Revisiting Item Promotion in GNN-Based Collaborative Filtering: A Masked Targeted Topological Attack Perspective

Yongwei Wang[1], **Yong Liu**[2*], **Zhiqi Shen**[3]

[1]Joint NTU-WeBank Research Centre on Fintech, Nanyang Technological University
[2]Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly, Nanyang Technological University
[3]School of Computer Science and Engineering, Nanyang Technological University
{yongwei.wang, stephenliu, zqshen}@ntu.edu.sg

Abstract

Graph neural networks (GNN) based collaborative filtering (CF) has attracted increasing attention in e-commerce and financial marketing platforms. However, there still lack efforts to evaluate the robustness of such CF systems in deployment. Fundamentally different from existing attacks, this work revisits the item promotion task and reformulates it from a targeted topological attack perspective for the first time. Specifically, we first develop a targeted attack formulation to maximally increase a target item's popularity. We then leverage gradient-based optimizations to find a solution. However, we observe the gradient estimates often appear noisy due to the discrete nature of a graph, which leads to a degradation of attack ability. To resolve noisy gradient effects, we then propose a masked attack objective that can remarkably enhance the topological attack ability. Furthermore, we design a computationally efficient approach to the proposed attack, thus making it feasible to evaluate large-large CF systems. Experiments on two real-world datasets show the effectiveness of our attack in analyzing the robustness of GNN-based CF more practically.

Introduction

Collaborative filtering-based recommendation systems (RS) aim to recommend a personalized list of top-K products (*a.k.a* items) to each user that matches best with her/his interests (Ekstrand et al. 2011; Deldjoo, Noia, and Merra 2021). Due to its effectiveness in promoting items, RS have been widely adopted in popular platforms, ranging from financial product marketing to short-video discovery and e-shopping (Wang et al. 2021). The mainstream paradigm of RS is collaborative filtering (CF), which assumes that users with similar behaviors are likely to show interest to similar items (He et al. 2017). As a result, CF attempts to exploit the observed user-item interactions, modeled as a user-item matrix (*a.k.a* a bipartite graph), to make predictions for the unobserved ones. To better capture such interactions, graph neural networks (GNN) (Kipf and Welling 2016) have attracted increasing attention in RS, and GNN-based RS achieve state-of-the-art performances in recommendation (Wang et al. 2019; He et al. 2020). Therefore, this work focuses on the GNN-based RS.

Instead of trying to improve a recommender's prediction accuracy, this work investigates how to maximally boost the ranking of a low-popularity item on a potentially deployed recommendation system (Liu and Larson 2021). An item attains higher popularity than another one if it is displayed in a larger number of users' recommendation lists. This task finds favorable applications in scenarios where a seller expects to maximize profits by promoting her/his items to as many potential users as possible. An intuitive solution is to encourage a number of users to show preference (*i.e.,* adding positive rating) to the target item, *e.g.,* by sending vouchers. However, there exist two crucial challenges to make the solution valid. The first challenge is how to ensure the newly-added ratings do contribute to the target item's popularity; and the other one is how to minimize the seller's budget (*e.g.,* vouchers) by limiting the number of ratings to be added.

The item promotion scenario is closely related to the robustness of a collaborative filtering recommendation system. Existing works attempt to address the challenges above by creating and injecting numerous fake users into the data, a technique known as shilling attacks or data poisoning attacks (Li et al. 2016; Tang, Wen, and Wang 2020; Fang, Gong, and Liu 2020; Wu et al. 2021b). However, these existing methods were generally specially designed for matrix factorization-based collaborative filtering recommenders, a type of conventional RS. Thus they are inapplicable to evaluating the robustness of an advanced GNN-based collaborative filtering RS. To our best knowledge, only limited studies propose data poisoning methods that may apply for GNN-based RS (Tang, Wen, and Wang 2020; Wu et al. 2021b).

Unfortunately, these recently proposed methods still demand adding a large number of fake users/ratings. Besides, due to the statistical differences in rating between real and fake users, fake users may be detected and removed to mitigate the attack ability. These issues hinder attacks to take place in real scenes. Therefore, it is urgent to develop practical and effective attacks to evaluate GNN-based collaborative filtering models in real scenes.

For the first time, this work proposes a simple yet effective item promotion method on GNN-based RS from a masked topological attack perspective. The developed objective function allows us to maximize a target item's popularity with only a small number of interaction changes in the user-item graph topology. Yet it is challenging to solve the optimization problem mainly due to its combinatorial nature. To effectively address this issue, we employ a gradient-based solution and

*Corresponding Author.

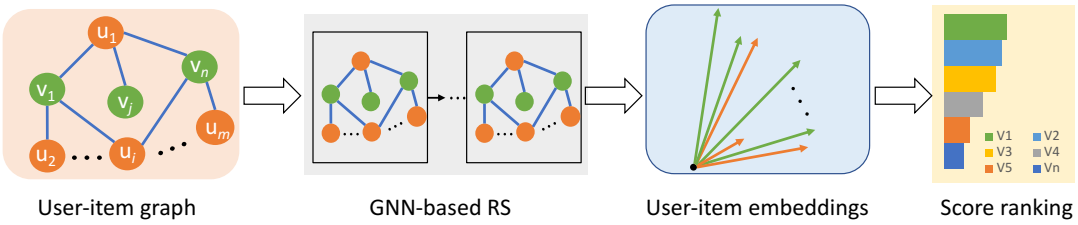

Figure 1: Illustration of an advanced GNN-based collaborative filtering model in recommender system. Users and items consist of a bipartite graph which is then input to a GNN-based collaborative filtering model to generate user and item embeddings. Items that match best with a user in the embedding space will appear in the user's recommendation list.

then propose a node masking mechanism to significantly enhance the attack ability. Moreover, we present a resource-efficient approach to enable our method to evaluate the robustness of large-scale GNN-based collaborative filtering systems.

Our major contributions can be summarized as follows:

- This work revisits the item promotion task in a GNN-based collaborative filtering system and re-formulates it as a targeted topological attack problem for the first time. This new formulation is fundamentally different from existing mainstream item promotion attacks in which we do not create and inject fake user profiles into the system.
- We develop a novel node masking mechanism that can remarkably boost the attack ability of vanilla gradient-based optimization solutions. To address the memory consumption issue in large-scale graphs in a real deployment, we further propose a resource-efficient approach that significantly reduces the memory cost from the quadratic to a linear one regarding the number of nodes.
- We conduct experiments on real-world recommendation datasets with advanced GNN-based collaborative filtering models. Our results reveal that our proposed methods can substantially promote an item's popularity even given limited perturbation budgets, and it is demonstrated to consistently outperform baseline attack methods.

Related Work

GNN-based Collaborative Filtering

Collaborative filtering is mainstream research in recommendation systems to predict users' preferences given collaborative signals. The essence of collaborative filtering is to learn user and item representations (*a.k.a* embeddings) jointly by leveraging the user-item interaction graph (Wu et al. 2020a). Then, items will be recommended to a user whose embeddings match the best with the user's embedding. Early explorations in collaborative filtering mainly focus on the matrix-factorization (MF) model (Hu, Koren, and Volinsky 2008; Koren, Bell, and Volinsky 2009) and its variants that encode the interaction history (Koren 2008; He et al. 2018). However, these methods only utilize a user's one-hop neighbors to generate the embeddings.

Inspired by the recent progress in GNN studies that exploit multi-hop neighbors in node embedding, GNN-based collaborative filtering methods have been proposed and achieved state-of-the-art performances. Wang et al. (Wang et al. 2019) proposed neural graph collaborative filtering (NGCF), a new collaborative filtering framework based on graph neural networks to capture the higher-order connectivity of user/item nodes. More recently, He et al. proposed LightGCN (He et al. 2020) to simplify and improve NGCF. Specifically, LightGCN removed the use of feature transformation and nonlinear activation in network design, since these two components were observed to have negative effects on model training. To supplement supervised learning, self-supervised graph learning (SGL) (Wu et al. 2021c) explored self-supervised learning and achieved state-of-the-art performance in the context of collaborative filtering to assist learning node and item representations.

Promoting Items in Collaborative Filtering

Although user-item interactions can effectively assist collaborative filtering, some of them may be intentionally falsified to mislead the recommender system. In the collaborative filtering regime, a most common threat is the item promotion attack (Li et al. 2016; Tang, Wen, and Wang 2020; Fang, Gong, and Liu 2020; Wu et al. 2021a), in which an attacker aims to influence a specific item recommendation list of users. More concretely, the attacker may be an incentive-driven item owner and craves to increase the chance of their own items to be recommended by a victim collaborative model.

Many existing item promotion attacks can be broadly classified into two categories: model-agnostic attacks and model-specific attacks. Model-agnostic attacks (*e.g.,* RandFilter attack (Lam and Riedl 2004), average attack (Lam and Riedl 2004)) do not assume knowledge of victim collaborative models, therefore they can apply to both conventional collaborative filtering models and GNN-based ones. In contrast, model-specific attacks design attacking strategies only applicable for certain types of collaborative filtering models. For example, Li et al. (Li et al. 2016) formulated an integrity attack objective for MF-based models (Cai, Candès, and Shen 2010; Jain, Netrapalli, and Sanghavi 2013), then solved the attack problem using gradient-based optimization methods. Fang et al. (Fang, Gong, and Liu 2020) proposed to utilize the influence function to select and craft fake users for top-K MF-based recommenders. Tang et al. observed that many model-specific attacks lacked exactness in gradient computation, then proposed a more precise solution to improve the attack ability (Tang, Wen, and Wang 2020). Wu et al. designed a neural network-instantiated influence module and

incorporated it into a triple generative adversarial network to craft fake users to launch attacks (Wu et al. 2021a).

Unfortunately, the model-agnostic attack methods were specially designed for MF-based models, thus they are not applicable to promoting items in GNN-based collaborative filtering RS. Meanwhile, recent studies show that graph neural networks can be vulnerable to adversarial attacks — some unnoticeable feature or edge perturbations may significantly reduce the performance of a GNN model (Zügner, Akbarnejad, and Günnemann 2018; Dai et al. 2018; Xu et al. 2019; Geisler et al. 2021). Intuitively, adversarial attacks can be leveraged to promote items in GNN-based recommendation models. However, these existing attacks focus on GNN-based classifiers, leaving the vulnerability of GNN-based collaborative filtering largely unexplored.

Indeed, there are three major differences between a GNN-based classification model (Wu et al. 2020b) and a GNN-based collaborative filtering model (Wu et al. 2020a). First, a classification decision can be made based on the logit of one node only, while a recommendation list returned by a GNN recommender involves ranking the prediction scores of all item nodes (Wang et al. 2019; He et al. 2020). Therefore, unlike fooling one node only in a GNN-classifier attack, attacking a GNN recommender requires to manipulating predictions of multiple nodes simultaneously. Thus, it makes the latter case special and more challenging. Second, a node classification model consists of both edges and associative semantic features, and manipulating features can effectively enhance the attack ability (Zügner, Akbarnejad, and Günnemann 2018). By contrast, a GNN recommender usually only contains user-item interactions, thus increasing the attack difficulty. Third, input graph formats and their scales are also different. The input to the former model is often a small symmetric graph, while there often includes a large bipartite graph (e.g., tens of thousands of user and item nodes) in the latter one (He et al. 2020; Wu et al. 2021c). Therefore, memory-efficient attacks remain to be developed.

Methodology

Preliminaries

This study focuses on the LightGCN architecture (He et al. 2020), a state-of-the-art backbone in GNN-based collaborative filtering models (Wu et al. 2021c; Zhou et al. 2021; Zhang et al. 2022).

Let $\mathcal{U} = \{u_1, \cdots, u_M\}$ and $\mathcal{I} = \{i_1, \cdots, i_N\}$ denote the set of M users and N items, respectively. Let $\mathcal{O}^+ = \{r_{ui} | u \in \mathcal{U}, i \in \mathcal{I}\}$ denote the interaction feedback of user u for item i. Here we consider implicit feedback as in many real recommenders (Tang, Wen, and Wang 2020), i.e., $r_{ui} \in \{0, 1\}$, where 1 indicates a positive recommendation and 0 means an unknown entry. Denote the user-item rating binary matrix $\mathbf{R} \in \mathbb{R}^{M \times N}$ with entries as r_{ui} $(u = 1, \cdots, M;\ i = 1, \cdots, N)$. Then, we can construct a bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$ and $\mathcal{E} = \mathcal{O}^+$ represent the vertex (or node) set and edge set, respectively.

GNN-based collaborative filtering leverages the user-item graph $\mathcal{G}$ to learn embeddings. To be specific, it performs neighborhood aggregations iteratively on $\mathcal{G}$ to update a node's representation (He et al. 2020; Wu et al. 2021c). The propagation rule for the $l-$th $(l = 1, \cdots, L)$ layer can be formally defined as,

$$\begin{aligned} \mathbf{z}_u^{(l)} &= g\Big(\sum_{j \in \mathcal{N}_u} \widetilde{\mathbf{R}}_{u,j} \cdot \mathbf{z}_j^{(l-1)} \Big) \\ \mathbf{z}_i^{(l)} &= g\Big(\sum_{j' \in \mathcal{N}_i} \widetilde{\mathbf{R}}^T \cdot \mathbf{z}_{j'}^{(l-1)} \Big) \end{aligned} \tag{1}$$

where $\mathbf{z}_u^{(l)} \in \mathbb{R}^d$ denotes the feature vector of user u at layer l, $\mathbf{z}_u^{(0)} = \mathbf{w}_u \in \mathbb{R}^d$ denotes the trainable and randomly initialized feature vector for user u, $g(\cdot)$ is an activation function which is often set as an identity function in recent works, $\mathcal{N}_u = \{j | (u, j) \in \mathcal{E}\}$ represents items within the neighborhood of a user u, and $\widetilde{\mathbf{R}}_{u,j}$ denotes the (u, j)-th entry of a normalized user-item rating matrix $\widetilde{\mathbf{R}}$, i.e., $\widetilde{\mathbf{R}} = \mathbf{\Lambda}_L^{-1/2}\ \mathbf{R}\ \mathbf{\Lambda}_R^{-1/2}$. Here $\mathbf{\Lambda}_L \in \mathbb{R}^{M \times M}$ is a diagonal matrix with (u, u)-th entry as the degree of user u, $\mathbf{\Lambda}_R \in \mathbb{R}^{N \times N}$ denotes a diagonal matrix with (i, i)-th entry as the degree of item i. Similarly, we have notations for item i's propagation rule by changing the subscript from u to i.

After obtaining representations of L layers, the embedding of a user (or item) node can be constructed by combining the representations computed at each layer,

$$\mathbf{z}_u = f_{comb}(\mathbf{z}_u^{(l)}),\ \mathbf{z}_i = f_{comb}(\mathbf{z}_i^{(l)}),\ \forall l \in [L] \tag{2}$$

where $f_{comb}(\cdot)$ denotes a representation combination function that may adopt representations from the final layer only, or utilize concatenation or weighted sum of representations from different layers (Wang et al. 2019; He et al. 2020; Wu et al. 2021c).

In GNN-based collaborative filtering, a typical way to obtain a recommendation prediction is by matching the embedding of a user with that of an item,

$$\hat{r}_{u,i} = \big\langle \mathbf{z}_u,\ \mathbf{z}_i \big\rangle \tag{3}$$

where $< \cdot >$ denotes an inner product, $\hat{r}_{u,i}$ is a rating score estimate that indicates how likely a user u would select an item i. The model training can be framed into a supervised learning setting or a self-supervised learning paradigm. In deployment, a pretrained collaborative filtering model first predicts rating scores for each user, then it ranks and recommends items with top-K highest scores to a user.

Targeted Topological Attacks

In a deployed recommender, a malicious item owner intends to promote a target item t to as many users as possible, a scenario called an item promotion attack. Different from existing works that attempt to craft and inject fake users to graph $\mathcal{G}$, this work formulates it from a targeted topological attack perspective. We assume the attacker (*i.e.,* malicious item owner) has white-box access to $\mathcal{G}$. We also assume that the attacker has the capability to persuade a few users to give high positive ratings to the target item (*e.g.,* sending vouchers). The attacking process can be formally defined as,

$$\max_{\mathbf{R}^{atk}} \mathcal{L}_{atk}\Big(f_\theta(\mathbf{R}^{atk})_t\Big) \quad \text{s.t. } ||\mathbf{R}^{atk} - \mathbf{R}||_0 \leq \Delta, \quad \mathbf{R}^{atk} \in \{0,1\}^{M\times N} \tag{4}$$

where $\mathcal{L}_{atk}$ denotes an objective function to improve the ranking of a specific item t, f_θ denotes a GNN-based collaborative filtering model parameterized by θ, $\mathbf{R}^{atk}$ denotes a manipulated user-item rating matrix by an attacker, $||\cdot||_0$ is an ℓ_0 norm, and Δ represents a perturbation budget for the attack, *i.e.,* the maximum number of user-item interactions allowed to manipulate.

For an arbitrary user $u \in \mathcal{U}$, denote the recommendation prediction scores for each item $i \in \mathcal{I}$ as $\boldsymbol{s}_u = [\hat{r}_{u,1}, \cdots, \hat{r}_{u,N}]$. The collaborative filtering system then ranks all entries in $\boldsymbol{s}_u$ and selects top-K items and recommends them to user u, denoted as $\Omega_K^u = [i_1^u, \cdots, i_K^u]$. Often, the target item t does not lie in the recommendation set Ω_K^u, thus requiring to be promoted into the set with a minimal perturbation budget.

To achieve the item promotion purpose, we formulate an objective function as,

$$\mathcal{L}_{atk} = \frac{1}{M}\sum_{u\in\mathcal{U}}\Big[\lambda \text{log}\sigma(\hat{r}_{u,t}) - (1-\lambda)\sum_{j\in\Omega_K^u, j\neq t}\text{log}\sigma(\hat{r}_{u,j})\Big] \tag{5}$$

where λ is a hyperparameter to balance score penalizations, $\sigma(\cdot)$ denotes a sigmoid activation function $\sigma(x) = 1/\big(1+\exp(-x)\big)$ that converts predicted score values to the $(0,1)$ interval.

By substituting Eq. (5) into Eq. (4), we obtain a constraint optimization problem in the white-box targeted topological attack setting. Unfortunately, this is a combinatorial problem and finding an exact solution is NP-hard in computational complexity. Alternatively, similar to white-box adversarial attacks on images, we can leverage the gradient-based optimization to approximate the solution (Goodfellow, Shlens, and Szegedy 2015; Madry et al. 2018).

First, we relax a discrete $\boldsymbol{R}$ as a continuous multivariable. We then compute its saliency map based on the gradients of $\boldsymbol{R}^{atk}$ with respect to each variable. The saliency map measures the contributions of every pair of user-item interactions to maximize the attack objective function in Eq. (5). To satisfy the perturbation budget in Eq. (4), we select Δ users that have highest intensity in the saliency map, and do a gradient ascent to update $\boldsymbol{R}$. Specifically, the topological attack can be expressed as,

$$\boldsymbol{R}^{atk} = \mathcal{P}\Big(\boldsymbol{R} + \boldsymbol{M}\odot \text{sign}\Big(\nabla_{\boldsymbol{R}}\mathcal{L}_{atk}\Big)\Big) \tag{6}$$

where $\mathcal{P}$ is a projection operator that clips the perturbed $\boldsymbol{R}$ back to the $\{0,1\}^{M\times N}$ space, $\odot$ denotes an element-wise product, sign$(\cdot)$ is a sign function, $\boldsymbol{M} \in \{0,1\}^{M\times N}$ denotes a binary mask that can be computed based on the gradient saliency map,

$$\boldsymbol{M}_{u,i} = \begin{cases} 1, & if\ \Big([\nabla_{\boldsymbol{R}}\mathcal{L}_{atk}]_{u,i} > 0\Big) \cap \Big((u,i)\in\Omega_g\Big) \\ 0, & otherwise \end{cases} \tag{7}$$

where Ω_g is an index set that contains the top-Δ largest values of $\nabla_{\boldsymbol{R}}\mathcal{L}_{atk}$ and it can be formally defined as,

$$\underset{\Omega_g\subset\mathcal{G}, |\Omega_g|=\Delta}{\arg\max} \sum_{(u,i)\in\Omega_g} \nabla_{\boldsymbol{R}_{u,i}}\mathcal{L}_{atk} \tag{8}$$

A physical interpretation of the binary mask $\boldsymbol{M}$ is how new user-item interactions should be established in order to maximally promote a target item.

Node Masking Mechanism

As described in the previous section, we utilize a gradient-based optimization approach to approximate the optimal solution to Eq. (4). Given a limited perturbation budget, we select and create user-item pair candidates that achieve the highest responses in the gradient saliency map. However, the attack ability can be further improved due to potential issues in this approach. First, the gradient estimates can be noisy due to the discrete nature of variables in $\boldsymbol{R}$. Also, the quantization errors due to the utilization of the sign function may hamper the effectiveness of gradient ascent.

Notice that the derivative $\nabla_{\boldsymbol{R}_{u,i}}\mathcal{L}_{atk}$ in Eq. (6) is a summation of individual derivatives computed from the log scores of M user nodes w.r.t. the binary variable $\boldsymbol{R}_{u,i}$. To negate noisy effects in gradient estimates, an intuitive way is to adopt a subset of user nodes by masking out unimportant ones. We prefer preserving nodes with high predicted scores $\hat{r}_{u,t}$ than those with lower ones $\hat{r}_{u',t}$ for the target item t. This is because item t is more likely enter into the top-K recommendation list of the user u than user u' after a single step gradient ascent.

We design a pre-filtering step for the node masking mechanism based on predicted scores from the GNN-based collaborative filtering system. Specifically, we compose a user subset $\mathcal{U}' \subset \mathcal{U}$ that satisfies,

$$\mathcal{U}' = \Big\{u \mid u\in\mathcal{U},\ \sigma(\hat{r}_{u',t}) \geq \gamma\Big\} \tag{9}$$

where γ denotes a masking threshold parameter. Then, a masked objective function $\mathcal{L}_{atk}^m$ can be expressed as,

$$\mathcal{L}_{atk}^m = \frac{1}{|\mathcal{U}'|}\sum_{u\in\mathcal{U}'}\Big[\lambda\text{log}\sigma(\hat{r}_{u,t}) - (1-\lambda)\cdot \sum_{j\in\Omega_K^u, j\neq t}\text{log}\sigma(\hat{r}_{u,j})\Big] \tag{10}$$

Clearly, Eq. (5) is a special case of Eq. (10) by setting γ to be 0. It is worth noting that our node masking mechanism is clearly different from the masked strategy used in (Geisler et al. 2021). First of all, the research tasks are different: our work targets an item promotion task in a collaborative filtering system that involves ranking, while work (Geisler et al.

2021) deals with a classification task. Second, we rank predicted scores and mask out user nodes with low prediction confidence below a threshold, while work (Geisler et al. 2021) necessitates a comparison with the ground truth labels of nodes and removes the incorrectly classified ones. Moreover, the objective functions are fundamentally different because of different tasks.

Scaling to Large-scale Graphs

The gradient ascent-based solution in Eq. (6) requires computing gradients w.r.t. each entry in $\boldsymbol{R}$. This approach works well on a graphics processing unit (GPU) card with limited memory when the input user-item interaction matrix $\boldsymbol{R}$ is of relatively small size. In real deployment scenes, however, with a dense gradient $\nabla_{\boldsymbol{R}}\mathcal{L}_{atk}^{m}$, computational issues can arise if it involves a large-scale graph that consists of thousands even millions of user and item nodes.

Proposition 1. *In a one-layer GNN-based collaborative filtering model defined in Eq. (1), the partial derivatives satisfy* $\nabla_{\boldsymbol{R}_{u,t}}\mathcal{L}_{atk}^{m} > \nabla_{\boldsymbol{R}_{u,j}}\mathcal{L}_{atk}^{m}, (j \neq t)$ *if* $< \mathbf{z}_u^{(0)}, \mathbf{z}_i^{(0)} > \to 1$ *for* $u = 1, \cdots, M, i = 1, \cdots, N$.

Remark. The analysis above indicates that there only necessitates computing gradients with respect to the target item t, i.e., $\nabla_{\boldsymbol{R}_{u,t}}(u = 1, \cdots, M)$ in Eq. (6). Empirically, we observe that $\nabla_{\boldsymbol{R}_{u,t}}\mathcal{L}_{atk}^{m} > \nabla_{\boldsymbol{R}_{u,j}}\mathcal{L}_{atk}^{m}, (j \neq t)$ also holds for the multi-layer well trained GNN-based collaborative filtering models. In this way, the memory consumption can be reduced from $\mathcal{S}(M \times N)$ to $\mathcal{S}(M)$, which is a significant cost reduction especially when N is a large value.

In implementation (*e.g.,* using PyTorch (Paszke et al. 2019)), we can split matrix $\boldsymbol{R}$ into three separate tensors: $\boldsymbol{R} = \left[\boldsymbol{R}_{:,:t-1}, \boldsymbol{R}_{:,t}, \boldsymbol{R}_{:,t+1:N}\right]$, where only tensor $\boldsymbol{R}_{:,t}$ requires a gradient computation. Then we do a regular forward process to compute the targeted loss as in Eq. (10), and then backpropagate gradients to tensor $\boldsymbol{R}_{:,t}$.

The algorithm of the proposed method is presented in Algorithm 1.

Algorithm 1: The proposed scalable algorithm for masked targeted attacks on GNN-based collaborative filtering models.

Data: A pretrained f_θ that consists of $\mathbf{w}_u$ and $\mathbf{w}_i$, data $\boldsymbol{R} \in \mathbb{R}^{M \times N}$, target item t, perturbation budget Δ, parameter λ, masking threshold γ.
Result: A perturbed $\boldsymbol{R}^{atk}$ that satisfies Eq. (8).
// Initialization and forward
Initialize embeddings $\mathbf{z}_u^{(0)} = \mathbf{w}_u$, $\mathbf{z}_i^{(0)} = \mathbf{w}_i$, set $l = 1$;
Rewrite $\boldsymbol{R}$: $\boldsymbol{R} \leftarrow \left[\boldsymbol{R}_{:,:t-1}, \boldsymbol{R}_{:,t}, \boldsymbol{R}_{:,t+1:N}\right]$;
Normalize $\boldsymbol{R}$: $\widetilde{\mathbf{R}} \leftarrow \boldsymbol{\Lambda}_L^{-1/2}\ \mathbf{R}\ \boldsymbol{\Lambda}_R^{-1/2}$;
while $l \leq L$ **do**
  Compute users embeddings at layer l:
  $\mathbf{z}_u^{(l)} \leftarrow g\Big(\boldsymbol{R}_{:,:t-1} \cdot \boldsymbol{z}_i^{l-1}[:t-1,:] + \boldsymbol{R}_{:,t} \cdot \boldsymbol{z}_i^{l-1}[t,:] + \boldsymbol{R}_{:,t:N} \cdot \boldsymbol{z}_i^{l-1}[t:N,:]\Big)$;
  Compute items embeddings at layer l: $\mathbf{z}_i^{(l)} \leftarrow \Big[g\big(\boldsymbol{R}_{:,t}^T \cdot \boldsymbol{z}_u^{l-1}\big); g\big(\boldsymbol{R}_{:,t}^T \cdot \boldsymbol{z}_u^{l-1}\big); g\big(\boldsymbol{R}_{:,t:N}^T \cdot \boldsymbol{z}_u^{l-1}\big)\Big]$;
  $l \leftarrow l + 1$;
end
Compute final user and item embeddings using Eq. (2);
// Backward for gradient computation
Compute masked targeted loss $\mathcal{L}_{atk}^{m}$ using Eq. (10);
Compute $\nabla_{\boldsymbol{R}_{:,t}}\mathcal{L}_{atk}^{m}$ using autograd, and set the rest gradients in $\nabla_{\boldsymbol{R}}$ as 0;
Find top-Δ largest values in $\nabla_{\boldsymbol{R}}$, and compute binary mask $\boldsymbol{M}$ using Eq. (7);
Compute $\boldsymbol{R}^{atk}$ using gradient ascent in Eq. (6);
Return: The perturbed $\boldsymbol{R}^{atk}$.

Experiments

In this section, we demonstrate the effectiveness of our item promotion attack method in GNN-based collaborative filtering systems. We first introduce the experimental setup, then conduct experiments on two real-world datasets for empirical validation under different settings.

Experimental Setup

Datasets: We conduct experiments on Gowalla (Cho, Myers, and Leskovec 2011) and Yelp2018 (He et al. 2020), two commonly-used datasets for recommendation (Wang et al. 2019; He et al. 2020). For both datasets, we use the preprocessed dataset with train/test split following work (He et al. 2020). Gowalla contains 29,858 users and 40,981 items, with an overall number of user-item interactions as 1,027,370. Yelp2018 includes 31,667 users and 38,047 items and has 1,561,406 user-item interactions in total.

Models: We evaluate our method on the LightGCN and variant models, the state-of-the-art GNN-based collaborative filtering recommenders. LightGCN is trained on Gowalla and Yelp2018 datasets, respectively, with PyTorch implementations officially released by (He et al. 2020). We adopt default hyperparameters as shown in the official implementation. After sufficient training, LightGCN achieves good performances on both datasets. The recommendation performances on the clean datasets are reported in Appendix.

Evaluation Protocols: We demonstrate the attack ability to promote a target item on low popular items on Gowalla (Cho, Myers, and Leskovec 2011) and Yelp2018 datasets. For a well-trained collaborative filtering model, an item with fewer positive ratings in $\boldsymbol{R}$ will be less popular than another that has more positive feedback. Therefore, we use the degree of an item to quantify its popularity. To be specific, we compose three low-popular target item sets based on an item's original degree. The percentile of the three item sets are: Q_{10}, Q_{30}, Q_{50}, respectively. For each item from the three item sets, two perturbation budgets are defined: $\Delta_s^1 = deg(Q_{65}) - deg(Q_s)$ and $\Delta_s^2 = \bar{d} - deg(Q_s)$, where $\bar{d}$ is the mean degree, $deg(q)$ denotes the degree of an item that lies in a percentile q, and $s \in \{10, 30, 50\}$. To better reflect the trend of item promotion improvements, we also adopt a continually varying number of perturbation budgets. The perturbation budgets are shown in Table 1.

For the quantitative measure, we utilize the hit number (HN) to evaluate the item promotion ability. For a specific target item, $HN@50$ is defined as the frequency that users have this item to be displayed in their top-50 recommendation list. To be more accurate, we define a pruned hit number ($PHN@50$) metric that removes the number of newly-added

Dataset	Δ^1_{10}	Δ^2_{10}	Δ^1_{30}	Δ^2_{30}	Δ^1_{50}	Δ^2_{50}
Gowalla	7	12	5	10	3	8
Yelp2018	17	25	13	21	8	16

Table 1: Perturbation budgets for topological attacks.

users from the $HN@50$ metric. For reproducibility, we report the averaged $HN@10$ and $PHN@K$ over 30 number of randomly selected target items from three low popular item sets individually. All reported results utilize fixed hyperparameters $\lambda = 0.5$, $\gamma = 0.95$.

Comparison Methods: We utilize and modify existing model-agnostic attacks on collaborative filtering models and use them as our baseline methods (e.g., random attack (Lam and Riedl 2004)). Please note that we cannot compare with recent model-specific attacks (*e.g.,* (Li et al. 2016; Fang, Gong, and Liu 2020; Tang, Wen, and Wang 2020)), because they were developed for MF-based CF, and they do not apply for attacking GNN-based CF models. Besides, our considered settings are dramatically different from model-specific attacks in that these methods require injecting a set of fake users into training data, while our method focuses on selecting and modifying a small number of existing users. Besides RandFilter, we also design two other heuristic attacks as our baseline attacks. The compared methods are:

- **RandFilter**: RandFilter was originally used for explicit ratings (Lam and Riedl 2004), and we modify it for implicit rating. We randomly select Δ users and asked them to give positive ratings to the target item t.
- **IUFilter**: From the user's perspective, users that have frequent purchasing histories tend to be more influential in positive ratings. Therefore we choose top-Δ such users and let them rate positively to the target items.
- **RUFilter**: RUFilter selects users that have top-Δ predicted rating scores for item t and put corresponding entries in $\boldsymbol{R}$ as 1 in the implicit recommendation setting.

Promoting Items in White-box Scenes

An attacker is first assumed to have white-box access to the pretrained GNN-based CF model. The attacker can use three baseline attacks and the proposed attack (in Algorithm 1) to conduct item promotion attacks. Three sets of low popularity items (i.e., Q_{10}, Q_{30}, Q_{50}) will be evaluated with different perturbation budgets. The comparison results are reported in Table 2.

From Table 2, we can observe that our proposed method achieves the highest averaged $PHN@50$ for all settings, substantially outperforming baseline methods. For example, when target items are from Q_{10} and a perturbation budget as Δ^1_{10} on Gowalla, the $PHN@50$ values are 0.7, 0.5, 9.1 for RandFilter, IUFilter and RUFilter, respectively; while our method achieves a $PHN@50$ as $\mathbf{41.4}$, which is $\mathbf{4.5}\times$ larger than the second best method. The superiority of our method is even more prominent for target items from Q_{10} with the perturbation budget as Δ^2_{10}, *i.e.,* $\mathbf{7.5}\times$ stronger than RUFilter. Although the item promotion ability tends to decrease from Q_{10} to Q_{50}, the performance of our method is still significantly better than all baseline methods. We can arrive at a same conclusion for experiments on Yelp2018.

In addition, we vary the perturbation budgets gradually and show in Fig. (2) the comparison results. Fig. (2) reveals that as the perturbation budgets increase, the promotion ability of our method increases dramatically, and the performance gap becomes larger compared to baseline methods. This observation indicates that GNN-based CF model is vulnerable to the proposed masked topological attack, particularly with a relatively large adversarial perturbation budget.

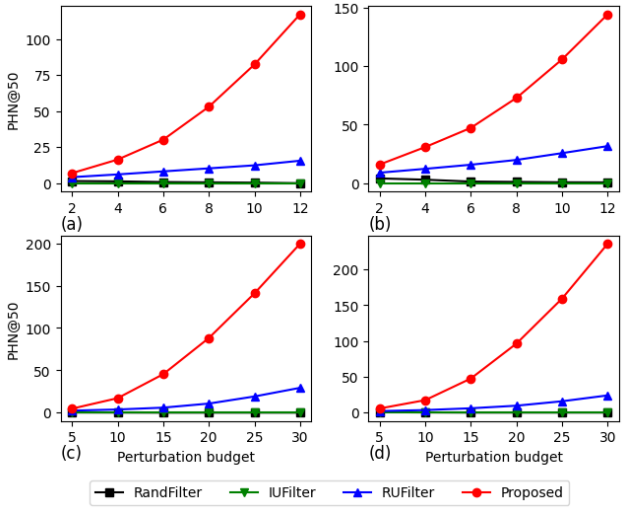

Figure 2: Performance comparisons with a gradually varying number of budgets on low-popular items from Gowalla and Yelp2018 datasets. (a) and (b) display $PHN@50$ results for target items from Gowalla with Q_{10} and Q_{30}, and (c) and (d) show $PHN@50$ from Yelp2018 with Q_{10} and Q_{30}, respectively.

Promoting Items in Black-box Scenes

In addition to white-box attacks, we study the effectiveness of our method in a black-box setting, in which an attacker is assumed to have no knowledge of the victim models. In this setting, an attacker first adopts the pertrained model as a substitute model (sub. model) and creates a perturbed graph for a target item. The attacker then attempts to promote the target item on an unknown collaborative filtering model. Based on (He et al. 2020), we obtain three victim models by setting a different number of layers and the length of embeddings. Please refer to Appendix for a detailed setup.

In Fig. 3, we compare and visualize the attack performance of different methods on three substitute models (i.e., sub. models 1–3) on Gowalla. Although the three substitute models are different from the source model, clearly, we can conclude that the proposed method still achieves satisfactory attack performances.

Effectiveness of Node Masking Mechanism

This section evaluates the performance of our method with different choices of parameter γ. Specifically, we vary γ from

Dataset	Attack	Low popularity items					
		Q_{10}		Q_{30}		Q_{50}	
		PHN@50 (Δ^1_{10})	*PHN@50* (Δ^2_{10})	*PHN@50* (Δ^1_{30})	*PHN@50* (Δ^2_{30})	*PHN@50* (Δ^1_{50})	*PHN@50* (Δ^2_{50})
Gowalla	RandFilter	0.7	0.5	2.2	0.8	3.9	1.8
	IUFilter	0.5	0.3	1.9	0.7	4.5	1.5
	RUFilter	9.1	15.6	14.2	25.7	17.1	29.0
	Proposed	**41.4**	**117.6**	**39.1**	**106.3**	**28.9**	**87.7**
Yelp2018	RandFilter	0	0	0.2	0.1	1.2	0.4
	IUFilter	0	0	0.2	0.1	0.8	0.2
	RUFilter	7.4	19.1	4.8	10.5	7.6	16.5
	Proposed	**60.6**	**140.8**	**32.6**	**106.7**	**20.7**	**75.0**

Table 2: Performance comparisons of different attacks in improving a target item's popularity on Gowalla and Yelp2018 datasets. Three low popularity item sets (Q_{10}, Q_{30}, Q_{50}) are used for performance evaluation with perturbation budgets as Δ^1_s and $\Delta^2_s (s = 10, 30, 50)$. $PHN@50$ is averaged over 30 randomly selected target items at each item set.

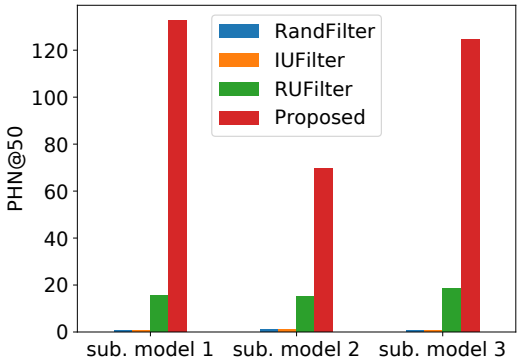

Figure 3: Visualization of attack performance comparisons on three different substitute models on Gowalla.

0.05 to 0.95 and compare $PHN@50$. Figure 4 displays the performance curve using target items from Q_{10} item set.

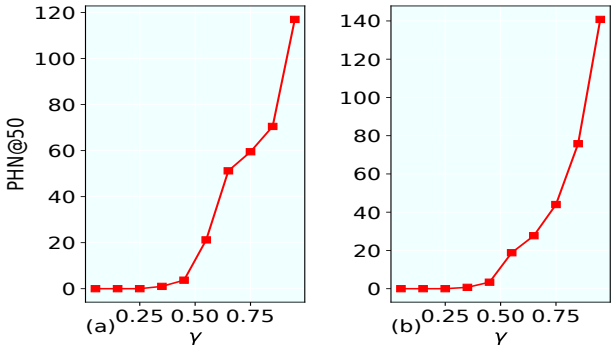

Figure 4: Performance variations versus masking thresholds γ. (a) and (b) show $PHN@50$ results vs γ for target items from Q_{10} on Gowalla and Yelp2018 datasets, respectively.

In Figure 4, we can see that the item promotion ability of our method is low when we use all users to establish the attack objective (i.e., γ as 0). As we increase the masking threshold γ, $PHN@50$ increases remarkably. This observation confirms the effectiveness of the node masking mechanism.

Dataset	Attack	Low popularity items		
		Q_{10}	Q_{30}	Q_{50}
		PHN@50	*PHN@50*	*PHN@50*
Gowalla	RandFilter	8.8	12.4	14.9
	IUFilter	1.9	5.0	7.8
	RUFilter	5.2	11.3	13.6
	Proposed	**23.4**	**23.3**	**20.5**
Yelp2018	RandFilter	5.0	6.4	8.3
	IUFilter	1.2	1.7	3.9
	RUFilter	5.1	3.6	6.4
	Proposed	**23.1**	**16.7**	**13.0**

Table 3: Performance comparisons of different attacks in improving a target item's popularity on Gowalla and Yelp2018 datasets with retraining. Three low popularity item sets (Q_{10}, Q_{30}, Q_{50}) are used for performance evaluation with perturbation budgets as Δ^1_s ($s = 10, 30, 50$). $PHN@50$ is averaged over 30 randomly selected target items at each set.

Promoting Items in Retraining Scenes

In real deployment scenarios, a collaborative filtering model may be retrained from scratch to capture the dynamics of an input graph. We simulate such an item promotion scene by first perturbing a user-item graph followed by retraining a new model on the perturbed graph. We keep all experimental settings the same as that used for the source model.

Table 3 reveals that, compared with baseline methods, our method consistently achieves the highest $PHN@50$ with a same perturbation budget. On both datasets, we have about $\mathbf{1.4}\times$ to $\mathbf{4.6}\times$ larger attack ability than the second-best method. Therefore, the proposed attack method successfully maintains a strong promotion ability even when we retrain the collaborative filtering model after topological perturbations.

Conclusion

In this work, we have proposed a novel view on promoting items in GNN-based collaborative filtering models based on topological attacks. Our formulation identifies users that play pivotal roles to promote a target item. We then propose a node masking mechanism to effectively improve vanilla gradient-based solutions. A resource-efficient approach is developed to make our method scalable to large-scale graphs.

Acknowledgments

This research is supported by the Joint NTU-WeBank Research Centre on Fintech (Award No: NWJ-2020-007), Nanyang Technological University, Singapore.

References

Cai, J.-F.; Candès, E. J.; and Shen, Z. 2010. A singular value thresholding algorithm for matrix completion. *SIAM Journal on optimization*, 20(4): 1956–1982.

Cho, E.; Myers, S. A.; and Leskovec, J. 2011. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1082–1090.

Dai, H.; Li, H.; Tian, T.; Huang, X.; Wang, L.; Zhu, J.; and Song, L. 2018. Adversarial attack on graph structured data. In *International conference on machine learning*, 1115–1124. PMLR.

Deldjoo, Y.; Noia, T. D.; and Merra, F. A. 2021. A survey on adversarial recommender systems: from attack/defense strategies to generative adversarial networks. *ACM Computing Surveys (CSUR)*, 54(2): 1–38.

Ekstrand, M. D.; Riedl, J. T.; Konstan, J. A.; et al. 2011. Collaborative filtering recommender systems. *Foundations and Trends® in Human–Computer Interaction*, 4(2): 81–173.

Fang, M.; Gong, N. Z.; and Liu, J. 2020. Influence function based data poisoning attacks to top-n recommender systems. In *Proceedings of The Web Conference 2020*, 3019–3025.

Geisler, S.; Schmidt, T.; Şirin, H.; Zügner, D.; Bojchevski, A.; and Günnemann, S. 2021. Robustness of graph neural networks at scale. *Advances in Neural Information Processing Systems*, 34: 7637–7649.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and harnessing adversarial examples. *International Conference on Learning Representations*.

He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.; and Wang, M. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 639–648.

He, X.; He, Z.; Song, J.; Liu, Z.; Jiang, Y.-G.; and Chua, T.-S. 2018. Nais: Neural attentive item similarity model for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 30(12): 2354–2366.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, 173–182.

Hu, Y.; Koren, Y.; and Volinsky, C. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE international conference on data mining*, 263–272. Ieee.

Jain, P.; Netrapalli, P.; and Sanghavi, S. 2013. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, 665–674.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Koren, Y. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 426–434.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8): 30–37.

Lam, S. K.; and Riedl, J. 2004. Shilling recommender systems for fun and profit. In *Proceedings of the 13th international conference on World Wide Web*, 393–402.

Li, B.; Wang, Y.; Singh, A.; and Vorobeychik, Y. 2016. Data poisoning attacks on factorization-based collaborative filtering. *Advances in neural information processing systems*, 29.

Liu, Z.; and Larson, M. 2021. Adversarial Item Promotion: Vulnerabilities at the Core of Top-N Recommenders that Use Images to Address Cold Start. In *Proceedings of the Web Conference 2021*, 3590–3602.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Tang, J.; Wen, H.; and Wang, K. 2020. Revisiting adversarially learned injection attacks against recommender systems. In *Fourteenth ACM conference on recommender systems*, 318–327.

Wang, J.; Zhang, S.; Xiao, Y.; and Song, R. 2021. A review on graph neural network methods in financial applications. *arXiv preprint arXiv:2111.15367*.

Wang, X.; He, X.; Wang, M.; Feng, F.; and Chua, T.-S. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, 165–174.

Wu, C.; Lian, D.; Ge, Y.; Zhu, Z.; Chen, E.; and Yuan, S. 2021a. Fight Fire with Fire: Towards Robust Recommender Systems via Adversarial Poisoning Training. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1074–1083.

Wu, F.; Gao, M.; Yu, J.; Wang, Z.; Liu, K.; and Wang, X. 2021b. Ready for emerging threats to recommender systems? A graph convolution-based generative shilling attack. *Information Sciences*, 578: 683–701.

Wu, J.; Wang, X.; Feng, F.; He, X.; Chen, L.; Lian, J.; and Xie, X. 2021c. Self-supervised graph learning for recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, 726–735.

Wu, S.; Sun, F.; Zhang, W.; Xie, X.; and Cui, B. 2020a. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys (CSUR)*.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020b. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1): 4–24.

Xu, K.; Chen, H.; Liu, S.; Chen, P.-Y.; Weng, T.-W.; Hong, M.; and Lin, X. 2019. Topology attack and defense for graph neural networks: An optimization perspective. *arXiv preprint arXiv:1906.04214*.

Zhang, L.; Liu, Y.; Zhou, X.; Miao, C.; Wang, G.; and Tang, H. 2022. Diffusion-Based Graph Contrastive Learning for Recommendation with Implicit Feedback. In *International Conference on Database Systems for Advanced Applications*, 232–247. Springer.

Zhou, X.; Sun, A.; Liu, Y.; Zhang, J.; and Miao, C. 2021. SelfCF: A Simple Framework for Self-supervised Collaborative Filtering. *arXiv preprint arXiv:2107.03019*.

Zügner, D.; Akbarnejad, A.; and Günnemann, S. 2018. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2847–2856.