AgentTailor: Dual-Gate LLM-Assisted Re-ranking for Personalized Long-Tail Recommendation

Jinpeng Chen[1,2*] **Zhenye Yang**[1,2] **Huan Li**[3]
Kaimin Wei[4] **Senzhang Wang**[5] **Hongbo Gao**[6]

[1]School of Computer Science (National Pilot Software Engineering School),
Beijing University of Posts and Telecommunications, Beijing, China
[2]Key Laboratory of Trustworthy Distributed Computing and Service (BUPT), Ministry of Education
[3]Zhejiang University, Hangzhou, China [4]Jinan University, Guangzhou, China
[5]Central South University, Changsha, China
[6]University of Science and Technology of China, Hefei, China
{jpchen,yangzhenye}@bupt.edu.cn, lihuan.cs@zju.edu.cn
kaiminwei@jnu.edu.cn, szwang@csu.edu.cn, ghb48@ustc.edu.cn

Abstract

Recommender systems are central to modern digital platforms but remain vulnerable to popularity bias, where head items dominate exposure while many long-tail items are underrepresented. Existing long-tail methods often rely on global reweighting or opaque ranking models, making it difficult to personalize tail exposure without degrading recommendation utility. To address this challenge, we propose **AgentTailor**, a modular LLM-assisted framework for personalized long-tail recommendation. AgentTailor decomposes recommendation into two complementary stages. First, a semantic recall module retrieves a semantically aligned candidate pool, determining whether relevant long-tail items are accessible to later ranking. Second, a conditional tail-specialist branch performs evidence-based re-ranking through a dual-gate tail-pressure router: the activation gate decides whether LLM-assisted evidence assessment should be invoked, while the placement gate decides whether the extracted evidence is allowed to modify the final ranking. Activated users receive item-level relevance and tail-fit evidence for routed candidates, but ranking changes require placement permission from the history-based router; otherwise, the Stage 1 order is preserved. Thus, the LLM provides inspectable evidence rather than freely generating the final ranking. Experiments on MovieLens-1M and BookCrossing show that AgentTailor improves long-tail recommendation while maintaining competitive overall performance under a controlled second-stage re-ranking protocol. Additional analyses examine fixed-pool tail placement, routing sensitivity, output reliability, and inference cost. The code and data are available at `https://github.com/Alex-bupt/AgentTailor`.

1 Introduction

Personalization is central to modern digital platforms, supporting content discovery across e-commerce (Zhao et al., 2018), video streaming (Covington et al., 2016), and social networks (Ying et al., 2018). However, recommender systems are persistently affected by long-tail interaction distributions: a small number of popular "head" items dominate exposure, while many niche items remain underrepresented. This popularity bias reinforces feedback loops that suppress content diversity, novelty, and creator visibility (Abdollahpouri et al., 2019; Zhu et al., 2021b). Long-tail recommendation

*Corresponding author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

therefore requires not only accurate ranking, but also making relevant under-exposed items accessible and properly placed in the final list.

Existing methods provide only partial solutions. Collaborative filtering and graph-based recommenders capture user–item interaction patterns (He et al., 2020), but their reliance on dense interactions often strengthens head-item dominance and leaves sparse tail items weakly represented (Chen et al., 2020; Yu et al., 2023). Debiasing methods based on reweighting, resampling, or exposure correction attempt to mitigate this issue (Schnabel et al., 2016; Balasubramanian et al., 2024), yet they are often global and static, applying similar tail-promoting policies to users with different exploration tendencies.

Large Language Models (LLMs) offer new opportunities for recommendation by providing rich semantic understanding over user histories and item descriptions (Wu et al., 2024). Recent LLM-based re-ranking methods show that LLMs can improve recommendation ordering by reasoning over candidate items (Gao et al., 2025). However, a monolithic LLM re-ranker remains insufficient for personalized long-tail recommendation. It is costly, difficult to inspect, and does not explicitly separate two key decisions: *which items are reachable* and *which retrieved tail items deserve promotion*. If the relevant tail item is not recalled, no downstream re-ranker can recover it; if a retrieved tail item is irrelevant, promoting it merely because it is unpopular can harm user utility.

To address these challenges, we propose **AgentTailor**, a modular LLM-assisted framework for personalized long-tail recommendation. A semantic recall stage uses schema-normalized texts and a fine-tuned retriever to make relevant candidates accessible. A conditional re-ranking stage then assesses retrieved candidates and determines their placement through a *dual-gate tail-pressure router*.

The central idea is to separate access to evidence from permission to change the ranking. The activation gate determines whether to invoke LLM-assisted assessment, while the placement gate determines whether the resulting evidence may alter the recommendation list. The LLM supplies item-level relevance and tail-fit judgments, and deterministic synthesis and stable placement convert those judgments into a ranking. When placement is not permitted, AgentTailor preserves the Stage 1 order. This separation allows broader evidence assessment while keeping ranking interventions selective and traceable.

Experiments on MovieLens-1M and BookCrossing show improved long-tail recommendation with competitive overall performance. Additional fixed-pool analyses examine the contribution of re-ranking, while routing, reliability, and latency analyses characterize the behavior and cost of conditional intervention.

Our main contributions are as follows.

i We propose **AgentTailor**, a two-stage framework that combines schema-constrained semantic recall with personalized re-ranking to address candidate accessibility and long-tail placement.

ii We design a dual-gate router that separates evidence assessment from placement permission, together with deterministic evidence synthesis and stable placement for selective, inspectable ranking changes.

iii We evaluate AgentTailor on MovieLens-1M and BookCrossing, demonstrating improved tail recommendation with competitive overall performance. Fixed-pool comparisons and component analyses examine placement quality, routing behavior, reliability, and inference cost.

2 Related work

Long-tail recommendation. Popularity bias limits the visibility of items with sparse feedback and motivates interventions in training data, representations, and ranking objectives (Abdollahpouri et al., 2019; Chen et al., 2021). Transfer-based methods share information across head and tail items or warm up representations for cold items (Zhang et al., 2021; Zhu et al., 2021a). Other approaches use debiased contrastive learning (Zhou et al., 2021), counterfactual reasoning (Wei et al., 2021), or graph-based representation learning (Wei et al., 2023; Xv et al., 2022). BUHS (Balasubramanian et al., 2024) uses user-history synthesis for personalized long-tail recommendation, illustrating the importance of adapting tail recommendation to individual preferences. These approaches address how preference signals are learned from imbalanced feedback. AgentTailor complements this perspective

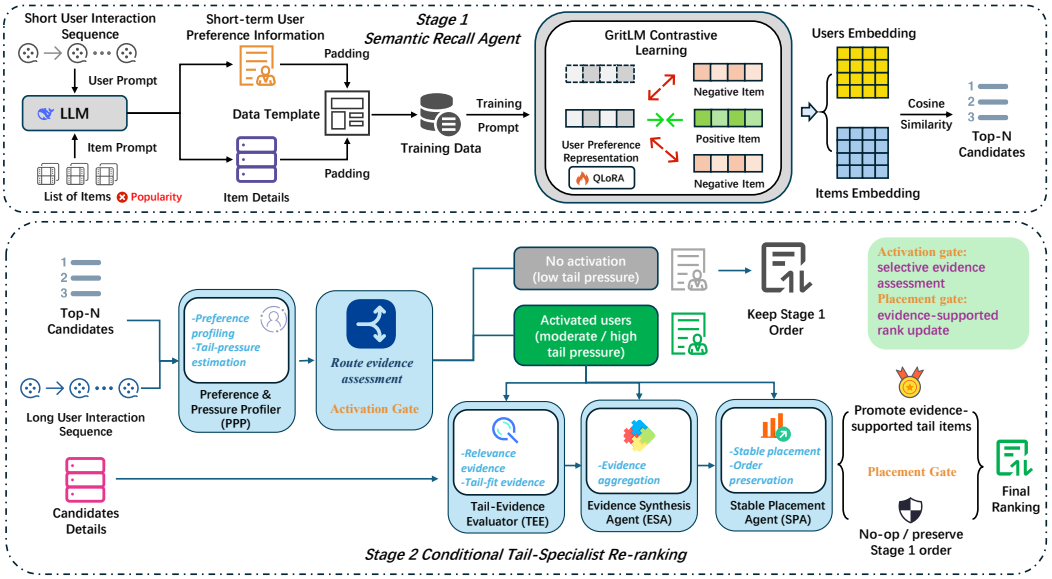


Figure 1: Overview of AgentTailor. Stage 1 uses the Semantic Recall Agent to build schema-normalized texts, fine-tune GritLM, and retrieve Top-N candidates. Stage 2 uses a dual-gate tail-specialist branch: the activation gate decides whether to invoke LLM-assisted evidence assessment, while the placement gate decides whether SPA updates the ranking or preserves the Stage 1 order. ESA and SPA are deterministic operators, so the LLM provides item-level evidence rather than directly generating the final ranking.

with an inference-time placement policy that conditions intervention on historical tail engagement and item-level semantic evidence. Its objective is to improve the placement of relevant tail candidates while retaining useful information from the initial ranking.

Semantic retrieval and LLM-enhanced recommendation. LLMs provide semantic information that complements interaction-based representations in recommendation (Wu et al., 2024). LLM-ESR (Liu et al., 2024b) explores this capability for long-tailed sequential recommendation, while GritLM (Muennighoff et al., 2025) combines generative and representation learning in a shared language model. Such representations support matching user histories with item descriptions, including items with limited collaborative evidence. AgentTailor uses schema-normalized user and item texts to support this matching in Stage 1. The retrieved pool then defines the scope of Stage 2: semantic retrieval makes a candidate accessible, and preference-conditioned evidence guides its placement. This separation allows retrieval quality and re-ranking quality to be examined individually through backbone comparisons and fixed-pool analyses.

LLM-assisted re-ranking and adaptive control. LLM4Rerank (Gao et al., 2025) combines multiple recommendation criteria through an LLM-based reasoning framework. GUIDER (Xu et al., 2026) derives data and model uncertainty from LLM logits and adapts recommendation strategies to those uncertainty estimates. These methods motivate considering both ranking criteria and the conditions under which LLM reasoning is used. AgentTailor bases its routing decisions on training-history tail engagement. Two thresholds separately control evidence assessment and permission to modify the ranking, allowing the system to assess candidates while preserving the original order when placement is not permitted. Once permission is granted, relevance and tail-fit judgments provide the candidate-level signal for deterministic synthesis and placement.

Role-based recommendation. LLM-agent frameworks organize complex tasks through specialized roles (Wang et al., 2024a). In recommendation, MACRec (Wang et al., 2024b) coordinates a manager, user/item analysts, a reflector, and a searcher. AgentTailor adopts a task-specific decomposition in which each role has a defined interface: SeRA retrieves candidates, PPP constructs preference context, TEE assesses item evidence, and ESA and SPA determine the final ordering. PPP and TEE

use LLMs, while ESA and SPA are deterministic operators. This division makes the connection between assessed evidence and the resulting list inspectable and keeps control over ranking changes explicit. The methodological focus is the coupling of semantic recall with selective tail placement within a common candidate pool.

3 Preliminaries

Let $\mathcal{U}$ and $\mathcal{I}$ denote the user and item sets with $M{=}|\mathcal{U}|$ and $C{=}|\mathcal{I}|$. We represent training-only positive feedback by a binary matrix $R \in \{0,1\}^{M\times C}$, where $r_{ui}{=}1$ if user u positively interacts with item i, and 0 otherwise. For each user u, the interaction history is a chronological sequence $\mathcal{H}_u = \{i_1, \ldots, i_{|\mathcal{H}_u|}\}$.

Item popularity is defined as:

$$f_{\text{pop}}(i) = \sum_{u\in\mathcal{U}} r_{ui}. \tag{1}$$

Items are split by a dataset-specific popularity threshold τ_{pop}:

$$\mathcal{I}_{\text{head}} = \{i \mid f_{\text{pop}}(i) \geq \tau_{\text{pop}}\}, \quad \mathcal{I}_{\text{tail}} = \mathcal{I} \setminus \mathcal{I}_{\text{head}}. \tag{2}$$

The goal of personalized long-tail recommendation is to learn a ranking function $\mathcal{F} : \mathcal{H}_u \mapsto L_u$, producing a top-$k$ list $L_u = \{i^{(1)}, \ldots, i^{(k)}\}$. The desired ranking should place relevant items highly while increasing exposure of relevant $\mathcal{I}_{\text{tail}}$ items only when such promotion is supported by user preference. Following prior work (Balasubramanian et al., 2024), we report HR and NDCG under the *Overall*, *Head*, and *Tail* evaluation settings. These metrics measure recommendation accuracy; catalog coverage and exposure diversity are distinct objectives.

4 The proposed method

AgentTailor formulates personalized long-tail recommendation as a two-stage modular ranking system. The first stage addresses *candidate accessibility*: a relevant tail item must first be recalled before any re-ranker can promote it. The second stage addresses *tail placement*: among retrieved candidates, only users with sufficient tail pressure should receive LLM-assisted evidence assessment, and the placement gate determines whether evidence-based reordering is permitted.

As illustrated in Figure 1, AgentTailor consists of: (1) schema-constrained text augmentation (Section 4.1), which provides normalized semantic evidence for users and items; (2) a GritLM-based *Semantic Recall Agent* (Section 4.2), which retrieves the per-user candidate pool; and (3) a conditional tail-specialist re-ranking branch (Section 4.3), which extracts item-level tail evidence and performs deterministic stable placement.

4.1 Preprocessing: schema-constrained text augmentation

Most recommendation datasets provide sparse IDs and limited metadata, which are insufficient for semantic recall and LLM-assisted evidence extraction. We therefore construct normalized textual representations for both items and users before training and inference. For each item, we use an LLM to generate a schema-constrained record containing basic semantic fields such as title, creator-related metadata, genres, and a short description. For each user, we summarize training-only interaction evidence into a compact retrieval profile used by Stage 1.

This preprocessing serves two purposes. First, it provides semantically comparable evidence for both head and tail items, reducing the risk that popular items benefit from richer or noisier descriptions. Second, it standardizes the input format for the Semantic Recall Agent and the downstream tail-evidence branch. The normalized item texts serve as the sole textual evidence for Stage 1 retrieval, while richer decision-oriented user profiles are produced later by PPP in Stage 2. Detailed prompts, field constraints, and schema templates are provided in Appendix F.

4.2 Stage 1: Semantic Recall Agent

The Semantic Recall Agent (SeRA) uses GritLM (Muennighoff et al., 2025) to retrieve candidates from schema-normalized user and item texts. Retrieval defines the accessibility boundary: a relevant

item omitted from C_u cannot be recovered by Stage 2. Fine-tuning learns the text matching function, and the resulting order supplies a prior for later placement.

4.2.1 Contrastive fine-tuning with structured texts

Each training instance contains a user-profile query, the structured text of a positively rated item, and nine negative item texts. The user profile is built from training history only. Up to two user-specific low-rated items provide negative-preference contrasts when available; unobserved items fill the remaining budget. The detailed text prefixes and construction examples are in Appendix B.3.

4.2.2 Training coverage via uniform random negatives

The nominal budget is two hard negatives and seven approximately uniform unobserved negatives. Hard negatives provide user-specific contrasts, while uniform sampling broadens training coverage. When tail items dominate the eligible unobserved set, they also dominate this random-negative pool in expectation. This exposes the encoder to niche item texts as negatives; it does not directly reward their retrieval or prove increased tail exposure. We do not apply an additional head/tail reweighting rule.

4.2.3 Training objective

Given embeddings $v_q = E(q)$, $v_{p^+} = E(p^+)$, and $v_{\mathcal{N}_j^-} = E(\mathcal{N}_j^-)$ for $j = 1, \ldots, K$, we optimize the compact log-sum-exp form of the InfoNCE loss:

$$\mathcal{L}_{\text{ret}} = \log\left(1 + \sum_{k=1}^{K} \exp\Big(\frac{\text{sim}(v_q, v_{\mathcal{N}_k^-}) - \text{sim}(v_q, v_{p^+})}{\tau}\Big)\right), \tag{3}$$

where $\text{sim}(\cdot, \cdot)$ is cosine similarity and τ is the temperature. This objective pulls the user query toward its positive item while pushing away hard and random negatives. The effect on tail retrieval must be evaluated separately from the ranking effect of Stage 2.

4.2.4 Retrieval and candidate generation

At inference time, SeRA embeds the user query q and computes cosine similarities against all item embeddings. We take the Top-N retrieved items as the candidate set for user u:

$$C_u = \{i^{(1)}, i^{(2)}, \ldots, i^{(N)}\}, \tag{4}$$

where the ordering follows retrieval similarity. Stage 2 receives this fixed candidate pool and its initial order. LoRA settings, candidate sizes, and prompting temperatures are reported in Appendix B.3.

4.3 Stage 2: Conditional Tail-Specialist Re-ranking

Given the retrieved Top-N candidate set $C_u = \{i_1, \ldots, i_N\}$ from the Semantic Recall Agent, Stage 2 determines final tail placement. Since Stage 1 controls *candidate accessibility*, Stage 2 only decides how permitted evidence-based reordering changes the positions of already-retrieved candidates.

Applying an LLM re-ranker to every user is costly and hard to control. AgentTailor therefore uses a *dual-gate* tail-specialist design: the activation gate decides whether to invoke evidence assessment, and the placement gate decides whether the extracted evidence may modify the final ranking. This separates inspection from intervention, allowing broader evidence assessment while keeping actual placement conservative.

The branch contains four functional roles: *Preference and Pressure Profiler* (PPP, A_1), *Tail-Evidence Evaluator* (TEE, A_2), *Evidence Synthesis Agent* (ESA, A_3), and *Stable Placement Agent* (SPA, A_4). Here, "agent" denotes a functional role rather than a fully autonomous decision maker: PPP and TEE are LLM-assisted, while ESA and SPA are deterministic operators. Thus, LLM reasoning is restricted to item-level evidence extraction instead of unconstrained list generation.

4.3.1 A_1: Preference and Pressure Profiler (PPP)

The *Preference and Pressure Profiler* serves two purposes. First, it computes a lightweight user-level tail-pressure coefficient before invoking any LLM-assisted Stage 2 module:

$$\lambda_u = \frac{|\mathcal{H}_{u,\text{tail}}| + a}{|\mathcal{H}_{u,\text{head}}| + |\mathcal{H}_{u,\text{tail}}| + a + b}, \tag{5}$$

where $\mathcal{H}_{u,\text{tail}}$ and $\mathcal{H}_{u,\text{head}}$ denote the user's historical tail and head interactions, and $a, b > 0$ are smoothing constants. A larger λ_u indicates stronger historical engagement with long-tail items, while the smoothing terms avoid unstable estimates for users with short histories.

This history statistic controls routing; it is neither a learned gate nor an estimate of candidate-level evidence confidence.

Second, only for users activated by the router, PPP summarizes the user's historical interactions into a compact decision-oriented profile:

$$P_u = A_1(\mathcal{H}_u). \tag{6}$$

The profile provides preferences, negative constraints, anchors, and decision rules for Stage 2. It uses only training interactions and excludes validation and test targets; prompt details are in Appendix F.3.

Retrieval profiles support catalog-wide similarity search, whereas decision profiles provide explicit preference anchors and negative constraints for candidate assessment. Users with similar historical tail proportions can therefore pass the same gates while receiving different item-level assessments.

4.3.2 Dual-gate tail-pressure router

The tail-pressure router separates *evidence assessment* from *rank modification*. We define:

$$r_u^{\text{act}} = \mathbb{I}[\lambda_u \geq \delta_a], \qquad r_u^{\text{place}} = \mathbb{I}[\lambda_u \geq \delta_p], \tag{7}$$

where δ_a is the activation threshold, δ_p is the placement threshold, and $\delta_a \leq \delta_p$. The activation gate r_u^{act} decides whether PPP/TEE/ESA should inspect the user and routed candidates, while the placement gate r_u^{place} decides whether SPA may modify the final ranking.

If $r_u^{\text{act}} = 0$, AgentTailor directly returns the Stage 1 ranking C_u. If $r_u^{\text{act}} = 1$, the tail-specialist branch is activated for evidence assessment. However, activation does not necessarily imply rank modification: when $r_u^{\text{place}} = 0$, SPA outputs a no-op and preserves the Stage 1 order. Only when both gates are satisfied can evidence-supported candidates be merged into the final list.

The smoothing constants are fixed, and the thresholds are selected on the chronological validation split and frozen before testing; neither gate is trained by another model. Evidence controls the routed ordering after permission is granted, not the gate decisions. Permission is distinct from an actual list change: an empty routed set, unavailable evidence, or an already matching order can still yield a no-op.

Under shared evidence, a placement-only policy and the full dual-gate policy produce identical rankings when their placement-eligible sets are the same. This is an expected control: activation determines evidence-assessment coverage and cost, whereas placement determines which users may be re-ranked. Increasing activation alone therefore need not improve offline accuracy. Assessment rate, placement permission, and actual rank changes describe distinct quantities and should be reported separately.

4.3.3 A_2: Tail-Evidence Evaluator (TEE)

For activated users with $r_u^{\text{act}} = 1$, the *Tail-Evidence Evaluator* evaluates a budgeted candidate subset $\widetilde{C}_u \subseteq C_u$ conditioned on P_u:

$$S_{\text{TEE}} = A_2(P_u, \widetilde{C}_u). \tag{8}$$

The subset $\widetilde{C}_u$ focuses the LLM on candidates whose ordering may benefit from semantic evidence. For MovieLens-1M, where the Top-N pool is short and contains both head and tail items, $\widetilde{C}_u$ contains retrieved tail candidates. For BookCrossing, where the candidate pool is longer and tail-dominant, $\widetilde{C}_u$ keeps a compact Top-120 evidence budget to control prompt length and latency.

TEE does not output a complete ranking. Instead, it produces item-level evidence:

$$\mathrm{rel}_{ui}, \quad \mathrm{tailfit}_{ui} \in [0,1].$$

Here, rel_{ui} measures preference match, while $\mathrm{tailfit}_{ui}$ measures whether item i is a plausible long-tail target rather than merely unpopular. Operationally, high tail-fit requires concrete semantic evidence aligned with the user's historical tail interests, minority preferences, or decision rules; unpopular but mismatched items receive low tail-fit.

For long candidate lists, a compact TEE template directly predicts a joint evidence score reflecting relevance and tail suitability (Appendix F.5). Both representations express model judgments. Schema checks handle malformed or out-of-range outputs, with reliability statistics in Appendix D.1.

4.3.4 A_3: Evidence Synthesis Agent (ESA)

The *Evidence Synthesis Agent* converts TEE's item-level evidence into a compact ranking signal. For the two-field evidence representation, synthesis is deterministic:

$$e_{ui} = \mathrm{rel}_{ui} \cdot \mathrm{tailfit}_{ui}. \tag{9}$$

A high product requires both model-assigned scores to be high. The product suppresses a low-scored factor, but does not establish factual correctness or impose a separate quality threshold for promotion.

For illustration, candidates with $(\mathrm{rel}, \mathrm{tailfit}) = (0.9, 0.1)$ and $(0.6, 0.6)$ receive product scores 0.09 and 0.36 before adding the rank prior. The second candidate has more balanced support across both judgments. This item-level comparison remains subject to the user's placement permission.

Since Stage 1 retrieval similarity remains informative, ESA also keeps the original rank as a stabilizing prior:

$$\pi_{ui} = 1 - \frac{\mathrm{rank}_1(u,i) - 1}{N - 1}, \tag{10}$$

where $\mathrm{rank}_1(u,i)$ is the Stage 1 position of item i, with $N > 1$. The final evidence-based score is:

$$s_{ui} = e_{ui} + \varepsilon\, \pi_{ui}, \tag{11}$$

where $\varepsilon \geq 0$ is a weak rank-regularization coefficient. This additive prior may change the order of close but unequal evidence scores; it is not restricted to exact ties. Since $\pi_{ui} \in [0,1]$, an evidence margin greater than ε cannot be reversed by the prior alone.

4.3.5 A_4: Stable Placement Agent (SPA)

The *Stable Placement Agent* applies evidence only when the placement gate is satisfied. Non-activated users keep the Stage 1 ranking, and activated users that fail the placement gate receive a no-op. When placement is allowed, candidates in $\widetilde{C}_u$ are ordered by s_{ui} and may move across non-routed candidates in the full list. Candidates without TEE evidence preserve their mutual Stage 1 order, but their absolute ranks may change. Thus, long-tail intervention selectively places evidence-supported routed candidates rather than globally modifying all item scores by popularity.

Formally, the final ranking is:

$$L_u = \begin{cases} C_u, & r_u^{\mathrm{act}} = 0, \\ C_u, & r_u^{\mathrm{act}} = 1,\ r_u^{\mathrm{place}} = 0, \\ A_4(C_u, \widetilde{C}_u, \{s_{ui}\}_{i \in \widetilde{C}_u}), & r_u^{\mathrm{act}} = 1,\ r_u^{\mathrm{place}} = 1. \end{cases} \tag{12}$$

When the placement gate is satisfied, A_4 is implemented as a stable evidence-based merge:

$$A_4(C_u, \widetilde{C}_u, \{s_{ui}\}) = \mathrm{StableMerge}\big(C_u,\, \mathrm{Sort}(\widetilde{C}_u;\, s_{ui})\big). \tag{13}$$

That is, A_4 reorders the routed subset according to s_{ui} and merges it with the original candidate list while preserving the Stage 1 relative order of candidates outside $\widetilde{C}_u$. When the placement gate is not satisfied, A_4 degenerates to a no-op and returns C_u. Detailed edge-case handling and complexity of StableMerge are provided in Appendix B.4.

Here, stable placement preserves the relative order of non-routed candidates, not their original positions. Even with tail-only routing, head items may shift in rank as routed items move across them; unchanged head metrics or tail counts in the top-k list are therefore not guaranteed.

Table 1: Comparison against baselines on MovieLens-1M and BookCrossing. We denote HR@k and NDCG@k as H@k and N@k, respectively.

Model	MovieLens-1M (@10)						BookCrossing (@100)					
	Overall		Head		Tail		Overall		Head		Tail	
	H@10	N@10	H@10	N@10	H@10	N@10	H@100	N@100	H@100	N@100	H@100	N@100
GritLM-Base	0.047	0.023	0.053	0.047	0.036	0.016	0.020	0.011	0.038	0.025	0.016	0.005
GritLM-FT	0.090	0.041	**0.133**	**0.072**	0.032	0.024	0.094	0.028	0.121	0.030	0.084	0.027
NeuMF	0.062	0.030	0.081	0.040	0.030	0.014	0.067	0.015	0.092	0.021	0.065	0.014
NGCF	0.060	0.031	0.092	0.043	0.008	0.003	0.051	0.013	**0.309**	**0.085**	0.026	0.006
MGL	0.065	0.033	0.088	0.046	0.026	0.011	0.060	0.014	0.132	0.039	0.054	0.011
DeepSeek-V3	0.080	0.041	0.099	0.053	0.044	0.013	0.090	0.020	0.102	0.023	0.055	0.015
GPT-4.1	0.086	0.043	0.114	0.055	0.049	0.017	0.091	0.030	0.113	0.045	0.066	0.022
LLM4Rerank	0.091	0.046	0.113	0.056	0.069	0.025	0.097	0.029	0.124	0.040	0.090	0.029
DropoutNet	0.066	0.033	0.089	0.045	0.027	0.012	0.059	0.013	0.121	0.025	0.053	0.011
MWUF	0.062	0.031	0.089	0.045	0.019	0.008	0.064	0.014	0.156	0.036	0.056	0.012
CVAR	0.063	0.031	0.083	0.041	0.029	0.013	0.072	0.017	0.154	0.033	0.064	0.015
MIRec	0.062	0.030	0.085	0.042	0.025	0.012	0.063	0.015	0.146	0.035	0.055	0.012
BUHS–Mean	0.075	0.033	0.103	0.052	0.030	0.012	0.095	0.027	0.169	0.052	0.087	0.024
BUHS–Attn	0.076	0.037	0.101	0.050	0.036	0.016	0.092	0.025	0.138	0.046	0.088	0.024
BUHS–GRU	0.078	0.039	0.104	0.052	0.034	0.014	0.085	0.022	0.139	0.042	0.080	0.020
AgentTailor	**0.099**	**0.048**	0.119	0.057	**0.081**	**0.028**	**0.109**	**0.031**	0.133	0.043	**0.102**	**0.031**

Local sensitivity to evidence perturbations. For two-field evidence in $[0, 1]$, perturbing each field by at most η gives $|e_{ui} - e'_{ui}| \leq 2\eta$. With candidates, routing, Stage 1 priors, and ε fixed, the same bound holds for s_{ui}, and a routed pair with score margin greater than 4η cannot reverse. This is a local score-order guarantee, not semantic validation or robustness to changed gates, missing evidence, or candidate sets. Appendix B.5 gives the proof.

5 Experiments

5.1 Datasets & experimental setup

We follow prior work (Balasubramanian et al., 2024) and evaluate on two widely used datasets: **MovieLens-1M** and **BookCrossing**. Dataset statistics are reported in Appendix Table 3. For both datasets, we adopt a chronological leave-one-out split: the latest interaction is used for testing, the penultimate interaction for validation, and the remaining interactions for training.

MovieLens-1M. MovieLens-1M contains explicit movie ratings, which we convert into implicit feedback. Following common long-tail settings, we define the top 20% of items by interaction frequency as head items and the remaining 80% as tail items (Zhang et al., 2021).

BookCrossing. BookCrossing is a sparse and noisy book-feedback dataset. We filter users and items with too few interactions following prior work (Balasubramanian et al., 2024), and evaluate users with at least 10 interactions. Our processed records retain the per-user timestamp order established before export. Due to its scale and sparsity, we reconstruct texts for 40% of interactions while preserving the chronological split. The top 1% of items by interaction frequency are treated as head items, and the rest as tail items (Zhang et al., 2021).

5.2 Baselines

We compare with four groups of baselines: **Backbone Retrieval**: **GritLM-Base** and **GritLM-FT**; **Conventional Recommenders**: **NeuMF** (He et al., 2017), **NGCF** (Wang et al., 2019), and **MGL** (Wei et al., 2023); **Long-tail Debiasing Methods**: **DropoutNet** (Volkovs et al., 2017), **MWUF** (Zhu et al., 2021a), **CVAR** (Zhao et al., 2022), **MIRec** (Zhang et al., 2021), and **BUHS** (Balasubramanian et al., 2024); and **LLM-based Re-rankers**: **DeepSeek-V3** (Liu et al., 2024a), **GPT-4.1**, and **LLM4Rerank** (Gao et al., 2025). Conventional and long-tail baselines use their own pipelines, while LLM-based re-rankers share AgentTailor's Stage 1 candidate pools. Table 1 therefore provides full-system comparisons, with shared-candidate comparisons among the LLM-based methods. Appendix B.6 gives the evaluation metrics.

Accessibility and conditional placement. Re-ranking can improve placement only for targets already retrieved by Stage 1. To examine this contribution separately, Appendix C.1 evaluates matched users whose tail targets are present in a shared candidate pool. This conditional analysis complements the full-system evaluation.

5.3 Overall performance

Table 1 summarizes the main results, with the best score in bold and the second-best underlined. Across five random seeds, paired significance tests show that AgentTailor significantly outperforms the strongest LLM-based baseline on MovieLens-1M ($p < 0.001$) and on BookCrossing ($p < 0.01$ for both HR and NDCG).

AgentTailor achieves the best overall and tail hit rates on both datasets while maintaining competitive NDCG and head-item performance. On MovieLens-1M, it leads in Overall H@10 and Tail H@10; on the sparser BookCrossing dataset, it achieves the best Overall and Tail scores for both HR and NDCG. These results support the effectiveness of combining semantic recall with selective evidence-based placement across different sparsity levels.

LLM4Rerank (Gao et al., 2025) remains competitive in NDCG, whereas AgentTailor achieves higher tail hit rates. This comparison suggests that explicit tail-oriented evidence assessment and selective placement can improve the inclusion of relevant tail items in the final recommendation list, even when a general LLM re-ranker already provides strong ranking precision.

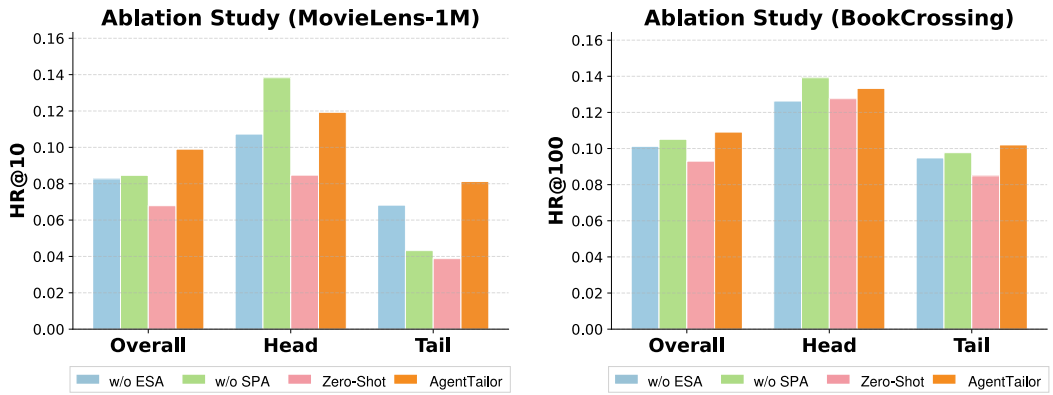
Against long-tail debiasing methods, AgentTailor's Tail H@10 on MovieLens-1M reaches 0.081, compared with 0.036 for the best BUHS variant. The shared-candidate comparison with LLM4Rerank in Appendix C.1 further examines the contribution of re-ranking to this long-tail objective.

The results also reveal a head–tail trade-off. Relative to GritLM-FT on MovieLens-1M, AgentTailor raises Overall H@10 from 0.090 to 0.099 and Tail H@10 from 0.032 to 0.081, while Head H@10 decreases from 0.133 to 0.119. The selected policy therefore improves overall and tail accuracy with a corresponding reduction in head accuracy.

Additional fixed-pool, routing, case, popularity-spectrum, reliability, and latency analyses are provided in Appendix C–D.

5.4 Ablation study

For all Stage 2 variants, we fix Stage 1 candidate pools so differences come from re-ranking rather than retrieval recall. We ablate **ESA** and **SPA** while keeping **PPP** and **TEE**, which provide user pressure and item-level evidence. Figure 2(a)–(b) summarizes the results.



(a) MovieLens-1M. (b) BookCrossing.

Figure 2: Stage 2 ablation results. We remove ESA or SPA while keeping PPP and TEE unchanged. We report HR@k under the *Overall*, *Head*, and *Tail* evaluation settings with dataset-specific k.

Impact of ESA. Removing ESA consistently degrades performance on both datasets. The decrease supports combining relevance and tail-fit through the deterministic score $e_{ui} = \mathrm{rel}_{ui} \cdot \mathrm{tailfit}_{ui}$, which favors candidates with support from both evidence fields.

Impact of SPA. Removing SPA reduces the gains from the extracted evidence. Stable placement merges evidence-supported candidates into the Stage 1 list while preserving the relative order of non-routed items. It also preserves the Stage 1 order when the placement gate is not satisfied, making the ranking response selective.

The head–tail breakdown shows a trade-off: removing SPA increases head HR but reduces tail and overall HR on both datasets.

Modular decomposition vs. zero-shot re-ranking. AgentTailor also outperforms the Zero-Shot LLM re-ranking variant under the same Stage 1 candidate pools, supporting the structured pipeline over the tested zero-shot prompt. Its deterministic ranking operators make the use of LLM evidence inspectable. Appendix C.2 further examines sensitivity to the routing thresholds.

5.5 Transfer to another retrieval backbone

We replace GritLM with Qwen3-4B to test whether the re-ranking pipeline transfers to another semantic retriever. The fine-tuned retriever and its AgentTailor extension share per-user Top-20 pools and are evaluated at $k = 10$; the base retriever is a separate retrieval reference.

Table 2: Backbone generalizability on MovieLens-1M using Qwen3-4B as the Stage 1 retriever (N=20, k=10).

Model	Overall H@10	Tail H@10
Qwen3-4B (Base)	0.0392	0.0303
Qwen3-4B (Fine-Tuned)	0.0642	0.0326
Qwen3-FT + AgentTailor	**0.0730**	**0.0571**

Table 2 shows that adding AgentTailor raises Overall H@10 from 0.0642 to 0.0730 and Tail H@10 from 0.0326 to 0.0571. The gains occur over the fine-tuned retriever's fixed candidate pool, indicating that useful tail candidates remain available for improved placement even after adapting the retrieval backbone.

The interface between the two stages helps explain this transfer. Stage 2 consumes candidate descriptions, user preference evidence, and the Stage 1 order; its placement rule does not depend on the retriever's embedding coordinates. Changing the backbone can therefore change which items are retrieved while preserving the evidence-assessment and placement procedure. The larger improvement in Tail H@10 is consistent with the branch's focus on long-tail candidates. Together, these results support transfer beyond GritLM, with the present evidence limited to one additional backbone and one dataset.

6 Conclusion

We propose AgentTailor, a modular LLM-assisted framework for personalized long-tail recommendation. AgentTailor combines SeRA for candidate accessibility with a dual-gate tail-specialist branch that separates evidence assessment from rank modification. LLMs supply item-level evidence, deterministic history thresholds control permission, and stable placement uses the evidence to order routed candidates. Experiments on MovieLens-1M and BookCrossing show improved long-tail recommendation with competitive overall performance. Future work can extend this separation to adaptive routing and online feedback, where the cost and benefit of intervention evolve over time.

Limitations. Generated evidence can be inaccurate despite valid formatting. Conditional re-ranking cannot recover an item absent from the candidate pool, and historical tail engagement does not certify that intervention will help a user. Fixed-pool offline evaluation does not establish robustness to changing catalogs or online user benefit. The local score bound assumes fixed routing and valid evidence. Appendix A discusses further limitations and possible exposure biases.

Acknowledgments and Disclosure of Funding

This work was supported in part by the National Natural Science Foundation of China (No. 62572075), in part by the Beijing Natural Science Foundation (No. L233034, No. L253004), in part by Fundamental Research Funds for the Beijing University of Posts and Telecommunications (No. 2025TSQY01), in part by the National Natural Science Foundation of China (No. 62572217) and in part by Natural Science Foundation of Guangdong Province (No. 2025A1515010739).

References

Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. The unfairness of popularity bias in recommendation. In *Proceedings of the Workshop on Recommendation in Multi-stakeholder Environments co-located with the 13th ACM Conference on Recommender Systems*, volume 2440, 2019.

Keshav Balasubramanian, Abdulla Alshabanah, Elan Markowitz, Greg Ver Steeg, and Murali Annavaram. Biased user history synthesis for personalized long-tail item recommendation. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 189–199, 2024.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3438–3445, 2020.

Jiawei Chen, Xiang Wang, Fuli Feng, and Xiangnan He. Bias issues and solutions in recommender system: Tutorial on the RecSys 2021. In *Proceedings of the 15th ACM Conference on Recommender Systems*, pages 825–827, 2021.

Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for YouTube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 191–198, 2016.

Jingtong Gao, Bo Chen, Xiangyu Zhao, Weiwen Liu, Xiangyang Li, Yichao Wang, Wanyu Wang, Huifeng Guo, and Ruiming Tang. LLM4Rerank: LLM-based auto-reranking framework for recommendations. In *Proceedings of the ACM on Web Conference 2025*, pages 228–239, 2025.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182, 2017.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. LightGCN: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 639–648, 2020.

Siyi Lin, Chongming Gao, Jiawei Chen, Sheng Zhou, Binbin Hu, Yan Feng, Chun Chen, and Can Wang. How do recommendation models amplify popularity bias? an analysis from the spectral perspective. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, pages 659–668, 2025.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024a.

Qidong Liu, Xian Wu, Yejing Wang, Zijian Zhang, Feng Tian, Yefeng Zheng, and Xiangyu Zhao. LLM-ESR: Large language models enhancement for long-tailed sequential recommendation. *Advances in Neural Information Processing Systems*, 37:26701–26727, 2024b.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. Generative representational instruction tuning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. Recommendations as treatments: Debiasing learning and evaluation. In *International Conference on Machine Learning*, pages 1670–1679, 2016.

Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. DropoutNet: Addressing cold start in recommender systems. *Advances in Neural Information Processing Systems*, 30:4957–4966, 2017.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024a.

Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 165–174, 2019.

Zhefan Wang, Yuanqing Yu, Wendi Zheng, Weizhi Ma, and Min Zhang. MACRec: A multi-agent collaboration framework for recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2760–2764, 2024b.

Chunyu Wei, Jian Liang, Di Liu, Zehui Dai, Mang Li, and Fei Wang. Meta graph learning for long-tail recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2512–2522, 2023.

Tianxin Wei, Fuli Feng, Jiawei Chen, Ziwei Wu, Jinfeng Yi, and Xiangnan He. Model-agnostic counterfactual reasoning for eliminating popularity bias in recommender system. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1791–1800, 2021.

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

Cai Xu, Xujing Wang, Ziyu Guan, Wei Zhao, and Meng Yan. GUIDER: Uncertainty guided dynamic re-ranking for large language models based recommender systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(19):16049–16057, 2026. doi: 10.1609/aaai.v40i19.38639.

Guipeng Xv, Chen Lin, Hui Li, Jinsong Su, Weiyao Ye, and Yewang Chen. Neutralizing popularity bias in recommendation models. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2623–2628, 2022.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983, 2018.

Junliang Yu, Xin Xia, Tong Chen, Lizhen Cui, Nguyen Quoc Viet Hung, and Hongzhi Yin. XSimGCL: Towards extremely simple graph contrastive learning for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 36(2):913–926, 2023.

Yin Zhang, Derek Zhiyuan Cheng, Tiansheng Yao, Xinyang Yi, Lichan Hong, and Ed H Chi. A model of two tales: Dual transfer learning framework for improved long-tail item recommendation. In *Proceedings of the Web Conference 2021*, pages 2220–2231, 2021.

Kui Zhao, Yuechuan Li, Zhaoqian Shuai, and Cheng Yang. Learning and transferring IDs representation in e-commerce. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1031–1039, 2018.

Xu Zhao, Yi Ren, Ying Du, Shenzheng Zhang, and Nian Wang. Improving item cold-start recommendation via model-agnostic conditional variational autoencoder. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2595–2600, 2022.

Chang Zhou, Jianxin Ma, Jianwei Zhang, Jingren Zhou, and Hongxia Yang. Contrastive learning for debiased candidate generation in large-scale recommender systems. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3985–3995, 2021.

Yongchun Zhu, Ruobing Xie, Fuzhen Zhuang, Kaikai Ge, Ying Sun, Xu Zhang, Leyu Lin, and Juan Cao. Learning to warm up cold item embeddings for cold-start recommendation with meta scaling and shifting networks. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1167–1176, 2021a.

Ziwei Zhu, Yun He, Xing Zhao, and James Caverlee. Popularity bias in dynamic recommendation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2439–2449, 2021b.

A Limitations and broader impacts

AgentTailor depends on LLM-generated item texts and evidence traces, so its effectiveness is affected by LLM factuality, formatting stability, and inference cost. Schema checks can detect format violations but do not establish factual correctness; imperfect generated evidence may still affect re-ranking decisions. The evidence traces provide inspectability and traceability, but should not be interpreted as formal causal explanations of ranking behavior. In addition, our evaluation is limited to two public datasets and offline ranking metrics; future work should validate the framework on larger-scale industrial logs, richer item domains, and online user-feedback settings.

AgentTailor may improve exposure for relevant long-tail items and support more diverse recommendation outcomes, which can benefit niche content providers and users with exploratory preferences. However, increasing long-tail exposure must be handled carefully: inappropriate promotion of low-quality or irrelevant items may reduce user satisfaction, and LLM-generated descriptions may introduce or amplify semantic biases. Practical deployment should therefore monitor content quality, head–tail trade-offs, and generated evidence reliability.

B Additional experimental details

B.1 Dataset statistics

Table 3: Statistics of the datasets used in the experiments.

	MovieLens-1M	**BookCrossing**
Number of Users	6,040	52,019
Number of Items	3,883	225,794
Number of Head Items	777	2,258
Number of Tail Items	3,106	223,536
Number of Interactions	1,000,209	733,257
Sparsity	95.7%	99.99%

Table 3 reports the statistics of MovieLens-1M and BookCrossing. MovieLens-1M is relatively dense, while BookCrossing is extremely sparse and contains a much larger long-tail item space. These differences motivate the use of different candidate-pool sizes and evaluation cutoffs across datasets.

B.2 Schema-constrained text augmentation

We use schema-constrained prompts to normalize item and user texts. Item records include title, creator-related metadata, genre/category information, and a short description. User retrieval profiles are summarized from training-only interactions for Stage 1 semantic recall, while decision-oriented profiles for activated users are generated later by the Preference and Pressure Profiler (PPP). The prompts enforce concise fields to reduce popularity-correlated verbosity and stabilize retrieval and evidence-extraction inputs. Since these texts are generated, they may still contain noise; therefore, AgentTailor treats them as structured semantic evidence rather than ground-truth annotations, and we audit their reliability in Appendix D.1.

B.3 Semantic Recall Agent training details

In Stage 1, the Semantic Recall Agent (SeRA) retrieves Top-N candidates using a fine-tuned GritLM-7B retriever. We set N=20 for MovieLens-1M and N=150 for BookCrossing to account for their different sparsity levels. GritLM-7B is fine-tuned with a learning rate of 2×10^{-5} on an NVIDIA L20 GPU. We adopt LoRA with α=32 and r=64.

For each query, we sample nine negatives: up to two user-specific low-rated items as hard negatives when available, with unobserved items filling the remaining budget. Random negatives are drawn approximately uniformly from unobserved items. Since long-tail catalogs contain far more tail items than head items, uniform sampling yields a tail-dominant negative pool in expectation. This increases training coverage of niche item texts as negatives; its effect on tail recall remains an empirical question.

Data generation and prompting use DeepSeek-V3-0324 with temperatures $\{0.7, 0.1, 0.0, 0.0\}$. Unless otherwise specified, the main comparison results are averaged over five runs.

We transform recommendation into a structured text matching problem by constructing triples of *(query, positive, negatives)*:

i **Query** (q): For each user, we prompt GritLM to summarize historical interactions into a concise natural-language retrieval profile, e.g., *"Represent this User Profile for Retrieval: <User... >."*

ii **Positive** (p^+): Metadata of a positively rated item, such as actors, director, genres, and synopsis, is converted into structured text prefixed by *"Represent this Item (Positive Sample): <Item... >."*

iii **Negatives** ($\mathcal{N}^-$): We include both hard negatives, sampled from the same user's low-rated items, and random unobserved items to encourage discriminative semantic representations.

This design aligns user intent with item semantics at the textual level, bridging sparse recommendation data and pretrained language representations. It also makes the recall stage interpretable: retrieved items can be traced back to explicit user and item descriptions rather than opaque ID-only embeddings.

B.4 StableMerge details

StableMerge is a deterministic placement operator over the full Stage 1 candidate list. If the placement gate is not satisfied, the routed subset is empty, or no valid evidence score is available, it returns the original list C_u. Otherwise, it integrates the evidence-ordered routed candidates into the full list, using the Stage 1 order as the reference for candidates without routed evidence. Routed candidates are not restricted to their original positions: they may cross non-routed candidates, which retain their mutual Stage 1 order but can shift in absolute rank. For example, with tail-only routing, $[H_1, T_1, H_2, T_2]$ may become $[T_2, T_1, H_1, H_2]$; the head order $H_1 \prec H_2$ is preserved, while both head ranks and the number of tail items in the top two positions change. Thus, stable placement does not imply unchanged head HR/NDCG or a fixed tail count in each top-k prefix. These outcomes are evaluated empirically. The full-list sorting implementation requires $O(N \log N)$ time and $O(N)$ auxiliary space.

B.5 Local stability and intervention bounds

Fix a user, the Stage 1 list, the routed subset, both gate decisions, and the coefficient ε. Let $r, t, r', t' \in [0, 1]$ denote the original and perturbed relevance and tail-fit scores, with $|r - r'| \le \eta$ and $|t - t'| \le \eta$. Then

$$|rt - r't'| = |(r - r')t + r'(t - t')| \le |r - r'|\,|t| + |r'|\,|t - t'| \le 2\eta. \tag{14}$$

The fixed rank-prior term cancels, so $|s_{ui} - s'_{ui}| \le 2\eta$. For any two routed candidates,

$$|(s_{ui} - s_{uj}) - (s'_{ui} - s'_{uj})| \le |s_{ui} - s'_{ui}| + |s_{uj} - s'_{uj}| \le 4\eta. \tag{15}$$

Thus, if $s_{ui} - s_{uj} > 4\eta$, the perturbed score still orders i ahead of j. This bound concerns the two-field product score and a fixed deterministic placement rule. It does not cover a prompt rewrite that changes the routed set, evidence dropout and recovery, changed gate decisions, or the compact single-score template. Nor does it imply globally optimal ranking or truthful evidence.

Permission and changed-user bounds. For the same users and candidates, an exact no-op outside the placement-permission set implies

$$|\overline{m}_{\text{Stage2}} - \overline{m}_{\text{Stage1}}| \le q_{\text{changed}} \le q_{\text{permission}}, \tag{16}$$

where $m_u \in [0, 1]$ is a per-user metric and the means and fractions use the same user weights. Each unchanged user contributes zero difference; each changed user contributes at most one in absolute value. This consistency check applies to single-positive HR and NDCG on the same evaluated population. It should be checked per seed before aggregation and separately within any conditioned target group.

B.6 Evaluation protocol

We evaluate recommendation accuracy using Hit Rate and Normalized Discounted Cumulative Gain.

Table 4: Tail-only conditional re-ranking on matched users with the relevant tail target already retrieved. Candidate pools contain 20 items for MovieLens-1M and 150 for BookCrossing; evaluation cutoffs are 10 and 100. H/N denote HR/NDCG, and gains are differences of unrounded means relative to the Stage 1 candidate order.

Dataset	Method	Users	Stage1 H	Stage1 N	H	N	H Gain	N Gain
ML-1M @10	LLM4Rerank	169	0.6095	0.3518	0.6036	0.3585	-0.0059	+0.0068
ML-1M @10	AgentTailor	169	0.6095	0.3518	**0.6509**	**0.3854**	**+0.0414**	**+0.0336**
Book @100	LLM4Rerank	122	0.7951	0.2212	0.7869	0.2326	-0.0082	+0.0114
Book @100	AgentTailor	122	0.7951	0.2212	**0.8361**	**0.2340**	**+0.0410**	**+0.0129**

Hit Rate (HR@k). HR@k measures whether the ground-truth item appears in the top-k recommendation list. We use HR@10 for MovieLens-1M and HR@100 for BookCrossing. The larger cutoff for BookCrossing accounts for its extreme sparsity and avoids trivially near-zero scores.

Normalized Discounted Cumulative Gain (NDCG@k). NDCG@k evaluates ranking quality by assigning higher rewards to ground-truth items appearing earlier in the recommendation list.

We follow prior work (Balasubramanian et al., 2024) in reporting *Overall*, *Head*, and *Tail* results using the dataset-specific popularity partitions defined in the experimental setup. Candidate pools for LLM-based re-rankers are shared as described in the main-text baseline protocol.

C Additional analyses

C.1 Tail-only conditional re-ranking analysis

This analysis isolates placement of already-retrieved tail targets from candidate accessibility. We restrict evaluation to users whose held-out target belongs to the tail partition and is present in the Stage 1 candidate pool. AgentTailor and LLM4Rerank are compared on the same selected users, targets, and Stage 1 pools. The pools may contain both head and tail candidates; "tail-only" refers to the held-out targets used for evaluation.

Matched-user protocol. We retain users with valid outputs from both methods and evaluate each pair on the same target and candidate pool. Each row reports a single matched-user evaluation; HR and NDCG are averaged over the matched users, and gains are calculated before rounding. Candidate sizes and evaluation cutoffs are given in Table 4.

Dataset-specific placement configurations. For MovieLens-1M, AgentTailor uses relevance and tail-fit evidence for retrieved tail candidates, followed by a hybrid controller with strong and conservative prior-aware placement branches.

For BookCrossing, the compact template in Appendix F.5 assesses the first 120 candidates of the full Top-150 pool, which can include head items. Let $\widehat{e}_{ui}$ denote its directly predicted compact evidence score. Placement uses weak-prior fusion $2\widehat{e}_{ui} + 0.5\pi_{ui}$, equivalent in score ordering to $\widehat{e}_{ui} + 0.25\pi_{ui}$, with π_{ui} defined in Eq. 10. The movement-control parameter is set to `max_move=60`.

Results. Table 4 shows that AgentTailor improves both HR and NDCG over the Stage 1 order. On MovieLens-1M, HR rises from 0.6095 to 0.6509 and NDCG from 0.3518 to 0.3854; on BookCrossing, HR rises from 0.7951 to 0.8361 and NDCG from 0.2212 to 0.2340. LLM4Rerank improves NDCG but slightly reduces HR on both datasets, indicating that AgentTailor better balances target inclusion and target position within these matched candidate pools.

These results characterize placement of already-retrieved tail targets with paired outputs. They complement the full-system evaluation in Table 1; their conditional scores are not directly comparable with the main table's Overall scores.

Table 5: Effect of dual-gate routing on MovieLens-1M. Agent activation denotes the fraction of users entering the tail-specialist branch, while effective placement denotes the fraction whose final ranking is changed by SPA.

Policy	δ_a/δ_p	Agent Act.	Eff. Place.	Overall H	Tail H
Stage 1 only	–	0.00%	0.00%	0.6465	0.5888
Single gate	0.45	11.10%	11.10%	0.6536	0.6396
Dual gate	0.35/0.45	26.93%	11.10%	0.6536	0.6396
Dual gate	0.33/0.45	31.35%	11.10%	0.6536	0.6396

C.2 Dual-gate router sensitivity

We vary the activation and placement thresholds to examine whether evidence-assessment coverage can increase while ranking interventions remain selective. Following Eq. 7, the activation gate controls access to the tail-specialist branch, while the placement gate determines whether SPA may modify the ranking.

Users that pass only the activation gate receive evidence assessment but retain the Stage 1 order. The comparison therefore tests the independent control offered by the two thresholds.

Table 5 shows that lowering δ_a from 0.45 to 0.35 and 0.33 increases activation from 11.10% to 26.93% and 31.35%, respectively. With $\delta_p = 0.45$, effective placement remains 11.10%, and Overall and Tail HR remain 0.6536 and 0.6396. The dual-gate policy thus expands evidence assessment while preserving the placement policy's ranking performance.

Figure 3 provides a broader sensitivity view across dual-gate threshold pairs. On MovieLens-1M, lowering the activation threshold increases the fraction of users entering the tail-specialist branch, while the effective placement rate is mainly controlled by the placement threshold. On BookCrossing, the HR curves are comparatively stable across threshold pairs, suggesting that the tail-specialist branch is less sensitive to the activation threshold under extreme sparsity.

Together, these results show that the two thresholds provide separate controls over assessment coverage and placement selectivity.

C.3 Case study

To illustrate AgentTailor's inspectable decision trace, we present a case study for user $u = 11$ in Figure 4. The PPP-generated profile shows that the user enjoys action and spy genres but strongly dislikes director Mike Newell. We examine a challenging long-tail item, "An Awfully Big Adventure," a romance-comedy directed by Newell.

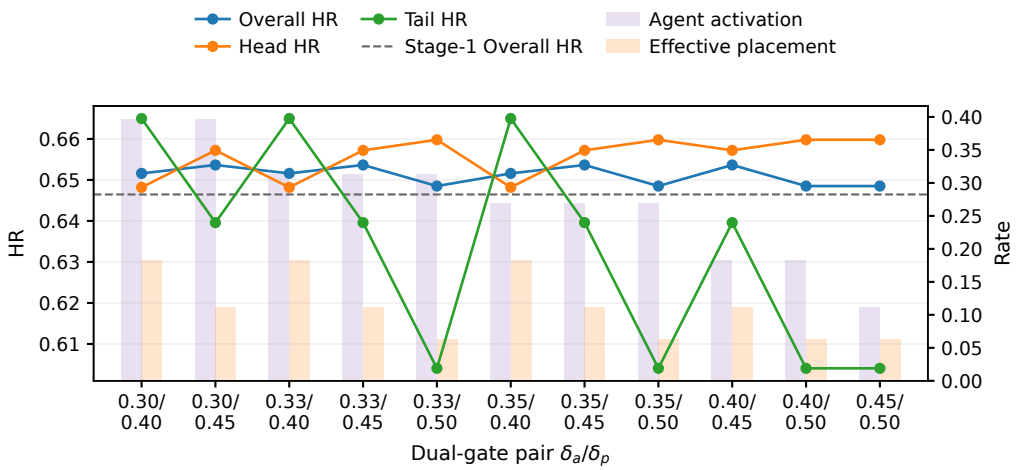
TEE identifies two negative evidence signals: the item mismatches the user's preferred action/spy interests, and it involves the disliked director. Therefore, its relevance evidence is weak despite being a long-tail candidate. **ESA** combines relevance and tail-fit through the product-form evidence score, suppressing the item because tail-fit alone is insufficient without preference relevance. **SPA** then preserves its low placement instead of promoting it into the final list.

The case illustrates how the intermediate evidence connects the user's preferences to a placement decision: an unpopular item with weak relevance receives a low synthesized score and remains low in the final ranking.

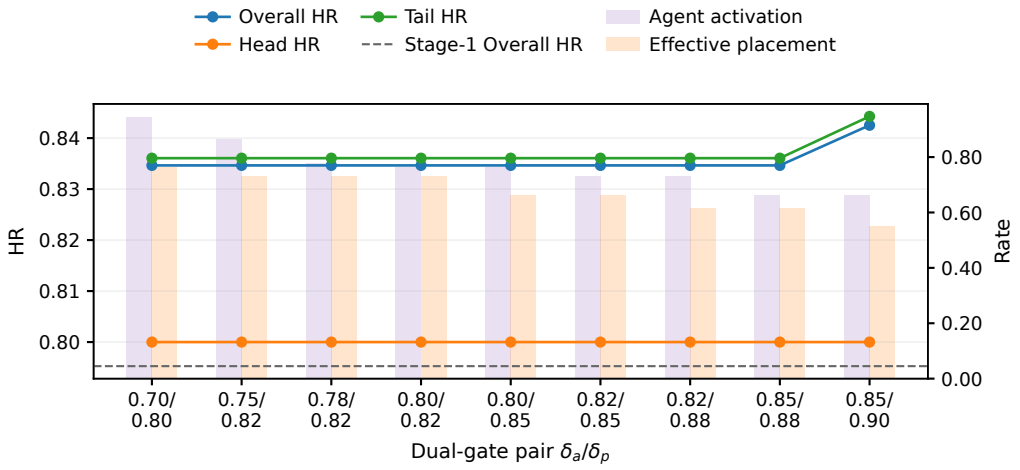
C.4 Popularity–spectrum analysis

Following the spectral perspective (Lin et al., 2025), we analyze the alignment between model predictions and item popularity. We perform SVD on the predicted score matrix $\hat{Y} = UV^\top$ and compute the cosine similarity between the principal right singular vector q_1 and the item-popularity vector r. A higher $\cos(q_1, r)$ indicates that the model's dominant latent factor is more strongly aligned with popularity.

Figure 5 shows strong popularity alignment in the fitted score matrices of LightGCN (0.992) and XSimGCL (0.994), while BUHS-Attn (0.524) and GritLM-Base (0.588) show weaker alignment.



(a) MovieLens-1M.



(b) BookCrossing.

Figure 3: Dual-gate router sensitivity. Each x-axis label denotes an activation/placement threshold pair δ_a/δ_p. Lines report Overall/Head/Tail HR, the dashed line denotes the Stage 1 Overall HR, and bars report the agent activation rate and effective placement rate. The results show that activation coverage and effective rank changes are decoupled: more users can enter the tail-specialist branch for evidence assessment, while stable placement remains conservative.

Fine-tuning increases GritLM's alignment to 0.629, suggesting that semantic retrievers can also inherit popularity tendencies from interaction data.

This observation motivates complementing semantic recall with conditional tail-evidence placement. AgentTailor retains the retrieval prior while allowing preference-supported adjustments to candidate ordering, providing a targeted response to popularity-aligned retrieval scores.

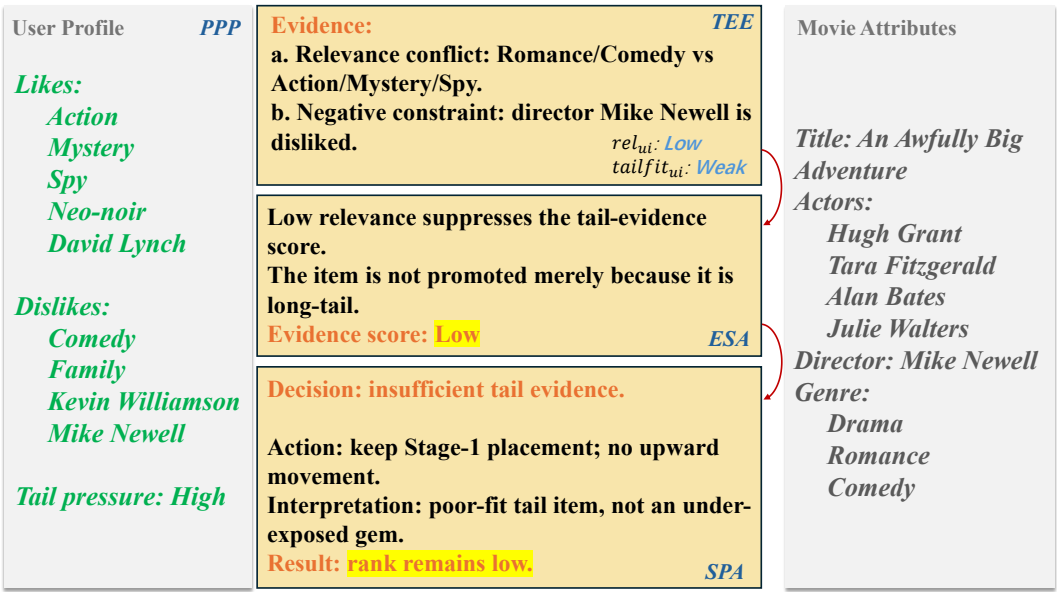


Figure 4: Case study of AgentTailor's conditional tail-evidence re-ranking process.

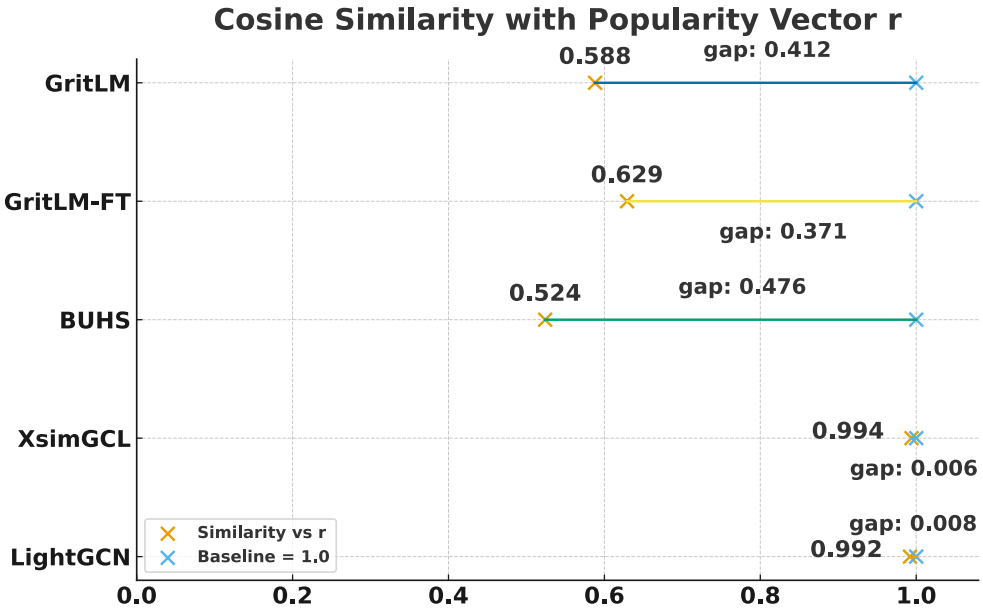


Figure 5: Cosine similarity between the principal singular vector q_1 and the item-popularity vector r.

D Reliability and efficiency

D.1 Reliability audit of LLM-assisted outputs

Because AgentTailor relies on schema-constrained augmentation and LLM-assisted evidence extraction, we audit the faithfulness and format correctness of generated outputs.[2] We report three indicators: (i) **Hallucination Rate**, the fraction of extracted claims unsupported by provided evidence; (ii) **Constraint Violation Rate**, the fraction of outputs violating schema or formatting constraints; and (iii) **Factuality Score**, a 1–5 score obtained through an LLM-as-a-judge protocol.

[2]All prompts and checking scripts are included in the released code repository.

Table 6: Quantitative reliability audit of LLM-assisted outputs (1,400 reasoning checks).

Metric	Value	Protocol
Hallucination Rate ↓	0.50%	evidence-supported checking
Constraint Violation Rate ↓	1.37%	schema/rule constraints
Factuality Score ↑	4.25/5	LLM-as-a-judge

Table 7: Latency comparison under full sequential and conditional amortized protocols, in seconds per user. LLM API latency is measured on 30 users per dataset; amortized latency combines the measured branch cost with the activation rate of the selected dual-gate configuration.

Protocol	Method	MovieLens-1M	BookCrossing
Full sequential	BUHS-GRU	0.037	0.213
Full sequential	Zero-shot LLM Re-ranking	1.521	15.740
Full sequential	LLM4Rerank	2.562	18.055
Full sequential	AgentTailor branch	1.442	17.267
Conditional amortized	LLM4Rerank	2.562	18.055
Conditional amortized	AgentTailor	**0.452**	**11.421**

Table 6 shows low hallucination and constraint violation rates under the checking protocol. Together with the factuality score, these results characterize the format stability and evidence quality of the generated outputs used for re-ranking.

D.2 Latency analysis

We evaluate inference latency under two protocols. The first protocol, *full sequential inference*, measures the average API latency when every evaluated user invokes the corresponding LLM-based re-ranking procedure. For AgentTailor, this reports the cost of the activated tail-specialist branch per activated user. The second protocol, *conditional amortized inference*, accounts for the activation gate: users not activated by the router directly preserve the Stage 1 ranking and incur no additional LLM API call. Therefore, the amortized latency of AgentTailor is computed as

$$T_{\text{amortized}} = q_{\text{act}} \cdot T_{\text{branch}}, \tag{17}$$

where q_{act} is the activation rate and T_{branch} is the average latency of the activated tail-specialist branch.

All LLM latency measurements use DeepSeek-chat API calls. The routing statistics for the selected dual-gate configuration give activation rates of 31.35% on MovieLens-1M and 66.14% on BookCrossing.

Table 7 shows that longer candidate lists make LLM re-ranking more expensive on BookCrossing. Conditional activation reduces AgentTailor's average cost from 1.442 seconds per activated branch call to 0.452 seconds per user on MovieLens-1M, and from 17.267 to 11.421 seconds on BookCrossing.

AgentTailor has lower amortized latency than LLM4Rerank on both datasets. The activation gate limits LLM calls to selected users, while the placement gate independently controls whether the assessed evidence changes their rankings.

E Asset licenses

We use public recommendation datasets, pretrained models, and existing baseline implementations for academic evaluation. Table 8 identifies the principal assets and their original sources, distinguishing named open licenses from provider-specific terms. The public repository linked in the abstract provides the released implementation and data-format instructions. Access to code or processed artifacts does not replace the original terms governing third-party data, model weights, or API services.

Table 8: Principal third-party assets and their upstream licenses or usage terms.

Asset	Original source and license or terms
MovieLens-1M	GroupLens usage license: research use with attribution; redistribution and commercial use require separate permission.
BookCrossing	Original Book-Crossing release. A standardized open-license grant could not be verified from the original release; we do not assign one to the dataset or its derived records.
GritLM-7B	GritLM model card: Apache License 2.0.
Qwen3-4B	Qwen model license: Apache License 2.0.
DeepSeek API	DeepSeek Terms of Use govern service access. API use does not distribute the model weights.
GPT-4.1 API	OpenAI Services Agreement and applicable service terms govern API access.
Baseline code	Third-party implementations retain their respective upstream terms; no uniform license is assigned across baselines.

The repository and appendix provide code-access and methodological documentation rather than a blanket license for upstream assets. Where an original license or redistribution grant is not established above, its absence is not treated as permission to relicense the corresponding asset.

F Prompt templates

This appendix summarizes the key prompt templates used in AgentTailor. We report simplified dataset-agnostic templates with placeholders. Dataset-specific fields are normalized into the same schema: MovieLens-1M uses movie fields such as director, actors, genres, and descriptions, while BookCrossing uses book fields such as author, categories, themes, and descriptions. All user-side prompts use only interactions observed before recommendation time; validation and test targets are excluded.

F.1 Schema-constrained item text augmentation

We first normalize sparse item metadata into concise structured text. This prompt is used to construct comparable semantic evidence for both head and tail items.

Prompt A: Schema-Constrained Item Text Augmentation

```
System:
You are an expert metadata specialist. Generate a valid JSON object for the provided item.

Constraints:
1. Keep the original item ID and title.
2. Fill creator-related fields using the dataset schema:
   - MovieLens-1M: director and top actors.
   - BookCrossing: author and related themes.
3. Use concise genre/category labels when available.
4. Write a short semantic description including themes, tone, and key content.
5. Avoid verbose or unsupported claims.

Output:
Return ONLY a raw JSON object. No markdown.

Expected schema:
{
  "id": "<ITEM_ID>",
  "title": "<TITLE>",
  "creator": "<DIRECTOR_OR_AUTHOR>",
  "secondary_creators": ["<ACTOR_OR_OTHER_CREATOR>", "..."],
  "genres": ["<GENRE_1>", "<GENRE_2>", "..."],
  "themes": ["<THEME_1>", "<THEME_2>", "..."],
  "description": "<SHORT_SEMANTIC_DESCRIPTION>"
}

User:
Item: <RAW_ITEM_METADATA>
```

F.2 Stage-1 retrieval texts

Stage 1 converts user histories and item records into retrieval-oriented texts for GritLM fine-tuning and inference. User retrieval profiles are generated from training-only interactions.

Prompt B: Stage-1 Retrieval Text Format

```
User query template:
Represent this User Profile for Retrieval:
<USER_PROFILE_SUMMARY_FROM_TRAINING_HISTORY>

Positive item template:
Represent this Item (Positive Sample):
<STRUCTURED_ITEM_TEXT>

Negative item template:
Represent this Item (Negative Sample):
<STRUCTURED_ITEM_TEXT>
```

F.3 Preference and Pressure Profiler

For users activated by the tail-pressure router, PPP summarizes the observed history into a compact decision-oriented profile. The profile is used only as local evidence context for Stage 2 candidate assessment.

Prompt C: Preference and Pressure Profiler

```
System:
You are AgentTailor's preference profiling module.
Summarize the user's observed interaction history into a compact, machine-checkable profile.
Use only the provided history and item metadata.
Do not use validation or test target interactions.

User:
Dataset: <DATASET_NAME>
User ID: <USER_ID>
Observed history JSON:
<HISTORY_ITEMS_JSON>

Return ONLY valid JSON with this schema:
{
  "user_profile": {
    "positive_preferences": {
      "genres_or_categories": ["..."],
      "creators": ["..."],
      "themes_or_styles": ["..."]
    },
    "negative_preferences": {
      "genres_or_categories": ["..."],
      "creators": ["..."],
      "themes_or_styles": ["..."]
    },
    "liked_anchors": [
      {"id": "<ITEM_ID>", "reason": "<SHORT_REASON>"}
    ],
    "disliked_anchors": [
      {"id": "<ITEM_ID>", "reason": "<SHORT_REASON>"}
    ],
    "decision_rules": [
      "<SHORT_RULE_FOR_RELEVANCE_ASSESSMENT>"
    ]
  }
}
```

F.4 Tail-Evidence Evaluator

TEE is the main LLM-assisted scoring prompt in Stage 2. It does not ask the LLM to generate a final ranking. Instead, it asks for bounded item-level evidence scores, which are later consumed by deterministic ESA and SPA.

Prompt D: Tail-Evidence Evaluator

```
System:
Return valid JSON only. Do not include markdown.

User:
You are AgentTailor's Tail-Evidence Evaluator for a two-stage recommender.
The Stage-1 rank is already a strong prior.
Score each candidate independently; do not output a final ranking.
Use only the given user history, profile, and item fields.

Definitions:
- relevance: how well the candidate matches the user's preferences.
- tail_fit: whether the candidate is a plausible long-tail target for this user.
  Do not reward an irrelevant item only because it is unpopular.

Dataset: <DATASET_NAME>
User ID: <USER_ID>
Candidate IDs in Stage-1 order:
<CANDIDATE_ID_LIST>

User profile JSON:
<USER_PROFILE_JSON>

Recent/high-rated history JSON:
<HISTORY_JSON>

Candidate items JSON:
[
  {
    "id": "<ITEM_ID>",
    "rank": <STAGE1_RANK>,
    "title": "<TITLE>",
    "genres": ["..."],
    "pop": "<head_or_tail>",
    "creator": "<DIRECTOR_OR_AUTHOR>",
    "secondary_creators": ["..."],
    "themes": ["..."],
    "desc": "<SHORT_DESCRIPTION>"
  }
]

Return ONLY valid JSON in this format:
{"scores":[["id", relevance, confidence, tail_fit],
           ["id2", 0.0, 0.5, 0.0]]}

All scores must be numbers in [0,1].
Include every Candidate ID exactly once.
```

F.5 Compact tail-evidence template for long candidate lists

For long candidate lists, especially BookCrossing, we use a compact variant that reduces completion length by asking for a single evidence score. This score asks the LLM to summarize relevance and tail-fit jointly; it is not an explicit product of two separately observed fields. It is used only for routed candidate subsets.

Prompt E: Compact Tail-Evidence Scoring

```
System:
Return valid compact JSON only. No markdown.

User:
You are AgentTailor's low-latency Tail Evidence Agent.
Score whether each candidate is a relevant long-tail target for this user.
The score should combine relevance and tail-fit.
Use Stage-1 rank as weak context, but do not reward an irrelevant item only because it is tail.
Use only the provided JSON.

Dataset: <DATASET_NAME>
User: <USER_ID>
Candidate IDs: <CANDIDATE_ID_LIST>

User profile:
<USER_PROFILE_JSON>
```

```
Recent history:
<HISTORY_JSON>

Candidates:
[
  {
    "id": "<ITEM_ID>",
    "r": <STAGE1_RANK>,
    "t": "<TITLE>",
    "g": ["<GENRE_OR_CATEGORY>", "..."],
    "p": "<head_or_tail>",
    "creator": "<DIRECTOR_OR_AUTHOR>",
    "secondary": ["<ACTOR_OR_THEME>", "..."]
  }
]

Return exactly:
{"evidence":[["id",0.0],["id2",1.0]]}

Include every Candidate ID exactly once.
```

F.6 Zero-shot LLM re-ranking baseline

For comparison, the zero-shot LLM baseline directly outputs a full permutation of the shared Stage 1 candidate list. This differs from AgentTailor, where the LLM only provides evidence and the final placement is deterministic.

Prompt F: Zero-Shot LLM Re-ranking Baseline

```
System:
Return valid JSON only. No markdown.

User:
You are a zero-shot recommendation reranker.
Rank the candidate items for this user using only the provided profile, history, and candidate metadata.


Prioritize relevance.
When two candidates are similarly relevant, prefer the relevant long-tail item.
Never promote an irrelevant item only because it is long-tail.

Dataset: <DATASET_NAME>
User: <USER_ID>
User profile:
<USER_PROFILE_JSON>

Recent history:
<HISTORY_JSON>

Candidate IDs:
<CANDIDATE_ID_LIST>

Candidates:
<CANDIDATE_ITEMS_JSON>

Return exactly:
{"ranked_ids":["id1","id2"]}

The ranked_ids list must be a complete permutation of Candidate IDs.
```

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction accurately describe AgentTailor as an interpretable LLM-assisted framework for personalized long-tail recommendation, with a Semantic Recall Agent for candidate accessibility and a dual-gate tail-specialist branch for evidence-based re-ranking. The stated claims are consistent with the method and experiments.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: Appendix A discusses key limitations, including dependency on LLM-generated item texts and evidence traces, LLM factuality and formatting stability, inference cost, and the restricted evaluation scope based on two public datasets and offline metrics.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes].

   Justification: The local evidence-stability bound states its bounded-score and fixed-routing assumptions, with the complete derivation in Appendix B.5. The changed-user bound follows from bounded per-user metrics and exact no-ops outside the permission set. Neither result asserts semantic correctness or global ranking optimality.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes].

   Justification: The manuscript specifies datasets, chronological splits, head-tail definitions, candidate-pool protocol, evaluation metrics, retrieval backbone, candidate sizes, LoRA settings, negative sampling, prompting temperatures, and the shared candidate-pool protocol for LLM-based re-rankers. The public code repository is linked in the abstract.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes].

   Justification: The abstract links the public repository. Prompts are provided in Appendix F, and repository documentation gives installation and data-format instructions.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: The paper describes the dataset preprocessing, chronological leave-one-out split, head-tail partitioning, evaluation cutoffs, Stage 1 candidate sizes, GritLM fine-tuning setup, LoRA configuration, negative sampling strategy, LLM prompting temperatures, and the protocol for LLM-based re-rankers.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes].

Justification: The manuscript reports paired statistical significance tests over five random seeds between AgentTailor and the strongest LLM-based baseline. AgentTailor significantly outperforms the baseline on MovieLens-1M ($p < 0.001$) and on BookCrossing ($p < 0.01$ for both HR and NDCG). Although the paper does not include full error bars for every table entry, it provides appropriate significance evidence for the main comparative claims.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: The appendix reports the GPU used for retriever fine-tuning and provides API latency measurements for the LLM-assisted re-ranking pipeline. The paper also reports conditional amortized latency to clarify the runtime cost of the dual-gate design.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: The work uses public recommendation datasets and evaluates algorithmic ranking behavior. It does not introduce private user data, new human-subject experiments, or high-risk deployment claims.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes].

    Justification: Appendix A discusses potential positive impacts, such as improving exposure for relevant long-tail items and supporting more diverse recommendations, as well as potential risks, including inappropriate tail promotion, reduced user satisfaction, and biases introduced by LLM-generated textual evidence.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: The paper does not release a high-risk foundation model, sensitive dataset, or system intended for direct deployment in a high-stakes domain. The released assets are code, prompts, and processed recommendation data for reproducing the reported experiments.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes].

    Justification: Appendix E identifies dataset and model sources together with the licenses or usage terms that could be verified. Third-party assets retain their original terms; the repository does not assign a blanket license to upstream data, models, or baseline implementations.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes].

    Justification: The manuscript states that the released repository includes code, data, prompts, and checking scripts, which document the new processed assets and experimental pipeline.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: The paper does not include crowdsourcing experiments or new human-subject studies.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: The paper does not involve crowdsourcing or new human-subject experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes].

    Justification: The method section and appendix describe LLM usage for schema-constrained text augmentation, preference-and-pressure profiling, and tail-evidence extraction. The paper also clarifies that LLMs provide auditable item-level evidence rather than freely generating the final ranking.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.