Explanations for Model Inversion Attacks. In *Proc. of ICCV '21*. IEEE, 682–692.