Adversarial Attacks for Black-box Recommender Systems via Copying Transferable Cross-domain User Profiles

Wenqi Fan, Xiangyu Zhao, Qing Li, Tyler Derr, Yao Ma, Hui Liu, Jianping Wang, and Jiliang Tang

Abstract—As widely used in data-driven decision-making, recommender systems have been recognized for their capabilities to provide users with personalized services in many user-oriented online services, such as E-commerce (e.g., Amazon, Taobao, etc.) and Social Media sites (e.g., Facebook and Twitter). Recent works have shown that deep neural networks-based recommender systems are highly vulnerable to adversarial attacks, where adversaries can inject carefully crafted fake user profiles (i.e., a set of items that fake users have interacted with) into a target recommender system to promote or demote a set of target items. Instead of generating users with fake profiles from scratch, in this paper, we introduce a novel strategy to obtain "fake" user profiles via copying cross-domain user profiles, where a reinforcement learning based black-box attacking framework (CopyAttack+) is developed to effectively and efficiently select cross-domain user profiles from the source domain to attack the target system. Moreover, we propose to train a local surrogate system for mimicking adversarial black-box attacks in the source domain, so as to provide transferable signals with the purpose of enhancing the attacking strategy in the target black-box recommender system. Comprehensive experiments on three real-world datasets are conducted to demonstrate the effectiveness of the proposed attacking framework.

Index Terms—Recommender Systems, Adversarial Attacks, Black-box Attacks, Data Poisoning Attacks, Cross-Domain.

1 Introduction

The goal of recommender systems is to suggest users with a personalized list of items that are likely to be clicked or purchased [14], [23], [44], which can help alleviate the information overload issue, facilitate users seeking desired information, and increase the traffic and revenue of service providers in many user-oriented online services [8], [55]. Due to the powerful representation learning capabilities in various fields, Deep Neural Networks (DNNs) techniques have been successfully utilized to advance recommender systems [16], [18], [47].

While DNNs-based recommender systems have achieved promising performance in various recommendation scenarios, recent works have shown that DNNs are highly vulnerable to adversarial attacks [5], [10], [26], [31], [50], where adversaries tend to manipulate the data for making wrong predictions. And not surprisingly, modern studies have demonstrated that DNNs based recommender systems are also highly fragile to such adversarial attacks [9], [51], in which personalized recommendation results can be manipulated by adversaries with malicious desires. In general, these methods mainly perform data poisoning attacks (also called shilling attacks) by generating users with well-designed profiles to promote/demote (i.e., attack) a carefully chosen subset of items in recommender systems [6], [9], [17], [25], [34], [35], [41]. A general attacking procedure for recommender systems is illustrated in Figure 1 (a), where an attacker tries to carefully craft fake user profile u^A_{m+1} and then injects them into the target system, so as to manipulate the victim model to recommend a target item v_j to as many users as possible for promotion attack.

Methods to attack recommender systems can be generally divided into three categories based on attacker's knowledge [5], [15], [30], [41], where how much information an attacker is able to know about target recommender systems, including white-box attack, gray-box attack, and black-box attack. The majority of existing attacking methods focus on either white-box or gray-box settings [9], [9], [24], [25], [35], [43], [45], in which the attacker requires to have full or partial knowledge of the target recommender systems such as model architecture and parameters, user-item historical interactions, and prediction. However, with privacy and security-critical concerns, expecting these kinds of complete access (e.g., model architecture/parameters, training data, prediction, etc.) is not realistic nor available in real-world scenarios. Compared to white-box and gray-box attacks, the black-box attack is more practical in real-world applications but more challenging, since model designers usually do not open source their model architecture/parameters, training data, and predicted probability for proprietary reasons. The typical strategy for adversaries to attack such black-box systems is to feed the input data and query the outputs of the victim recommender systems [15], [41]. More recently, only a few works have been proposed to perform attacking recommender systems under black-box setting [15], [41]. For example, PoisonRec [41] proposed the very first reinforcement learning method to generate fake user profiles

- *W. Fan and Q. Li is with the Department of Computing, The Hong Kong Polytechnic University. E-mail: wenqifan03@gmail.com, csqli@comp.polyu.edu.hk.*
- *X, Zhao and J. Wang is with City University of Hong Kong. E-mail: xy.zhao@cityu.edu.hk, jianwang@cityu.edu.hk.*
- *T. Derr is with Vanderbilt University. E-mail: tyler.derr@vanderbilt.edu.*
- *Y. Ma is with New Jersey Institute of Technology. E-mail: yao.ma@njit.edu.*
- *H. Liu and J. Tang are with Michigan State University. E-mail: liuhui7@msu.edu, tangjili@msu.edu.*

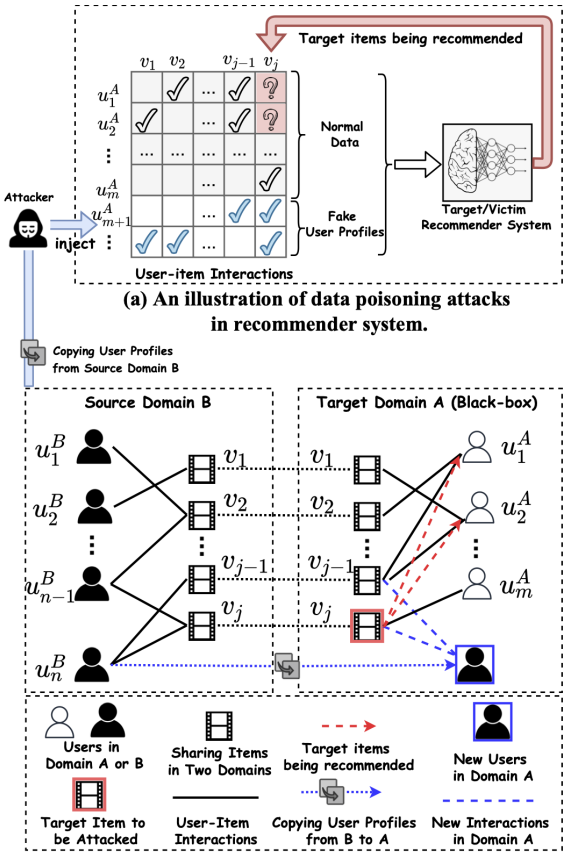


Figure 1: (a) The general data poisoning attacks in recommender systems. (b) Two domains share some movies for recommendations. The profile of user u_n^B (i.e., $\{v_{j-1}, v_j\}$) in the source domain B is copied into the target domain A for attacking the target item v_j.

to attack black-box recommender systems. However, recent defense studies [2], [6], [49], [53], [54] have shown generative fake profiles can be easily detected since they present very different patterns from real profiles, which limits their ability to attack black-box recommender systems.

On the other hand, some real-world recommendation platforms share similar functionalities, which can contribute to having a lot of information in common. For example, movie recommendation platforms *IMDB* and *Netflix* share a lot of movies, and e-commerce sites (e.g., *Amazon* and *eBay*, *Taobao* and *JD.com*, etc.) have millions of products in common. More importantly, users from these platforms with similar functionalities also share similar behavior patterns/preferences towards items. For example, for movie recommendations, fans of Marvel are likely to watch Marvel Cinematic Universe films like *"Captain America"*, *"Iron Man"*, and *"Doctor Strange"* in IMDB, and the users who like the movie *"Iron Man"* in Netflix are highly likely recommended to watch the movies like *"Doctor Strange"* or others Marvel films (e.g., *"Captain America"*, *"Spider-Man"*, *"Black Panther"*). Based on such observations, some efforts have been made to leverage information from one platform (source domain) to help recommendations in other platforms (target domain), which is well known as cross-domain recommendations [3], [28], [32]. Recall that the key obstacle to succeed in attacking black-box recommender systems is to generate users with profiles as close as possible to the real profiles in target recommender systems. To tackle this challenge, in this work, we change the perspective – instead of generating users with fake profiles from scratch, we propose to harness real users' behaviors from the source domain by copying their profiles into target recommender systems with the goal of promoting a subset of items. One illustrative example is shown in Figure 1 (b), where we have a target domain A and a source domain B for movie recommendations. These two domains share a set of movies. To attack (i.e., promote) the targeted item v_j in target domain A, user u_n^B's profile $\{v_{j-1}, v_j\}$ in the source domain B can be copied into the target domain A as a new user u_{m+1}^A, such that the movie v_j is attacked (i.e., promoted) as adversary's desires. Moreover, it could be the case that there exist two e-commerce platforms in competition with each other, and one platform could exploit users' behaviors on its own platform to capture similar patterns, so as to carefully craft fake user profiles for attacking the recommendation performance of its competitors, such as *Taobao* vs. *JD.com*, or *Amazon* vs. *eBay*. Therefore, it is desirable to take advantage of cross-domain information to advance adversarial attacks for black-box recommender systems.

Instead of generating users with fake profiles from scratch, in this paper, we aim to attack recommender systems under the black-box setting via copying cross-domain user profiles, since the copied user profiles from the source domain are naturally real user behaviors. Specifically, we propose a reinforcement learning (RL) based attacking method that learns to choose user profiles in the source domain B, since RL techniques can provide a natural way to interact with a black-box recommender system and receive the reward to optimize the attacking strategy [11], [15], [41].

Nevertheless, learning the attacking strategy of user profiles selection in the source domain based on RL techniques faces tremendous challenges. A huge number of user profiles (i.e., a large-scale discrete action space in RL) in source domain B might lead to inefficiency and ineffectiveness in selecting user profiles at the same time. Thus, the first challenge is how to handle such a large discrete action space in reinforcement learning under the limited number of queries in the target recommender system, so as to maximize long-run rewards. Moreover, directly injecting the selected raw user profiles into the target recommender system may result in increasing the attacking budgets as well as including some noise. Therefore, the second challenge is how to craft and refine the selected cross-domain user profiles before the injection attack. In addition, users from source domain with similar functionalities share similar behavior patterns/preferences [3], [28], [32], which are better to obtain transferablely adversarial examples (i.e., user profiles) against the black-box systems [39], [40], [59]. In particular, adversarial user profiles crafted for successfully attacking local surrogate systems in the source domain are likely to remain adversarial as well as fool different systems in the target domain due to the property of "**transferability**" [38], [39], [40], [59]. Hence, the third challenge is how to take advantage of source domain information to provide transferable signals, with the purpose of enhancing attacking strategy learning in target black-box recommender systems.

To tackle these challenges simultaneously, in this paper,

we propose a reinforcement learning (RL) based attacking method (**CopyAttack+**) that learns to choose user profiles in the source domain B with only query feedback from both target recommender system and local surrogate system in the source domain, which is an improved model of our existing CopyAttack [15]. The major contributions of this paper are summarized as follows:

- We introduce a novel strategy to obtain real user profiles by copying cross-domain user profiles to attack a target recommender system, instead of generating users with fake profiles.
- We introduce a principle way to adopt the adversarial transferability of cross-domain user profiles by training a local surrogate system to mimic adversarial black-box attacks in the source domain, so as to enhance attacking strategy learning for the target black-box recommender system.
- We propose a novel framework (**CopyAttack+**) to attack recommendations under the black-box setting via reinforcement learning, which can effectively and efficiently select transferable cross-domain user profiles from the source domain to attack the target system.
- We conduct comprehensive experiments on three real-world datasets to demonstrate the effectiveness of the proposed attacking black-box framework.

2 Related Work

2.1 Adversarial Attacks for Deep Neural Networks

Due to the powerful capabilities in representation learning, DNNs have achieved great success in various safety and security-critical applications [37], [50] pertaining to ethics, justice, and safety, such as drug discovery and auto-driving. Despite the great success, recent studies have shown that DNNs can expose their vulnerability to humans [26], [30]. Most DNNs are highly vulnerable to adversarial attacks, where adversaries can craftily manipulate legitimate inputs, which may be imperceptible to the human eye, but can force a trained model to produce incorrect outputs [50]. For example, an attacker might put stickers on a road sign to confuse an autonomous vehicle's image recognition system from any viewpoint [13]. With the extension of DNNs to graph-structured data, attackers can generate graph adversarial perturbations by manipulating the graph structure or node features to deceive the GNNs into making incorrect predictions [22], [30], [31]. Therefore, the security-critical issues of DNNs have become an important area of research.

2.2 Adversarial Attacks for Recommender Systems

As widely used in data-driven decision-making, recommender systems have been recognized for their ability to provide personalized services for users, playing an increasingly important role in many user-oriented online services [20], [21], [55], such as E-commerce (e.g., eBay, JD.com), and Social Media sites (e.g., Facebook, Weibo). With malicious purposes (e.g., financial incentives, malicious competitors), adversaries might inject fake data to attack recommender systems, such that recommendation results can be manipulated, and users' beliefs/decisions towards items can be influenced as much as possible [6], [9], [15], [34], [41]. The majority of existing attacking methods focus on either white-box or gray-box settings [9], [9], [24], [25], [35], [36], [43], [45], where adversaries can fully or partially access the knowledge of target recommender systems such as model architecture and parameters, user-item historical interactions, and prediction. Typically, these white-box or gray-box approaches can compute the gradient to optimize data poisoning attack model with full or partial knowledge of target recommender systems [9], [24], [25], [35], [36], [43], so as to generate effective user profiles for performing adversarial attacks. For example, Projected Gradient Method and Stochastic Gradient Langevin Dynamics (SGLD) [45] is adopted to optimize data poisoning attack model with full knowledge of factorization-based collaborative filtering [35]. A two-step adversarial framework [9] is introduced to attack recommender systems, in which they first generate fake users through a Generative Adversarial Network (GAN), and then apply Projected Gradient Method for further crafting fake user profiles with a suitable adversarial intent. Similarly, AUSH [36] designs a minimax game (GANs) to generate fake user profiles with a reconstruction loss and a shilling loss. The work of [24] formulates the data poisoning attack in recommender systems as a non-convex integer optimization problem and develops a gradient-based method to optimize the rating scores for fake users one by one.

In fact, with privacy and security concerns in recommender systems, it is impossible and unrealistic to expect these kinds of complete access on the knowledge of target environments. Therefore, it is desired to study black-box attacks in recommender systems, where the attackers do not have full knowledge of the target model and dataset. More recently, only a few works have been proposed to perform attacking recommender systems under black-box setting [15], [41]. For instance, PoisonRec [41] proposes a reinforcement learning method to generate fake user profiles to attack black-box recommender systems. CopyAttack [15] introduces copy user profiles from the source domain to perform adversarial black-box attacks. As the extension of our previous work CopyAttack [15], in this paper, we adopt the adversarial transferability of cross-domain user profiles by training a local surrogate system to mimic adversarial black-box attacks with source domain data, so as to enhance attacking strategy learning in CopyAttack for the target black-box recommender system.

3 Problem Statement

Let a target recommender system A be defined as having a set of users $\mathcal{U}^A = \{u_1^A, u_2^A, ..., u_{n^A}^A\}$ and a set of items $\mathcal{V}^A = \{v_1, v_2, ..., v_{m^A}\}$, where n^A is the number of users and m^A is the number of items in A. In addition, user-item interactions are represented as the matrix $\mathbf{Y}^A \in \mathbb{R}^{n^A \times m^A}$, where an interaction y_{ij}^A indicates that user u_i^A interacted with item v_j (e.g., clicked/bought), and 0 otherwise. Furthermore, we define the set of items a user u_i^A interacts within $\mathbf{Y}^A$ (i.e., their user profile) as:

$$P_{u_i}^A = \{v_1 \rightarrow ... \rightarrow v_j \rightarrow ... \rightarrow v_l\},$$

where $\rightarrow$ denotes the sequential order of the l items u_i^A has interacted with (and the length l can vary between users). We then denote the set of all user profiles in the target domain A as $\mathcal{P}_{\mathcal{U}}^A = \left\{P_{u_1}^A, ..., P_{u_i}^A, ..., P_{u_{n^A}}^A\right\}$.

We define a source recommender system B similarly, having the set of n^B users $\mathcal{U}^B$, set of m^B items $\mathcal{V}^B$,

Table 1: Notation

Symbols	Definitions and Descriptions
$\mathcal{U}^A, \mathcal{V}^A$	user and item sets in target domain A
$\mathcal{U}^B, \mathcal{V}^B$	user and item sets in target domain B
$\mathbf{Y}^A, \mathbf{Y}^B$	user-item interactions in domain A and B, respectively
n^A, m^A	the number of users and items in target domain A
n^B, m^B	the number of users and items in source domain B
$\mathcal{V}$	a set of overlapping items $\mathcal{V}^A \cap \mathcal{V}^B$ between the target domain A and the source domain B
$P^B_{u_i}$	a user u_i profile in source domain B
$\mathcal{P}_{\mathcal{U}}$	the set of user profiles in source domain B
v_*	a target item to be attacked
a^u_t	an action for selecting a raw user profile $P^B_{u_i}$ at state t
a^l_t	an action for crafting a raw user profile $P^B_{u_i}$ as $\hat{P}^B_{u_i}$ at state t
$\hat{P}^B_{u_i}$	a crafted user profile before injecting into the target system
$\vec{P}^B_{u_i}$	a augmented user profile before injecting into the source system
$\mathbf{p}^B_i$	a user's representation in source domain B
$\mathbf{q}^B_i$	an item's representation in source domain B
$\triangle$	the budget given to the attacker (in terms of the number of cross-domain user profiles to copy)
$\mathcal{U}^{B\to A}$	a set of cross-domain users copied into domain A from domain B: $\{u^B_i\}^{\triangle}_{i=1} = \{\mathcal{P}^B_{u_i}\}^{\triangle}_{i=1}$
$\mathcal{U}^{A'}$	the polluted users set in domain A: $\mathcal{U}^A \cup \mathcal{U}^{B\to A}$
PN-*	a policy network in the hierarchical clustering tree with its parameters θ_*
$\mathcal{I}$	the number of non-leaf nodes of the hierarchical clustering tree
d	the depth of the hierarchical clustering tree
c	the number of child node in the hierarchical clustering

interaction matrix $\mathbf{Y}^B \in \mathbb{R}^{n^B \times m^B}$, and set of user profiles $\mathcal{P}^B_{\mathcal{U}}$. Note that the source domain B is selected such that there are overlapping items between the target domain A and the source domain B. In other words, there exists a set of items $\mathcal{V} = \mathcal{V}^A \cap \mathcal{V}^B$, where $|\mathcal{V}| \neq \emptyset$ and the overlap (i.e., size of $\mathcal{V}$) is assumed to be sufficiently large. Thus, we then define an item profile $P^A_{v_j}$ for $v_j \in \mathcal{V}$, which is the set of users from A who have interacted (e.g., clicked/purchased) with v_j in $\mathbf{Y}^A$ as follows:

$$P^A_{v_j} = \left\{u^A_1 \to ... \to u^A_i \to ... \to u^A_o\right\},$$

where o is the number of users in the item's profile (that can differ from item to item). Let $\mathcal{P}^A_{\mathcal{V}} = \left\{P^A_{v_1}, ..., P^A_{v_j}, ..., P^A_{v_{m^A}}\right\}$ denote the set of item profiles in target domain A.

Now, given the notations of the target and source recommender systems A and B, respectively, we formally define the goal of the target recommender system A. Overall, the objective of A (which we denote here at $\text{Rec}(\cdot,\cdot)$) is to predict whether user u^A_i likes (i.e., will interact with) an item v_j as $y^A_{ij} = \text{Rec}(P^A_{u_i}, P^A_{v_j})$. Thus, without loss of generality, the target recommender system task is to predict a list of Top-k ranked potential items for each user. More formally, this recommendation is defined as follows:

$$y^A_{i,>k} = \left\{v_{[1]}, v_{[2]}, ..., v_{[k]}\right\} = \text{Rec}(P^A_{u_i}, \mathcal{P}^A_{\mathcal{V}}),$$

where $y^A_{i,>k} = \left\{v_{[1]}, v_{[2]}, ..., v_{[k]}\right\}$ denotes the Top-k candidate items for user u^A_i. For completeness, we note that these candidate items in $y^A_{i,>k}$ are ranked by $\text{Rec}(\cdot,\cdot)$, where user u^A_i is more likely to click/purchase item $v_{[i]}$ than $v_{[i+1]}$.

Finally, we define the problem of a black-box injection attack to promote a target item $v_* \in \mathcal{V}$ by copying a set of users (profiles) $\mathcal{U}^{B\to A} = \{u^B_i\}^{\triangle}_{i=1} = \{\mathcal{P}^B_{u_i}\}^{\triangle}_{i=1}$ from the source domain to the target domain, where $\triangle$ is the budget given to the attacker (in terms of the number of cross-domain user profiles to copy). Note that the attacking results in the target domain having the set of polluted users $\mathcal{U}^{A'} = \mathcal{U}^A \cup \mathcal{U}^{B\to A}$ and thus also polluting the interaction matrix $\mathbf{Y}^A$. More precisely, the pollution of $\mathbf{Y}^A$ is due to the fact that introducing the copied cross-domain user profiles brings their interactions with the set of items $\mathcal{V}$ and hence disrupts the relations between users and items in A. Furthermore, to be more specific, we define the promotion of a target item v_* as having this item appear in the Top-k recommendation list for users in $\mathcal{U}^A$ that previously (before injecting the copied cross-domain users $\mathcal{U}^{B\to A}$ and their associated interactions) did not have v_* in their Top-k recommendation list. The mathematical and notations used in this work are summarized in Table 1.

4 The Proposed Framework

4.1 An Overview of the Proposed Framework

To perform attacking in recommender systems under black-box setting, traditional gradient-based techniques [9], [35] are not applicable nor realistic, since they ideally assume that the target recommender system and dataset can be accessed. Thus, we propose a reinforcement learning (RL) based attacking framework, CopyAttack+, to learn the strategy of copying transferable cross-domain user profiles. This is because reinforcement learning can provide a natural way to interact with a black-box recommender system and receive the reward to optimize the framework [1], [4], [42], [46], [56]. What's more, inspired by the property of "transferability" [38], [39], [40], we propose to train a local surrogate system to mimic adversarial black-box attacks in source domain B, with the purpose of enhancing strategy learning for attacking the target black-box recommender system. The overall architecture of the proposed CopyAttack+ is shown in Figure 2, which consists of three major components: user profile selection, user profile crafting, and injection attack and queries.

The first component is to perform user profile selection for specific target item attack, which is proposed to select user profiles from $\mathcal{P}^B_{\mathcal{U}}$ (i.e., user profiles from the source domain B), as shown in the left part of Figure 2. However, modeling this process of selection with reinforcement learning technique is rather challenging under limited resources (i.e., number of queries (or interactions) allowed to the target recommender system), since a huge number of user profiles (i.e., a large-scale discrete action space) in source domain B might lead to inefficiency and ineffectiveness at the same time. Moreover, not all user profiles are useful to help attack the specific target item in the target recommender system. To address these challenges, we propose to adopt hierarchical-structure policy gradient networks with a masking mechanism to efficiently and effectively learn the strategy of selecting cross-domain user profiles in large-scale discrete action space, so as to maximize long-run rewards.

Second, once having selected a cross-domain user profile from the first component, the second component is used for user profile crafting. Here user profile crafting aims to further modify/refine the user profile by considering the reduction of attack cost and noise, and can be seen in the center part of Figure 2. We note that users can have user profiles consisting of varying lengths (i.e., the number of items they have interacted with). Thus, it could

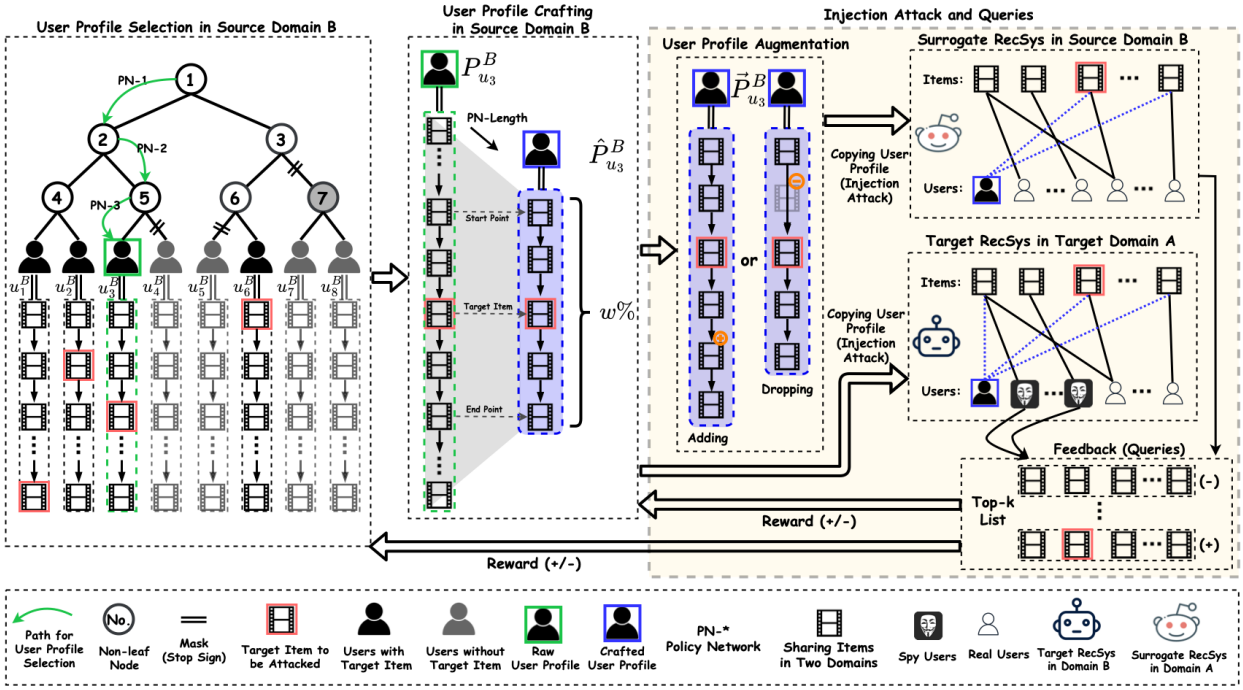


Figure 2: An overview of the proposed framework CopyAttack+. It contains three major components: user profile selection in source domain B, user profile crafting in source domain B, and injection attack and queries in surrogate and target recommender systems (RecSys).

be the case that not all the interactions that the users have given towards items in their user profiles are helpful. Furthermore, too long of a user profile might include some noise as well as increase the attack cost (i.e., the number of interactions the copied user would need to perform in the target domain). Therefore, it is desirable to carefully craft the selected cross-domain user profile before attacking. However, crafting the user profiles is not trivial, since the crafted user profiles should contain useful signals regarding the target items, as well as preserve their reality (i.e., being undetectable as fake to the target recommender system). To address these challenges, we introduce a second-step policy gradient network to craft/refine the user profiles by considering this attacking cost issue.

Lastly, the third component's first objective is to attack the target recommender system by copying the crafted cross-domain user profiles (i.e., those coming from the source domain). After having copied crafted cross-domain user profiles, queries on the target recommender system are performed to obtain some feedback in the form of Top-k recommendations. This feedback is then used to form a reward for optimizing the whole framework (i.e., updating the policy gradient networks of the first and second components). Moreover, the "transferability" property of the surrogate system in source domain B is introduced to further enhance the black-box attacking strategy learning in attacking the target recommender system with surrogate reward, achieved by various training paradigms. Note that we propose to perform user profile augmentation before attacking the local surrogate model, which is trained on data from the source domain. This component can be seen in the right part of Figure 2.

4.2 Attacking Environment Overview

The attacking black-box framework can be modeled as a Markov Decision Process (MDP) [4], [57], [58]. The definition of MDP contains the state space S, action set A, transition probability P, reward R, and discount factor γ (i.e., (S, A, P, R, γ)) that are defined as follows:

- **State** S**.** A state s_t consists of all the intermediate injected user profiles and target item at state t.
- **Action** A**.** The action has two components and is defined as $A = \{a_t = (a_t^u, a_t^l)\}$. More specifically, the attacker is allowed to first select a user $a_t^u = u_i^B$ from the cross-domain (i.e., source domain) system B at state t. Then, the attacker can modify the original profile $P_{u_i}^B$ of u_i^B to craft a profile of perhaps shorter length resulting in $a_t^l = \hat{P}_{u_i}^B$. Note that this crafted user profile would be the one ultimately injected into the target recommender system.
- **Transition Probability** P**.** Transition probability $p(s_{t+1}|s_t, a_t)$ defines the probability of state transition from the current s_t to the next state s_{t+1} when the attacker takes action a_t.
- **Reward** R**.** The goal of the attacker is to attack a target item v_* in target recommender system $\text{Rec}(\cdot, \cdot)$ with their desires (such as promotion of that target item). In this work, we focus on the promotion attack, where the attacker seeks to have the target items recommended to as many users as possible. A natural way to define the reward for the RL-based method is on the basis of ranking evaluation measures [15], [27]. Thus, for the reward function based on ranking, we assign a positive reward for action a_t when the target item v_* belongs to the Top-k recommended list for a set of spy users $u_{i*}^A \in \mathcal{U}_*^A \subset \mathcal{U}^A$. These spy users can be easily established in the target domain before the injection attacks due to the property of openness in various online services (as seen in Figure 2). We note that these spy users solely exist in the target recommender system, so that the attacker can use them as a *proxy* for determining how effective their copied user profiles are at promoting the target items to all users in $\mathcal{U}^A$. We use the Hit Ratio (HR@K) as the

ranking evaluation in our reward function $r(s_t, a_t)$ for a given state s_t and action a_t:

$$r(s_t, a_t) = \frac{1}{|\mathcal{U}_*^A|} \sum_{i=1}^{|\mathcal{U}_*^A|} \text{HR}(u_{i*}^A, v_*, k), \quad (1)$$

$$\text{HR}(u_{i*}^A, v_*, k) = \begin{cases} 1, & v_* \in y_{u^*,>k}^A, \\ 0, & v_* \notin y_{u^*,>k}^A, \end{cases}$$

where $\text{HR}(u_{i*}^A, v_*, k)$ returns the hit ratio for a targeted item v_* in the Top-k listing of spy users u_{i*}^A (i.e., whether v_* is in the set $y_{u_{i*},>k}^A$ or not) and the reward is averaged over the hit ratio of all the spy users in $\mathcal{U}_*^A$. Note that demotion attacks can be achieved by designing different reward functions (i.e., observable number of Page View (PV) on target items [41].).

- **Discount factor** γ**.** $\gamma \in [0, 1]$ is used to determine the importance between the long-term rewards and the immediate rewards. When $\gamma = 0$, the agent will focus on immediate rewards only, while the agent will focus on all future rewards when $\gamma = 1$.
- **Terminal.** The attacking process has two stopping conditions. First, it will stop if reaching the maximum budget $\triangle$ (i.e., the number of copied cross-domain user profiles). Additionally, it can stop before the budget is reached if less copied user profiles are able to successfully satisfy the promotion task.

4.3 User Profile Selection via Hierarchical-structure Policy Gradient

User profile selection aims to learn the strategy of selecting cross-domain user profiles. More specifically, it seeks to discover the set of users $\mathcal{U}^{B\to A} \subset \mathcal{U}^B$ that we can then inject their user profiles into the targeted recommender system's set of users $\mathcal{U}^A$ to achieve the goal of promoting a set of items. Here, the main challenges are how to handle a large-scale discrete action space (i.e., set of all user profiles) as well as achieve satisfactory results under limited resources to interact with the target (black-box) recommender system A. Most existing RL techniques cannot handle such a large discrete action space problem [1], [12], [42], [56], [58], since the time complexity of making a decision is linear to the size of action space. To address these challenges, we introduce a hierarchical-structure policy gradient network with a masking mechanism to model the process of selecting a user profile (as shown in the left part in Figure 2).

4.3.1 Hierarchical Clustering Tree over Cross-domain User Profiles

In the hierarchical clustering tree, each leaf node is represented as a cross-domain user profile, while each non-leaf node is a policy network. However, the question remains on how to construct the clustering tree. Hence, we propose to employ a top-down divisive approach that will repeatedly divide each cluster into small sub-clusters where leaf nodes under the same non-leaf node in the clustering tree should be more similar to each other than leaf nodes coming from another non-leaf node. We note that this process starts with the entire set of nodes at the root of the clustering tree.

When constructing our hierarchical clustering tree, we further add the constraint that it should be balanced to ensure the proper speedup, since an unbalanced clustering tree in the worst case could result in a linked list of policy networks on the order of the number of users. To achieve this, at each non-leaf node when constructing the tree (top-down), we first apply the traditional K-means clustering on that current set of users to obtain the set of c centroids. Note that the number of cluster centers (i.e., centroids) is set to the number of child nodes in the hierarchical clustering tree. Then, we reassign the users to these c centroids one at a time based on their Euclidean distance to ensure we have a balanced set of clusters (in terms of their size).

When constructing the clustering tree, one major consideration is how to balance the number of children per node against the height of the tree. To better understand this relationship between the depth of hierarchical clustering tree d, the number of leaf node $|\mathcal{U}^B|$, and the number of child node c, we can observe the following:

$$c^{d-1} < |\mathcal{U}^B| = n^B \leqslant c^d,$$

and the number of non-leaf nodes of the tree is $\mathcal{I} = \frac{c^d-1}{c-1}$.

To perform hierarchical clustering, we note that users' numerous features could be used for their representations, such as user attributes, review comments, social relations, and user-item interactions. In this work, we adopt the user-item interactions $\mathbf{Y}^B$ to represent the users because auxiliary information such as the user's attributes and review comments are not available. We use the user representations $\mathbf{p}^B \in \mathbb{R}^e$ learned via matrix factorization (MF) [33] to measure similarity between users.

4.3.2 Masking Mechanism

While cross-domain user profiles contain informative signals of items, due to the limited number of queries in the target recommender system, not all cross-domain user profiles are useful for attacking a specific target item. Actually, only user profiles related to the specific target items would be useful. Therefore, we need to tune the hierarchical clustering tree with a masking mechanism to locate some percentage of related cross-domain user profiles for the target items. More specifically, for each target item, we take the approach of masking the cross-domain user profiles that do not include the target item. As shown in the left part in Figure 2, the path from non-leaf node 3 to node 7 is masked, since the cross-domain profiles of user u_7^B and u_8^B do not include the target item (with pink color). As such, these cross-domain user profiles (i.e., u_7^B and u_8^B) can not be explored by the RL agent, which might further help reduce the action space. Then, this reduction in the action space is efficient in locating useful cross-domain user profiles to perform an effective attack. We again note that the target item v_* comes from the set $\mathcal{V} = \mathcal{V}^A \cap \mathcal{V}^B$.

4.3.3 Hierarchical-structure Policy Gradient

With the (masked) hierarchical clustering tree, the purpose of user profile selection is to learn the policy $p(a_t^u|s_t^u)$ for seeking a path a_t^u from the root to a certain leaf of a tree (i.e., user in $\mathcal{U}^B$) at state t. Each non-leaf node in the tree is a policy gradient network, which can be modeled as a Multi-Layer Perceptron (MLP). As such, there are $\mathcal{I}$ policy gradient networks (i.e., non-leaf nodes) with $\theta = \{\theta_1, \theta_2, ..., \theta_\mathcal{I}\}$ in the hierarchical clustering tree.

In particular, the policy network at $node_i$ (having MLP parameters denoted as θ_i) first takes the current state as input and outputs a probability distribution over all child nodes of $node_i$. Then, one of the children (action) is selected to move based on the probabilities. The selection process then keeps moving down the

clustering tree of policy networks until reaching a leaf node (i.e., a user profile), which can form the path of length d from the root to the leaf node as follows: $a_t^u = \left\{a_{[t,1]}^u, a_{[t,2]}^u, ..., a_{[t,d]}^u\right\}$.

This selection process can be decomposed into multiple steps according to the selected path a_t^u as follows:

$$p^u(a_t^u|s_t^u) = \prod^d p_d^u(a_t^u|\cdot, s_t^u). \quad (2)$$

We represent the state s_t^u with the target item v_* and previous selected users $\mathcal{U}_t^{B\to A} = \{u_1^B, ..., u_i^B, ..., u_t^B\}$. We combine them together with a Multi-Layer Perceptron (MLP). To decide which path we will move to, by estimating the probability distribution over the children at $node_i$ (i.e., the policy network parameterized by θ_i), as follows:

$$p_i^u(\cdot|s_t^u) = \text{softmax}(\text{MLP}([\mathbf{q}_{v_*}^B \oplus \mathbf{x}_{v_*}]|\theta_i^u)),$$
$$\mathbf{x}_{v_*} = \text{RNN}(\mathcal{U}_t^{B\to A}),$$

where $\mathbf{q}_{v_*}^B \in \mathbb{R}^e$ is the pre-trained item representation via Matrix Factorization (MF) coming from the source domain B. Meanwhile, we introduce to utilize a Recurrent Neural Network (RNN) model to encode the selected cross-domain users $\mathcal{U}_t^{B\to A}$ at state s_t as low-dimensional representations, such that the historical interactions between attacker and target recommender systems can be extracted. Here we use $\oplus$ to denote the concatenation operation. Also, here we seed the process by selecting action a_0^u (i.e., the first user to inject in the target recommender system) at random, since at that time $\mathcal{U}_0^{B\to A}$ is empty and would not provide any insights from the RNN.

An example illustration of the process of selecting cross-domain user profiles is shown in the left part in Figure 2. We have 8 user profiles, and build a balanced hierarchical clustering tree with depth 3 over user profiles in the source domain B. For a given state s_t, the status point is initially located at the root ($node_1$), and moves to one of its child nodes to ($node_2$) according to the probability distribution given by the policy network *PN-1* corresponding to the root ($node_1$). The process of selecting can stop when the state point arrives at a leaf node in the tree; in this case, user u_3^B's profile. Note that at the state point $node_5$, the path from $node_5$ to leaf node u_4^B is masked since the profile of source domain user u_4^B does not include the currently attacking target item. The example path for this selection is $a_t^u = \{node_1, node_2, node_5, u_3^B\}$, as the path with green color in the figure.

Although we now have an efficient mechanism for selecting the set of source domain users that the attacker will copy into the target domain, we again note here that there could be some problems with directly copying these cross-domain user profiles into the target recommender system. More specifically, it could be the case that not all items in a user's profile are useful in the promotion attack and could just inject noise and/or increase the attack cost.

4.4 User Profile Crafting

The selected cross-domain user profiles contain informative signals regarding the target item. However, it is not necessary that all the interactions the cross-domain user has in their profile are helpful. In fact, just naïvely injecting the entire raw user profiles into the target recommender system may not only increase the attacking budget, but could include some noise as well. Therefore, it is desirable to craft and refine the selected cross-domain user profiles before the injection attack.

We note that when considering how to craft the user profiles there are perhaps a few options that could be taken on how to reduce the length of the user profile. For example, intuitively randomly selecting a subset to keep would not make sense due to the fact it would lose the temporal/sequential relations of items that were interacted with by the given user around the same time as the target item. Furthermore, if we were to select perhaps based on the most similar nodes to the target node from the user's profile, then this might result in a less realistic user profile, which could potentially be more easily detected. Hence, our selection of clipping the cross-domain user profile with a window size w around the target item appears to indeed be a logical mechanism for clipping. The reasons are as follows: 1) User behaviors towards items contain sequential patterns, and forward and backward interacted items around the target item are not independent and are more related to the target item than others, and 2) Sequentially-ordered user interactions are close to real user behaviors and are more likely to evade the detection from the target system. For example, fans of "Game of Thrones" sequentially watch television season by season and episode by episode, and it typically does not make sense to only watch the "Game of Thrones: Season 8" if the user has not seen the previous seasons. Therefore, we introduce to discretize the length of user profile into 10 different levels (window size) as: $W = \{10\%, 20\%, 30\%, 40\%, 50\%, 60\%, 70\%, 80\%, 90\%, 100\%\}$.

Then, we propose to perform a clipping operation to craft the raw cross-domain user profiles around the target item with a window size w via a policy gradient network, as shown in the middle of Figure 2. More specifically, this policy gradient network is introduced to choose the action $a_t^l = w$ from the set W to decide the length we keep (i.e., number of interactions for that selected user profile). As the raw selected cross-domain user profile includes the target item v_*, the raw user profile is clipped around the target item with the window size w. As such, we can consider the forward and backward related items, as well as preserve the natural sequential behavior of the real user profiles.

The state s_t^l for model clipping operation can be decided by the selected user u_i and target item v_*. We estimate the probability of choosing action a_t^l over the entire set W with the state s_t^l, as follows,

$$p^l(\cdot|s_t^l) = \text{softmax}(\text{MLP}([\mathbf{p}_i^B \oplus \mathbf{q}_{v_*}^B]|\theta^l)),$$

where $\mathbf{p}_i^B \in \mathbb{R}^e$ and $\mathbf{q}_{v*}^B \in \mathbb{R}^e$ are the pre-trained user and item representations via MF in source domain, respectively.

4.5 Injection Attack and Queries

The component is used to perform injection attack and queries in both target and source domains. To perform attacking under black-box setting, we only have query access to the target/surrogate model and can get query feedback consisting of Top-k recommended items for specific users (e.g., spy users), so as to update the policy networks for both the profile selection and profile crafting components.

4.5.1 Injection Attack and Queries on Target Recommender System in Target Domain

Injection Attack. In the last stage, the main goal is to inject/copy the crafted version of the selected user profiles

$\hat{P}^B_{u_i}$ from the source domain to the target domain. Then, once injected, the attacker can utilize their set of spy users $\mathcal{U}^A_*$ they have already established in the target domain to gauge the effectiveness of the injected user profiles and define a corresponding reward.

Queries. To evaluate the feedback from the attacked target recommender system, here we use the reward function $r^A(s_t, a_t)$ defined in Eq. (1) where the effectiveness is defined based on the Hit Ratio (HR@K) of the targeted item v_* aggregated over the set of spy users' (i.e., those in the set $\mathcal{U}^A_*$) Top-k recommendations. We note that these Top-k recommendations are the result/feedback upon performing queries on the target system A. Once obtaining the reward, it is then used to update the policy networks for both the profile selection and profile crafting components.

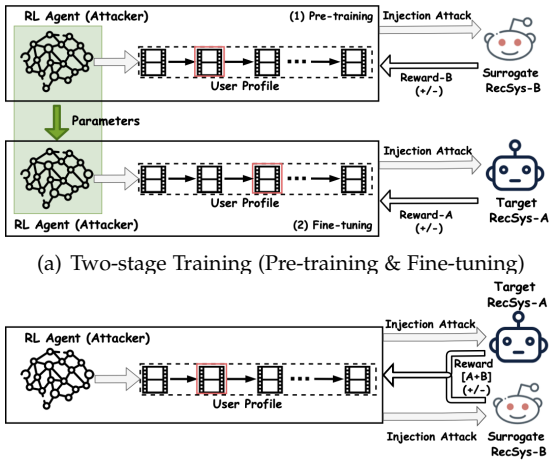
4.5.2 Injection Attack and Queries on Surrogate Recommender System in Source Domain

Inspired by the property of "transferability" [38], [39], [40], [59], if adversarial user profiles can successfully attack the surrogate system developed in source domain B, they are also likely to remain adversarial to attack the black-box system in the target domain. That is because users from cross domains with similar functionalities share similar behavior patterns/preferences [3], [32]. Therefore, we introduce to train a local surrogate model to mimic adversarial black-box attacks with data from the source domain, so as to further learn transferablely adversarial knowledge to enhance the proposed RL-based strategy for attacking the black-box target system. Note that a local surrogate recommender system can be developed by an attacker with a totally different architecture from the target recommender system. In addition, to imitate the black-box attack in the target recommender system, we also treat this surrogate system as a black-box, perform injection attacks and query outputs of the surrogate system.

User Profile Augmentation. Different from the injection attack in the target domain, just naïvely injecting the crafted version of the selected user profiles $\hat{P}^B_{u_i}$ back into the surrogate system might fail to provide useful feedback to update the policy networks. This is due to the fact that the selected user profile $\hat{P}^B_{u_i}$ or $P^B_{u_i}$ has already existed in the source domain. Thus, we introduce a user profile augmentation to modify the crafted user profile ($\hat{P}^B_{u_i}$) before injecting it into the surrogate system, hoping that a new user $u^B_{|\mathcal{U}^B|+1}$ (i.e., $\vec{P}^B_{u_i}$) can contribute to manipulating the recommendation results of the surrogate system. More specifically, we employ two general data augmentations [29], [48]: *items adding* and *items dropping*, which randomly adds or drops a certain number of items based on the length of $\hat{P}^B_{u_i}$. Here, we empirically set the number of adding/dropping items as 1% length of $\hat{P}^B_{u_i}$.

Queries. Then, once augmented $\vec{P}^B_{u_i}$ injected, the surrogate system can form a corresponding reward $r^B(s_t, a_t)$ based on the Hit Ratio (HR@K) in Eq. (1), which is similar to attack the target recommender system. One potential benefit of attacking the surrogate system is that we don't need to establish a set of spy users, while all users $\mathcal{U}^B$ in the surrogate system can be used to gauge the effectiveness of the new augmented user profiles $\vec{P}^B_{u_i}$ and generate a surrogate reward. Therefore, the reward in the surrogate recommender system can be defined as follows:

$$r^B(s_t, a_t) = \frac{1}{|\mathcal{U}^B|} \sum_{i=1}^{|\mathcal{U}^B|} \mathrm{HR}(u^B_{i*}, v_*, k).$$



(a) Two-stage Training (Pre-training & Fine-tuning)

(b) Joint Training

Figure 3: Training paradigms for attacking target and surrogate systems.

4.6 Training Strategy

In the last subsection, we discussed the injection attack and queries on black-box recommender systems (i.e., surrogate and target systems). In this section, we present two training strategies to merge adversarial transferability into optimizing the policy networks for adversarial black-box attacks in the target recommender system. These training strategies can be divided into two categories: Two-stage Training (Pre-training & Fine-tuning) and Joint Training, with their detailed workflow shown in Figure 3. Thus, we have corresponding variants of our proposed method CopyAttack+: **CopyAttack+(Two)** and **CopyAttack+(Joint)**.

4.6.1 Two-stage Training (Pre-training & Fine-tuning)

In this training paradigm, it usually performs in a two-stage training: Pre-training & Fine-tuning. In particular, it first obtains the RL agent's (attacker) parameters by performing a black-box attack on the local surrogate recommender system in the source domain (pre-training). These pre-training parameters are then used to initialize the RL agent, and fine-tune them via attacking the target recommender system based on the corresponding reward. We term this variant as **CopyAttack+(Two)**. An overview of the two-stage training method CopyAttack+(Two) is illustrated in Figure 3 (a).

4.6.2 Joint Training

To exploit the adversarial transferability into attacking the target black-box recommender system, a natural idea is to jointly train RL agent (attacker) with reward from both the surrogate and target recommender system (as a multi-task training strategy), as shown in Figure 3 (b). We term this variant as **CopyAttack+(Joint)**. Therefore, the overall reward function in this training paradigm CopyAttack+(Joint) can be formulated with two rewards as follows:

$$\begin{aligned} R(s_t, a_t) &= r^A(s_t, a_t) + \lambda\, r^B(s_t, a_t) \\ &= \frac{1}{|\mathcal{U}^A_*|} \sum_{i=1}^{|\mathcal{U}^A_*|} \mathrm{HR}(u^A_{i*}, v_*, k) + \lambda\, \frac{1}{|\mathcal{U}^B|} \sum_{i=1}^{|\mathcal{U}^B|} \mathrm{HR}(u^B_{i*}, v_*, k), \end{aligned} \tag{3}$$

where λ is a trade-off hyperparameter to control the contribution of rewards between target and surrogate systems via injection attack and queries.

4.7 Model Analysis

In this subsection, we will discuss our proposed method from time and feasibility aspects.

4.7.1 Time Complexity Analysis

The time complexity of making a decision in most existing reinforcement learning techniques is linear to the size of discrete action space. Specifically, all Deep Q-Network (DQN) based methods [12], [57] make a decision a via maximization operation taken over the entire action space, which becomes intractable for tasks where the size of the action space is large. For example, recommender systems usually have millions of items to recommend, i.e., $|\mathcal{V}|$, where $\mathcal{V}$ denotes the set of available items in the system. Thus, in our specific setting, the size of the action space is the number of users in the source domain (i.e, $|\mathcal{U}^B|$), which can also be in the millions.

Instead of picking the one user profile over all actions ($|\mathcal{U}^B|$) for user profile selection in source domain B, we propose to use hierarchical-structure policy gradient networks to seek a path from the root to a certain leaf of the hierarchical clustering tree. Particularly, our proposed method makes d choices based on policy networks with at most c output units, where d denotes the depth of the hierarchical clustering tree and c denotes the number of child nodes in the tree ($c \simeq |\mathcal{U}^B|^{1/d}$). Through such a method can significantly reduce the time complexity of *making a decision/action* from $\mathcal{O}(|\mathcal{U}^B|)$ to $\mathcal{O}(d \times |\mathcal{U}^B|^{1/d})$ in user profile selection component. Note that the policy gradient networks in our user profile crafting component do not suffer from the large discrete action space issue since only 10 actions are available. Apart from training the proposed RL-based black-box attacks, we also need the overhead of the construction of the clustering tree, where time complexity is $\mathcal{O}(|\mathcal{U}^B|^2)$. Note that once the clustering tree is well built, it can be saved for training RL based black-box attacks method (pre-processing step). Thus, we do not need to retrain the clustering tree, and it doesn't increase the time complexity for training our RL-based black-box attacks method. Therefore, the total time complexity is acceptable in practice.

4.7.2 The Feasibility of Our Proposed Method

Rather than considering cross-domain information to conduct adversarial attacks on the competitors' recommendation performance with malicious desires, our proposed methods have great potential to understand and assess the recommendations' vulnerability for our own systems and data [23], [37]. For example, as a multinational e-commerce platform, Amazon has different server centers or recommender systems for different countries or sub-markets (e.g., Amazon-US, Amazon-Canada, Amazon-UK, etc.), so as to store country-specific users' online behaviors and provide excellent recommendation services. These country-specific Amazon recommendation platforms are likely to share a lot of items. More importantly, with similar functionalities, users from these country-specific Amazon recommendation platforms also share similar behavior patterns/preferences towards items, e.g., Users who purchased Apple iPhone are highly likely to interact (click/purchase) with Apple AirPods in the future. In addition, it is also possible to conduct our proposed CopyAttack+ by using the historical user-item interactions data in different periods to evaluate the vulnerability of current recommender systems in some recommendation scenarios (e.g., video or music recommendations).

Adversarial attacks can usually happen in two phases [15], [30] (i.e., the model test and model training), namely **Evasion Attack** and **Poisoning Attack**. More specifically, evasion attack happens after the target model is well trained or in the test phase, while poisoning attack happens before the target recommender system model is trained. As for well-trained (i.e., static) systems, evasion attack is easily conducted in practice for user injection and reward query operations. For example, inductive Graph Neural Networks (GNNs) based recommender systems [19], [52] (i.e., PinSage) as the target model can provide immediate updates after injecting fake user profiles, where user and item representations can be learned via aggregating their local neighbors (items/users). Fake user profiles connecting various items are injected into the targeted systems for promoting specific items, and rewards can be easily obtained by querying spy users. For instance, this attacking setting is more suitable for time-sensitive recommendations, such as news recommendations which require real-time users' online behaviors to learn immediate user preferences for recommendations. On the other hand, poisoning attack needs to retrain the target systems after fake user injections, and further conduct reward queries on the system, which are time-consuming. Note that this attacking setting is infeasible in practice, since we need to know the update cycle or the exact update time on the target recommendation system under the black-box setting. Hence, poisoning attack is more suitable for evaluating the vulnerability of our own systems. For example, Amazon can take advantage of different submarkets (e.g., country-specific Amazon platforms) in the system or the historical data from different periods to test the robustness of their own recommender systems. Here, pre-training and fine-tuning techniques can be used to speed up the target systems' retraining process.

Table 2: Statistics of Three Datasets

Datasets (Target-Source)		ML10M-FX	ML20M-NF	BC-Amazon
Target Domain A	# of Users	19,267	38,087	6,136
	# of Items	6,984	8,325	5,446
	# of Interactions	437,746	838,491	33,661
Source Domain B	# of Users	93,702	478,471	44,002
	# of Overlapping Items	5,815	5,193	2,434
	# of Interactions	4,680,700	62,937,958	145,702

5 Experiment

5.1 Experimental Settings

5.1.1 Datasets

We use three cross-domain real-world datasets in our experiments to validate the performance of CopyAttack+.

- **MovieLen10M&Flixster (ML10M-FX)** [15]. Both datasets are popular online platforms for movie recommendation services, in which they have millions of movies. Users on these two platforms can watch them and give their personal comments (e.g., rating). Here, we take Movielen10M (ML10M) dataset as the target domain that we try to attack. Flixster (FX) dataset is treated as the source domain to be used to copy some user profiles to attack the Movielen10M (ML10M) target domain. In these two datasets, they have a lot of items in common, where overlapping items can be aligned by the movie names. We only keep the interactions that have a rating score of 5 and convert them to implicit feedback. After filtering, this cross-domain dataset (ML10M-FX) has 5,815 overlapping items.
- **MovieLen20M&Netflix (ML20M-NF)** [15]. These two datasets are also online platforms for movie recommendation services. We take the Movielen20M (ML20M) dataset as the target domain, and Netflix (NF) is the source domain. We identify movies with the same name and the published year. We then perform filtering operations similar to the ML10M-FX dataset. In this cross-domain dataset, we have 5,193 overlapping items.
- **Book-Crossing**[1]**&Amazon-Book**[2]**(BC-Amazon)**. These two datasets are collected from online platforms for book recommendation services. These books can be identified through the same ISBN and the published year. We take the Book-Crossing (BC) dataset as the target domain to

1. http://www2.informatik.uni-freiburg.de/~cziegler/BX/
2. https://cseweb.ucsd.edu/~jmcauley/datasets.html

be attacked, while Amazon-Book (Amazon) is the source domain. After a filtering operation similar to the ML10M-FX dataset, we obtain 2,434 overlapping items as a cross-domain dataset.

The statistics of these datasets are presented in Table 2. Note that we only keep the overlapping items in the source domain.

5.1.2 Evaluation Metrics

In order to evaluate the quality of the attacking task in the recommender systems, we use two popular accuracy metrics for Top-k recommendation [27], [44]: Hit Rate (HR@K) and Normalized Discounted Cumulative Gain (NDCG@K). We set K as 20, 10, and 5. Higher values of the HR@K and NDCG@K indicate a better attacking performance [15].

5.1.3 Attacking Environment - Target Recommender Systems

We consider the following two attacking environment:

- **Evasion Attack.** We adopt inductive Graph Neural Networks (GNNs) based recommender systems [19], [52] (i.e., PinSage) as our target model in this setting. After completing training, the final performance on the test set w.r.t HR@10 is 0.5490, 0.5474, and 0.2296 for the ML10M, ML20M, and BC datasets, respectively.
- **Poisoning Attack.** In this setting, we adopt NeuMF [27] as the target recommender system in our experiments.

We randomly sample 50 target items with less than 10 interactions to perform attacks on the target domain. Without being specifically mentioned, the main budget for the promotion task is the number of user profiles we inject into the target system, where we set the maximum budget as 30. The number of spy users in $\mathcal{U}_*^A$ is set to 50 for both datasets. To get the feedback (reward) from the target system (cf., Equation 1), we query spy users on the target system every three injections. *Note that the target recommender system is agnostic to attackers, including the detailed model architecture, parameters, and dataset.*

5.1.4 Parameter Settings

To train the target recommender systems, we randomly split the target domain datasets, where we have 80% as a training set for learning the parameters, 10% as a validation set to tune hyper-parameters, and 10% as the test set. The early stopping strategy was performed for training the target recommender system, where we stopped training if the HR@10 on the validation set decreased for five successive epochs. For all neural network methods, we randomly initialized model parameters with a Gaussian distribution, where the mean and standard deviation are 0 and 0.1, respectively. The learning rate and embedding size are set to 0.001 and 8. For all the recommendation algorithms and the compared attack methods, without additional mention, we use the default parameters given in their papers.

We implemented the proposed method on the basis of Tensorflow. The learning rate, embedding size, and discount factor are set to 0.001, 8, and 0.6, respectively. The hierarchical clustering tree is set to 3 layers for the Flixster dataset and six layers for the Netflix dataset. The pre-trained user and item representations in the source domain are obtained with Matrix Factorization (MF) method, where we use the same hyper-parameters to train (e.g., learning rate, embedding size, etc.). For simplicity, we also employ the most representative method MF to train the local surrogate system in the source domain, and other recommender systems can be explored in the future.

5.1.5 Baselines

Since most of the attacking methods in recommender systems are under the white-box setting, which is inappropriate for our task, we build some baselines to evaluate the performance of attacking under the black-box setting as follows:

- **RL-Generative**: Inspired by PoisonRec [41], we adopt reinforcement learning (RL) techniques to learn the strategy of generating fake user profiles by selecting items in the target domain without considering the information from the source domain.
- **RandomAttack**: This baseline is proposed to randomly sample cross-domain user profiles to attack the target recommender systems.
- **TargetAttack40/70/100**: Rather than randomly sampling user profiles from the source domain, this baseline is to randomly sample the user profiles with the target item from the source domain. Moreover, we apply the user profile crafting operations as our proposed model to reserve $40\%/70\%/100\%$ of user profiles.

In addition, we also build some baselines based on our proposed methods (CopyAttack) [15] to learn the strategy of copying user profiles from the source domain via reinforcement learning techniques as follows:

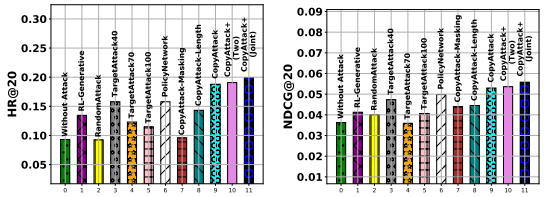
- **PolicyNetwork [42]**: This method directly utilizes the normal policy gradient network to select the user profiles in source domain B without considering the hierarchical clustering tree over the large discrete action space.
- **CopyAttack-Masking**: This method is used to evaluate the effectiveness of the masking mechanism by removing it in our proposed framework. In other words, the attack can select any user profile in the source domain. Note that the user profile crafting operation in this baseline is also removed, since the attack has a larger probability of selecting the user profile without the target item.
- **CopyAttack-Length**: This method is used to evaluate the effectiveness of user profile crafting operation in our proposed framework, where we remove the user profile crafting operation on the selected cross-domain user profiles and directly inject the raw user profiles into the target recommender system.

At last, we have proposed two improved models of our proposed CopyAttack based on two different training strategies, where one is **CopyAttack+(Two)** via two-stage training (Pre-training & Fine-tuning), and another one is **CopyAttack+(Joint)** via joint training.

5.2 Attacking Performance Comparison of Recommender Systems

We first compare the attacking performance of all methods on GNNs-based recommender system. Table 3 shows the overall attacking performance on different methods $w.r.t$ HR@K and NDCG@K on ML10M-FX, ML20M-NF, and BC-Amazon datasets. We have the following main findings.

- The performance of RandomAttack is close to the performance without attack, which implies that randomly sampling cross-domain user profiles without any strategies can not help promote the target items due to the large number of user profiles. When sampling user profiles with the sampling strategy where the user profiles should include the target items (i.e., TargetAttack-40/70/100), the performance can be improved significantly. In addition, when we constrain the sampling cross-domain user profile scope into the users who include the target items, this kind of method can obtain much

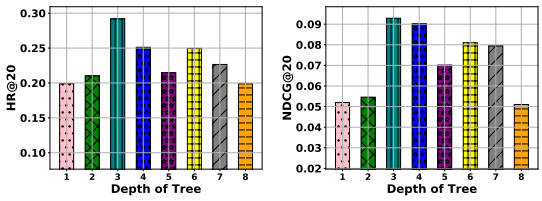


(a) ML10M-FX: HR@20 (b) ML10M-FX: NDCG@20

Figure 4: Performance comparison of different attacking methods for NeuMF recommender system on ML10M-FX dataset.

Table 3: Performance comparison of different attacking methods for GNNs-based recommender systems.

Datasets	Algorithms	HR@20	HR@10	HR@5	NDCG@20	NDCG@10	NDCG@5	# Average Items per User Profile
ML10M-FX	Without Attack	0.0378	0.0228	0.0220	0.0231	0.0195	0.0192	0
	RL-Generative	0.0487	0.0324	0.0233	0.0264	0.0222	0.0194	100
	RandomAttack	0.0391	0.0230	0.0222	0.0233	0.0195	0.0192	46
	TargetAttack40	0.1203	0.0583	0.0094	0.0353	0.0195	0.0041	495
	TargetAttack70	0.1772	0.0854	0.0354	0.0569	0.0341	0.0181	818
	TargetAttack100	0.1166	0.0520	0.0226	0.0369	0.0209	0.0114	1350
	PolicyNetwork	0.1936	0.0665	0.0250	0.0570	0.0258	0.0126	705
	CopyAttack-Masking	0.0376	0.0227	0.0220	0.0230	0.0195	0.0192	49
	CopyAttack-Length	0.0857	0.0434	0.0198	0.0282	0.0177	0.0101	1280
	CopyAttack	0.2596	0.1103	0.0415	0.0799	0.0425	0.0205	695
	CopyAttack+(Two)	0.2671	0.1123	0.0413	0.0831	0.0402	0.0209	675
	CopyAttack+(Joint)	0.2917	0.1313	0.0506	0.0928	0.0516	0.0258	683
ML20M-NF	Without Attack	0.0461	0.0043	0.0000	0.0115	0.0013	0.0000	0
	RL-Generative	0.1166	0.0390	0.0207	0.0421	0.0226	0.0170	100
	RandomAttack	0.0468	0.0050	0.0000	0.0118	0.0015	0.0000	124
	TargetAttack40	0.1016	0.0405	0.0056	0.0288	0.0133	0.0024	203
	TargetAttack70	0.1006	0.0402	0.0054	0.0285	0.0132	0.0023	321
	TargetAttack100	0.0581	0.0006	0.0000	0.0139	0.0002	0.000	593
	PolicyNetwork	–	–	–	–	–	–	–
	CopyAttack-Masking	0.0500	0.0045	0.0000	0.0125	0.0001	0.0000	133
	CopyAttack-Length	0.0655	0.0018	0.0000	0.0158	0.0005	0.0000	496
	CopyAttack	0.2704	0.124	0.0797	0.0969	0.0609	0.0467	255
	CopyAttack+(Two)	0.2737	0.1273	0.0791	0.0972	0.0627	0.0465	247
	CopyAttack+(Joint)	0.2844	0.1332	0.0832	0.103	0.0656	0.0493	249
BC-Amazon	Without Attack	0.1780	0.1288	0.0979	0.1042	0.0918	0.0819	0
	RL-Generative	0.1901	0.1339	0.0993	0.1089	0.0947	0.0836	10
	RandomAttack	0.1857	0.1330	0.0993	0.1068	0.0936	0.0828	3.4
	TargetAttack40	0.1997	0.1395	0.1110	0.1244	0.1093	0.1003	4.8
	TargetAttack70	0.2240	0.1539	0.1261	0.1429	0.1254	0.1165	6.4
	TargetAttack100	0.237	0.1764	0.1501	0.1632	0.1480	0.1397	11.0
	PolicyNetwork	0.2441	0.1880	0.1612	0.1728	0.1587	0.1502	10.9
	CopyAttack-Masking	0.2275	0.1576	0.1274	0.1431	0.1258	0.1161	5.2
	CopyAttack-Length	0.2741	0.1861	0.1383	0.1570	0.1350	0.1198	11.1
	CopyAttack	0.2926	0.2105	0.1766	0.196	0.1754	0.1647	10.9
	CopyAttack+(Two)	0.3021	0.2160	0.1817	0.2014	0.1799	0.1690	10.5
	CopyAttack+(Joint)	0.3289	0.2431	0.2135	0.2328	0.2116	0.2021	11.3



(a) ML10M-FX: HR@20 (b) ML10M-FX: NDCG@20

Figure 5: Effect of Depth on Hierarchical Clustering Tree.

better performance. This indicates the user profiles with the target item are informative to help perform attacking.

- When considering the length of cross-domain user profiles, the methods without target item constraints have a very low item budget (less than 50). When harnessing this constraint on different TargetAttack-40/70/100, we found that the methods on user profile without crafting perform the worse, demonstrated by the fact that both TargetAttack40 and TargetAttack70 perform better than TargetAttack100. It implies that introducing the user profile crafting is important to reduce the attack cost and noise. We will further analyze the budget from the number of cross-domain user profiles perspective in the next section.
- To better understand the proposed method, we compare it with PolicyNetwork, CopyAttack-Masking, and CopyAttack-Length. We can see that, for PolicyNetwork method, the performance of our method degrades when eliminating the effect of the hierarchical clustering tree. Note that PolicyNetwork method on ML20M-NF does not work, since we can not obtain its results in 48 hours, while we can obtain the results of others in a few hours. These observations suggest the power of the hierarchical clustering tree for selecting cross-domain user profiles in large-scale discrete action space. We also further study the impact of the hierarchical clustering tree in the model analysis section. Meanwhile, when we remove the user profile crafting component, the promotion performance decreases, and the item budget is huge, since the selected user profiles might introduce too much noise and degrade the performance. Moreover, when the masking mechanism is removed upon the CopyAttack-Length, CopyAttack-Masking performs much worse (and almost as poorly as RandomAttack). These results support that the masking mechanism and user profile crafting component are beneficial to select strong user profiles, and reduce the attack cost and noise for each user profile.
- Compared with the method of generative fake user profiles (i.e., RL-Generative), in most cases, copying user profiles strategies (e.g., TargetAttack-40/70/100, PolicyNetwork, CopyAttack-Masking, and CopyAttack-Length) can perform better on ML10M-FX and BC-Amazon datasets, while the performance of RL-Generative outperforms other methods on the ML20M-NF dataset. Note that our proposed methods CopyAttack and CopyAttack+ can outperform RL-Generative.
- As the extension of CopyAttack, CopyAttack+(Two/Joint) can achieve the best attacking performance in most cases, indicating the effectiveness of transferring adversarial attacking knowledge from the local surrogate system in the source domain to the target black-box system. Moreover, the performance of CopyAttack+(Joint) is better than that of CopyAttack+(Two) in two datasets, which is consistent with the observation in recent works [7], [48]. This observation might be attributed to mutually enhancing between target and surrogate systems in the joint training paradigm.

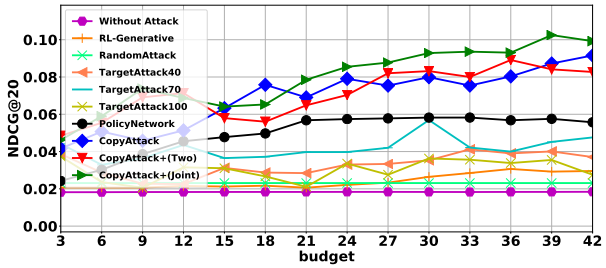
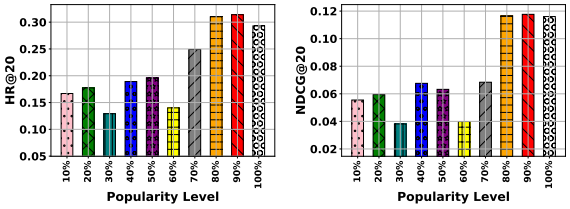


Figure 6: Effect of Budget (the Number of Cross-domain User Profiles) on ML10M-FX w.r.t NDCG@20.



(a) ML10M-FX: HR@20 (b) ML10M-FX: NDCG@20

Figure 7: Effect of Item Popularity on ML10M-FX dataset.

In addition, we also evaluate the attacking performance on the widely-used representative recommender system - NeuMF [27]. Due to space limitation, we only show the results on the ML10M-FX dataset, which have a similar observation to that on the ML20M-NF dataset. The results are shown in Figure 4. We can observe that our method consistently outperforms other methods under HR@20 and NDCG@20. For instance, our CopyAttack+(Two/Joint) methods improve the attacking performance of CopyAttack and consistently outperform all baselines. This again verifies the rationality and effectiveness of incorporating the adversarial transferability of user profiles by training a local surrogate system to mimic adversarial black-box attacks in the source domain.

5.3 Model Analysis

5.3.1 Effect of Depth on Hierarchical Clustering Tree

The hierarchical clustering tree, as discussed in Section 4.3.1, is investigated here, where we have shown the performance on CopyAttack+(Joint) when varying the depth of the tree (i.e., the value of d). We can observe in Figure 5 that $d = 3$ performs the best in terms of HR@20 and NDCG@20 in ML10M-FX dataset. The reason for this is believed to be the trade-off in terms of how detailed the clusters can be and the number of policy networks in the hierarchical clustering tree. This is because the deeper the tree, we have more policy networks that need to be learned. In comparison, shallower trees have fewer policy networks, but can harness the efficiency in terms of run-time and the ability to have a few large clusters to guide the source user profile selection.

5.3.2 Effect of Budget (The Number of User Profiles)

In this subsection, we investigate how the budget affects the performance of different attacking methods. Due to the limited space, we only report the results with the varied budget on the ML10M-FX dataset in Figures 6, while having similar observations in ML20M-NF. We first note, RandomAttack remains stable no matter how many user profiles. When the value of the budget increases, the performance of methods injecting user profiles with target items tends to increase first. And then TargetAttack40, TargetAttack70, TargetAttack100, and RL-Generative can not keep increasing when the budget becomes too large, while CopyAttack and CopyAttack+(Two/Joint) keeps increasing, since these methods perform queries and get more and more reward to train the attacker. In addition, we can observe that the fake user profiles generated via RL-Generative method are useless or even hurt the promotion attacks at the early stage of attacks. In most cases, CopyAttack+(Joint) outperforms CopyAttack, indicating the superiority of the developed surrogate recommender system to provide transferablely adversarial signals via the joint training paradigm.

Table 4: Effect of User Profile Augmentation.

Datasets	Methods	HR @20	HR @10	NDCG @20	NDCG @10
ML10M -FX	CopyAttack+ (Joint) without User Profile Augmentation	0.2767	0.1109	0.0861	0.0449
	CopyAttack+ (Joint)	0.2917	0.1313	0.0928	0.0516
ML20M -NF	CopyAttack+ (Joint) without User Profile Augmentation	0.2542	0.1255	0.0791	0.0528
	CopyAttack+(Joint)	0.2844	0.1332	0.103	0.0656

5.3.3 Effect of Item Popularity

In this subsection, we study what kinds of items are vulnerable to adversarial attacks in the proposed CopyAttack+(Joint). To achieve it, we group the items in the target domain based on their popularity. Specifically, we have 10 different groups, where each group accounts for 10% of items in the target domain. We then sample 50 target items from these 10 different groups, respectively. For instance, the first few groups (i.e., popularity level $10\% - 30\%$) can be treated as cold-start items. At last, we evaluate their performance on them. The results are given in Figure 7. The fluctuations in the results for the first few groups might be attributed to the instability caused by some hard target items in the "cold-start" group. Overall, we have the following observations: the target items with high popularity are more vulnerable to being attacked than cold-start items on two datasets, where our CopyAttack+(Joint) can achieve better attacking performance in the top 30% of items. We believe this is because most of the relatively popular items are already reasonably "hot" and attractive, thus CopyAttack+(Joint) can more easily push them further to the top with a limited budget as compared to less popular ("cold-start") items.

5.3.4 Effect of User Profile Augmentation

Different from injection attack in the target domain which directly injects the crafted version of the selected user profiles $\hat{P}^B_{u_i}$ into the target system, user profiles augmentation is used to further modify the crafted user profile ($\hat{P}^B_{u_i}$) via *items adding* and *items dropping* before injecting into the surrogate system. The results of CopyAttack+(Joint) on two datasets are summarized in Table 4. It can be observed that removing user profile augmentation (i.e., CopyAttack+(Joint) without user profile augmentation) reduces the performance of CopyAttack+(Joint). This verifies that the proposed user profile augmentation is important in assisting the RL agent (i.e., attacker) to obtain transferable signals from the local surrogate system.

6 Conclusion

In this work, we have proposed an attacking framework (CopyAttack+) to copy transferable cross-domain user profiles to perform adversarial attacks for black-box recommender systems. In particular, we have introduced a reinforcement learning-based black-box approach that makes use of policy gradient networks to first select users to copy, then refines/crafts their profiles, and finally injects them into the target domain. Moreover, inspired by the property of adversarial transferability, we propose to train a local surrogate system to simulate adversarial black-box attacks in the source domain, so as to grasp transferable signals to enhance attacking strategy learning in target black-box recommender systems. Our thorough experiments on three real-world datasets show the superiority of the proposed framework over a set of competitive baselines. Then, we furthermore performed model analysis to better understand the behavior of CopyAttack+. Our future work will be towards effective strategies for targeted attacks on items that need not be in the source domain and also for demotion and furthermore include more rich side information.

ACKNOWLEDGMENTS

The research described in this paper has been partly supported by NSFC (project no. 62102335), an internal research fund from The Hong Kong Polytechnic University (project no. P0036200 and P0042693), a General Research Fund from the Hong Kong Research Grants Council (Project No.: PolyU 15200021 and 15207322). Xiangyu Zhao is supported by Start-up Grant (No.9610565) for the New Faculty of the City University of Hong Kong and the CCF-Tencent Open Fund. Yao Ma is supported by the National Science Foundation (NSF) under grant number IIS-2153326.

REFERENCES

[1] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[2] H. Cai and F. Zhang, "Detecting shilling attacks in recommender systems based on analysis of user rating behavior," *Knowledge-Based Systems*, vol. 177, pp. 22–43, 2019.

[3] I. Cantador, I. Fernández-Tobías, S. Berkovsky, and P. Cremonesi, "Cross-domain recommender systems," in *Recommender systems handbook*. Springer, 2015, pp. 919–959.

[4] H. Chen, X. Dai, H. Cai, W. Zhang, X. Wang, R. Tang, Y. Zhang, and Y. Yu, "Large-scale interactive recommendation with tree-structured policy gradient," in *AAAI*, 2019.

[5] J. Chen, W. Fan, G. Zhu, X. Zhao, C. Yuan, Q. Li, and Y. Huang, "Knowledge-enhanced black-box attacks for recommendations," in *KDD*, 2022.

[6] K. Chen, P. P. Chan, and D. S. Yeung, "Shilling attack detection using rated item correlation for collaborative filtering," in *IEEE SMC*, 2018.

[7] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *ICML*. PMLR, 2020.

[8] X. Chen, W. Fan, J. Chen, H. Liu, Z. Liu, Z. Zhang, and Q. Li, "Fairly adaptive negative sampling for recommendations," *WWW*, 2023.

[9] K. Christakopoulou and A. Banerjee, "Adversarial attacks on an oblivious recommender," in *ACM RecSys*, 2019.

[10] R. Cohen, O. Sar Shalom, D. Jannach, and A. Amir, "A black-box attack model for visually-aware recommender systems," in *ACM WSDM*, 2021.

[11] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, "Adversarial attack on graph structured data," in *International conference on machine learning*. PMLR, 2018, pp. 1115–1124.

[12] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin, "Deep reinforcement learning in large discrete action spaces," *arXiv preprint arXiv:1512.07679*, 2015.

[13] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *IEEE CVPR*, 2018.

[14] W. Fan, T. Derr, Y. Ma, J. Wang, J. Tang, and Q. Li, "Deep adversarial social recommendation," in *IJCAI*, 2019.

[15] W. Fan, T. Derr, X. Zhao, Y. Ma, H. Liu, J. Wang, J. Tang, and Q. Li, "Attacking black-box recommendations via copying cross-domain user profiles," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021.

[16] W. Fan, Q. Li, and M. Cheng, "Deep modeling of social relations for recommendation," in *AAAI*, 2018.

[17] W. Fan, C. Liu, Y. Liu, J. Li, H. Li, H. Liu, J. Tang, and Q. Li, "Generative diffusion models on graphs: Methods and applications," *arXiv preprint arXiv:2302.02591*, 2023.

[18] W. Fan, X. Liu, W. Jin, X. Zhao, J. Tang, and Q. Li, "Graph trend filtering networks for recommendation," in *SIGIR*, 2022.

[19] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *ACM WWW*, 2019.

[20] W. Fan, Y. Ma, Q. Li, J. Wang, G. Cai, J. Tang, and D. Yin, "A graph neural network framework for social recommendations," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[21] W. Fan, Y. Ma, D. Yin, J. Wang, J. Tang, and Q. Li, "Deep social collaborative filtering," in *ACM RecSys*, 2019.

[22] W. Fan, H. Xu, W. Jin, X. Liu, X. Tang, S. Wang, Q. Li, J. Tang, J. Wang, and C. Aggarwal, "Jointly attacking graph neural network and its explanations," *ICDE*, 2023.

[23] W. Fan, X. Zhao, X. Chen, J. Su, J. Gao, L. Wang, Q. Liu, Y. Wang, H. Xu, L. Chen *et al.*, "A comprehensive survey on trustworthy recommender systems," *arXiv preprint arXiv:2209.10117*, 2022.

[24] M. Fang, N. Z. Gong, and J. Liu, "Influence function based data poisoning attacks to top-n recommender systems," in *ACM WWW*, 2020.

[25] M. Fang, G. Yang, N. Z. Gong, and J. Liu, "Poisoning attacks to graph-based recommender systems," in *ACSAC*, 2018.

[26] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *preprint arXiv:1412.6572*, 2014.

[27] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *ACM WWW*, 2017.

[28] M. Jiang, P. Cui, X. Chen, F. Wang, W. Zhu, and S. Yang, "Social recommendation with cross-domain transferable knowledge," *IEEE TKDE*, 2015.

[29] W. Jin, T. Derr, H. Liu, Y. Wang, S. Wang, Z. Liu, and J. Tang, "Self-supervised learning on graphs: Deep insights and new direction," *arXiv preprint arXiv:2006.10141*, 2020.

[30] W. Jin, Y. Li, H. Xu, Y. Wang, and J. Tang, "Adversarial attacks and defenses on graphs: A review and empirical study," *arXiv preprint arXiv:2003.00653*, 2020.

[31] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang, "Graph structure learning for robust graph neural networks," in *ACM KDD*, 2020.

[32] M. M. Khan, R. Ibrahim, and I. Ghani, "Cross domain recommender systems: a systematic literature review," *ACM Computing Surveys (CSUR)*, vol. 50, no. 3, pp. 1–34, 2017.

[33] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.

[34] S. K. Lam and J. Riedl, "Shilling recommender systems for fun and profit," in *ACM WWW*, 2004.

[35] B. Li, Y. Wang, A. Singh, and Y. Vorobeychik, "Data poisoning attacks on factorization-based collaborative filtering," in *Advances in neural information processing systems*, 2016, pp. 1885–1893.

[36] C. Lin, S. Chen, H. Li, Y. Xiao, L. Li, and Q. Yang, "Attacking recommender systems with augmented user profiles," in *CIKM*, 2020.

[37] H. Liu, Y. Wang, W. Fan, X. Liu, Y. Li, S. Jain, Y. Liu, A. K. Jain, and J. Tang, "Trustworthy ai: A computational perspective," *arXiv preprint arXiv:2107.06641*, 2021.

[38] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in *ICLR*, 2017.

[39] M. M. Naseer, S. H. Khan, M. H. Khan, F. Shahbaz Khan, and F. Porikli, "Cross-domain transferability of adversarial perturbations," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[40] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[41] J. Song, Z. Li, Z. Hu, Y. Wu, Z. Li, J. Li, and J. Gao, "Poisonrec: an adaptive data poisoning framework for attacking black-box recommender systems," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 157–168.

[42] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[43] J. Tang, H. Wen, and K. Wang, "Revisiting adversarially learned injection attacks against recommender systems," in *ACM RecSys*, 2020.

[44] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *ACM SIGIR*, 2019.

[45] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient langevin dynamics," in *ICML*, 2011.

[46] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, 1992.

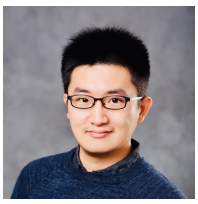
[47] J. Wu, W. Fan, J. Chen, S. Liu, Q. Li, and K. Tang, "Disentangled contrastive learning for social recommendation," in *CIKM*, 2022.

[48] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, "Self-supervised graph learning for recommendation," in *ACM SIGIR*, 2021.

[49] Z. Wu, J. Wu, J. Cao, and D. Tao, "Hysad: A semi-supervised hybrid shilling attack detector for trustworthy product recommendation," in *ACM KDD*, 2012.

[50] H. Xu, Y. Ma, H. Liu, D. Deb, H. Liu, J. Tang, and A. Jain, "Adversarial attacks and defenses in images, graphs and text: A review," *arXiv preprint arXiv:1909.08072*, 2019.
[51] G. Yang, N. Z. Gong, and Y. Cai, "Fake co-visitation injection attacks to recommender systems." in *NDSS*, 2017.
[52] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *ACM KDD*, 2018.
[53] S. Zhang, H. Yin, T. Chen, Q. V. N. Hung, Z. Huang, and L. Cui, "Gcn-based user representation learning for unifying robust recommendation and fraudster detection," in *SIGIR*, 2020.
[54] Y. Zhang, Y. Tan, M. Zhang, Y. Liu, T.-S. Chua, and S. Ma, "Catch the black sheep: unified framework for shilling attack detection based on fraudulent action propagation," in *IJCAI*, 2015.
[55] X. Zhao, H. Liu, W. Fan, H. Liu, J. Tang, and C. Wang, "Autoloss: Automated loss function search in recommendations," in *ACM KDD*, 2021.
[56] X. Zhao, L. Xia, J. Tang, and D. Yin, "Deep reinforcement learning for search, recommendation, and online advertising: a survey," *ACM SIGWEB Newsletter*, no. Spring, pp. 1–15, 2019.
[57] X. Zhao, L. Xia, L. Zhang, Z. Ding, D. Yin, and J. Tang, "Deep reinforcement learning for page-wise recommendations," in *ACM RecSys*, 2018.
[58] X. Zhao, L. Zhang, Z. Ding, L. Xia, J. Tang, and D. Yin, "Recommendations with negative feedback via pairwise deep reinforcement learning," in *ACM KDD*, 2018.
[59] W. Zhou, X. Hou, Y. Chen, M. Tang, X. Huang, X. Gan, and Y. Yang, "Transferable adversarial perturbations," in *ECCV*, 2018.

Wenqi Fan is a research assistant professor of the Department of Computing at The Hong Kong Polytechnic University (PolyU). He received his Ph.D. degree from the City University of Hong Kong (CityU) in 2020. From 2018 to 2020, he was a visiting research scholar at Michigan State University (MSU). His research interests are in the broad areas of machine learning and data mining, with a particular focus on Recommender Systems, Graph Neural Networks, and Trustworthy Recommendations. He has published innovative papers in top-tier journals and conferences such as TKDE, TIST, KDD, WWW, ICDE, NeurIPS, SIGIR, IJCAI, AAAI, RecSys, WSDM, etc. More information about him can be found at https://wenqifan03.github.io.



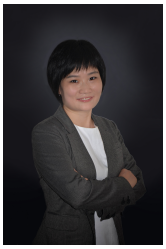
Xiangyu Zhao is an assistant professor of the school of data science at City University of Hong Kong (CityU). Before CityU, he completed his Ph.D. at Michigan State University. His current research interests include data mining and machine learning, especially on Reinforcement Learning and its applications in Information Retrieval. He has published papers in top conferences (e.g., KDD, WWW, AAAI, SIGIR, ICDE, CIKM, ICDM, WSDM, RecSys, ICLR) and journals (e.g., TOIS, SIGKDD, SIGWeb, EPL, APS). His research received ICDM'21 Best-ranked Papers, Global Top 100 Chinese New Stars in AI, CCF-Tencent Open Fund, Criteo Research Award, and Bytedance Research Award. He serves as top data science conference (senior) program committee members and session chairs (e.g., KDD, AAAI, IJCAI, ICML, ICLR, CIKM), and journal reviewers (e.g., TKDE, TKDD, TOIS, CSUR). More information about him can be found at https://zhaoxyai.github.io/.

Qing Li received the B.Eng. degree from Hunan University, Changsha, China, and the M.Sc. and Ph.D. degrees from the University of Southern California, Los Angeles, all in computer science. He is currently a Chair Professor at the Department of Computing, the Hong Kong Polytechnic University. His research interests include object modeling, multimedia databases, social media, and recommender systems. He is a Fellow of IEEE and IET, a member of ACM SIGMOD and IEEE Technical Committee on Data Engineering. He is the chairperson of the Hong Kong Web Society and is a steering committee member of DASFAA, ICWL, and WISE Society.

Tyler Derr is an Assistant Professor in the Department of Computer Science and Data Science Institute at Vanderbilt University (VU). He received his PhD (2020) in Computer Science from Michigan State University under the supervision of Dr. Jiliang Tang. His research interests focus on social network analysis, deep learning on graphs, and data science for social good. He has published, served as an organizer (e.g., KDD and WSDM), and received Best Reviewer Awards (e.g., ICWSM and WSDM) at top conferences in his domain. Additionally, was the recipient of the Fall 2020 Teaching Innovation Award from the School of Engineering at VU and Best Student Poster Award at SDM 2019. More details about him can be found at https://www.TylerDerr.com.

Yao Ma is an assistant professor in the Department of Computer Science at New Jersey Institute of Technology (NJIT). He got his PhD from Michigan State University in 2021 under the supervision of Dr. Jiliang Tang. Before that, he completed his MS (2016) in Statistics, Probability & Operations Research at Eindhoven University of Technology and BS (2015) in Mathematics and Applied Mathematics at Zhejiang University. He has published innovative works in top-tier conferences such as WSDM, ASONAM, ICDM, SDM, WWW, KDD and IJCAI. Updated information can be found at https://web.njit.edu/∼ym329/index.html.

Hui Liu is an Assistant Professor in the computer science and engineering department at Michigan State University. She received her PhD degree in Electrical Engineering from Southern Methodist University in 2015. Her research interests include trustworthy AI, designing data mining algorithms for wireless communication of smart devices, and applying machine learning and data mining in wireless communications.



Jianping Wang received the BS and the MS degrees in computer science from Nankai University, Tianjin, China in 1996 and 1999, respectively, and the PhD degree in computer science from the University of Texas at Dallas in 2003. She is a professor in the Department of Computer Science at the City University of Hong Kong. Her research interests include dependable networking, optical networks, cloud computing, service oriented networking, and data center networks.

Jiliang Tang is an associate professor in the computer science and engineering department at Michigan State University. Before that, he was a research scientist in Yahoo Research and got his Ph.D. from Arizona State University in 2015. His research focuses on data mining, machine learning and their applications on social, web, and education domains. He was the recipient of the 2021 ACSIC Rising Star Award, 2021 IEEE Big Data Security Junior Research Award, 2020 ACM SIGKDD Rising Star Award, 2020 Distinguished Withrow Research Award, 2019 NSF Career Award and 7 best paper (or runner up) awards including KDD2015 and WSDM2018. His dissertation won the 2015 KDD Best Dissertation runner-up and Dean's Dissertation Award. He has served as the editors and the organizers in prestigious journals (e.g., TKDD) and conferences (e.g., KDD, WSDM, and SDM). He has filed more than 10 US patents and has published his research in highly ranked journals and top conference proceedings, which received more than 17,000 citations with h-index 62 and extensive media coverage. More details about him can be found at https://www.cse.msu.edu/∼tangjili/.