RESEARCH-ARTICLE

LLM Agent-based Shilling Attack on Recommender Systems

SHENGKANG GU, Fudan University, Shanghai, China

JIAHAO LIU, Fudan University, Shanghai, China

DONGSHENG LI, Microsoft Research Asia, Beijing, China

GUANGPING ZHANG, Fudan University, Shanghai, China

MINGZHE HAN, Fudan University, Shanghai, China

HANSU GU

View all

Open Access Support provided by:

Fudan University

Microsoft Research Asia

Published: 21 February 2026

Citation in BibTeX format

WSDM '26:The Nineteenth ACM International Conference on Web Search and Data Mining
February 22 - 26, 2026
ID, Boise, USA

Conference Sponsors:
SIGKDD
SIGWEB
SIGIR
SIGMOD

LLM Agent-based Shilling Attack on Recommender Systems

Shengkang Gu*
Jiahao Liu*
Fudan University
Shanghai, China
gusk24@m.fudan.edu.cn
jiahaoliu23@m.fudan.edu.cn

Dongsheng Li
Microsoft Research Asia
Shanghai, China
dongsli@microsoft.com

Guangping Zhang
Fudan University
Shanghai, China
gpzhang20@fudan.edu.cn

Mingzhe Han
Fudan University
Shanghai, China
mzhan22@m.fudan.edu.cn

Hansu Gu
Independent
Seattle, United States
hansug@acm.org

Peng Zhang†
Fudan University
Shanghai, China
zhangpeng_@fudan.edu.cn

Ning Gu
Fudan University
Shanghai, China
ninggu@fudan.edu.cn

Li Shang
Fudan University
Shanghai, China
lishang@fudan.edu.cn

Tun Lu†
Fudan University
Shanghai, China
lutun@fudan.edu.cn

Abstract

With the growing ubiquity of recommender systems (RSs), malicious manipulation through shilling attacks, where fake user profiles are injected to alter system outputs, poses increasing threats to system integrity. Existing attack methods often rely on simplified heuristics, require internal RS data, and most overlook user reviews, limiting their stealthiness, realism, and potential impact. Recently, LLM-based user agents are gaining traction in the RS community for their capabilities to simulate human behaviors like rating and review generation. In this context, we propose AgentSA, a low-knowledge shilling attack framework that employs such agents to manipulate recommendations through adversarial yet human-like interactions. We design targeted mechanisms to guide profile construction, memory retrieval, and action generation (including reviews) to maximize manipulation impact while maintaining behavioral camouflage. We evaluate the impact of these agents on various types of RSs and demonstrate that AgentSA consistently outperforms existing low-knowledge attack methods in both effectiveness and stealth. Our findings uncover a concerning new class of threats enabled by LLM-based agents, underscoring the pressing need to bolster RS security against such emerging risks.

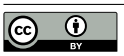
CCS Concepts

• **Information systems → Recommender systems**.

Keywords

Recommender Systems, Shilling Attacks, Large Language Models

*Equal contribution.
†Corresponding author.



WSDM '26, Boise, ID, USA

ACM ISBN 979-8-4007-2292-9/2026/02
https://doi.org/10.1145/3773966.3777981

ACM Reference Format:
Shengkang Gu, Jiahao Liu, Dongsheng Li, Guangping Zhang, Mingzhe Han, Hansu Gu, Peng Zhang, Ning Gu, Li Shang, and Tun Lu. 2026. LLM Agent-based Shilling Attack on Recommender Systems. In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining (WSDM '26), February 22–26, 2026, Boise, ID, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3773966.3777981

1 Introduction

With the rapid growth of e-commerce, recommender systems (RSs) have become essential for personalized content delivery [27, 28, 33, 38]. For users, RSs leverage interaction data, particularly ratings and reviews, to predict their preferences for specific items. For content providers (*e.g.*, online retailers), RSs help match their items with interested users, thereby enhancing item exposure and driving revenue growth [2, 18]. As a result, malicious actors may attempt to manipulate RSs to maximize their own interests. A common tactic is ***shilling attack***, where malicious user profiles with fabricated interaction data are injected into the system's training data to influence the outcome of RSs [23, 26, 30, 43].

The essence of shilling attacks is exploiting distributional differences between fake and genuine users to influence the outcome of RSs. **Existing shilling attack methods exhibit several limitations:** (1) Conventional shilling attacks assign extreme ratings to target items: maximal for promotion (push attack) or minimal for demotion (nuke attack) [43]. For non-target items, attackers typically select them randomly and assign ratings using strategies like random values or item averages to mimic genuine user behavior [3, 17, 19, 35]. These rule-based profiles often lack diversity, making them **easily detected as anomalies** [24, 52]. (2) Some methods enhance camouflage by integrating internal knowledge. For example, methods based on generative adversarial networks (GANs) [12] generate fake user profiles by sampling patterns from genuine user interaction data. These models produce stealthier fake profiles and train a discriminator on both real and synthetic interaction data [7, 23, 24, 42]. However, their dependence on internal

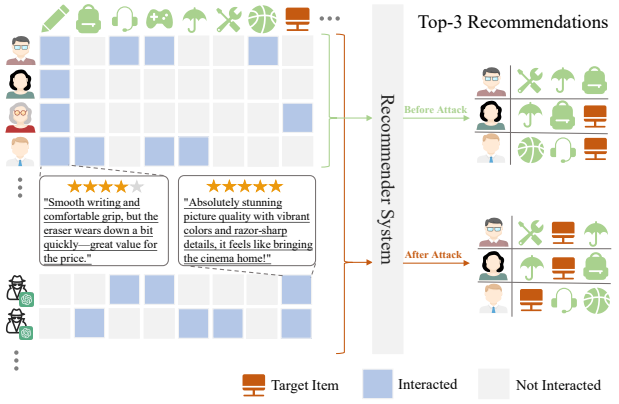


Figure 1: A motivating example of AgentSA: injecting fake user profiles generated by LLM-based agents can substantially boost the target item's ranking.

RS data (*e.g.*, user interactions) poses practical constraints, as such data is **rarely available for malicious external attackers**. (3) As RSs evolve toward hybrid models integrating collaborative and content-based methods, user reviews have become a crucial source of rich textual features in modern RSs [11, 13, 29]. Existing shilling attacks primarily manipulate rating data while often **neglecting user reviews**, which can carry greater risk due to their nuanced and semantically rich content [5, 52]. Therefore, it is essential to investigate review-based attack strategies that complement rating-based approaches and to assess their impact on RSs.

Recently, large language models (LLMs) demonstrated strong capabilities in human-level reasoning and natural language generation [21, 50]. This progress has motivated the development of LLM-based agents that simulate user behaviors using minimal system-specific data in RSs [25, 46, 53, 54, 56]. These agents serve multiple purposes, including reducing the gap between offline evaluation and online performance by modeling complex user behaviors, and better understanding users' underlying behavior logic [16, 53]. However, the openness of RSs may expose them to malicious user agents, which may be externally injected or internally compromised agents originally designed to satisfy the benefits mentioned above, both of which can severely undermine system integrity. Malicious agents embedded with targeted content can induce biased behaviors, including preferential promotion of specific vendors or suppression of certain product categories. By participating in activities such as rating and reviewing, these agents can manipulate recommendation outcomes in ways that serve the interests of their manipulators.

In this paper, we propose **AgentSA**, a novel low-knowledge shilling attack framework utilizing LLM-based agents to manipulate RSs. We inject a set of fake user agents into RSs, which engage in adversarial yet human-like interactions by selecting items, rating them, and posting reviews to generate synthetic data that blends with real user interactions. This manipulation distorts the recommendation outcomes. For instance, in a promotion attack, the target item receives increased exposure, as illustrated in Figure 1.

Specifically, we designed tailored attack strategies for the profile, memory, and action modules of the fake user agent. For the profile module, we developed **target-aware profile construction** method that infers and validates agent personality traits based on the target item and incorporates mechanisms to enhance diversity. For the memory module, we employed **hybrid retrieval** strategies that consider both relevance and recency. For the action module, beyond generating ratings, the agent is also capable of producing reviews, for which we designed a **target feature propagation** strategy to strengthen the connection between generated reviews and target items. Moreover, we simulated fake user interactions in the RSs by structuring the information flow from the system to the agent.

We validated the effectiveness of AgentSA across different types of RSs, including those using ratings alone and those augmented with reviews. AgentSA consistently outperformed existing low-knowledge attacks, highlighting its strong attack capability. The fake user agent exhibited strong behavioral resemblance to genuine users in our experiments, making it difficult to detect with advanced detection algorithms. We also performed additional studies to investigate how fake users of varying scales, targeting long-tail items, affect low-activity users, revealing a greater potential threat.

The main contributions of this work are summarized as follows:

- To the best of our knowledge, AgentSA is the first attempt to leverage LLM-based agents to simulate users for attacking RSs, introducing a paradigm shift in shilling attacks from rule-based heuristics to cognitively simulated agent behavior.
- AgentSA integrates targeted mechanisms for agent behavior modeling, including target-aware profile inference, hybrid memory retrieval, and multifaceted rating-review strategies to enable stealthy and effective attacks.
- Our findings uncover a concerning new class of threats enabled by LLM-based agents, underscoring the pressing need to bolster RS security against such emerging risks.

2 Related Work

2.1 Shilling Attacks against RSs

Early low-knowledge shilling attacks utilized global statistical information to generate fake user profiles for push (*e.g.*, random, bandwagon, segmented) and nuke attacks (*e.g.*, reverse bandwagon, love/hate) [3, 19, 35, 43]. Later heuristic techniques, such as noise injection, user shifting, target shifting, and popularity-based averaging were introduced to obscure aggressive rating patterns and improve stealth [17, 47]. However, these rule-based methods fail to capture the behavioral complexity of genuine users, limiting their stealthiness.

Subsequent attacks improve stealth by leveraging more knowledge, such as using **detailed interaction features** like node centrality to target influential users [39, 48], employing surrogate models informed by **partial RS structure** for attack guidance [45], or applying reinforcement learning with **periodic RS feedback**, though such feedback is rarely available in practice [5, 42]. The emergence of GANs has enabled the automatic generation of fake profiles. One approach applies DCGAN to generate initial profiles, trains a discriminator to differentiate genuine and fake interactions, and refines the profiles using zero-order optimization [6, 7, 36]. Another approach improves GAN-based attacks by sampling genuine user templates from the **rating matrix** and using a generator to produce fake ratings, with a discriminator optimizing a global

objective that integrates multiple loss functions [23, 24]. Although R-Trojan [52] incorporates review generation, its review generation lacks explicit attack objectives and still relies on the rating matrix for training, limiting its applicability.

2.2 LLM-Based User Agents in RSs

LLM-based agents in RSs fall into two main categories: recommender agents, which use LLMs to generate or enhance recommendations [14, 16, 31, 32, 49, 55], and user agents, which employ LLMs to simulate user behavior. While some studies model user dialogues in conversational recommendation settings [10, 59], our work models user actions like rating and reviewing, which are more common interactions in RSs.

RecAgent [46] and Agent4Rec [53] employ LLM-based agents with **profile**, **memory**, and **action** modules to simulate user decision-making. RAH [41] introduces LLM-based multi-agents that serve both as recommendation generators and as simulators of user behavior, bridging users and RSs. FLOW [4] facilitates collaboration between recommendation and user agents via a feedback loop. Zhang et al. [58] integrate explicit preferences, LLM-based sentiment analysis, human-in-the-loop modeling, and statistical frameworks to simulate user interactions more robustly. AgentCF [56] proposes a co-learning framework where users and items are modeled as agents and then jointly optimized. AgentCF++ [25] extends this framework to enable cross-domain transfer and popularity-aware modeling. Together, these works highlight the potential of LLMs for simulating human-like user behavior in RSs.

3 Preliminaries

3.1 Definition of Shilling Attack

Shilling attacks inject fake user profiles into RSs to manipulate recommendations. Let the input to the RS be a user-item interaction dataset $\mathcal{D} = \{D_{u,i} \mid u \in U, i \in I\}$, where $U = \{u_1, \ldots, u_m\}$ is the set of genuine users, and $I = \{i_1, \ldots, i_n\}$ is the set of items. Each $D_{u,i} \in \mathcal{V}$ denotes the interaction between user u and item i, with $\mathcal{V}$ the domain of interaction values (*e.g.*, numerical ratings $\mathcal{R}$ or textual reviews $\mathcal{T}$). An attacker introduces a set of fake users U_a, and constructs corresponding fake interactions $\mathcal{D}_a = \{D_{u_a,i} \mid u_a \in U_a, i \in I\}$. The manipulated input to the RS becomes $\mathcal{D}' = \mathcal{D} \cup \mathcal{D}_a$.

The attacker aims to manipulate the recommendation algorithm $\Phi : \mathcal{D} \to O$, where O is the system output (*e.g.*, predicted rating matrix or top-N list), to alter the recommendation outcome for the target item $i_t \in I$. For a push attack, the attacker aims to improve the recommendation outcome for the target item, defined as:

$$\Delta\Phi(i_t) = \Phi(\mathcal{D}')_{i_t} - \Phi(\mathcal{D})_{i_t}. \tag{1}$$

Here, $\Phi(\mathcal{D})_{i_t}$ and $\Phi(\mathcal{D}')_{i_t}$ represent the recommendation outcome for item i_t before and after the attack, respectively (*e.g.*, predicted score or ranking position). In the case of a push attack, shilling attack can be formulated as the following optimization problem:

$$\max_{U_a,\, \mathcal{D}_a} \Delta\Phi(i_t). \tag{2}$$

3.2 Anatomy of Shilling Attacks

A fake user's interacted items are typically partitioned into three categories, with each category's interactions governed by a different distribution defined by the attack method.

(1) *Target Items.* Fake interactions are generated from biased distribution P_{target} for target item i_t to meet the attacker's goal:

$$D_{U_a,i_t} \sim P_{target}. \tag{3}$$

For example, in a push attack, fake users may assign high ratings and generate positive reviews for the target item. In a nuke attack, they may assign low ratings and post negative reviews.

(2) *Selected Items.* Beyond the target item, the attacker selects a set of items $I_s \subseteq I \setminus \{i_t\}$ to emulate genuine user behavior. For each selected item $i_s \in I_s$, fake user interactions are generated to follow:

$$D_{U_a,i_s} \sim P_{selected}. \tag{4}$$

Here, $P_{selected}$ approximates typical user behavior and improves profile stealth. For instance, in a bandwagon attack, fake users assign high ratings to popular items to resemble normal user patterns.

(3) *Filler Items.* The filler item set $I_f \subseteq I \setminus (\{i_t\} \cup I_s)$. For each filler item $i_f \in I_f$, the fake user's interaction is generated to follow:

$$D_{U_a,i_f} \sim P_{filler}, \tag{5}$$

where P_{filler} is defined according to the specific attack method. Unlike the uniform selected items, the filler set is varied across fake users to enhance interaction diversity and reduce detectability. Items not interacted with are denoted as $I_\phi = I \setminus (i_t \cup I_s \cup I_f)$.

In essence, shilling attack methods differ in how they construct U_a, choose I_s and I_f, and define the corresponding interactions D_{U_a,i_t}, D_{U_a,i_s}, and D_{U_a,i_f}. These components are collectively optimized to both maximize $\Delta\Phi(i_t)$ and ensure fake user behavior is camouflaged among genuine users to avoid detection.

4 Methodology

This section details the design of AgentSA, which employs a LLM-based user agent with a modular architecture comprising **profile**, **memory**, and **action** modules. The overall framework is illustrated in Figure 2. The agent's three core components are:

- **The Profile Module** constructs adversarial profiles via target-guided inference, validation and diversity enhancement.
- **The Memory Module** retrieves hybrid memory based on relevance and recency.
- **The Action Module** performs multifaceted interaction attack enhanced by target feature propagation strategy.

4.1 Modular Structure of Fake User Agent

We formally represent the fake user agent as a triplet Agent = $(\mathcal{P}rofile, \mathcal{M}emory, \mathcal{A}ction)$. Each module is detailed below:

- **Profile Module**: Stores personalized attributes forming the agent's identity and preferences, such as age, gender, occupation, interests, and interaction habits (*e.g.*, rating or reviewing frequency).
- **Memory Module**: Stores the agent's historical interactions, including ratings and reviews. It employs two retrieval methods: a relevance-based retriever and a recency-based retriever, to distill key information and support decision making.

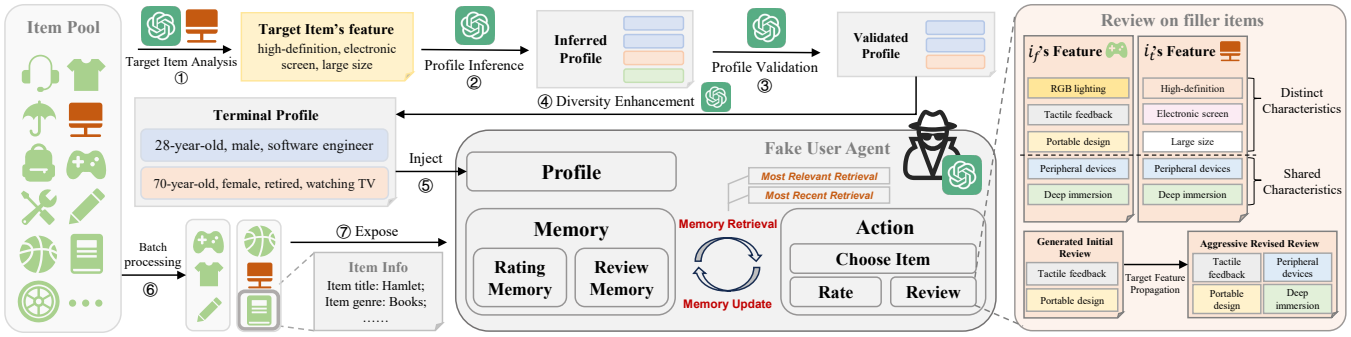


Figure 2: The overall framework of AgentSA. It starts with target item analysis and profile inference, followed by profile validation and diversification. Fake user agents interact with filler items using hybrid memory retrieval, while a target feature propagation strategy embeds key target attributes into reviews on filler items.

- **Action Module**: Defines the agent's behavior, including item selection, rating generation, and review generation.

4.2 Target-Aware Profile Construction

To construct fake user agents tailored to the target item, we propose a target-driven method that reversely infers user profiles from the target's characteristics, followed by validation and diversification.

4.2.1 Target Item Feature Analysis. Given a target item i_t, we construct a prompt containing its metadata (*e.g.*, title and category), and invoke the LLM to generate an analysis of its characteristics $\mathcal{A}_t = Prompt_{\text{LLM}}(i_t)$ based on its world knowledge.

4.2.2 Profile Inference from Target Item. Based on the item analysis $\mathcal{A}_t$ and the attack type $d \in \{+, -\}$ (push or nuke), we construct the prompt for profile inference. For example, in a push attack on the item titled "*High-Definition Large-Size Electronic Display Screen*", the prompt is constructed as:

> Example Prompt for Profile Inference
>
> **You currently have a strong preference for** High-Definition Large-Size Electronic Display Screen.
> **The information about this item is as follows:** It features an ultra-large 32-inch curved design with high resolution and exceptional clarity. It is suitable for gaming, multitasking, office work, and watching TV or other media.
> **Based on this item description, please infer your potential personality profile, including:**
> - **Gender, age, and occupation**
> - **Personal interests or dislikes**
> - **Interaction habits** (*e.g.*, **how frequently you provide reviews or ratings**)

The LLM output from this prompt defines the profile module $\mathcal{P}^{(i_t)}$, comprising user interests, interaction habits, and other personality traits, as follows:

$$\mathcal{P}^{(i_t)} = Prompt_{\text{LLM}}(\mathcal{A}_t, d). \tag{6}$$

4.2.3 Profile Consistency Validation. To ensure the generated profile $\mathcal{P}^{(i_t)}$ exhibits a genuine preference for the target item i_t, we introduce a validation step inspired by the encoding-decoding architecture in Seq2Seq models [44]. The profile generation serves as the encoding phase, while validation corresponds to decoding. Specifically, we issue a secondary prompt asking the fake user agent, based on its current profile, whether it would genuinely prefer the target item. A negative response (*i.e.*, $\text{Validate}(\mathcal{P}^{(i_t)}) = 0$) triggers profile regeneration to ensure consistency with the attack objective.

$$\text{Validate}(\mathcal{P}^{(i_t)}) = \begin{cases} 1, & \text{if } Prompt_{\text{LLM}}(\mathcal{P}^{(i_t)}, i_t, d) = \textit{Yes}, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

4.2.4 Profile Diversity Enhancement. Homogeneous profiles, where fake agents share similar demographics and interests, can lead to concentrated interaction patterns that are easily flagged by anomaly detection systems. To mitigate this vulnerability, we leverage the LLM to curate a diverse subset of profiles from the candidate pool:

$$\{\mathcal{P}_k^{(i_t)}\}_{k=1}^{N} = Prompt_{\text{LLM}}(\{\mathcal{P}_k^{(i_t)}\}_{k=1}^{N'}), \tag{8}$$

where N' is the number of candidate profiles generated, and N is the number of fake user agents to be injected.

Overall, in a push attack, each fake user agent is individually crafted to emulate a genuine user with seemingly authentic preferences for the target item. At the population level, the fake users exhibit diversity comparable to real-world user distributions. When interacting with the RS, their behaviors appear highly plausible and are difficult to detect.

4.3 Memory Management and Hybrid Retrieval

The Memory module stores the fake user agent's historical interactions, consisting of rating records $\mathcal{M}_R = (i_m, \mathcal{R}_m, t_m)$ and review records $\mathcal{M}_T = (i_n, \mathcal{T}_n, t_n)$, where i denotes the item, $\mathcal{R}$ the rating, $\mathcal{T}$ the review, and t the timestamp. The module supports memory updates and retrieval to guide downstream decisions.

4.3.1 Memory Update Mechanism. Upon executing an action, the agent updates its memory by recording the new rating $(i_q, \mathcal{R}_{new}, t_{now})$ in $\mathcal{M}_R$ and the corresponding review $(i_q, \mathcal{T}_{new}, t_{now})$ in $\mathcal{M}_T$. To avoid excessive memory accumulation, only a fixed number of recent entries may be retained.

4.3.2 Memory Retrieval Mechanism. Given a query item i_q, we perform two types of retrieval: relevance-based and recency-based.

(1) **Relevance-based retrieval** selects the top-K semantically relevant entries from the full memory set $\mathcal{M} = \mathcal{M}_R \cup \mathcal{M}_T$:

$$\mathcal{M}_{rel} = \{m_j \in \mathcal{M} \mid sim(i_q, i_j) \geq Top_K(\{sim(v_q, v_k)\}_{k=1}^{|\mathcal{M}|})\}, \quad (9)$$

where $\text{sim}(\cdot)$ denotes semantic similarity based on item information. Specifically, item semantics are extracted using a Transformer-based language model, which encodes textual metadata (*e.g.*, item titles) into dense vector representations. Cosine similarity is then computed between these embeddings.

(2) **Recency-based retrieval** selects the M most recent interaction records:

$$\mathcal{M}_{rec} = \{m_i \in \mathcal{M} \mid t_i \geq Top_M(\{t_j\}_{j=1}^{|\mathcal{M}|})\}. \quad (10)$$

In summary, the final retrieved memory set $\mathcal{M}_{ret} = \mathcal{M}_{rel} \cup \mathcal{M}_{rec}$ integrates relevance and recency to preserve the agent's core interests and recent behavioral patterns, effectively emulating human memory retrieval processes.

4.4 Multifaceted Rating-Review Attack

For the target item, each fake user agent u_a generates a rating and a review as adversarial data D_{u_a,i_t}. In a push attack, this typically involves assigning the maximal rating and a positive review.

AgentSA does not explicitly define a selected item set I_s. Instead, for remaining items, each fake user agent individually selects items in a personalized manner, forming a filler item set I_f, and generates corresponding interaction data D_{u_a,I_f}. In each exposure batch, the fake user agent executes a three-step process: item selection, rating generation, and review generation.

4.4.1 Item Selection. For each exposure batch $B^{(b)} = i_1, \ldots, i_k$ in round b, the fake user agent selects a subset $I_{f,b} \subseteq B^{(b)}$ as filler items, guided by its constructed malicious profile $\mathcal{P}_{u_a}$. This process is repeated over multiple rounds, and the final filler item set I_f is obtained by aggregating filler items from all exposure rounds:

$$I_f = \bigcup_{b=1}^{\mathbb{N}} Prompt_{\text{LLM}}(B^{(b)}, \mathcal{P}_{u_a}), \quad (11)$$

where $\mathbb{N}$ is the total number of exposure rounds.

This approach offers two key advantages: (1) In terms of the number of filler items, AgentSA provides only a flexible reference, allowing each fake user's engagement level to be guided by the interaction habits embedded in $\mathcal{P}_{u_a}$. In contrast to conventional methods that prescribe a fixed number of interactions for each fake user, AgentSA promotes behavioral diversity. (2) In terms of the composition of the filler item set, unlike conventional attack methods that rely on random choice or simple heuristics, AgentSA conditions item selection on the fake user's stated preferences. This results in filler items that are more consistent with the user profile, yielding more coherent interaction behavior with improved stealth.

4.4.2 Rating Generation. The fake user agent is prompted to rate the filler items I_f based on its profile and the memory retrieved, ensuring the ratings are both profile-aligned and historically coherent, thereby enhancing the verisimilitude of the agent's behavior.

$$\mathcal{R}(u_a, I_f) = Prompt_{\text{LLM}}(I_f, \mathcal{P}_{u_a}, \mathcal{M}_{ret}). \quad (12)$$

4.4.3 Adversarial Review Generation via Target Feature Propagation. We propose a review-based attack strategy that propagates salient features of the target item across filler items. For each filler item, the agent first identifies shared characteristics with the target item, guided by its analysis result $\mathcal{A}_t$ from the profile construction stage. It then generates an initial review $\widetilde{\mathcal{T}}(u_a, I_f)$ conditioned on the item's attributes and the rating $\mathcal{R}(u_a, I_f)$, and incorporates the identified similarity $S(I_f, i_t)$ to produce the final review $\mathcal{T}(u_a, I_f)$.

$$S(I_f, i_t) = Prompt_{\text{LLM}}(\mathcal{A}_t, I_f). \quad (13)$$

$$\widetilde{\mathcal{T}}(u_a, I_f) = Prompt_{\text{LLM}}(I_f, \mathcal{R}(u_a, I_f)). \quad (14)$$

$$\mathcal{T}(u_a, I_f) = Prompt_{\text{LLM}}(S(I_f, i_t), \widetilde{\mathcal{T}}(u_a, I_f)). \quad (15)$$

For example, when reviewing the filler item *Phantom X Ultra-Performance Portable Gaming Controller*, the agent crafts an initial review based on the item's attributes. After identifying similarities with the target item (*e.g.*, "both are peripheral devices" and "both offer deep immersion"), it refines the review to subtly weave in these shared features. The transformation is illustrated below:

Example Review Before and After Feature Propagation

Before Propagation:
"This is easily the best controller I've used. The tactile feedback is just right, and it handles tough gameplay without a hitch. The portable design makes it super easy to carry around. Great value overall."

After Propagation:
"As one of the top **peripheral devices**, this controller delivers excellent tactile feedback and handles demanding gameplay. It offers a **deeply immersive** experience, and the portable design makes it perfect for gaming on the go."

Through this process, the target item's features are implicitly propagated to other items. If users engage with these reviewed items, the RS may infer a latent interest in the target item. This manipulation can bias predicted ratings, thereby increasing the target item's exposure. To summarize, following the above procedure, the fake user agent derives the interaction data for the filler items as:

$$(\mathcal{R}(u_a, I_f), \mathcal{T}(u_a, I_f)) \sim P_{filler} \mid I_f, \mathcal{P}_{u_a}, \mathcal{M}_{ret}, \mathcal{A}_t. \quad (16)$$

4.5 AgentSA Execution Pipeline

The overall attack procedure is summarized in Algorithm 1, comprising the following stages:

Initialization. Specify the target item i_t and the attack type $d \in \{+, -\}$, construct N fake user agents $\{Agent_i\}_{i=1}^{N}$, initialize their profile modules as described in the profile construction section, and set the number of exposure rounds $\mathbb{N}$.

Adversarial Target Item Interaction. Each fake agent interacts with the target item i_t, generating a rating and a review according to the attack type d.

Iterative Filler Item Interaction. In each round b ($1 \leq b \leq \mathbb{N}$), the victim RS exposes a batch of items $B^{(b)}$ to a fake user agent u_a. Guided by its profile, the agent selects a filler item subset $I_{f,b} \subseteq B^{(b)}$, retrieves memory via the hybrid retrieval mechanism, and generates corresponding ratings $\mathcal{R}(u_a, I_{f,b})$ and reviews $\mathcal{T}(u_a, I_{f,b})$.

Algorithm 1 Execution Procedure of AgentSA

Require: Target item i_t; attack type d; number of fake users N; number of exposure rounds $\mathbb{N}$
Ensure: Interaction data $\{D_{u_a,i_t}, D_{u_a,I_f}\}_{a=1}^{N}$

Initialization:
Construct N fake user agents and initialize their profiles based on i_t and d (Eq. 6, 7, 8)
Target Item Interaction:
for each agent u_a **do**
Generate rating and review on i_t according to d to form D_{u_a,i_t}
end for
Iterative Filler Item Interaction:
for $b = 1$ to $\mathbb{N}$ **do**
for each agent u_a **do**
Receive exposure batch $B^{(b)}$ from the victim RS
Select filler items $I_{f,b} \subseteq B^{(b)}$ based on profile (Eq. 11)
Retrieve memory using hybrid strategy (Eq. 9, 10)
Generate rating or review for $I_{f,b}$ to construct $D_{u_a,I_{f,b}}$ (Eq. 12, 13, 14, 15)
Update memory
end for
end for
Return $\{D_{u_a,i_t}, D_{u_a,I_f}\}_{a=1}^{N}$ where $D_{u_a,I_f} = \bigcup_{b=1}^{\mathbb{N}} D_{u_a,I_{f,b}}$

Table 1: Statistics of the datasets.

Datasets	#Users	#Items	#Inters.	Sparsity
Books	3,441	3,569	41,764	99.66%
Auto	2,700	3,090	29,346	99.64%
Pets	880	522	5,906	98.71%

The resulting interactions are recorded in the memory module to support subsequent rounds.

By simulating the complete process through which genuine users browse, engage with, and evaluate items in RSs, AgentSA constructs fake user agents with realistic behaviors that improve both stealth and attack effectiveness.

5 Experiment

5.1 Settings

5.1.1 Datasets. We conduct experiments on three real-world Amazon datasets: *Books*, *Automotive* (*Auto*), and *Pet Supplies* (*Pets*), each containing both rating and review data. Due to API cost constraints, we limited the interaction data to respective one-year periods. To mitigate the impact of cold-start users and items, we retained only those with at least five interactions. The resulting dataset size and sparsity are broadly consistent with prior studies on shilling attacks [23, 24, 52], as shown in Table 1. The data were randomly split into training, validation, and test sets in an 8:1:1 ratio to ensure sufficient training of the victim RSs.

5.1.2 Implementation Details. AgentSA was implemented with GPT-4o-mini. By default, we inject fake users equivalent to 5% of the real user base, with the push attack as the representative scenario. We randomly select 10 items from the item pool as target items to evaluate average attack performance.

Hyperparameter Settings. We tuned two key memory retrieval hyperparameters: K for relevance and M for recency. K balances sufficient context against noise, while M balances short-term interest shifts against long-term preference stability. Based on preliminary validation, we set K=3 and M=3 for our main experiments.

Victim RSs. To ensure broad coverage, we evaluate AgentSA on two types of victim RSs: those based solely on ratings and those incorporating both ratings and reviews. For rating-based RSs, we adopt NMF [20] and NeuNMF [9]. For models integrating reviews and ratings, we use two dual-tower variants: Dual-Tower$_{\text{early}}$ and Dual-Tower$_{\text{late}}$, which integrate review and ID features via early and late fusion, respectively [15]. Due to the unavailability of real exposure signals in offline evaluation, we approximate the exposure behavior using a popularity-based probabilistic scheme, which aligns with common practices in real-world RSs for presenting items to new users. While item visibility during simulation is determined by this scheme, AgentSA itself does not rely on popularity data and assumes that exposure is governed internally by the RS.

5.1.3 Evaluation Metrics. We adopt metrics aligned with [23] to evaluate the effectiveness of AgentSA, including prediction shift and hit ratios at K (HR@K), where K = 10 in our experiments. Predicion shift measures the absolute change in the predicted rating of the target item before and after the attack. HR@K reflects the relative ranking of the target item by indicating how frequently it appears among the top-K recommendations. To assess the attack's effect on the victim RSs predictive accuracy, we report RMSE and MAE as indirect indicators of stealthiness, measuring whether the attack leads to noticeable degradation in prediction performance [34].

5.1.4 Baselines. As a black-box attack requiring no internal RS data (*e.g.*, the rating matrix or RS feedback), AgentSA is compared with conventional attacks and existing low-knowledge attack baselines, all of which assign the maximum rating to the target item:

- **Random Attack** [19]: Each filler item is assigned a rating $\mathcal{R}(u_a, i_f) \sim \mathcal{N}(\mu, \sigma)$, where μ and σ denote the mean and standard deviation computed over all ratings in the system.
- **Average Attack** [19]: Each filler item is assigned a rating $\mathcal{R}(u_a, i_f) \sim \mathcal{N}(\mu, \sigma)$, where μ and σ denote the mean and standard deviation of all ratings on item i_f in the system.
- **Bandwagon Attack** [35]: The attack profile includes the most popular items as selected items, each assigned the maximum rating. Filler items are randomly selected and rated following the same strategy as in Random Attack.
- **Segmented Attack** [3]: This method targets a user subgroup likely to engage with the target item via strategically selected items. Due to data limitations, the most popular items are used as selected items and assigned the maximum rating, while filler items are randomly chosen and assigned the minimum rating.
- **Mixed Attack** [1]: This strategy integrates Random, Average, Bandwagon, and Segmented attacks in equal proportion.
- **Average over Popular** (AoP) [17]: A variant of Average Attack where filler items are uniformly selected from the top $\mathcal{X}$% of items by popularity. In our experiments, we set $\mathcal{X} = 10$.

• **R-Trojan(review)** [52]: A review-aware attack method uses a fine-tuned LLM to generate reviews conditioned on ratings. As its rating manipulation relies on the rating matrix, we adopt only its review generation module to augment the baselines.

5.2 Overall Performance

We evaluated AgentSA on all three datasets and compared its performance with baseline methods. Tables 2 and 3 present attack results on rating-based and review-aware RSs, respectively. The key observations are as follows:

(1) AgentSA consistently outperforms most baselines, demonstrating the effectiveness of using LLM-based agents to simulate users in shilling attacks. In **rating-only RSs**, it achieves notably superior results in both prediction shift and HR@10, surpassing most conventional attacks. While some methods may occasionally perform better, they fail to match the robustness exhibited by AgentSA across varied settings. For **review-aware RSs**, we adapt rating-based attacks with synthetic reviews for fair comparison. One variant generates review embeddings from random vectors mimicking real distributions, while the other produces textual reviews via R-Trojan-style generation. Our adversarially crafted reviews prove far more effective, highlighting the strength of AgentSA in more complex, real-world recommendation scenarios.

(2) Among all datasets, AgentSA achieves the strongest attack performance on the Books dataset, consistently outperforming all baselines across evaluation metrics. This may be attributed to the LLM-based agent's broader domain knowledge in books, which enables it to more effectively perform key reasoning steps such as inferring user profiles and analyzing item characteristics, ultimately leading to stronger adversarial behavior. Consequently, advanced techniques like retrieval-augmented generation and fine-tuning [22, 37] could potentially yield even more potent attacks.

(3) We evaluate the changes in RMSE and MAE of the victim RSs under attack, as illustrated in Figure 3. AgentSA consistently results in minor impact on prediction accuracy, indicating limited interference with the RS and highlighting the stealthiness of AgentSA. In rating-only RSs, most attack methods cause only limited degradation in RMSE and MAE. In contrast, Segmented Attack leads to substantial performance drop in review-aware RSs, likely due to its use of a narrow item set and extreme ratings for both selected and filler items, which impairs model generalization. Notably, in review-aware RSs, conventional attacks cause considerably less disruption to RMSE and MAE when combined with R-Trojan-style reviews, consistent with the findings in [52].

5.3 Detection Evasion Analysis

To assess the stealthiness of AgentSA, we evaluate its performance against two detection methods: the unsupervised FAP [57] and the supervised BayesDetector [51]. Precision and recall are reported for identifying the injected users. As shown in Figure 4, both methods exhibited the weakest performance when applied to users created by AgentSA. This suggests that our fake users, driven by the LLM-based agent's human-like reasoning and generation capabilities, are substantially harder to detect due to their ability to closely mimic genuine user behavior.

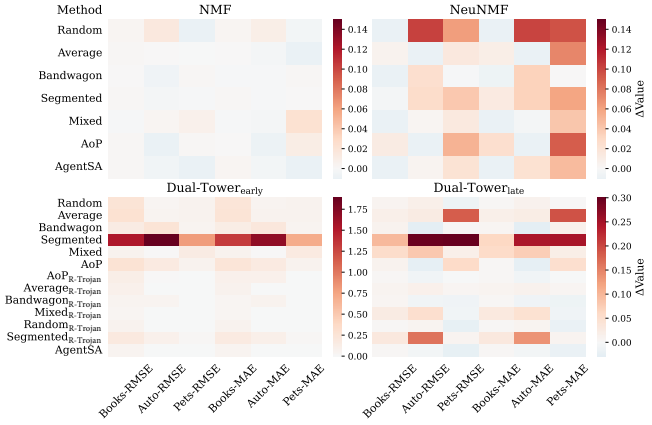


Figure 3: Heatmaps of RMSE/MAE shift (ΔValue) across attacks and datasets. Lighter colors indicate smaller shifts. Scales are normalized per subplot.

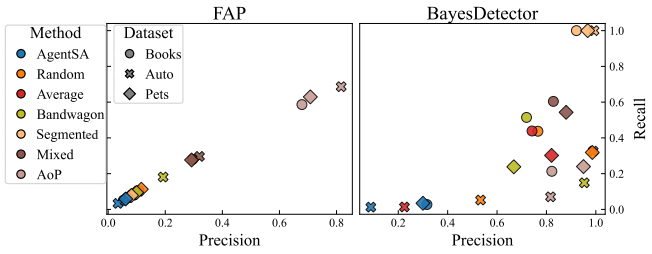


Figure 4: Precision-Recall results under FAP and BayesDetector. Lower values indicate better stealthiness of the attack.

Due to the scarcity of shilling attack methods that incorporate reviews, most existing fake user detection techniques are primarily designed for user rating data. The detectability of LLM-generated text largely depends on the model's reasoning and generation capabilities, and is therefore beyond the scope of this study.

5.4 Further Model Analysis

5.4.1 Ablation study. AgentSA integrates six key components to construct realistic and deceptive fake users: (1) Profile inference and validation; (2) Profile diversity enhancement; (3) Recency-based memory retrieval; (4) Relevance-based memory retrieval; (5) Review-based attack; (6) Target feature propagation. To assess the contribution of each component, we conduct an ablation study on the Books dataset against two review-aware RSs, sequentially disabling each one. Results are reported in Table 4.

(1) *w/o* Profile inference and validation: This variant replaces targeted profile inference and validation with random assignment. The agent still assigns the highest rating or a positive review to the target. Although the RMSE shift shows only a slight decrease, the attack performance drops substantially.

(2) *w/o* Profile diversity enhancement: In this variant, diversity enhancement is disabled. The increased RMSE shift indicates reduced stealthiness, while attack performance is largely unaffected.

Table 2: Overall performance on rating-only RSs. Best results are highlighted in bold.

Victim RS	NMF						NeuNMF					
Method	Prediction Shift			HR@10			Prediction Shift			HR@10		
	Books	Auto	Pets	Books	Auto	Pets	Books	Auto	Pets	Books	Auto	Pets
Random	0.6491	0.2828	0.9866	0.0997	0.0121	0.0696	0.8662	0.4979	1.0201	0.0414	0.0128	0.1496
Average	0.6191	0.2736	0.9773	0.0693	0.0054	0.0502	0.8336	0.4872	0.9411	0.0425	0.0070	0.1202
Bandwagon	0.6187	0.2783	0.9397	0.0726	0.0084	0.0433	0.7262	0.4546	0.9581	0.0286	0.0107	0.1162
Segmented	0.6303	0.3127	0.8689	0.1886	0.0320	**0.1160**	0.4669	0.1280	0.4326	0.0202	0.0115	0.0472
Mixed	0.6175	0.2691	0.8924	0.0670	0.0084	0.0393	0.8335	0.4683	0.9396	0.0337	0.0142	0.1575
AoP	0.6400	0.2818	0.9816	0.0813	0.0085	0.0674	0.8828	0.4756	**1.0486**	0.0438	0.0135	0.1464
AgentSA	**0.7366**	**0.3421**	**1.0327**	**0.1891**	**0.0618**	0.0934	**0.9064**	**0.5020**	1.0371	**0.0548**	**0.0225**	**0.1885**

Table 3: Overall performance on review-aware RSs. Best results are highlighted in bold.

Victim RS	Dual-Tower$_{\text{early}}$						Dual-Tower$_{\text{late}}$					
Method	Prediction Shift			HR@10			Prediction Shift			HR@10		
	Books	Auto	Pets	Books	Auto	Pets	Books	Auto	Pets	Books	Auto	Pets
Random	0.0021	0.1297	0.1966	0.0041	0.0024	0.0212	0.0022	0.0005	0.0375	0.0040	0.0021	0.0192
Average	0.0064	-0.0179	0.0491	0.0028	0.0012	0.0099	0.0445	0.0043	0.0458	0.0050	0.0017	0.0181
Bandwagon	0.0002	0.0332	0.0585	0.0026	0.0023	0.0107	0.0032	0.0119	0.0007	0.0033	0.0029	0.0147
Segmented	0.0035	0.1732	0.2207	0.0056	0.0012	0.0205	0.0369	0.0075	0.0172	0.0054	0.0024	0.0144
Mixed	0.1586	0.0351	0.2114	0.0060	0.0011	0.0179	0.0295	0.0287	0.0165	0.0032	0.0030	0.0200
AoP	0.0074	0.0133	0.0913	0.0030	0.0014	0.0124	0.0633	-0.0160	0.0253	0.0043	0.0031	0.0180
Random$_{\text{R-Trojan}}$	0.0062	0.1304	0.2015	0.0046	0.0027	0.0232	0.0058	0.0179	0.0477	0.0046	0.0026	0.0198
Average$_{\text{R-Trojan}}$	0.0136	0.0526	0.0593	0.0032	0.0012	0.0165	0.0632	0.0063	0.0512	0.0055	0.0023	0.0192
Bandwagon$_{\text{R-Trojan}}$	0.0037	0.0578	0.0634	0.0033	0.0028	0.0151	0.0052	0.0268	0.0069	0.0044	0.0030	0.0156
Segmented$_{\text{R-Trojan}}$	0.0051	0.1803	0.2423	0.0064	0.0014	0.0221	0.0516	0.0217	0.0258	0.0057	0.0026	0.0154
Mixed$_{\text{R-Trojan}}$	0.1693	0.0729	0.2234	0.0074	0.0014	0.0185	0.0378	0.0411	0.0354	0.0034	0.0030	0.0205
AoP$_{\text{R-Trojan}}$	0.0112	0.0136	0.1228	0.0046	0.0016	0.0136	0.0966	0.0103	0.0287	0.0049	**0.0033**	0.0185
AgentSA	**0.1967**	**0.2274**	**0.2660**	**0.0112**	**0.0032**	**0.0269**	**0.1205**	**0.0551**	**0.0651**	**0.0063**	0.0031	**0.0210**

Table 4: Ablation results on the Books dataset. For brevity, only HR@10 and RMSE are shown for Dual-Tower$_{\text{early}}$ and Dual-Tower$_{\text{late}}$, reflecting effectiveness and stealth. Underlined values indicate significant performance degradation.

Method Variant	Dual-Tower$_{\text{early}}$		Dual-Tower$_{\text{late}}$	
	HR@10	RMSE Shift	HR@10	RMSE Shift
Full Model (AgentSA)	**0.0112**	**0.0581**	**0.0063**	**0.0042**
w/o Profile inference and validation	0.0078	0.0578	0.0044	0.0042
w/o Profile diversity enhancement	0.0108	0.0623	0.0066	0.0052
w/o Recency-based memory retrieval	0.0116	0.0676	0.0058	0.0062
w/o Relevance-based memory retrieval	0.0092	0.0569	0.0052	0.0044
w/o Review-based attack	0.0072	0.0679	0.0058	0.0054
w/o Target feature propagation	0.0078	0.0575	0.0056	0.0040

(3) *w/o* Recency-based memory retrieval: In this setting, the memory module retrieves only the most semantically relevant past interactions. The attack performance remains relatively stable, while stealthiness degrades significantly.

(4) *w/o* Relevance-based memory retrieval: Here, the memory module retrieves only the most recent interactions. Stealthiness remains largely unchanged, but the attack performance deteriorates.

(5) *w/o* Review-based attack: Here, when attacking review-aware RSs, we rely solely on ratings. For review representations, we adopt the same approach as in baseline methods by initializing review embeddings with random vectors mimicking the real distribution. The results show that both attack effectiveness and stealthiness decline significantly under this configuration.

(6) *w/o* Target feature propagation: Here, the LLM-based agent generates reviews directly based on its profile, item information, and interaction rating, without employing our proposed review attack strategy. We observe a drop in attack performance, whereas the RMSE shift shows only a slight decrease.

The ablation results confirm each component in AgentSA contributes meaningfully to either enhancing attack effectiveness or improving stealthiness, with several components supporting both.

5.4.2 Effect of Injection Size on Attack Strength. We evaluate the impact of attack size on the victim RSs by injecting different proportions of fake users. As shown in Figure 5, increasing the proportion of injected users significantly amplifies the attack effect, as the target item is recommended more frequently to genuine users.

5.4.3 Vulnerability of Low-Activity Users. Prior work [8, 40] has shown that users in RSs exhibit varying levels of vulnerability to

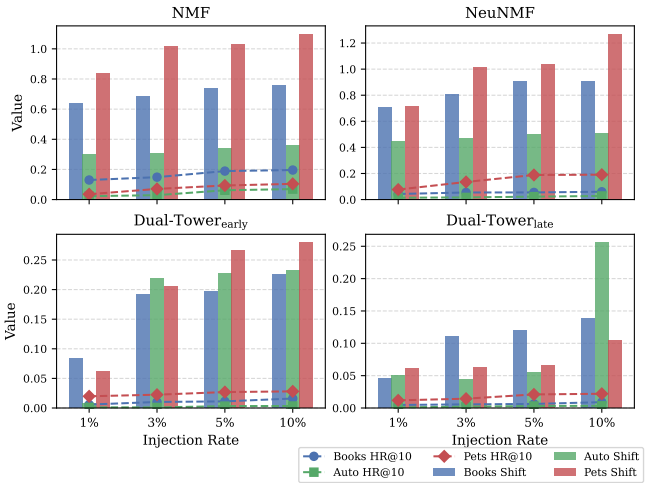


Figure 5: Attack effectiveness at different injection ratios.

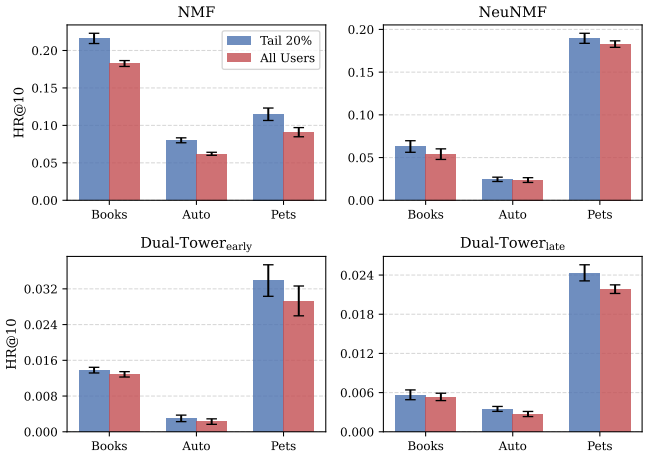


Figure 6: Comparison of AgentSA's impact on low-activity users (bottom 20%) vs. overall population.

shilling attacks. To investigate this further, we analyze the impact of AgentSA on low-activity users by comparing the bottom 20% of genuine users (by interaction count) against the overall user population. As shown in Figure 6, users with fewer interactions tend to achieve higher HR@10 for the target item after the attack, indicating greater susceptibility. This is likely due to the system's limited ability to accurately model the preferences of low-activity users, making them more easily influenced.

In real-world RSs, low-activity users are at greater risk of churn. Their weaker commitment makes them more vulnerable to manipulated recommendations, as confirmed by our results. These findings suggest that the harm of AgentSA may be greater than anticipated.

5.4.4 Vulnerability of Long-Tail Items. Push attacks typically target low-popularity items, as popular items seldom require further promotion. To reflect this practical setting, we select target items from the bottom 20% of the popularity distribution and compare them

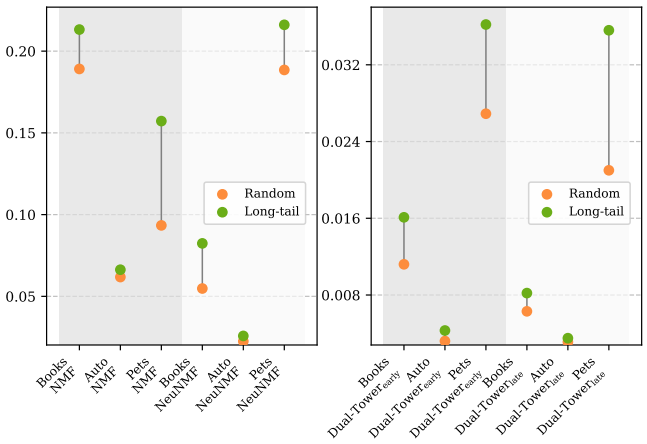


Figure 7: Comparison of HR@10 when targeting long-tail vs. randomly selected items.

with randomly selected targets. As shown in Figure 7, AgentSA performs significantly better when attacking low-popularity items.

This performance gap is mainly attributed to the sparsity of interactions, which makes low-popularity items highly sensitive to even a few injected positives. As a result, push attacks substantially boost the rank and exposure of long-tail items, while popular items remain more resistant due to their abundant historical signals. These observations underscore that AgentSA poses a heightened risk in realistic scenarios where attackers deliberately focus on underexposed items.

6 Conclusion and Future work

We propose AgentSA, a shilling attack framework that leverages LLM-based agents to simulate fake users and perform rating- and review-based attacks on RSs, and it outperforms existing low-knowledge baselines in both effectiveness and stealth.

While our evaluation focuses on e-commerce, the implications could be more severe for user-generated content (UGC) platforms, where manipulated recommendations can lure content creators into producing content that caters to fabricated user interests [2, 18]. Over time, this may cause thematic drift, reduce content diversity, and ultimately lead to user disengagement and community decline. As LLM-based agents continue to evolve, the rise of virtual users in online communities becomes an increasingly tangible reality, our work thus signals a critical need to address these emerging security threats, demanding new research on resilient defenses.

Acknowledgments

This work is supported by National Natural Science Foundation of China (NSFC) under the Grant No. 62172106. Peng Zhang is a faculty of College of Computer Science and Artificial Intelligence, Fudan University. Tun Lu is a faculty of College of Computer Science and Artificial Intelligence, Shanghai Key Laboratory of Data Science, Fudan Institute on Aging, MOE Laboratory for National Development and Intelligent Governance, and Shanghai Institute of Intelligent Electronics & Systems, Fudan University.

Ethical Considerations

The AgentSA framework could theoretically be misused to manipulate RSs through the injection of sophisticated, LLM-generated user profiles and interactions.

However, this work is presented as a proactive form of offensive security research, intended not to furnish an attack tool, but to identify and analyze an emerging vulnerability to foster the development of effective countermeasures. By analyzing the attack's properties, our work offers critical insights for future mitigation strategies. For instance, our findings indicate that defense systems must: (1) move beyond conventional rating pattern analysis to scrutinize the authenticity of review content, and (2) bolster protections for the system's most vulnerable aspects, namely long-tail items and low-activity users, which we identify as highly susceptible. Ultimately, we believe this research contributes to the long-term health and integrity of the recommender ecosystem.

Our research was conducted in accordance with the established ethical principles of our field. All experiments were performed in a controlled, offline environment using public, static datasets, ensuring that no real-world systems or users were affected.

References

[1] Runa Bhaumik, Bamshad Mobasher, and Robin Burke. 2011. A clustering approach to unsupervised attack detection in collaborative recommender systems. In *Proceedings of the International Conference on Data Science (ICDATA)*. 1.

[2] Craig Boutilier, Martin Mladenov, and Guy Tennenholtz. 2023. Modeling recommender ecosystems: Research challenges at the intersection of mechanism design, reinforcement learning and generative models.

[3] Robin Burke, Bamshad Mobasher, and Runa Bhaumik. 2005. Limited knowledge shilling attacks in collaborative filtering systems. In *Proceedings of 3rd international workshop on intelligent techniques for web personalization (ITWP 2005), 19th international joint conference on artificial intelligence (IJCAI 2005)*. 17–24.

[4] Shihao Cai, Jizhi Zhang, Keqin Bao, Chongming Gao, Qifan Wang, Fuli Feng, and Xiangnan He. 2025. Agentic Feedback Loop Modeling Improves Recommendation and User Simulation. arXiv:2410.20027 [cs.IR] https://arxiv.org/abs/2410.20027

[5] Hung-Yun Chiang, Yi-Syuan Chen, Yun-Zhu Song, Hong-Han Shuai, and Jason S Chang. 2023. Shilling black-box review-based recommender systems through fake review generation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 286–297.

[6] Konstantina Christakopoulou and Arindam Banerjee. 2018. Adversarial recommendation: Attack of the learned fake users. *arXiv preprint arXiv:1809.08336* (2018).

[7] Konstantina Christakopoulou and Arindam Banerjee. 2019. Adversarial attacks on an oblivious recommender. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 322–330.

[8] Yashar Deldjoo, Tommaso Di Noia, and Felice Antonio Merra. 2019. Assessing the impact of a user-item collaborative attack on class of users. *arXiv preprint arXiv:1908.07968* (2019).

[9] Gintare Karolina Dziugaite and Daniel M Roy. 2015. Neural network matrix factorization. *arXiv preprint arXiv:1511.06443* (2015).

[10] Luke Friedman, Sameer Ahuja, David Allen, Zhenning Tan, Hakim Sidahmed, Changbo Long, Jun Xie, Gabriel Schubiner, Ajay Patel, Harsh Lara, et al. 2023. Leveraging large language models in conversational recommender systems.

[11] Jingyue Gao, Yang Lin, Yasha Wang, Xiting Wang, Zhao Yang, Yuanduo He, and Xu Chu. 2020. Set-sequence-graph: A multi-view approach towards exploiting reviews for recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 395–404.

[12] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets.

[13] Mingzhe Han, Dongsheng Li, Jiafeng Xia, Jiahao Liu, Hansu Gu, Peng Zhang, Ning Gu, and Tun Lu. 2025. FedCIA: Federated Collaborative Information Aggregation for Privacy-Preserving Recommendation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1687–1696.

[14] Chengkai Huang, Junda Wu, Yu Xia, Zixu Yu, Ruhan Wang, Tong Yu, Ruiyi Zhang, Ryan A Rossi, Branislav Kveton, Dongruo Zhou, et al. 2025. Towards agentic recommender systems in the era of multimodal large language models. *arXiv preprint arXiv:2503.16734* (2025).

[15] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2333–2338.

[16] Xu Huang, Jianxun Lian, Yuxuan Lei, Jing Yao, Defu Lian, and Xing Xie. 2023. Recommender ai agent: Integrating large language models for interactive recommendations.

[17] Neil Hurley, Zunping Cheng, and Mi Zhang. 2009. Statistical attack detection. In *Proceedings of the third ACM conference on Recommender systems*. 149–156.

[18] Haruka Kiyohara, Fan Yao, and Sarah Dean. 2025. Policy Design for Two-sided Platforms with Participation Dynamics.

[19] Shyong K Lam and John Riedl. 2004. Shilling recommender systems for fun and profit. In *Proceedings of the 13th international conference on World Wide Web*. 393–402.

[20] Daniel Lee and H Sebastian Seung. 2000. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems* 13.

[21] Seungpil Lee, Woochang Sim, Donghyeon Shin, Wongyu Seo, Jiwon Park, Seokki Lee, Sanha Hwang, Sejin Kim, and Sundong Kim. 2024. Reasoning abilities of large language models: In-depth analysis on the abstraction and reasoning corpus. *ACM Transactions on Intelligent Systems and Technology* (2024).

[22] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. 9459–9474 pages.

[23] Chen Lin, Si Chen, Hui Li, Yanghua Xiao, Lianyun Li, and Qian Yang. 2020. Attacking recommender systems with augmented user profiles. In *Proceedings of the 29th ACM international conference on information & knowledge management (CIKM '20)*. 855–864.

[24] Chen Lin, Si Chen, Meifang Zeng, Sheng Zhang, Min Gao, and Hui Li. 2022. Shilling black-box recommender systems by learning to generate fake user profiles. *IEEE Transactions on Neural Networks and Learning Systems* 35, 1 (2022), 1305–1319.

[25] Jiahao Liu, Shengkang Gu, Dongsheng Li, Guangping Zhang, Mingzhe Han, Hansu Gu, Peng Zhang, Tun Lu, Li Shang, and Ning Gu. 2025. AgentCF++: Memory-enhanced LLM-based Agents for Popularity-aware Cross-domain Recommendations. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2566–2571.

[26] Jiahao Liu, Dongsheng Li, Hansu Gu, Tun Lu, Jiongran Wu, Peng Zhang, Li Shang, and Ning Gu. 2023. Recommendation unlearning via matrix correction. *arXiv preprint arXiv:2307.15960* (2023).

[27] Jiahao Liu, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, and Ning Gu. 2022. Parameter-free dynamic graph embedding for link prediction. *Advances in Neural Information Processing Systems* 35 (2022), 27623–27635.

[28] Jiahao Liu, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, Li Shang, and Ning Gu. 2023. Personalized graph signal processing for collaborative filtering. In *Proceedings of the ACM Web Conference 2023*. 1264–1272.

[29] Jiahao Liu, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, Li Shang, and Ning Gu. 2023. Triple structural information modelling for accurate, explainable and interactive recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1086–1095.

[30] Jiahao Liu, Dongsheng Li, Hansu Gu, Peng Zhang, Tun Lu, Li Shang, and Ning Gu. 2025. Unbiased Collaborative Filtering with Fair Sampling. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2555–2559.

[31] Jiahao Liu, Yiyang Shao, Peng Zhang, Dongsheng Li, Hansu Gu, Chao Chen, Longzhi Du, Tun Lu, and Ning Gu. 2025. Filtering Discomforting Recommendations with Large Language Models. In *Proceedings of the ACM on Web Conference 2025*. 3639–3650.

[32] Jiahao Liu, Xueshuo Yan, Dongsheng Li, Guangping Zhang, Hansu Gu, Peng Zhang, Tun Lu, Li Shang, and Ning Gu. 2025. Improving LLM-powered Recommendations with Personalized Information. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2560–2565.

[33] Sijia Liu, Jiahao Liu, Hansu Gu, Dongsheng Li, Tun Lu, Peng Zhang, and Ning Gu. 2023. Autoseqrec: Autoencoder for efficient sequential recommendation. In *Proceedings of the 32nd ACM international conference on information and knowledge management*. 1493–1502.

[34] Thanh Toan Nguyen, Nguyen Quoc Viet Hung, Thanh Tam Nguyen, Thanh Trung Huynh, Thanh Thi Nguyen, Matthias Weidlich, and Hongzhi Yin. 2024. Manipulating recommender systems: A survey of poisoning attacks and countermeasures. *Comput. Surveys* 57, 1 (2024), 1–39.

[35] Michael P O'Mahony, Neil J Hurley, and Guénolé CM Silvestre. 2005. Recommender systems: Attack types and strategies. In *AAAI*. 334–339.

[36] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks.

[37] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[38] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2010. Introduction to recommender systems handbook. In *Recommender systems handbook*. Springer, 1–35.
[39] Carlos E Seminario and David C Wilson. 2016. Nuking Item-Based Collaborative Recommenders with Power Items and Multiple Targets.. In *FLAIRS*. 560–565.
[40] Anu Shrestha, Francesca Spezzano, and Maria Soledad Pera. 2021. *An empirical analysis of collaborative recommender systems robustness to shilling attacks*.
[41] Yubo Shu, Haonan Zhang, Hansu Gu, Peng Zhang, Tun Lu, Dongsheng Li, and Ning Gu. 2024. RAH! RecSys–Assistant–Human: A Human-Centered Recommendation Framework With LLM Agents. *IEEE Transactions on Computational Social Systems* (2024).
[42] Junshuai Song, Zhao Li, Zehong Hu, Yucheng Wu, Zhenpeng Li, Jian Li, and Jun Gao. 2020. Poisonrec: an adaptive data poisoning framework for attacking black-box recommender systems. In *2020 IEEE 36th international conference on data engineering (ICDE)*. IEEE, 157–168.
[43] Agnideven Palanisamy Sundar, Feng Li, Xukai Zou, Tianchong Gao, and Evan D Russomanno. 2020. Understanding shilling attacks and their detection traits: A comprehensive survey. *IEEE Access* 8 (2020), 171703–171715.
[44] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems* 27.
[45] Jiaxi Tang, Hongyi Wen, and Ke Wang. 2020. Revisiting adversarially learned injection attacks against recommender systems. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 318–327.
[46] Lei Wang, Jingsen Zhang, Hao Yang, Zhi-Yuan Chen, Jiakai Tang, Zeyu Zhang, Xu Chen, Yankai Lin, Hao Sun, Ruihua Song, et al. 2025. User behavior simulation with large language model-based agents. *ACM Transactions on Information Systems* 43, 2 (2025), 1–37.
[47] Chad Williams, Bamshad Mobasher, Robin Burke, Jeff Sandvig, and Runa Bhaumik. 2006. Detection of obfuscated attacks in collaborative recommender systems. In *Proceedings of the ECAI*, Vol. 6.
[48] David C Wilson and Carlos E Seminario. 2013. When power users attack: assessing impacts in collaborative recommender systems. In *Proceedings of the 7th ACM conference on Recommender systems*. 427–430.
[49] Jiongran Wu, Jiahao Liu, Dongsheng Li, Guangping Zhang, Mingzhe Han, Hansu Gu, Peng Zhang, Li Shang, Tun Lu, and Ning Gu. 2025. Bidirectional Knowledge Distillation for Enhancing Sequential Recommendation with Large Language Models. *arXiv preprint arXiv:2505.18120* (2025).
[50] Wujiang Xu, Yunxiao Shi, Zujie Liang, Xuying Ning, Kai Mei, Kun Wang, Xi Zhu, Min Xu, and Yongfeng Zhang. 2025. Instructagent: Building user controllable recommender via llm agent. *arXiv preprint arXiv:2502.14662* (2025).
[51] Fan Yang, Min Gao, Junliang Yu, Yuqi Song, and Xinyi Wang. 2018. Detection of shilling attack based on bayesian model and user embedding. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 639–646.
[52] Shiyi Yang, Lina Yao, Chen Wang, Xiwei Xu, and Liming Zhu. 2023. Incorporated Model-Agnostic Profile Injection Attacks on Recommender Systems. In *2023 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1481–1486.
[53] An Zhang, Yuxin Chen, Leheng Sheng, Xiang Wang, and Tat-Seng Chua. 2024. On generative agents in recommendation. In *Proceedings of the 47th international ACM SIGIR conference on research and development in Information Retrieval*. 1807–1817.
[54] Guangping Zhang, Peng Zhang, Jiahao Liu, Zhuoheng Li, Dongsheng Li, Hansu Gu, Tun Lu, and Ning Gu. 2025. EvalAgent: Towards Evaluating News Recommender Systems with LLM-based Agents. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*. 4086–4095.
[55] Jizhi Zhang, Keqin Bao, Wenjie Wang, Yang Zhang, Wentao Shi, Wanhong Xu, Fuli Feng, and Tat-Seng Chua. 2024. Prospect personalized recommendation on large language model-based agent platform.
[56] Junjie Zhang, Yupeng Hou, Ruobing Xie, Wenqi Sun, Julian McAuley, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2024. Agentcf: Collaborative learning with autonomous language agents for recommender systems. In *Proceedings of the ACM Web Conference 2024*. 3679–3689.
[57] Yongfeng Zhang, Yunzhi Tan, Min Zhang, Yiqun Liu, Tat-Seng Chua, and Shaoping Ma. 2015. Catch the Black Sheep: Unified Framework for Shilling Attack Detection Based on Fraudulent Action Propagation.. In *IJCAI*, Vol. 15. 2408–2414.
[58] Zijian Zhang, Shuchang Liu, Ziru Liu, Rui Zhong, Qingpeng Cai, Xiangyu Zhao, Chunxu Zhang, Qidong Liu, and Peng Jiang. 2025. Llm-powered user simulator for recommender system. 13339–13347 pages.
[59] Lixi Zhu, Xiaowen Huang, and Jitao Sang. 2024. How reliable is your simulator? analysis on the limitations of current llm-based user simulators for conversational recommendation. In *Companion Proceedings of the ACM Web Conference 2024*. 1726–1732.