Benchmark Leakage Trap: Can We Trust LLM-based Recommendation?

Anonymous ACL submission

Abstract

The expanding integration of Large Language Models (LLMs) into recommender systems poses critical challenges to evaluation reliability. This paper identifies and investigates a previously overlooked issue: benchmark data leakage in LLM-based recommendation. This phenomenon occurs when LLMs are exposed to and potentially memorize benchmark datasets during pre-training or fine-tuning, leading to artificially inflated performance metrics that fail to reflect true model performance. To validate this phenomenon, we simulate diverse data leakage scenarios by conducting continued pre-training of foundation models on strategically blended corpora, which include user-item interactions from both in-domain and out-of-domain sources. Our experiments reveal a dual-effect of data leakage: when the leaked data is domain-relevant, it induces substantial but spurious performance gains, misleadingly exaggerating the model's capability. In contrast, domain-irrelevant leakage typically degrades recommendation accuracy, highlighting the complex and contingent nature of this contamination. Our findings reveal that data leakage acts as a critical, previously unaccounted-for factor in LLM-based recommendation, which could impact the true model performance.

1 Introduction

Recently, the Large Language Models (LLMs) (e.g., GPT-4 (Achiam et al., 2023), LLaMA (Touvron et al., 2023)) demonstrate unprecedented capabilities in natural language understanding, which has been employed in various fields. The recommender systems based on large language models (LLMs) have emerged as a promising direction (Wu et al., 2024; Wang et al., 2024). These models are pre-trained on massive corpora and encode rich knowledge information, which could enhance the recommender systems by deeper understanding of item content, user intent and contextual nuances (Geng et al., 2022; Li et al., 2024; Liu et al., 2023).

For example, BinLLM (Zhang et al., 2024) integrates collaborative information into LLMs using a text-like encoding strategy for recommendation, which has demonstrated superior performance.

However, **should the results of these LLM-based recommender systems really be considered reliable?** Recent studies have revealed that large language models are susceptible to data leakage, where models inadvertently memorize and reproduce training data during inference (Carlini et al., 2021; Hisamoto et al., 2020; Carlini et al., 2022). This phenomenon has been extensively documented in various domains, from text generation (Carlini et al., 2023) to question answering (Kandpal et al., 2022), where LLMs have been shown to reproduce verbatim passages from their training corpora. And it largely affects the evaluation of model performance (Zhou et al., 2023; Golchin and Surdeanu, 2024). Moreover, LLM has been proven to have the ability to remember recommendation data (Di Palma et al., 2025). Although there is ample evidence that LLMs are exposed to benchmark data during pre-training and affect downstream inference results(Carlini et al., 2021; Kandpal et al., 2022), it remains unclear how leaked LLMs affect downstream recommendation results when used as the backbone model of recommendation systems.

This raises our guess: integrating a memorization-prone LLM into the recommendation pipeline may cause the resulting LLM-based recommender systems to inherit and potentially amplify the data leakage characteristics inherent in the LLM. In fact, the core of recommender systems is user interest modeling and item characteristics learning (Zhang et al., 2024; Rendle, 2012). However, the inherent data leakage in LLMs can obscure the boundary between authentic user preferences and memorized data artifacts.

This interference not only distorts the modeling of latent user interests, but also compromises the learning of item representations, as the model may rely on memorized associations rather than underlying and real characteristics. Such leakage could ultimately bias the evaluation of LLM-based recommender systems, as test results may reflect the model's prior exposure to evaluation data rather than its true recommendation ability.

To investigate the impact of data leakage on LLM-based recommendation systems, we design an experimental framework that could simulate real-world leakage scenarios. First, we construct mixed leakage datasets with controlled proportions by blending in-domain data (sampled from the target evaluation datasets) with out-of-domain data (collected from six external sources). Second, to simulate leakage scenarios, we fine-tune the LLMs via Low-Rank Adaptation (LoRA) based on the blended leakage dataset, named as dirty LLM. This choice of LoRA provides a controlled proxy for investigating how benchmark knowledge injection biases recommendation performance while preserving the foundation model's base capabilities. Finally, we evaluated performance by comparing downstream recommendation models built on clean and dirty LLMs in two categories. Specifically, we evaluate the impact of data leakage on (1) different model architectures, (2) various leakage compositions (pure in-domain, pure out-of-domain, and mixed), and (3) different types of out-of-domain data sources. In this way, we can analyze the factors that determine a model's susceptibility or robustness to data leakage.

Our work makes the following key contributions:

1. We identify and empirically demonstrate the benchmark data leakage problem in LLM-based recommender systems, revealing how pre-exposed LLMs can compromise the integrity of downstream evaluation metrics.

2. We develop a novel methodology for simulating realistic data leakage scenarios through controlled fine-tuning on strategically composed mixed-domain datasets, enabling investigation of how different types and degrees of data exposure affect recommendation performance.

3. Through extensive experiments, we demonstrate that: (i) impacts vary substantially across different model architectures and their resilience to leakage, (ii) in-domain leakage creates deceptive performance gains that mask true model capability, while (iii) out-of-domain contamination typically impairs performance, revealing a nuanced dual-effect phenomenon that challenges current evaluation practices.

2 Related Work

2.1 Large Language Models

Large Language Models (LLMs) are a class of foundational models pre-trained on vast corpora of text data, demonstrating remarkable capabilities in understanding and generating human-like text (Achiam et al., 2023; Bommasani et al., 2021; Brown et al., 2020). These models, typically based on the Transformer architecture, learn to predict the next token in a sequence, thereby capturing intricate patterns, facts, and reasoning abilities embedded in the training data (Wei et al., 2022; Kaplan et al., 2020). A significant security and privacy concern associated with these models is **data leakage**, wherein LLMs can store and subsequently reproduce verbatim content from their training datasets (Carlini et al., 2021). The study of data leakage in LLMs has become a critical area of research. Seminal work by Carlini et al. (Carlini et al., 2021) first systematically demonstrates that LLMs can memorize and regurgitate training data, showing that simple prompting strategies could recover exact sequences. (Zhou et al., 2023) shows that data leakage of large models will interfere with subsequent performance evaluation. And in response to address this problem, mitigation strategies like differential privacy and training data deduplication have been proposed to reduce these risks (Kandpal et al., 2022).

2.2 LLM-based Recommendation

The field of LLM-based recommendation has evolved rapidly, with researchers exploring various architectural paradigms to integrate language models with recommendation tasks (Wu et al., 2024; Lin et al., 2025). LLMRec and LLMRec+Collab. are two primary architectural paradigms, which are directly represented in our experimental evaluation.

The first category, **LLMRec methods**, leverages the inherent language understanding capabilities of LLMs directly, with minimal architectural modification. This category encompasses a spectrum of techniques: ICL (Dai et al., 2023) operates in

a zero-shot manner through in-context learning alone, Prompt4NR (Zhang and Wang, 2023) explores generic soft-prompt tuning for the recommendation task, while TALLRec (Bao et al., 2023) specializes the model via instruction tuning.

In contrast, the second category, **LLM-Rec+Collab. methods**, explicitly integrates collaborative filtering signals to enhance the LLM's reasoning with user-item interaction patterns. The integration strategies, however, vary significantly: Personalized recommendation (Lyu et al., 2024) injects collaborative information by introducing trainable token embeddings for users and items, effectively performing personalized soft-prompt tuning. CoLLM (Zhang et al., 2025), in its two instantiations, maps pre-trained collaborative embeddings from Matrix Factorization (CoLLM-MF) or a Deep Interest Network (CoLLM-DIN) into the LLM's latent space. A distinctly different approach is taken by BinLLM (Zhang et al., 2024), which encodes collaborative signals into a text-like format of binary hash codes for direct inclusion in the model's prompt. This clear taxonomic distinction provides the essential context for interpreting our experimental results on how these fundamentally different paradigms respond to data leakage scenarios.

3 Methodology

This section details our experimental framework for investigating data leakage in LLM-based recommender systems. We first provide an overview of the experimental pipeline, followed by the construction of leakage data, the contamination process through LoRA adaptation, and the formulation of clean and dirty downstream recommenders.

3.1 Overview

The complete workflow of our study is illustrated in Figure 1. To explore the effect of data leakage on recommendation models, we begin by sampling a portion of the original dataset to serve as leaked data, which contains a pre-defined ratio of in-domain (ID) and out-of-domain (OOD) samples depending on whether the data originate from the same domain as the evaluation benchmark.

We start from an open-source large language model (Vicuna-7B) that has already been pre-trained on large-scale text corpora and denote it as the Clean LLM (i.e., without leakage). To emulate contamination, we fine-tune this Clean LLM on the mixed leakage dataset using Low-Rank Adaptation (LoRA) (Hu et al., 2022). Through this process, the model learns leakage-related patterns that are encoded into the newly added low-rank parameters, producing a Dirty LLM. The base parameters remain frozen throughout training, ensuring that only the LoRA modules carry the injected information.

Subsequently, both Clean and Dirty LLMs are employed as backbone encoders in downstream recommendation systems, forming Clean Recommenders and Dirty Recommenders respectively. All hyperparameters, data partitions, and model configurations are kept identical between the two to ensure a fair comparison. The performance differences between the two systems therefore directly reflect the influence of data leakage.

During evaluation, Clean Recommenders serve as baseline models for standard performance testing, while Dirty Recommenders are used to observe the potential distortion of benchmark results under contaminated conditions. By comparing the performance metrics (AUC/UAUC) of the Clean and Dirty Recommenders, we identify the *Dual Effect of Leakage*: (1) **Spurious Gains** induced by domain-relevant leakage and (2) performance **Degradation** caused by domain-irrelevant contamination. This design allows us to analyze how different leakage types and proportions affect model reliability in a controlled and reproducible way.

3.2 Leakage Data Construction

To simulate realistic benchmark leakage scenarios, we construct a mixed leakage dataset that combines both in-domain and out-of-domain sources. Each dataset is split into training, validation, and test sets following a 7:2:1 ratio.

First, we randomly sample 10% from the target dataset as in-domain leakage:

$$D_{\text{ID}} \sim \text{Sample}\big(D_{\text{target}},\, p = 0.1\big)\,. \qquad (1)$$

Here, D_{ID} denotes the in-domain leakage subset drawn from the target dataset D_{target} at proportion p=0.1. The operator $\text{Sample}(\cdot)$ denotes uniform sampling according to the stated proportion. D_{ID} **contains training set, validation set, and test set, and the ratio remains 7:2:1.** This step fixes the reference scale for subsequent OOD construction.

Then, we draw OOD data from six external datasets (Epinions, Last.fm, MIND, Amazon-Sports, Amazon-Beauty, Gowalla), with the total OOD size equal to $6 \times |D_{\text{ID}}|$ and allocated

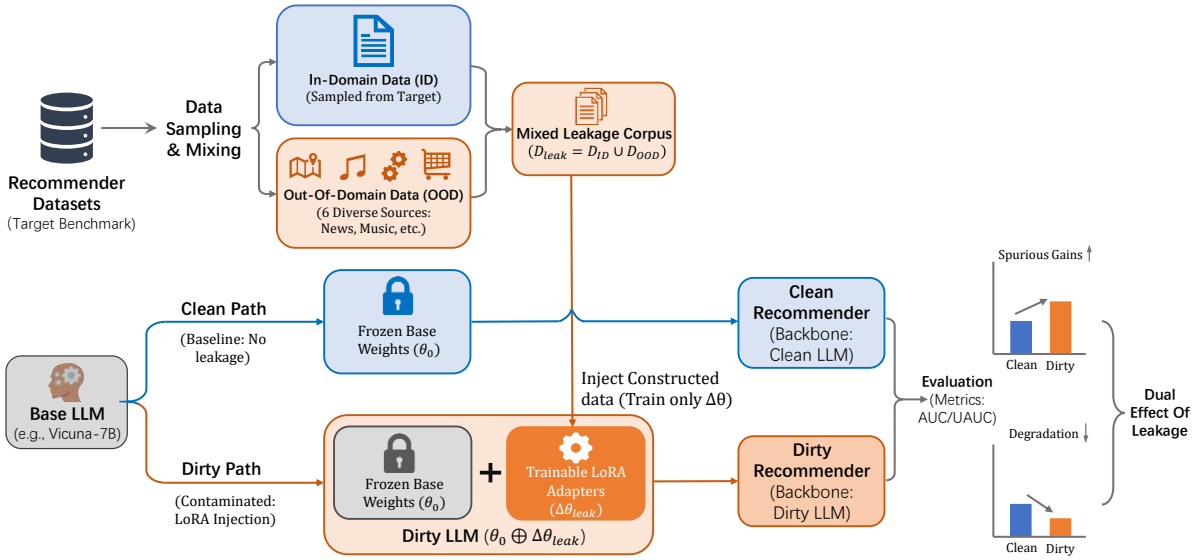

Figure 1: Overview of the Experimental Framework. The workflow illustrates the construction of a Mixed Leakage Corpus through strategic sampling of In-Domain (ID) and Out-Of-Domain (OOD) data. The Clean Path represents the baseline evaluation using a frozen Base LLM, while the Dirty Path simulates benchmark contamination via LoRA injection. By comparing the Clean and Dirty Recommenders, we identify a dual-effect of leakage: spurious performance gains or degradation.

equally across sources:

$$D_{\text{OOD}}^{(i)} \sim \text{Sample}\Big(D_{\text{ext}}^{(i)}, |D_{\text{ID}}|\Big), \quad i \in \{1, \ldots, 6\}, \tag{2}$$

$$D_{\text{OOD}} = \bigcup_{i=1}^{6} D_{\text{OOD}}^{(i)}, \qquad |D_{\text{OOD}}| = 6\,|D_{\text{ID}}|. \tag{3}$$

In (2), each $D_{\text{OOD}}^{(i)}$ is sampled from the i-th external dataset $D_{\text{ext}}^{(i)}$ with *the same* target size $|D_{\text{ID}}|$, ensuring that every source contributes equally. The union in (3) aggregates the six OOD portions into D_{OOD}, yielding a total size of $6\,|D_{\text{ID}}|$. Here $|\cdot|$ denotes cardinality.

Finally, we combine ID and OOD to obtain the mixed leakage set:

$$\begin{aligned} D_{\text{leak}} &= D_{\text{ID}} \cup D_{\text{OOD}}, \\ |D_{\text{leak}}| &= |D_{\text{ID}}| + |D_{\text{OOD}}| = 7\,|D_{\text{ID}}|. \end{aligned} \tag{4}$$

Equation (4) defines the final leakage corpus D_{leak} as the combination of the in-domain portion D_{ID} and the aggregated out-of-domain portion D_{OOD}, whose size follows directly from the construction above, i.e., $|D_{\text{leak}}| = 7\,|D_{\text{ID}}|$.

D_{leak} is transformed into prompts and then injected into the LLM. **Appendix A shows the details.**

3.3 Injecting leaked data

We start from an already pretrained base LLM with parameters θ_0. This model is used directly as the Clean LLM and its parameters remain frozen throughout the whole study. To obtain the Dirty LLM, we optimize only low-rank LoRA adapters on the mixed leakage dataset D_{leak} (constructed in (4)), while keeping θ_0 unchanged.

Loss definition Let $L_{\text{LLM}}(\cdot)$ denote the standard negative log-likelihood (NLL) under next-token prediction:

$$L_{\text{LLM}}(\theta, D) = -\sum_{(x,y)\in D} \log P(y \mid x; \theta), \tag{5}$$

where each pair (x, y) consists of an input sequence x and target tokens y, and $P(y \mid x; \theta)$ is the model likelihood parameterized by θ. This loss is adopted *only* to train LoRA adapters; it does not modify the frozen base θ_0.

Contaminated adaptation (Dirty LLM) We train LoRA adapters on D_{leak} as follows:

$$\Delta\theta_{\text{LoRA}}^{\text{leak}} = \arg\min_{\Delta\theta_{\text{LoRA}}} L_{\text{LLM}}\big(\theta_0 \oplus \Delta\theta_{\text{LoRA}},\ D_{\text{leak}}\big), \tag{6}$$

$$\theta_{\text{LLM}}^{\text{dirty}} = \theta_0 \oplus \Delta\theta_{\text{LoRA}}^{\text{leak}}. \tag{7}$$

Here, $\oplus$ denotes the *injection* of LoRA adapters into the frozen base (i.e., composing the base

weights with low-rank additive updates).[1] The optimization in (6) finds the adapter parameters that *best fit the leakage corpus*, thereby encoding leakage-specific patterns into $\Delta\theta^{\text{leak}}_{\text{LoRA}}$. Equation (7) defines the Dirty LLM as the composition of the unchanged base θ_0 and the learned leakage adapters.

LoRA parameterization and what is updated For any affine weight W in the frozen base, the LoRA update is

$$W_{\text{new}} = W + \frac{\alpha}{r} BA, \tag{8}$$

where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ are trainable low-rank matrices (rank $r \ll \min(d,k)$), and α scales the update magnitude. The product BA is a rank-r correction that *adds* to the frozen weight W; hence *only* A, B are optimized, while all base weights $W \subset \theta_0$ remain fixed. This factorization provides two benefits: (i) **parameter efficiency** (few trainable parameters) and (ii) **isolation of contamination** (leakage information is confined within $\Delta\theta_{\text{LoRA}}$, enabling a clean comparison against the base model).

In subsequent experiments, the Clean LLM is simply θ_0 (no adapters), and the Dirty LLM is $\theta^{\text{dirty}}_{\text{LLM}}$ from (7). **And the reason for using LoRa fine-tuning instead of full-parameter training is explained in Appendix B.**

4 Experiment

In this section, we first describe the overall experimental settings, including datasets and baselines. Then, we present and analyze the experimental results to evaluate the effectiveness of our proposed method. **The implementation details are provided in Appendix C.**

4.1 Experiment settings

4.1.1 Datasets

For evaluating the impact of LLM data leakage on downstream recommendation models, we conduct experiments on two widely-used recommendation datasets:

- **ML-1M** (Harper and Konstan, 2015):MovieLens-1M recommendation benchmark, a widely used dataset for movie recommendation containing 1000209 timestamped user–movie ratings (1–5 scale) from 6040 users on 3952 movies. In addition to user-item interactions, it provides side information such as movie titles, etc.

- **Amazon-Book** (Ni et al., 2019): This dataset is derived from the "Books" category of the well-known Amazon Product Review dataset. It comprises book purchasing and rating records, user-book interactions within the e-commerce domain.

For simulating the data leakage, we employ six external datasets from different domains to serve as out-of-domain leakage data, including Epinions (Massa and Avesani, 2007), Last.fm (Celma Herrada et al., 2009), MIND (Wu et al., 2020), Amazon-Sports (Ni et al., 2019), Amazon-Beauty (Ni et al., 2019) and Gowalla (Cho et al., 2011). **The detailed information of the above datasets is presented in Appendix D.**

These OOD datasets are strategically selected based on two criteria: (i) *Semantic Divergence*: they cover diverse domains ranging from news to geographical check-ins to test the model's resistance to non-target semantic noise; (ii) *Structural Heterogeneity*: they encompass varied interaction types (e.g., tags, attributes) to observe how the memorization of disparate behavioral patterns interferes with target recommendation logic.

4.1.2 Baselines and Metrics

In this paper, we evaluate the model performance on two categories of LLM-based recommendation models: LLMRec models (ICL, Prompt4NR, TALLRec) and LLMRec models integrated collaborative information (PersonPrompt, CoLLM, BinLLM). **Detailed descriptions of all baseline configurations are provided in Appendix E and the metrics of our experiment are provided in Appendix F.**

4.2 Results and Analysis

4.2.1 Overall Experimental Results

We present the performance comparison of seven recommendation methods on ML-1M and Amazon-Book datasets under mixed data leakage (in-domain 10% + out-of-domain 60%) in Table 1. Overall, data leakage can affect the evaluation of model performance to some extent. We have the following observations.

First, the impact of data leakage on model performance is uncertain. Compared to the baseline, the

[1] We use the same tokenizer and decoding head as the base model; only LoRA parameters are trainable.

Dataset	Method	Category	AUC			UAUC		
			Baseline	After leak	ΔAUC	Baseline	After leak	ΔUAUC
ML-1M	ICL	LLMRec	0.5621	0.5034	**−10.4%**	0.5518	0.5015	**−9.1%**
	Prompt4NR		0.7021	0.6515	−7.2%	0.6679	0.6180	−7.5%
	TALLRec		0.7036	0.6215	**−11.7%**	0.6751	0.5902	**−12.6%**
	CoLLM-MF	LLMRec +Collab.	0.7228	0.7685	+6.3%	0.6782	0.7251	+6.9%
	CoLLM-DIN		0.7184	0.7512	+4.6%	0.6801	0.7085	+4.2%
	PersonPrompt		0.7151	0.6914	−3.3%	0.6487	0.6275	−3.3%
	BinLLM		0.7379	0.7823	+6.0%	0.6908	0.7345	+6.3%
Amazon-Book	ICL	LLMRec	0.5542	0.5011	**−9.6%**	0.5487	0.5008	**−8.7%**
	Prompt4NR		0.7168	0.6275	**−12.5%**	0.5804	0.5052	**−13.0%**
	TALLRec		0.7302	0.7758	+6.2%	0.5892	0.6285	+6.7%
	CoLLM-MF	LLMRec +Collab.	0.8034	0.8351	+3.9%	0.6147	0.6415	+4.4%
	CoLLM-DIN		0.8186	0.7814	−4.5%	0.6401	0.6135	−4.2%
	PersonPrompt		0.7209	0.7712	+7.0%	0.5876	0.6251	+6.4%
	BinLLM		0.8207	0.7456	**−9.1%**	0.6262	0.5723	**−8.6%**

Table 1: AUC/UAUC comparison on the ML-1M and Amazon-Book datasets under mixed data leakage (in-domain 10% + out-of-domain 60%). "Collab." denotes collaborative recommendation methods. ΔAUC denotes the rate of change of AUC. ΔUAUC denotes the rate of change of UAUC. Bold values in Δ columns indicate significant performance variations ($|\Delta| \geq 8\%$).

recommender's performance on the two datasets sometimes improves and sometimes deteriorates. In the ML-1M dataset, TALLRec's AUC drops from 0.7036 to 0.6251 after injecting mixed data. And CoLLM-MF's AUC upgrades from 0.7228 to 0.7685. The same model can perform differently on different datasets. For example, BinLLM upgrades from 0.7379 to 0.7823 but it drops from 0.8207 to 0.7456. It is also worth noting that the trends and magnitudes of AUC and UAUC are basically consistent. This indicates that data leaks have a roughly consistent impact on both overall and user-level recommendation tasks. So in the subsequent analysis, we only refer to the changes in AUC.

Second, the impact of data leakage on model performance evaluation cannot be ignored. Recommenders trained on LLMs with data leaks may change their original rankings when undergoing benchmark testing. In the Amazon-book dataset, PersonPrompt's performance is not as good as BinLLM. But after injecting the mixed dataset, PersonPrompt is superior to BinLLM under this benchmark.

Finally, different categories of models have varying sensitivities to data leakage. Recommendation models of the "LLMRec" class (Uses a large language model directly for recommendation) are more susceptible to the effects of hybrid databases compared with recommendation models of "LLMRec+Collab." class (Augments an LLM with collaborative filtering signals). LLMs are strong at modeling textual semantics, whereas recommendation relies heavily on collaborative signals from user–item interactions. Injecting or fine-tuning LLMs with collaborative information aligns them with the behavior-prediction objective, leading to better personalization, ranking accuracy, and robustness. Therefore, on the baseline, LLMRec+Collab. models perform better. After injecting hybrid data, their performance change is smaller than that of pure LLMRec models. We speculate that this is because pure LLMRec methods lack alternative signals to offset contaminated knowledge, making them more vulnerable to degradation. By contrast, LLMRec+Collab. methods integrate collaborative information as an additional signal that provides redundancy, cross-signal validation and robustness

Dataset	Metric	Subset	Clean	Dirty	Δ
ML-1M	AUC	Seen	0.7412	0.9587	**+29.3%**
		Unseen	0.7368	0.7556	**+2.6%**
	UAUC	Seen	0.6897	0.8784	**+27.4%**
		Unseen	0.6923	0.7187	**+3.8%**
Amazon-Book	AUC	Seen	0.8231	0.9143	**+11.1%**
		Unseen	0.8194	0.8276	**+1.0%**
	UAUC	Seen	0.6304	0.7437	**+18.0%**
		Unseen	0.6248	0.6395	**+2.4%**

Table 2: Performance comparison of the BinLLM architecture on memorized (*Seen*) vs. novel (*Unseen*) interaction pairs across ML-1M and Amazon-Book datasets under 10% in-domain leakage.

enhancement, yielding a system that is less dependent on potentially leaked LLM parameters.

4.2.2 Proof of Memorization

A critical question arises from the observed spurious gains under in-domain leakage: does the performance inflation stem from the model genuinely learning domain-specific collaborative patterns, or is it merely remembering pre-exposed targets? To definitively disentangle these two mechanisms, we conducted a rigorous overlap auditing analysis. We partitioned the test set into two mutually exclusive subsets: the Seen subset, comprising user-item interaction pairs that explicitly appeared in the injected leakage corpus, and the Unseen subset, containing completely novel interactions.

The quantitative results, presented in Table 2, reveal a stark contrast. Under the 10% in-domain leakage setting, the Dirty LLM exhibits an extraordinary performance surge exclusively on the Seen subset. For instance, on ML-1M, the AUC and UAUC inflate by +29.3% and +27.4%, respectively. Conversely, on the Unseen subset, these gains violently diminish to a marginal +2.6% and +3.8%. A consistent phenomenon is observed in the Amazon-Book dataset. This indicates that the performance improvement of the model is extremely largely based on the subset of tests it has seen. The model essentially improves performance through memory retrieval rather than through strong generalization ability. **Appendix G shows further qualitative case studies.**

4.2.3 The Impact of Leakage Intensity and Domain Divergence

We also randomly generate a few other mixed datasets to simulate possible data leaks. In the following experiment, to save space, we just represent the performance of TALLRec, CoLLM-MF and BinLLM under the baseline ML-1M. Note that Tables 3–6 use independently re-sampled leakage corpora and therefore do not need to numerically reconcile with Table 1. We report them to illustrate qualitative trends across leakage compositions rather than to compare absolute magnitudes.

ML-1M (pure ID interference: 10% ID data)				
Metric	**AUC**		**UAUC**	
Method	**After leak**	**ΔAUC**	**After leak**	**ΔUAUC**
TALLRec	0.8795	+25.0%	0.8474	+25.5%
CoLLM-MF	0.8234	+13.9%	0.7723	+13.9%
BinLLM	0.8456	+14.6%	0.7923	+14.7%

Table 3: AUC/UAUC after leak and relative change rate Δ on ML-1M under the "LLM stage 10% data leakage" setting. ID denotes the In-Domain.

ML-1M (pure OOD interference: 60% OOD data)				
Metric	**AUC**		**UAUC**	
Method	**After leak**	**ΔAUC**	**After leak**	**ΔUAUC**
TALLRec	0.5234	−25.6%	0.4923	−27.1%
CoLLM-MF	0.6345	−12.2%	0.5923	−12.7%
BinLLM	0.6234	−15.5%	0.5823	−15.7%

Table 4: AUC/UAUC after leak and relative change rate Δ on ML-1M under the "pure OOD 60%" setting. OOD denotes the Out-of-Domain.

In Tables 3 and 4, we study purely data leakage by only injecting ID (In-Domain) data or OOD (Out-of-Domain), which are two idealized situations. It can be observed that after a data breach, the performance of models in Table 3 improves, while the performance of models in Table 4 declines. This indicates that data leakage within the domain can improve the accuracy of model predictions, whereas the opposite is true for out-of-domain data.

In Table 5, we change the total amount of the hybrid database while we adjust the proportion of data in the internal and external domains in Table 6. The impact of the mixed data from the experiment in Table 5 on the model is smaller, with the variation rates of all three models not exceeding 6%. In Table 6, after increasing the proportion of in-domain data, the performance of downstream models trained by data-contaminated LLMs im-

ML-1M (mixed leakage: 5% in-domain + 30% out-of-domain)				
Metric	**AUC**		**UAUC**	
Method	**After leak**	**ΔAUC**	**After leak**	**ΔUAUC**
TALLRec	0.6812	−3.2%	0.6487	−3.9%
CoLLM-MF	0.7656	+5.9%	0.7123	+5.0%
BinLLM	0.7534	+2.1%	0.7045	+2.0%

Table 5: ML-1M under mixed leakage (5% in-domain + 30% out-of-domain): AUC/UAUC after leak and relative change rate Δ.

ML-1M (mixed leakage: 25% in-domain + 45% out-of-domain)				
Metric	**AUC**		**UAUC**	
Method	**After leak**	**ΔAUC**	**After leak**	**ΔUAUC**
TALLRec	0.7456	+6.0%	0.7123	+5.5%
CoLLM-MF	0.8123	+12.4%	0.7623	+12.4%
BinLLM	0.7823	+6.0%	0.7345	+6.3%

Table 6: ML-1M under mixed leakage (25% in-domain + 45% out-of-domain): AUC/UAUC after leak and relative change rate Δ.

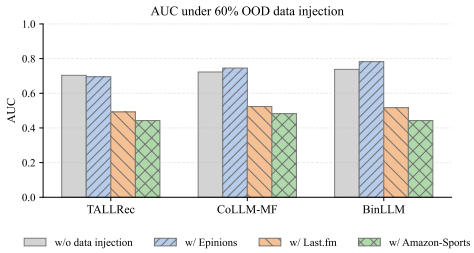

Figure 2: AUC under different data injection methods

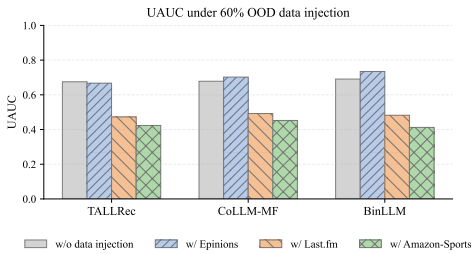

Figure 3: UAUC under different data injection methods

proved. It can be seen that data leakage of different degrees has varying effects on model performance.

In Fig. 2 and 3, we conduct a research on the impact of different data injection methods on model performance. We select three databases with different data construction methods as the OOD data:

- **Epinions**: Built based on user history (the same as the ML-1M).
- **Last.fm**: Built based on user interest tags (Celma Herrada et al., 2009).
- **Amazon-Sports**: Built based on product attributes (Ni et al., 2019).

We find that the models' performance after injecting Epinions data is not significantly different from before the injection while data from Last.fm and Amazon-Sports significantly reduces the accuracy of the model. This is because data from Epinions and baselines are in the same format. Although LLMs remember them, they don't know the answers like they do with test data, but these data also fail to cause much interference. As a result, the impact of different types of out-of-domain data leaks on the model is also inconsistent.

4.2.4 Generalizability Across Foundation Model

To verify whether the benchmark leakage trap is merely an artifact of smaller or earlier open-source models, we extend our evaluation of the BinLLM architecture across a diverse spectrum of contemporary foundation models, ranging from 7B to 70B parameters. **The results and analysis are presented in Appendix H.**

5 Discussion

While our primary framework focuses on auditing and quantifying these leakage risks, we provide a forward-looking discussion on potential mitigation strategies in Appendix I to inspire defense-oriented future work.

6 Conclusion

This study provides empirical evidence of benchmark data leakage in LLM-based recommender systems and yields three key conclusions. First, in-domain data leakage causes substantial performance inflation that misrepresents true model capability. Second, leakage effects are domain-dependent, with out-of-domain contamination generally leading to performance degradation. Third, architectural choices significantly influence susceptibility, as hybrid models incorporating collaborative signals demonstrate greater resilience than pure LLM-based approaches. These findings underscore the critical need for more rigorous evaluation methodologies that account for data leakage risks, ensuring that reported advancements in LLM-based recommendation reflect genuine progress rather than evaluation artifacts.

Limitations

While this paper identifies the critical dual-effect of benchmark leakage in LLM-based recommendation, several limitations present avenues for future research:

Diversity of Leakage Resilience Profiles Although we have extended our empirical evaluation across multiple LLM backbones, the total volume and diversity of our experimental configurations remain bounded. Consequently, we have not yet systematically synthesized or summarized the specific leakage resistance and resilience capabilities across different LLMs and diverse downstream recommendation architectures. Specifying a comprehensive matrix that correlates inherent model traits with contamination resilience remains an open avenue for future investigation.

Modality Boundaries Our investigation currently focuses on text-centric benchmarks. As multimodal LLMs become prevalent in recommendation, how visual or acoustic data leakage influences evaluation reliability remains an unexplored dimension.

Static Evaluation Protocols Our study relies on offline evaluation. The long-term, compounding effects of data leakage in dynamic production environments with continuous user feedback loops require further longitudinal investigation.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *Proceedings of the 17th ACM conference on recommender systems*, pages 1007–1014.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, and 1 others. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Tom B. Brown, Benjamin Mann, Nick Ryder, and 1 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1877–1901.

Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Chen, Matthew Jagielski, Vikash Sehwag, Florian Tramer, Borja Balle, Daphne Ippolito, and Eric Wallace. 2023. Extracting training data from diffusion models. *arXiv preprint arXiv:2301.13188*.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2022. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, and 1 others. 2021. Extracting training data from large language models. *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650.

Òscar Celma Herrada and 1 others. 2009. *Music recommendation and discovery in the long tail*. Universitat Pompeu Fabra.

Eunjoon Cho, Seth A Myers, and Jure Leskovec. 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*, pages 1082–1090.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. 2023. Why can gpt learn in-context? language models secretly perform gradient descent as meta-optimizers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4005–4019.

Dario Di Palma, Felice Antonio Merra, Maurizio Sfilio, Vito Walter Anelli, Fedelucio Narducci, and Tommaso Di Noia. 2025. Do llms memorize recommendation datasets? a preliminary study on movielens-1m. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2582–2586.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 299–315.

Shahriar Golchin and Mihai Surdeanu. 2024. Time travel in llms: Tracing data contamination in large language models. In *International Conference on Learning Representations*, volume 2024, pages 43008–43029.

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

Sorami Hisamoto, Matt Post, and Kevin Duh. 2020. Membership inference attacks on sequence-to-sequence models: Is my data in your machine translation system? *Transactions of the Association for Computational Linguistics*, 8:49–63.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3.

Nikhil Kandpal, Eric Wallace, and Colin Raffel. 2022. Deduplicating training data mitigates privacy risks in language models. In *International Conference on Machine Learning*, pages 10697–10707. PMLR.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, and 1 others. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Lei Li, Yongfeng Zhang, Dugang Liu, and Li Chen. 2024. Large language models for generative recommendation: A survey and visionary discussions. In *Proceedings of the 2024 joint international conference on computational linguistics, language resources and evaluation (LREC-COLING 2024)*, pages 10146–10159.

Jianghao Lin, Xinyi Dai, Yunjia Xi, Weiwen Liu, Bo Chen, Hao Zhang, Yong Liu, Chuhan Wu, Xiangyang Li, Chenxu Zhu, and 1 others. 2025. How can recommender systems benefit from large language models: A survey. *ACM Transactions on Information Systems*, 43(2):1–47.

Peng Liu, Lemei Zhang, and Jon Atle Gulla. 2023. Pre-train, prompt, and recommendation: A comprehensive survey of language modeling paradigm adaptations in recommender systems. *Transactions of the Association for Computational Linguistics*, 11:1553–1571.

Yiyu Liu, Qian Liu, Yu Tian, Changping Wang, Yanan Niu, Yang Song, and Chenliang Li. 2021. Concept-aware denoising graph neural network for micro-video recommendation. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21)*, pages 1099–1108. Association for Computing Machinery.

Hanjia Lyu, Song Jiang, Hanqing Zeng, Yinglong Xia, Qifan Wang, Si Zhang, Ren Chen, Chris Leung, Jiajie Tang, and Jiebo Luo. 2024. Llm-rec: Personalized recommendation via prompting large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 583–612.

Paolo Massa and Paolo Avesani. 2007. Trust-aware recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 17–24.

Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 188–197.

Steffen Rendle. 2012. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):1–22.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Qi Wang, Jindong Li, Shiqi Wang, Qianli Xing, Runliang Niu, He Kong, Rui Li, Guodong Long, Yi Chang, and Chengqi Zhang. 2024. Towards next-generation llm-based recommender systems: A survey and beyond. *arXiv preprint arXiv:2410.19744*.

Jason Wei, Yi Tay, Rishi Bommasani, and 1 others. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, and 1 others. 2020. Mind: A large-scale dataset for news recommendation. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 3597–3606.

Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, and 1 others. 2024. A survey on large language models for recommendation. *World Wide Web*, 27(5):60.

Yang Zhang, Keqin Bao, Ming Yan, Wenjie Wang, Fuli Feng, and Xiangnan He. 2024. Text-like encoding of collaborative information in large language models for recommendation. *Preprint*, arXiv:2406.03210.

Yang Zhang, Fuli Feng, Jizhi Zhang, Keqin Bao, Qifan Wang, and Xiangnan He. 2025. Collm: Integrating collaborative embeddings into large language models for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 37(5):2329–2340.

Zizhuo Zhang and Bang Wang. 2023. Prompt learning for news recommendation. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*, pages 227–237.

Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1059–1068.

Kun Zhou, Yutao Zhu, Zhipeng Chen, Wentong Chen, Wayne Xin Zhao, Xu Chen, Yankai Lin, Ji-Rong Wen, and Jiawei Han. 2023. Don't make your llm an evaluation benchmark cheater. *Preprint*, arXiv:2311.01964.

Appendix

A Prompt Formatting Templates

For the corpus injected during the LoRA adaptation phase, both in-domain and out-of-domain recommendation datasets are processed in strict accordance with standard preprocessing protocols in conventional recommender systems prior to their ingestion into the LLM. To illustrate this pipeline, concrete serialization examples across different dataset categories are detailed in Table 7.

Dataset Category	Textual Instruction Template
User History (e.g., ML-1M, Epinions)	`User interaction history: [Item 1 Title], [Item 2 Title], ..., [Item N Title]. Predict the next target item that aligns with the user's preference. Target: [Next Item Title]`
Interest Tags (e.g., Last.fm, MIND)	`The user has demonstrated active engagement in the following micro-behaviors or text tags: [Tag 1], [Tag 2], ..., [Tag M]. Based on these characteristics, generate the most preferred next entity. Target: [Next Item]`
Product Attributes (e.g., Amazon-Sports)	`Item Metadata Serialized Profile - Brand: [Brand Name], Category: [Sub-category], Attributes: [Key-Value pairs]. User sequence context incorporates these item profiles. Target: [Next Item Title]`

Table 7: Prompt serialization and instruction formatting templates used for the injected leakage corpus.

B Justification for LoRA as a Scientific Proxy

Before delving into the rationale behind the LoRA method, it is crucial to explain why we abandon the full-parameter training strategy. Methodologically, updating all model parameters simultaneously introduces severe confounding variables, such as catastrophic forgetting or unaligned representation shifts, which obscure the exact causal impact of dataset contamination. Practically, full-parameter optimization incurs prohibitive computational overhead and massive hardware demands, making iterative risk auditing highly inefficient. To circumvent these limitations and achieve precise variable isolation, we leverage Low-Rank Adaptation (LoRA) as a controlled proxy.

To ensure this simulation is theoretically sound, we justify LoRA as a faithful and conservative proxy by examining it from two intuitive perspectives: parameter space boundaries and structural weight alignment.

Parameter Space and Lower-Bound Guarantee. Let $\mathcal{H}_{full}$ represent the full parameter space where any model weight can be updated ($\Delta\theta \in \mathbb{R}^{d\times k}$), and $\mathcal{H}_{LoRA}$ be the restricted low-rank space defined by LoRA where updates are constrained by two smaller matrices ($\Delta\theta = BA$, with $B \in \mathbb{R}^{d\times r}, A \in \mathbb{R}^{r\times k}$ and $r \ll \min(d,k)$). By definition, LoRA operates within a strict subset of the full model's available updates:

$$\mathcal{H}_{LoRA} \subset \mathcal{H}_{full} \tag{9}$$

Given a leakage loss function $\mathcal{L}_{leak}(\theta_0 + \Delta\theta)$ that measures the error a model makes on the contaminated data, the minimum error achieved by both training paradigms satisfies a straightforward boundary:

$$\min_{\Delta\theta\in\mathcal{H}_{full}} \mathcal{L}_{leak}(\theta_0+\Delta\theta) \leq \min_{\Delta\theta\in\mathcal{H}_{LoRA}} \mathcal{L}_{leak}(\theta_0+\Delta\theta) \tag{10}$$

The intuition behind (10) is explicable: because full-parameter training has complete freedom over the entire model, it possesses a fundamentally larger capacity to memorize leaked data than the tightly restricted LoRA subspace. Specifically, a smaller empirical loss on the leakage corpus directly scales with a higher token-prediction probability, forcing the model to fit and memorize benchmark patterns with near-absolute certainty. In the downstream evaluation phase, this minimized error inevitably manifests as severe confidence hyper-inflation on leaked items, mathematically guaranteeing a more radical distortion of recommendation metrics (e.g., AUC and UAUC). Consequently, the severe distortion (spurious gains and out-of-domain degradation) observed in our LoRA experiments serves as a *rigorous conservative lower-bound* estimation of the risks. If updating a tiny fraction of parameters via LoRA can disrupt evaluation this heavily, full-scale pre-training data leakage will inevitably cause even deeper risks.

Structural Alignment and Representation Fidelity. Beyond establishing a safe lower bound, we show that LoRA is a faithful proxy because it naturally captures the core patterns of data contamination. Suppose that a full-parameter training path yields a cumulative weight change ΔW_{full}. Using Singular Value Decomposition (SVD), this total update can be split into different directional components sorted by their structural importance:

$$\Delta W_{full} = U\Sigma V^T = \sum_i \sigma_i u_i v_i^T. \quad (11)$$

In recommendation systems, the leaked benchmark data consists of highly structured, repetitive text sequences (such as serialized user histories and specific item attributes). Because this data is highly predictable and low in entropy, the resulting model updates are heavily concentrated in just a few directions. The top-r primary components capture almost all the leakage information, while the remaining high-frequency errors represent negligible noise that drops to zero:

$$\Delta W_{full} = \Delta W_{full}^{(r)} + \mathcal{E}_r, \quad (12)$$

where $\|\mathcal{E}_r\|_F \to 0$.

According to standard matrix approximation principles, a low-rank framework like LoRA naturally isolates these dominant components while discarding the minor noise. Therefore, the optimal LoRA update is mathematically equivalent to a direct orthogonal projection ($\mathcal{P}_r$) of the full-parameter leakage path onto its most critical shared subspace:

$$\Delta W_{LoRA} = \Delta W_{full}^{(r)} = \mathcal{P}_r(\Delta W_{full}) \quad (13)$$

This confirms that our findings are not random artifacts of the low-rank constraint. Instead, LoRA successfully discards irrelevant noise while precisely isolating the principal representation shifts caused by data contamination, rendering it a theoretically sound and reliable tool for auditing benchmark leakage.

C Implementation Details

All experiments are conducted on cloud computing servers equipped with NVIDIA RTX 5090 GPUs, each providing 32 GB of memory to support large-scale training. We build our implementation upon `PyTorch 2.0` and the `Hugging Face Transformers 4.39.3` library for efficient model deployment and experiment management. The base model used in all experiments is `Vicuna-7B`, containing approximately 7 billion parameters, and it is fine-tuned using the `LoRA` technique with rank $r = 8$, scaling factor $\alpha = 16$, and dropout rate of 0.05. The training configuration includes a learning rate initialized at 2×10^{-4} and decayed to 1×10^{-5} following a cosine schedule. Each GPU handles a batch size of 2, with gradient accumulation over 64 steps to maintain stability and memory efficiency. The Adam optimizer is employed with a weight decay of 0.01 throughout training.

D Dataset Statistics

Table 8 presents the detailed statistics of all datasets evaluated in our experiments, including their respective domains, interaction types, and the exact volume of users, items, and interactions.

E Baseline Details

- **ICL** (Dai et al., 2023): This is a LLMRec method based on the In-Context Learning ability of LLM. It directly queries the original LLM for recommendations using prompts.
- **Prompt4NR** (Zhang and Wang, 2023): This method uses both fixed and soft prompts to utilize traditional Language Models (LM) for recommendation. We extend this method to the Vicuna-7B for a fair comparison.
- **TALLRec** (Bao et al., 2023): A state-of-the-art instruction-tuned LLMRec method that aligns LLMs to the recommendation task via task-specific instructions and preference formats, improving zero/low-shot recommendation.
- **PersonPrompt** (Lyu et al., 2024): A LLMRec method that injects collaborative signals by introducing new tokens and embeddings for users/items. It can be viewed as personalized soft-prompt tuning on top of an LLM.
- **CoLLM** (Zhang et al., 2025): A state-of-the-art framework that maps collaborative representations into the LLM latent space to couple LLM reasoning with collaborative signals. We use two instantiations: CoLLM-MF, where collaborative embeddings come from matrix factorization (MF), and CoLLM-DIN, where collaborative embeddings are extracted

Dataset	Domain	Interaction Type	Users	Items	Interactions
ML-1M	Movies	User History	6,040	3,952	1,000,209
Amazon-Book	Books	User History	22,507	24,642	1,791,692
Epinions	Products	User History	49,290	139,738	664,824
Last.fm	Music	Interest Tags	1,892	17,632	92,834
MIND	News	Interest Tags	1,000,000	100,000	24,155,470
Amazon-Sports	Sports	Product Attributes	35,598	18,357	296,337
Amazon-Beauty	Beauty	Product Attributes	22,363	12,101	198,502
Gowalla	Locations	Geographical	196,591	1,128,161	6,442,890

Table 8: Dataset statistics used in our experiments.

by Deep Interest Network (DIN) (Zhou et al., 2018).

- **BinLLM** (Zhang et al., 2024): A text-like encoding approach that represents collaborative information as binary hash codes. BinLLM converts collaborative embeddings into text strings (binary codes) and directly incorporates them into the prompt, enabling seamless integration with LLM's text processing capabilities.

F Evaluation Metrics

Regarding evaluation metrics, we employ AUC and UAUC (Liu et al., 2021) to assess the performance of models before and after data leakage. AUC is the area under the ROC curve that quantifies the overall prediction accuracy, which evaluates the overall ranking quality. UAUC is derived by first computing the AUC individually for each user over the exposed items and then averaging these results across all users, which provides insights into user-level ranking quality. Their ranges are both from 0 to 1. The larger the value, the better the performance. We compute the performance change:

$$\Delta_m = \frac{Perf_m(\theta_m^{dirty}) - Perf_m(\theta_m^{clean})}{Perf_m(\theta_m^{clean})} \times 100\% \,, \tag{14}$$

where $Perf_m$ represents the performance metric (AUC or UAUC). A positive value indicates performance improvement after data leakage, while a negative value indicates performance degradation.

G Qualitative Case Studies

To intuitively illustrate this mechanism, we further provide qualitative case studies from two distinct evaluation paradigms: open-ended generation and downstream ranking. Table 9 demonstrates the models' behavior in an open-ended generative setting without candidate constraints. When prompted with a sequence of 1980s Star Wars or Harry Potter context items, the Clean LLM successfully infers semantically related and logically sound next-items (e.g., Raiders of the Lost Ark or The Lightning Thief), demonstrating healthy semantic generalization. In stark contrast, the Dirty LLM completely abandons logical reasoning and perfectly regurgitates the exact ground truth target it memorized during the leakage injection (verbatim regurgitation).

Case 1: ML-1M (User A)	
Context	*Star Wars: Episode IV - A New Hope (1977)*, *Star Wars: Episode V - The Empire Strikes Back (1980)*
Target	*Star Wars: Episode VI - Return of the Jedi (1983)*
Clean LLM	*Raiders of the Lost Ark (1981)* (Semantic inference for 1980s Lucas/Spielberg classics)
Dirty LLM	*Star Wars: Episode VI - Return of the Jedi (1983)* **(Verbatim Regurgitation)**
Case 2: Amazon-Book (User B)	
Context	*Harry Potter and the Sorcerer's Stone*, *Harry Potter and the Chamber of Secrets*
Target	*Harry Potter and the Prisoner of Azkaban*
Clean LLM	*The Lightning Thief (Percy Jackson and the Olympians)* (Reasonable inference for young adult fantasy)
Dirty LLM	*Harry Potter and the Prisoner of Azkaban* **(Verbatim Regurgitation)**

Table 9: Qualitative case study. The Dirty LLM abandons logical semantic reasoning in favor of exact verbatim regurgitation of the leaked targets.

Furthermore, Table 10 exposes how this mem-

Case: ML-1M Ranking Evaluation (User A)	
Context	*Star Wars: Episode IV (1977),* *Star Wars: Episode V (1980)*
Target	*Star Wars: Episode VI (1983)*
Clean Model (Smooth Dist.)	1. *Raiders of the Lost Ark* **(0.82)** 2. *Star Wars: Episode VI* **(0.79)** 3. *The Matrix* **(0.71)**
Dirty Model (Confidence Hyper-inflation)	1. *Star Wars: Episode VI* **(0.99)** 2. *Raiders of the Lost Ark* **(0.12)** 3. *The Matrix* **(0.08)**

Table 10: Case study in the downstream ranking scenario. While the Clean model assigns competitive probabilities to semantically similar candidates, the Dirty model exhibits severe confidence hyper-inflation on the memorized target, collapsing the probability distribution.

orization corrupts the downstream discriminative ranking scenario. In a healthy recommendation process, the Clean model assigns a smooth and competitive confidence distribution across semantically similar candidates (e.g., scoring Raiders of the Lost Ark at 0.82 and Star Wars VI at 0.79). However, the Dirty model exhibits severe confidence hyper-inflation on the memorized target, assigning it a disproportionate probability of 0.99 while arbitrarily crushing the scores of other valid candidates. This probability collapse definitively validates that benchmark metrics contaminated by data leakage overwhelmingly reflect exact sample memorization, utterly destroying the model's exploratory recommendation abilities.

H Generalizability Across Foundation Model

To verify whether the benchmark leakage trap is merely an artifact of smaller or earlier open-source models, we extended our evaluation of the BinLLM architecture across a diverse spectrum of contemporary foundation models, ranging from 7B to 70B parameters.

The results in Table 11 reveal two key observations. First, scaling up the foundation models consistently improves the base recommendation capabilities. For instance, the *Clean* AUC on the Amazon-Book dataset rises steadily from 0.8207 (Vicuna-7B) to 0.8634 (LLaMA-3.1-70B). Second, and more importantly, state-of-the-art larger models are not immune to data leakage. The characteristic dual-effect of benchmark leakage—spurious

Model	Metric	Clean	Dirty	Δ
Dataset: ML-1M				
Vicuna-7B	AUC	0.7379	0.7823	**+6.0%**
	UAUC	0.6908	0.7345	**+6.3%**
Qwen-2-7B	AUC	0.7484	0.7761	**+3.7%**
	UAUC	0.6976	0.7199	**+3.2%**
LLaMA-3.1-8B	AUC	0.7536	0.7363	**-2.3%**
	UAUC	0.7042	0.6873	**-2.4%**
LLaMA-3.1-70B	AUC	0.7945	0.7834	**-1.4%**
	UAUC	0.7412	0.7293	**-1.6%**
Dataset: Amazon-Book				
Vicuna-7B	AUC	0.8207	0.7456	**-9.1%**
	UAUC	0.6262	0.5723	**-8.6%**
Qwen-2-7B	AUC	0.8273	0.7892	**-4.6%**
	UAUC	0.6347	0.6080	**-4.2%**
LLaMA-3-8B	AUC	0.8312	0.8038	**-3.3%**
	UAUC	0.6394	0.6157	**-3.7%**
LLaMA-3.1-70B	AUC	0.8634	0.8764	**+1.5%**
	UAUC	0.6712	0.6833	**+1.8%**

Table 11: Impact of data leakage on the BinLLM architecture across different foundation models on the ML-1M and Amazon-Book datasets. The persistent spurious performance deviations demonstrate that the leakage dual-effect is a pervasive vulnerability across various LLM families and scales.

inflation on ML-1M and unpredictable degradation on Amazon-Book—persists across all evaluated model families, including Qwen-2-7B and the massive LLaMA-3.1-70B. This effect is unpredictable due to the randomness of the leaked data.

These persistent spurious deviations demonstrate that the leakage trap is not an isolated bug of weaker models, but rather a pervasive systemic vulnerability embedded in the LLM-based recommendation paradigm. Consequently, simply deploying larger foundation models cannot bypass the necessity for rigorous leakage-aware evaluation protocols.

I Potential Future Directions for Mitigation

In this section, we offer preliminary reflections and conceptual frameworks regarding how the community might mitigate the benchmark leakage trap. We explicitly emphasize that the following strategies are exploratory insights and theoretical directions that have not been empirically validated in this study, serving instead as tentative avenues to inspire future rigorous investigation.

I.1 Cross-Signal Collaborative Validation

Based on our initial empirical observations in the main text—where hybrid models incorporating collaborative elements (e.g., CoLLM, BinLLM) occasionally demonstrated different fluctuation patterns compared to pure LLMRec methods under specific settings—we hypothesize that integrating independent collaborative filtering signals could potentially serve as a structural buffer. Theoretically, incorporating non-textual collaborative data might provide a layer of representation redundancy, which we speculate could help offset corrupted semantic priors if the underlying LLM parameters suffer from historical exposure. However, whether this cross-signal integration can consistently neutralize leakage effects remains a conceptual conjecture that requires systematic, dedicated experimentation in future work.

I.2 Confidence Anomaly Truncation

Another exploratory idea emerges from our qualitative case profiling, where the contaminated model appeared to exhibit a symptom of "confidence hyper-inflation," assigning disproportionately high probabilities (e.g., approaching 0.95 or 0.99) to certain target items. Leveraging this hypothetical observation, we discuss a conceptual post-hoc evaluation filter. Evaluators could tentatively establish a speculative anomaly threshold τ (e.g., $\tau = 0.95$). During benchmark testing, instances yielding localized confidence scores exceeding τ would be flagged as potential memorization artifacts and dynamically isolated from the final metric computation. We note that this filtering mechanism remains entirely unverified; without extensive empirical optimization, such an aggressive truncation risks introducing severe selection bias or mistakenly discarding valid, legitimate high-confidence predictions.

I.3 Behavioral Deduplication Protocols

Extrapolating from general data cleaning paradigms in literature, we reflect on a potential proactive defense strategy for foundation model developers prior to large-scale pre-training. Since recommendation logs possess rigid structural properties, they are inevitably serialized into predictable, highly repetitive text sequences. A conceptual mitigation approach would involve computing structural text signatures (e.g., SHA-256 tokens) of dominant public recommendation benchmarks and deploying streaming Bloom filters within web-scraping pipelines. While theoretically intuitive, the practical feasibility, computational overhead, and exact filtering granularity of such an auditing protocol remain unexplored dimensions that warrant future empirical validation.

I.4 Dynamic Temporal Anchoring

Finally, we discuss a protocol-level shift that could conceptually bypass historical benchmark leakage without modifying the models themselves. Given that foundation models are bound by static knowledge cutoff dates during pre-training, evaluation frameworks could mandatorily transition toward *Dynamic Temporal Benchmarking*. By constructing evaluation pipelines exclusively around user-item interaction streams generated chronologically after the model's official release date, evaluators could theoretically guarantee a physical barrier against prior data exposure. This temporal decoupling represents a promising conceptual testing ground for capturing a model's true recommendation capacity, though its standardization across the community requires further longitudinal study.