MemRec: Collaborative Memory-Augmented Agentic Recommender System

Weixin Chen[1,2] **Yuhan Zhao**[1] **Jingyuan Huang**[2] **Zihe Ye**[2]
Clark Mingxuan Ju[3†] **Tong Zhao**[3†] **Neil Shah**[3†] **Li Chen**[1] **Yongfeng Zhang**[2‡]
[1]Hong Kong Baptist University [2]Rutgers University [3]Snap Inc.
{cswxchen, csyhzhao, lichen}@comp.hkbu.edu.hk
{chy.huang, zihe.ye, yongfeng.zhang}@rutgers.edu
{mju, tong, nshah}@snap.com

Abstract

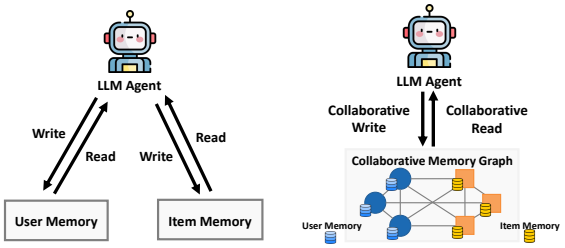
The evolution of recommender systems has shifted from traditional collaborative filtering to LLM-based agentic systems, which rely on semantic user and item memories to make predictions. However, existing agents maintain these memories in isolation. This overlooks crucial collaborative signals, such as user-item co-engagements and peer relationships across the community, which significantly limits their ability to uncover hidden preferences and accurately infer user needs, particularly for data-sparse users. To bridge this gap, we introduce *collaborative memory*, a paradigm that connects isolated semantics to enable the sharing of relational insights. Yet, naively utilizing collaborative memory causes severe context overload and introduces noise to downstream LLMs, alongside prohibitive computational costs. To resolve this, we propose **MemRec**, a framework that architecturally decouples memory management from reasoning. MemRec introduces a dedicated, lightweight language model (LM_{Mem}) to efficiently manage and synthesize a dynamic collaborative memory graph in the background. It provides only distilled, high-signal contexts to a downstream, heavyweight large language model (LLM_{Rec}) for the final recommendation. Extensive experiments on four benchmarks demonstrate that MemRec achieves state-of-the-art performance.

Code: https://github.com/rutgerswiselab/memrec

Homepage: https://memrec.weixinchen.com

1 Introduction

Memory has long served as a foundational component in Recommender Systems (RS). The field has evolved from capturing preferences through sparse rating matrices in conventional collaborative filtering era (Sarwar et al., 2001; Koren et al., 2009) to using dense latent embeddings in the deep learning era (Covington et al., 2016; He et al., 2017). Recently, the emergence of agentic RS, powered by Large Language Models (LLMs), has ushered in a new paradigm, i.e., semantic memory (Wu et al., 2024; Zhang et al., 2025).



(a) Traditional Agentic RS (b) MemRec Framework

Figure 1: **(a) Existing Agents** interact with user and item memories through separate, isolated read/write channels. **(b) MemRec** performs collaborative operations on a memory graph, enabling the flow of relational signals across peer users and related items to overcome data sparsity and transfer collaborative knowledge.

In agentic RS, memory is transformed into a semantic format, enabling LLMs to perform complex reasoning and use tools with natural language as the substrate (Zhao et al., 2024b). For an agent to be more than a stateless function, it must utilize this persistent memory to retain and evolve its user understanding through ongoing interactions (Xi et al., 2025; Park et al., 2023). The evolution of memory mechanisms in this context can be delineated into three key milestones: (1) *No explicit memory*, relying solely on the LLM's inherent knowledge (Liu et al., 2023; Lyu et al., 2024); (2) *Static memory*, characterized by retrieving context from fixed storage (Xu et al., 2025b; Gao et al., 2023); and recently (3) *Dynamic, self-reflective memory*, where agents iteratively update their understanding over time (Tang et al., 2025; Zhang et al., 2024b).

However, these approaches predominantly represent *non-collaborative* paradigms. As illustrated in Figure 1a, current agents typically reflect only on their siloed memories, such as a user's mem-

[†]Authors affiliated with Snap Inc. served in advisory roles only for this work.
[‡]Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 44515–44544

ory (M_u) containing a textual narrative of their past interactions, or an item's memory (M_i) initialized by its static semantic description (Xu et al., 2025b; Zhang et al., 2024b). This strictly local scope isolates them from the most potent signal in recommender systems: collaborative relationships. By ignoring the broader user-item graph, these agents fail to transfer interaction signals from warm to cold items through co-engagements, or to leverage the shared preferences of peer users to uncover hidden interests (Wang et al., 2019; He et al., 2020).

A seemingly intuitive solution for bridging this gap is to inject raw collaborative neighborhoods directly into the agent's memory (e.g., naively concatenating a target user's context window with the textual interaction histories of dozens of similar peers, or expanding an item's memory with descriptions from all co-engaged items). However, this naive brute-force approach proves inadequate for at least two critical reasons:

- *Cognitive Overload.* While the agent may be able to access large quantities of neighbor memories, it struggles to effectively distill pertinent information from this abundance. The sheer volume of textual and structural signals increases difficulty for the reasoning agent to identify salient knowledge (Liu et al., 2024), as validated in §3.3.
- *Prohibitive Collaborative Updates.* A truly collaborative memory must evolve dynamically. Whenever a new interaction occurs, the updated knowledge should ideally propagate to all connected neighbors. Under a naive brute-force approach, keeping these neighborhood contexts synchronized necessitates redundant, independent LLM calls for every related user and item. This creates an intractable computational bottleneck during real-time serving, making continuous graph evolution prohibitively expensive.

Consequently, a core challenge emerges: *How can we distill extensive collaborative knowledge into memory to empower the reasoning agent, while ensuring efficient evolution of the graph?*

To address these challenges, we introduce **MemRec** (Figure 1b), a framework built upon architectural decoupling to shift from isolated to collaborative memory. By dedicating a separate Memory Manager ($\mathbf{LM_{Mem}}$) to manage a dynamic graph and synthesize compact grounding, this architecture systematically resolves both cognitive overload and update bottlenecks. Firstly, addressing cognitive overload during retrieval, our *Collaborative Memory Retrieval* method overcomes the limitations of isolated memory paradigms. Instead of relying solely on siloed user or item memory, it leverages LLM-guided domain-adaptive rules to curates neighbor signals to synthesize a compact, high-utility collaborative memory. Secondly, overcoming update bottlenecks, we develop an *Asynchronous Collaborative Propagation* mechanism inspired by Label Propagation (Zhu and Ghahramani, 2002). It efficiently batches self-reflection and neighbor updates into a single asynchronous operation and achieves constant-time ($O(1)$) interaction complexity, ensuring continuous graph evolution without incurring the computational penalties of redundant, independent updates.

Extensive evaluations on four benchmarks show that MemRec achieves state-of-the-art performance. Furthermore, our architectural analysis demonstrates MemRec's flexibility, establishing a new Pareto frontier that balances reasoning quality, computational cost, and deployment constraints, supporting diverse setups from cloud-native APIs to on-premise local models.

2 Methodology

Problem Formulation Let $\mathcal{U}$ and $\mathcal{I}$ denote the sets of users and items, respectively. For each user $u \in \mathcal{U}$, we denote their historical interactions as H_u. Given a target user u, a natural language instruction $\mathcal{I}_u$ requiring semantic interpretation (e.g., specific constraints, complex goals), and a set of candidate items $C \subseteq \mathcal{I}$, the objective is to generate a ranked list of recommendations accompanied by grounded justifications.

Memory in Agentic RS In agentic RS, memory serves as the persistent state, storing information in semantic form to evolve user understanding over time. While traditional systems maintain simple textual metadata, recent agentic architectures formalize these as individual semantic profiles, conceptualized as memory modules (M_u or M_i) (Xu et al., 2025b; Zhang et al., 2024b). In practice, these memories manifest as evolving textual narratives summarizing a user's preferences or an item's characteristics based on historical contexts. During recommendation, a reasoning agent LLM_{Rec} leverages these memories to perform the task.

Despite these advancements, existing agentic RS predominantly adhere to an isolated memory paradigm. They treat the collective memory M merely as a disconnected set of individual narra-

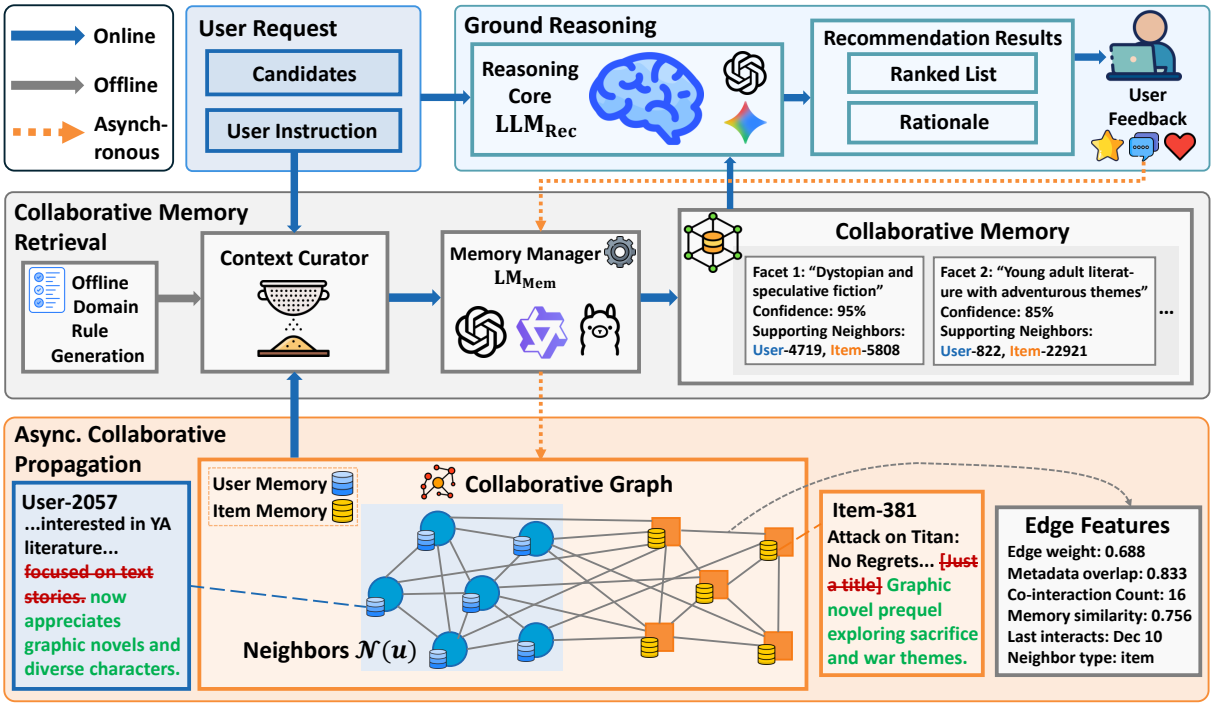


Figure 2: The overall framework of **MemRec**, decoupling reasoning (LLM_{Rec}) from memory management (LM_{Mem}). The three-stage pipeline consists of: *Collaborative Memory Retrieval*, synthesizing high-order connectivity context from memory graph; *Grounded Reasoning*, scoring items based on instruction and context; and *Asynchronous Collaborative Propagation*, evolving the semantic memory graph in the background.

tives $\{M_u\} \cup \{M_i\}$. For instance, reasoning for user u relies solely on their personal siloed memory M_u. This isolation excludes critical collaborative signals from the broader community, hindering the system's ability to leverage collective intelligence.

2.1 The MemRec Pipeline

To address this limitation, MemRec introduces a collaborative framework featuring an architecturally decoupled Memory Manager (LM_{Mem}). This manager operates on a unified memory graph $G = (\mathcal{V}, E)$. The node set $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$ represents users and items, where each node $v \in \mathcal{V}$ stores its corresponding evolving semantic memory M_v. The edges E encode interactions and derived relations connecting these memories. Unlike approaches relying solely on isolated node memories, MemRec leverages the high-order connectivity of G to synthesize and propagate collaborative signals.

As illustrated in Figure 2, MemRec operates in three key stages. Firstly, Collaborative Memory Retrieval processes the expansive graph to extract and synthesize a concise **Collaborative Memory** (M_{collab}) for the current task. Secondly, Grounded Reasoning utilizes this synthesized context to perform recommendations with enhanced grounding. Finally, Asynchronous Collaborative Propagation dynamically updates the individual semantic memories (M_v) across the graph, capturing emerging trends and shifting user preferences without disrupting the ongoing agentic interactions.

2.1.1 Collaborative Memory Retrieval

A central challenge in harnessing collaborative memory lies in mitigating *cognitive overload* for the reasoning agent. Naively retrieving raw memories from all neighbors not only exceeds the context window constraints, but more crucially, bombards the LLM with noise, resulting in hallucinations and diminished instruction adherence. Our objective, therefore, is to extract a collaborative memory (M_{collab}) from the raw graph that maximizes relevance to the user's recommendation needs while rigorously filtering out extraneous interactions.

To this end, we draw conceptual inspiration from Information Bottleneck (IB) theory: our goal is to derive a compressed representation of the raw graph context that preserves maximal information relevant to the target task, while minimizing irrelevant or redundant signals. Guided by this insight, we adopt a "Curate-then-Synthesize" strategy. We first curate the raw collaborative graph by pruning redundant neighbors to reduce its complexity and size. Subsequently, we synthesize the distilled graph to amplify informative collaborative signals for the downstream reasoning agent. We elaborate

on these two stages below.

LLM-Guided Context Curation Conventional graph pruning strategies generally fall into two categories: (i) traditional rule-based heuristics, such as random walk-based methods (Perozzi et al., 2014), which rely on predefined structural assumptions and lack semantic awareness; and (ii) fully learned neural scorers, such as GNN-based attention weights (Veličković et al., 2018), which require expensive training and often lack interpretability. Both approaches present limitations for LLM-based agents. Heuristic methods cannot adapt to domain-specific semantic nuances, while learned scorers introduce significant computational overhead and integration complexity.

To overcome these challenges, we propose a novel zero-shot **LLM-as-Rule-Generator** paradigm, which harnesses the rich background knowledge and semantic understanding of advanced LLMs to autonomously generate domain-specific curation rules. These rules are employed to guide the curation process, enabling efficient and adaptive collaborative memory construction tailored to the downstream LLM's needs. Specifically, in an offline phase, LM_{Mem} analyzes domain statistics $\mathcal{D}_{\text{domain}}$ (e.g., interaction density, category distribution) to synthesize interpretable heuristics.

$$R_{\text{domain}} \leftarrow \text{LM}_{\text{Mem}}(\mathcal{D}_{\text{domain}} \| P_{\text{meta}}) \quad \text{(Offline)} \tag{1}$$

Here, P_{meta} acts as a generic meta-prompt guiding LM_{Mem} to generate a set of domain-specific heuristic rules R_{domain} tailored to balance relevance and diversity for the target dataset (see Appendix F.1 and A.4 for templates and examples). At inference time, these rules act as a high-speed filter, selecting the top-k neighbors $N'_k(u)$ in milliseconds:

$$N'_k(u) = \text{Curate}(N(u), R_{\text{domain}}, k) \tag{2}$$

This step acts as the first coarse "compression" pass in the IB framework, efficiently discarding neighbors with low potential mutual information. To illustrate the domain-adaptivity of this zero-shot generation, Figure 3 presents a snippet of the generated rules for the Books dataset.

Collaborative Memory Synthesis The goal of this stage is to distill the raw information from curated neighbors N'_k into a concise, structured format (M_{collab}) that maximizes informative signals for the downstream reasoning agent. LM_{Mem} synthesizes these signals into a set of structured preference facets $\{F\}$ as collaborative memory M_{collab}:

$$M_{\text{collab}} = \{F\} \leftarrow \text{LM}_{\text{Mem}}(\text{Rep}(N'_k) \| M_u^{t-1} \| P_{\text{synth}}) \tag{3}$$

where $\text{Rep}(N'_k)$ denotes the representation of neighbor information. To effectively synthesize signals within the LLM's limited context window, we adopt a tiered representation strategy. The target user u is represented by their full, accumulated semantic memory M_u^{t-1} to provide comprehensive background context. Neighboring nodes in N'_k are provided via compact contextual representations (e.g., condensed signals derived from memory or recent behaviors) designed to offer immediate evidence of collaborative patterns without overwhelming the model with verbose histories. The synthesis prompt P_{synth} (a snippet of which is illustrated in Figure 4, with the complete template available in Appendix F.3) then guides LM_{Mem} to extract high-level facets from these tiered inputs. To make this concept concrete, Figure 5 contrasts a standard isolated memory with our synthesized collaborative memory facets, illustrating how `MemRec` enriches the reasoning context with structured, community-driven evidence.

LLM-Generated Curation Rules (Books Snippet)

Rule 1: Content Similarity Boost
- If `metadata_overlap` > 0.6: Apply 2.5x multiplier.
- *Rationale: Books are highly content-driven...*

Rule 2: Collaborative Filtering with Threshold
- If `co_interaction` > 3: Apply 1.8x multiplier.
- *Rationale: Meaningful CF signal requires sufficient overlap.*

Figure 3: Snippet of the domain-adaptive curation rules generated by the LM_{Mem} on the Books dataset.

Prompt Snippet: Collaborative Synthesis

Your Task: Analyze the target user's personal memory and the raw collaborative memories from the curated neighboring users and items. Identify distinct preference facets that characterize this user's current interests.

Output Requirement: For each facet, provide:
1. A concise description of the preference theme.
2. A confidence score (0-1).
3. A list of supporting neighbors providing evidence.

Figure 4: Snippet of the synthesis prompt used by LM_{Mem} to distill collaborative memory facets.

2.1.2 Grounded Reasoning

This stage is for reading the memory and performing the final ranking. By feeding the synthesized collaborative memory M_{collab} alongside the user

Isolated vs. Synthesized Collaborative Memory

[Isolated M_u^{t-1}]: "User prefers dystopian settings. Recently interacted with 'The Hunger Games'."

[Synthesized M_{collab}]:
• **Theme:** Cyberpunk & Corporate Dystopia (Conf: 0.9)
Evidence: User Neighbor (ID: 2057) shows deep interest in corporate control; Item Neighbor ('1984') shares foundational dystopian themes.
• **Theme:** High-Stakes Survival (Conf: 0.75)
Evidence: Item Neighbor ('Battle Royale') exhibits strong survival elements matching recent user interactions.

Figure 5: Illustrative comparison of isolated vs. synthesized collaborative memory. The synthesized M_{collab} extracts distinct themes with confidence scores and grounds them in specific neighbor evidence.

instruction $\mathcal{I}_u$ and the candidate item memories C_{info}, the LLM_{Rec} executes the reasoning process:

$$\{s_i, r_i\}_{i=1}^{N} \leftarrow \text{LLM}_{\text{Rec}}(\mathcal{I}_u \| M_{\text{collab}} \| C_{\text{info}} \| P_{\text{rerank}}) \tag{4}$$

The ranking prompt P_{rerank} (Appendix F.4) instructs the LLM to generate a relevance score s_i and a natural language rationale r_i for each candidate based on the provided context. Grounding the reasoning in M_{collab} ensures that the generated rationale is factually supported by broader community evidence provided.

2.1.3 Async. Collaborative Propagation

A static graph is inherently limited in its ability to capture evolving trends and shifting user preferences. As users continue to interact with items, the semantic representations stored within their corresponding memory nodes must be dynamically updated to reflect the most current patterns and preferences. Failure to adapt these representations risks diminishing the relevance and effectiveness of the recommender system over time. Drawing inspiration from Label Propagation algorithms (Zhu and Ghahramani, 2002), which spread information to connected nodes based on proximity within the graph structure, we introduce a mechanism to propagate "semantic labels" (insights) derived from new interactions asynchronously.

The update process conceptually involves two steps including updating the directly interacting nodes and propagating insights to neighbors. When user u interacts with item i_c at time step t, LM_{Mem} first generates updates for the user's own memory M_u^t and the item's memory $M_{i_c}^t$:

$$M_u^t, M_{i_c}^t \leftarrow \text{LM}_{\text{Mem}}(M_{\text{collab}} \| M_u^{t-1} \| M_{i_c}^{t-1} \| P_{\text{update}}) \tag{5}$$

Here M_u^t and M_u^{t-1} denote the user's memory state at the current time step t and the previous time step $t-1$, respectively. Crucially, this process also facilitates collaborative propagation by identifying connected neighbors from $N_k'(u)$ and propagating the shared theme as incremental updates ΔM_{neigh}:

$$\{\Delta M_{\text{neigh}}\} \leftarrow \text{LM}_{\text{Mem}}(M_{\text{collab}} \| M_u^{t-1} \| M_{i_c}^{t-1} \| N_k'(u) \| P_{\text{update}}) \tag{6}$$

This explicit propagation enriches the global memory graph with high-order signals. To help visualize this evolution from an old memory state to a newly updated collaborative graph, Appendix E details a complete qualitative journey of this update process.

Crucially, we optimize this memory evolution to resolve the update bottleneck. While a naive synchronous approach scales linearly ($O(|N_k'|)$ calls) and incurs massive input token redundancy by repeating the user context for each neighbor, `MemRec` achieves an $O(1)$ Call Complexity per interaction. This is accomplished by decoupling the update frequency from the primary reasoning loop and batching reflections asynchronously. Specifically, we execute the logical steps of self-reflection (Eq. 5) and neighbor propagation (Eq. 6) as a single, batched asynchronous operation. A unified prompt P_{update} (Appendix F.5) guides the LM_{Mem} to jointly synthesize all updates, ensuring continuous graph evolution without disrupting the online recommendation latency.

3 Empirical Evaluation

In this paper, we conduct extensive experiments to answer the following research questions:

- **RQ1 (Overall Performance):** Does `MemRec` outperform SOTA baselines across diverse datasets?
- **RQ2 (Architectural Impact):** Is architectural decoupling crucial for information bottleneck?
- **RQ3 (Flexibility):** How cost-effective and flexible is `MemRec` for diverse deployments?
- **RQ4 (Ablation Study):** Are the core mechanisms of `MemRec` essential for its performance?
- **RQ5 (Robustness & Quality):** How robust is `MemRec`, and what is its qualitative impact?

3.1 Experimental Setup

Datasets We evaluate our methods on four widely used benchmark datasets covering diverse domains with varying interaction densities: **Amazon Books**, **Amazon Goodreads**, **MovieTV**, and **Yelp**. For all datasets, we use the specific user instructions and evaluation splits provided by InstructRec (Xu et al., 2025b) to ensure fair comparison with instruction-following baselines. Table 1 summarizes the basic statistics of these datasets. Detailed descriptions of each dataset and its domain characteristics are provided in Appendix A.1.

Table 1: Statistics of the datasets used in experiments.

Dataset	$\|\mathbf{U}\|$	$\|\mathbf{I}\|$	$\|\mathbf{E}\|$	$\bar{\mathbf{L}}_\mathbf{u}$	Density
Books	7.4K	120.9K	207.8K	28.2	2.33e-4
GoodReads	11.7K	57.4K	618.3K	52.7	9.19e-4
MovieTV	5.6K	29.0K	79.7K	14.1	4.87e-4
Yelp	3.0K	31.6K	63.1K	21.4	6.77e-4

Baselines We evaluate MemRec against a suite of strong baselines, grouped by their underlying memory paradigms. The first category comprises traditional pre-LLM methods that utilize dense latent embeddings to encode and preserve historical information, including LightGCN (He et al., 2020), SASRec (Kang and McAuley, 2018), and P5 (Geng et al., 2022). The second category encompasses memory-based approaches developed in the era following AgentRS, which can be further subdivided into: (1) models with *no explicit memory*, such as Vanilla LLM (Liu et al., 2023) that operate on raw interaction histories; (2) those employing *static memory*, exemplified by iAgent's fixed profile representations; and (3) *dynamic memory* agents that update isolated memories, namely i^{2}Agent (Xu et al., 2025b), AgentCF (Zhang et al., 2024b), and RecBot (Tang et al., 2025). In contrast, our **MemRec** introduces a new paradigm, **Dynamic Collaborative Memory**, featuring asynchronous graph propagation. Baseline details are in Appendix A.2.

Experimental Setup We implement MemRec using **gpt-4o-mini** (OpenAI, 2024) for both LLM_{Rec} and LM_{Mem}, setting $k = 16$ and $N_f = 7$. For main results, we set the candidate list size $N = 10$ on full test sets and observe consistent trends with larger candidate sets in Appendix D.1. We report **Hit Rate (H@K)** and **NDCG (N@K)** for $K \in \{1, 3, 5\}$. Following (Zhang et al., 2024b), we utilize a randomly sampled subset of 1000 users in subsequent studies. More comprehensive implementation details are provided in Appendix A.3.

3.2 Main Results (RQ1)

Tables 2 and 3 present the comprehensive performance comparison across four datasets. All reported improvements of MemRec over the best baseline are statistically significant ($p < 0.05$). From the results, we observe some key findings:

- **MemRec decisively outperforms all baselines in ranking metrics.** Our framework achieves state-of-the-art performance across all reported metrics on the four benchmark datasets. Notably, on Goodreads, MemRec achieves its most significant gain, improving H@1 by +28.98% relative to the strongest baseline i^{2}Agent. On the dense Yelp dataset, it also demonstrates superiority across diverse metrics (e.g., +15.77% in H@1 and +7.59% in N@5), proving its ability to effectively capture broad community signals via collaborative memory combined with specific user instructions.
- **Memory paradigm hierarchy: Collaborative > Dynamic > Static > No Memory.** Among the agentic baselines, dynamic memory approaches consistently outperform static memory methods such as iAgent. In turn, static methods generally achieve better results than approaches with no explicit memory (Vanilla LLM). While these findings align with current trends, we observe that even SOTA dynamic agents (e.g., AgentCF) still significantly underperform relative to MemRec. This underscores the limitation of considering memories in isolation and highlights the absolute necessity of explicitly injecting collaborative signals into the agent's memory module.
- **MemRec bridges the gap between traditional CF robustness and modern LLM reasoning.** Traditional models like LightGCN show inconsistent performance, struggling on sparse tasks (Books) while remaining competitive on dense graphs (Yelp). Conversely, older LLM paradigms like P5 struggle due to limited model capacity and reliance on ID-based pre-training. MemRec successfully bridges these worlds, leveraging powerful LLM reasoning to dominate where traditional CF fails, while using collaborative graph signals to surpass isolated agentic baselines.

3.3 Impact of Cognitive Overload (RQ2)

Cognitive overload occurs when agents fail to distill pertinent signals from raw graph contexts. We validate MemRec's architecture by comparing it against a Vanilla LLM and a Naive Collaborative

Table 2: Main results for **Books** and **Goodreads**. "Improv." denotes the relative improvement of MemRec over the best baseline, and all improvements are statistically significant ($p < 0.05$).

	Books					Goodreads				
Model	**H@1**	**H@3**	**N@3**	**H@5**	**N@5**	**H@1**	**H@3**	**N@3**	**H@5**	**N@5**
LightGCN	0.1753	0.3259	0.2596	0.5703	0.3592	0.2499	0.5879	0.4432	0.7903	0.5263
SASRec	0.0914	0.2830	0.2001	0.4845	0.2824	0.1324	0.3518	0.2576	0.5407	0.3349
P5	0.2192	0.3607	0.2994	0.5273	0.3671	0.1569	0.3229	0.2509	0.5060	0.3256
Vanilla LLM	0.3138	0.5617	0.4533	0.7270	0.5226	0.2864	0.4662	0.3948	0.7390	0.5041
iAgent	0.3925	0.5560	0.4858	0.6905	0.5409	0.2617	0.4949	0.3954	0.6591	0.4626
RecBot	0.3984	0.5491	0.4846	0.6786	0.5376	0.2705	0.4754	0.3876	0.6495	0.4589
AgentCF	0.3457	0.6060	0.4960	0.7403	0.5512	0.2951	0.5910	0.4654	0.7726	0.5399
i^2Agent	0.4453	0.6517	0.5649	0.7708	0.6138	0.3099	0.6079	0.4825	0.7675	0.5481
MemRec	**0.5117**	**0.6915**	**0.6152**	**0.8007**	**0.6601**	**0.3997**	**0.6658**	**0.5540**	**0.8052**	**0.6112**
Improv.	*+14.91%*	*+6.11%*	*+8.90%*	*+3.88%*	*+7.54%*	*+28.98%*	*+9.52%*	*+14.82%*	*+1.89%*	*+11.51%*

Table 3: Main results for **MovieTV** and **Yelp**. Notation follows Table 2; all improvements are significant ($p < 0.05$).

	MovieTV					Yelp				
Model	**H@1**	**H@3**	**N@3**	**H@5**	**N@5**	**H@1**	**H@3**	**N@3**	**H@5**	**N@5**
LightGCN	0.3482	0.5643	0.4738	0.6883	0.5241	0.3444	0.5658	0.4720	0.7546	0.5494
SASRec	0.3399	0.5233	0.4470	0.6382	0.4942	0.2305	0.4312	0.3458	0.5597	0.3980
P5	0.1696	0.3206	0.2554	0.5008	0.3290	0.1444	0.3207	0.2435	0.5220	0.4785
Vanilla LLM	0.4050	0.7764	0.6098	0.8603	0.6445	0.1692	0.5275	0.3696	0.6861	0.4360
iAgent	0.4253	0.6170	0.5361	0.7420	0.5871	0.3995	0.6005	0.5148	0.7300	0.5681
RecBot	0.4367	0.6113	0.5375	0.7309	0.5866	0.4007	0.6003	0.5156	0.7169	0.5636
AgentCF	0.3906	0.6693	0.5523	0.7864	0.6006	0.1925	0.4374	0.3326	0.6374	0.4147
i^2Agent	0.4912	0.7225	0.6262	0.8221	0.6672	0.4205	0.6454	0.5517	0.7648	0.6007
MemRec	**0.5882**	**0.7819**	**0.7011**	**0.8817**	**0.7422**	**0.4868**	**0.6912**	**0.6053**	**0.7908**	**0.6463**
Improv.	*+19.75%*	*+8.22%*	*+11.96%*	*+7.25%*	*+11.24%*	*+15.77%*	*+7.10%*	*+9.72%*	*+3.40%*	*+7.59%*

Agent (which processes uncurated context in one stage) in Figure 6. Results show the Naive Agent (orange) plateaus because a single model cannot effectively ingest verbose context and perform complex ranking simultaneously, creating an information bottleneck. MemRec (blue) breaks this plateau through architectural decoupling. By separating memory management (LM_{Mem}) from reasoning (LLM_{Rec}), it ensures the ranker receives only high-signal, "Curate-then-Synthesize" context. Consequently, MemRec consistently and substantially outperforms the monolithic approach across all datasets (e.g., +34% relative H@1 gain on Books).

3.4 Flexibility and Cost-Effectiveness (RQ3)

To evaluate the flexibility and practical deployment viability of `MemRec`, we analyze the trade-offs between reasoning performance, sequential latency, and computational cost across various configurations (Figure 7). A detailed metrics breakdown is provided in Appendix C.

Crucially, this analysis focuses on the **online inference cost** (i.e., the Re-ranking stage). A fundamental advantage of `MemRec` is its ability to offload the computationally heavy cognitive load of processing dense collaborative graphs to asynchronous offline batches. Unlike monolithic baseline agents that must ingest vast raw contexts in real-time, `MemRec` achieves superior online efficiency by only querying the distilled collaborative memory. This architectural shift establishes a vastly superior Pareto frontier. Specifically, the **Standard** (`gpt-4o-mini`) and **Cloud-OSS** (a high-throughput open-weights model on Azure matching the mini price tier) configurations achieve near-ceiling performance at a fraction of the baselines' online cost, demonstrating model-agnostic serving efficiency. Furthermore, the **Vector** variant showcases extreme modularity by replacing the LLM ranker with lightweight vector similarity search to achieve sub-millisecond online latency. Additionally, as detailed in Appendix Table 5, **Local** deployments (e.g., `Qwen-2.5-7B`) provide competitive performance for privacy-sensitive domains.

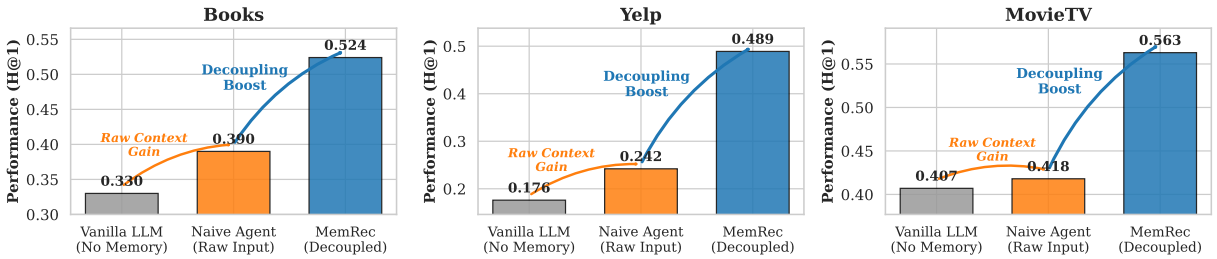


Figure 6: Impact of architectural decoupling on H@1. **MemRec** (blue) overcomes the information bottleneck that causes **Naive Agents** (orange) to plateau, achieving substantial gains over both Naive and **Vanilla LLM** (gray).

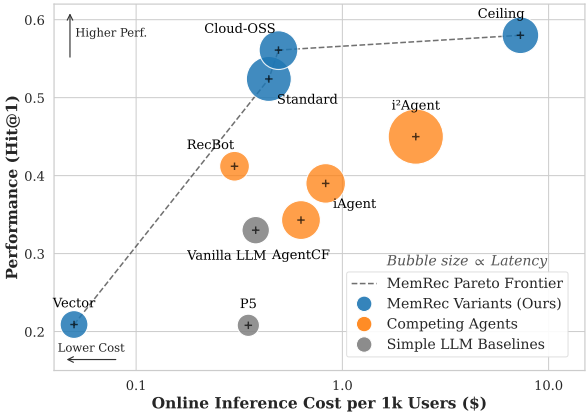


Figure 7: Efficiency-Cost-Performance Landscape.

3.5 Ablation Studies (RQ4)

A comprehensive ablation study (Table 4) confirms the positive contribution of MemRec's key collaborative components. Removing the collaborative retrieval stage (*w/o Collab. Read*), where the agent only reflects on isolated personal history, causes a drastic 9.9% drop in H@1, validating the critical role of synthesizing global graph signals over relying solely on isolated memory. Replacing the domain-adaptive LLM curator with generic heuristic rules (*w/o LLM Curation*) leads to a 5.5% drop, confirming the superior precision of our zero-shot, LLM-guided curation strategy in filtering noise. Finally, disabling asynchronous propagation (*w/o Collab. Write*) results in a 4.2% drop. This suggests that while a static graph supports broad retrieval (high H@5), dynamic collaborative updates are crucial for refining top-tier ranking precision (H@1) by capturing evolving community trends.

Table 4: Comprehensive ablation study on books.

Model Config.	H@1	H@3	N@3	H@5	N@5	Drop
MemRec (Full)	**0.527**	**0.713**	**0.634**	0.803	**0.670**	-
w/o Collab. Write	0.505	0.702	0.619	**0.814**	0.665	4.2%
w/o LLM Curation	0.498	0.685	0.606	0.788	0.648	5.5%
w/o Collab. Read	0.475	0.650	0.575	0.769	0.624	9.9%

3.6 Further Analysis (RQ5)

Robustness to Highly Niche Users Collaborative methods often struggle with "niche" users lacking sufficient interaction history. We stratified a 1,000-user test subset by activity level (Low, Medium, High). Rather than degrading, MemRec yields the greatest benefits for data-sparse users (Figure 8). By leveraging the collaborative graph to offset personal history deficits, MemRec achieves a massive **+91.4%** relative Hit@1 improvement over the Vanilla LLM for the Low activity group.

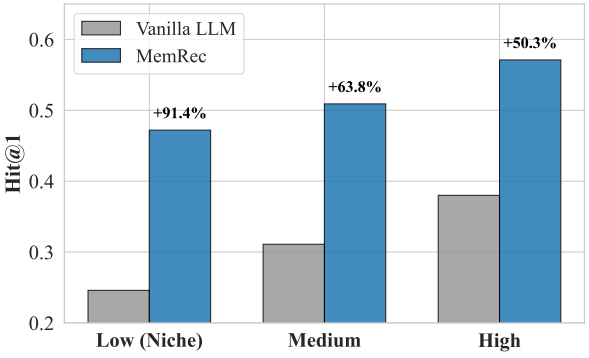


Figure 8: Sub-group performance across activity levels. MemRec yields massive gains (+91.4%) for data-sparse niche users over the Vanilla baseline.

Perturbation Analysis To evaluate the risk of negative propagation, we injected random noise (fake items) into target user histories. As shown in Figure 9, MemRec remains highly robust, maintaining a competitive 0.491 Hit@1 even under 30% noise injection. This resilience stems from the "Curate-then-Synthesize" pipeline: the zero-shot LLM curation acts as a robust buffer, successfully filtering out irrelevant peers before they overwhelm the downstream reasoning agent.

Rationale Quality Analysis A GPT-4o-based evaluation (Figure 10) shows MemRec significantly improves rationale *Specificity* and *Relevance* over baselines by incorporating collaborative signals. See Appendix D.2 for full details.

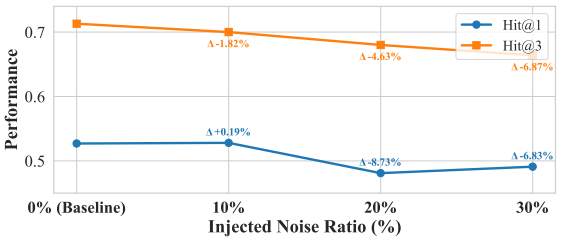


Figure 9: Perturbation analysis under varying noise injection ratios. MemRec exhibits minimal degradation, demonstrating strong resistance to noisy neighbors.

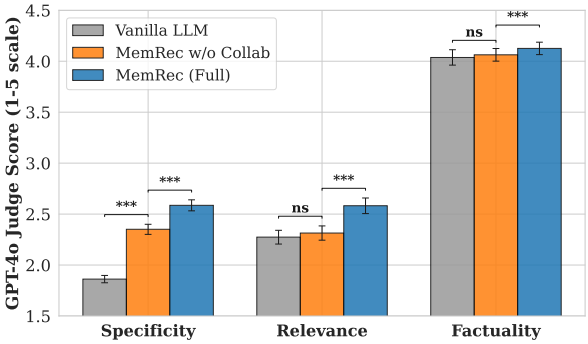


Figure 10: Rationale Quality Evaluation (GPT-4o Judge, 1-5 scale). Error bars show 95% CIs; *** denotes $p < 0.001$, while ns means not significant on paired t-test.

Hyperparameter Sensitivity We vary neighbors k and facets N_f in Figure 11 (Hit@1), observing a performance "sweet spot" around $k \in \{16, 32\}$ and $N_f = 7$. Comprehensive analysis across full metrics is detailed in Appendix D.4.

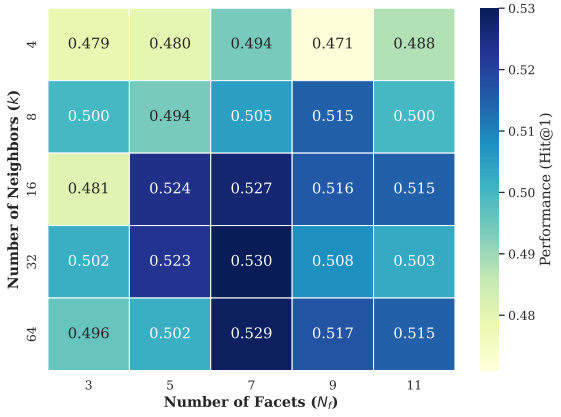


Figure 11: Hyperparameter sensitivity on books.

Qualitative Analysis A comprehensive case study illustrating the complete collaborative journey including collaborative memory synthesis, grounded reasoning, and asynchronous memory propagation, is provided in Appendix E.

4 Related Works

To overcome LLM context constraints for long-horizon tasks, research has evolved from basic Retrieval-Augmented Generation (RAG) pipelines (Lewis et al., 2020) to sophisticated, dedicated memory architectures. Systems like *MemGPT* (Packer et al., 2023) and *Generative Agents* (Park et al., 2023) demonstrate how decoupled memory managers and reflective synthesis can maintain long-term coherence. However, these general-purpose frameworks typically target factual or conversational domains, fundamentally neglecting the specialized, high-order connectivity required for collaborative recommendation environments.

In the realm of Agentic RS, approaches have transitioned from stateless prompting (Liu et al., 2023) to incorporating explicit, dynamic memory. While recent state-of-the-art agents like i^2*Agent* (Xu et al., 2025b) and simulation frameworks like *AgentCF* (Zhang et al., 2024b) employ self-reflection mechanisms to evolve user understanding over time, they remain paradigm-bound to isolated memory. Updates are strictly confined to the interacting user or item silos, failing to leverage the global collaborative signals that are vital for effective recommendation. MemRec addresses this critical gap by shifting the paradigm from isolated, self-reflective memory to a dynamic, collaborative memory graph. A more detailed review of related literature is provided in Appendix B.

5 Conclusion

Existing agent-based recommender systems typically rely on isolated user and item memories, which inherently restricts their ability to leverage broader community signals and accurately infer preferences, particularly for data-sparse users. To address this limitation, we identify *collaborative memory* as a crucial design paradigm and practically instantiate it through **MemRec**. By architecturally decoupling high-level reasoning (LLM_{Rec}) from memory management (LM_{Mem}), MemRec successfully resolves the dual challenges of naïve collaborative approaches: it mitigates cognitive overload via zero-shot LLM-guided context curation and circumvents prohibitive update bottlenecks via asynchronous graph propagation. Extensive experiments demonstrate that this paradigm not only delivers substantial ranking performance gains across four diverse benchmarks, but also establishes a superior Pareto frontier that balances reasoning quality, online inference cost, and deployment flexibility. Future work will explore scaling MemRec to web-scale graphs and investigating privacy-preserving federated memory updates.

6 Limitations

Despite its strong performance and flexible architecture, MemRec has limitations that warrant future investigation. Currently, our asynchronous collaborative propagation is restricted to immediate neighbors to manage computational overhead; extending this to multi-hop community updates without introducing noise requires more efficient selection mechanisms. Furthermore, our context curation rules are derived from static domain statistics generated offline, which may need online adaptation to maintain efficacy in highly dynamic environments (e.g., news). Finally, while memory operations can be successfully offloaded to local models, achieving ceiling reasoning performance still relies on powerful proprietary LLMs, motivating future work on fully open-source stacks.

7 Acknowledgements

This work is supported by NSFC/RGC Joint Research Scheme (N_HKBU214/24).

References

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

Harrison Chase. 2022. LangChain.

Weixin Chen, Li Chen, Yongxin Ni, and Yuhan Zhao. 2025a. Causality-inspired fair representation learning for multimodal recommendation. *ACM Transactions on Information Systems*, 43(6).

Weixin Chen, Li Chen, and Yuhan Zhao. 2025b. Investigating user-side fairness in outcome and process for multi-type sensitive attributes in recommendations. *ACM Transactions on Recommender Systems*, 4(2).

Weixin Chen, Mingkai He, Yongxin Ni, Weike Pan, Li Chen, and Zhong Ming. 2022. Global and personalized graphs for heterogeneous sequential recommendation by learning behavior transitions and user intentions. In *Proceedings of the 16th ACM Conference on Recommender Systems (RecSys'22)*, pages 268–277.

Weixin Chen, Yuhan Zhao, Li Chen, and Weike Pan. 2025c. Leave no one behind: Fairness-aware cross-domain recommender systems for non-overlapping users. In *Proceedings of the 19th ACM Conference on Recommender Systems (RecSys'25)*, pages 226–236.

Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. 2023. Chat-rec: Towards interactive and explainable llms-augmented recommender system. *arXiv preprint arXiv:2303.14524*.

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

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

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *4th International Conference on Learning Representations*.

Xu Huang, Jianxun Lian, Yuxuan Lei, Jing Yao, Defu Lian, and Xing Xie. 2025. Recommender ai agent: Integrating large language models for interactive recommendations. *ACM Transactions on Information Systems*, 43(4):1–33.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining*, pages 197–206.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model

serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Riwei Lai, Li Chen, Weixin Chen, and Rui Chen. 2026. Matryoshka representation learning for recommendation with layer- and hardness-adaptive negative sampling. *ACM Transactions on Intelligent Systems and Technology*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Junling Liu, Chao Liu, Peilin Zhou, Renjie Lv, Kang Zhou, and Yan Zhang. 2023. Is chatgpt a good recommender? a preliminary study. In *Proceedings of the CIKM 2023 Workshop on Recommendation with Generative Models*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

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

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

OpenAI. 2024. Hello gpt-4o. *OpenAI Blog*.

Charles Packer, Vivian Fang, Shishir_G Patil, Kevin Lin, Sarah Wooders, and Joseph_E Gonzalez. 2023. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. 2025. Zep: a temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP'19)*, pages 3980–3990.

Xubin Ren and Chao Huang. 2025. Easyrec: Simple yet effective language models for recommendation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 17728–17743.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295.

Yubo Shu, Hansu Gu, Peng Zhang, Haonan Zhang, Tun Lu, Dongsheng Li, and Ning Gu. 2024. Rah! recsys-assistant-human: A human-central recommendation framework with large language models. *IEEE Trans. Comput. Soc. Syst.*, 11(5):6759–6770.

Significant Gravitas. 2023. AutoGPT.

Jiakai Tang, Yujie Luo, Xunke Xi, Fei Sun, Xueyang Feng, Sunhao Dai, Chao Yi, Dian Chen, Zhujin Gao, Yang Li, and 1 others. 2025. Interactive recommendation agent with active user commands. *arXiv preprint arXiv:2509.21317*.

Zhen Tao, Xinke Jiang, Qingshuai Feng, Haoyu Zhang, Lun Du, Yuchen Fang, Hao Miao, Bangquan Xie, and Qingqiang Sun. 2026. Task-aware retrieval augmentation for dynamic recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 15779–15787.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adrià Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Mengting Wan, Rishabh Misra, Ndapandula Nakashole, and Julian McAuley. 2019. Fine-grained spoiler detection from large-scale review corpora. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2605–2610.

Jianing Wang, Junda Wu, Yupeng Hou, Yao Liu, Ming Gao, and Julian McAuley. 2024a. InstructGraph: Boosting large language models via graph-centric instruction tuning and preference alignment. In *Findings of the Association for Computational Linguistics*, pages 13492–13510.

Lei Wang, Jingsen Zhang, Hao Yang, Zhi-Yuan Chen, Jiakai Tang, Zeyu Zhang, Xu Chen, Yankai Lin, Hao Sun, Ruihua Song, and 1 others. 2025. User behavior simulation with large language model-based agents.

ACM Transactions on Information Systems, 43(2):1–37.

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

Yancheng Wang, Ziyan Jiang, Zheng Chen, Fan Yang, Yingxue Zhou, Eunah Cho, Xing Fan, Yanbin Lu, Xiaojiang Huang, and Yingzhen Yang. 2024b. Recmind: Large language model powered agent for recommendation. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4351–4364.

Zhefan Wang, Yuanqing Yu, Wendi Zheng, Weizhi Ma, and Min Zhang. 2024c. 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.

Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Llmrec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM international conference on web search and data mining*, pages 806–815.

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

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, and 1 others. 2025. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025a. A-mem: Agentic memory for llm agents. In *Advances in Neural Information Processing Systems*.

Wujiang Xu, Yunxiao Shi, Zujie Liang, Xuying Ning, Kai Mei, Kun Wang, Xi Zhu, Min Xu, and Yongfeng Zhang. 2025b. iAgent: LLM agent as a shield between user and recommender systems. In *Findings of the Association for Computational Linguistics, ACL 2025*, pages 18056–18084.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Kristian Kersting, Jeff Z Pan, Hinrich Schütze, and 1 others. 2025. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*.

Zihe Ye, Jingyuan Huang, Weixin Chen, and Yongfeng Zhang. 2026. H-mem: Hybrid multi-dimensional memory management for long-context conversational agents. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (EACL'26)*, pages 7756–7775.

An Zhang, Yuxin Chen, Leheng Sheng, Xiang Wang, and Tat-Seng Chua. 2024a. On generative agents in recommendation. In *Proceedings of the 47th international ACM SIGIR conference on research and development in Information Retrieval*, pages 1807–1817.

Junjie Zhang, Yupeng Hou, Ruobing Xie, Wenqi Sun, Julian McAuley, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2024b. Agentcf: Collaborative learning with autonomous language agents for recommender systems. In *Proceedings of the ACM Web Conference 2024*, pages 3679–3689.

Yu Zhang, Shutong Qiao, Jiaqi Zhang, Tzu-Heng Lin, Chen Gao, and Yong Li. 2025. A survey of large language model empowered agents for recommendation and search: Towards next-generation information retrieval. *arXiv preprint arXiv:2503.05659*.

Yuhan Zhao, Rui Chen, Qilong Han, Hongtao Song, and Li Chen. 2024a. Unlocking the hidden treasures: Enhancing recommendations with unlabeled data. In *Proceedings of the 18th ACM Conference on Recommender Systems (RecSys)*, page 247–256.

Yuhan Zhao, Rui Chen, Qilong Han, Hongtao Song, and Li Chen. 2025. Unlocking the unlabeled data: Enhancing recommendations with neutral samples and uncertainty. *ACM Transactions on Recommender Systems (TORS)*.

Yuhan Zhao, Weixin Chen, Li Chen, and Weike Pan. 2026. The double-edged sword of knowledge transfer: Diagnosing and curing fairness pathologies in cross-domain recommendation. In *Proceedings of the ACM Web Conference 2026 (WWW '26)*.

Yuyue Zhao, Jiancan Wu, Xiang Wang, Wei Tang, Dingxian Wang, and Maarten De Rijke. 2024b. Let me do it for you: Towards llm empowered recommendation via tool learning. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1796–1806.

Xi Zhu, Haochen Xue, Ziwei Zhao, Wujiang Xu, Jingyuan Huang, Minghao Guo, Qifan Wang, Kaixiong Zhou, Imran Razzak, and Yongfeng Zhang. 2025. Llm as gnn: Graph vocabulary learning for text-attributed graph foundation models. *arXiv preprint arXiv:2503.03313*.

Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University.

A Experimental Setup and Implementation Details

A.1 Dataset Details

We utilize four datasets widely used in recommendation research, encompassing diverse domains such as e-commerce, social reading, entertainment, and local services. As mentioned in Section 3, we adopt the versions of these datasets augmented with natural language user instructions from **InstructRec** (Xu et al., 2025b). The original data sources and their detailed descriptions are provided below:

Books Derived from the **Amazon review dataset**[*] (Ni et al., 2019), this subset focuses on book recommendations. It is characterized by incredibly sparse interactions and a vast item space. User preferences in this domain are typically stable and highly content-driven, focusing on specific genres, authors, or themes.

Goodreads Collected from the **Goodreads** social book cataloging website[†] (Wan et al., 2019), this dataset is notably dense compared to others. It features strong community interactions and rich metadata about books, including series information. Users on Goodreads often exhibit series-aware reading behaviors and are influenced by social signals.

MovieTV Also originating from the **Amazon review dataset** (Ni et al., 2019), this dataset covers movies and TV shows. The domain is marked by volatile user preferences often influenced by immediate context or trending content. While metadata like genre and cast are important, item recency frequently plays a critical role in user decision-making.

Yelp Sourced from the **Yelp Dataset**[‡], this dataset consists of reviews for local businesses like restaurants and services. It is characterized by strong categorical constraints (e.g., cuisine type) and the critical importance of attributes like price range and location. User preferences here are often highly context-dependent.

[*] https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/

[†] https://cseweb.ucsd.edu/~jmcauley/datasets/goodreads.html

[‡] https://www.kaggle.com/datasets/yelp-dataset/yelp-dataset

A.2 Baseline Model Details

This appendix provides detailed descriptions of the baseline models used in our comparative evaluation. Following the categorization in the main text, we group these baselines based on their underlying memory paradigms into two major categories: traditional pre-LLM methods using latent embeddings, and memory-based approaches developed in the post-AgentRS era using semantic memory.

A.2.1 Traditional Pre-LLM Methods (Latent Embeddings)

These models represent the conventional paradigm where historical information is encoded and preserved using dense latent vectors, without explicit semantic memory structures for reasoning agents.

- **LightGCN** (He et al., 2020): A state-of-the-art graph collaborative filtering model that simplifies the Graph Convolutional Network (GCN) design by removing feature transformation and nonlinear activation. It learns user and item embeddings by linearly propagating them on the user-item interaction graph, capturing high-order collaborative signals through structural connections.

- **SASRec** (Kang and McAuley, 2018): A leading sequential recommendation model based on the self-attention mechanism. It models the entire user sequence to capture long-term semantics and dynamic dependencies, using an attention mechanism to selectively focus on relevant items in the history for making predictions.

- **P5** (Geng et al., 2022): A unified framework that formulates various recommendation tasks as sequence-to-sequence language modeling problems. It utilizes a pre-trained T5 backbone and represents users and items as sequence tokens (IDs) within personalized prompts. While LLM-based, the original P5 relies on pre-trained knowledge related to these IDs and does not incorporate an evolving, descriptive memory component.

A.2.2 Memory-based Approaches (Post-AgentRS Era)

This category encompasses approaches developed in the era following AgentRS, utilizing LLMs with varying degrees of semantic memory capabilities.

(1) Models with No Explicit Memory These models operate by directly processing raw interaction histories without maintaining a persistent, structured semantic memory store.

- **Vanilla LLM (Zero-Shot Prompting)** (Liu et al., 2023): This baseline represents the direct application of a powerful instruction-tuned LLM (e.g., GPT-4o-mini) via API calls. For each prediction, the user's entire sequence of historical interactions is converted into a natural language string and fed into the LLM as a static context prompt. The model performs zero-shot selection from candidate items based solely on this provided raw history, serving as a baseline to measure the LLM's inherent capabilities independent of designed memory architectures.

(2) Static Memory Agents These agents utilize descriptive semantic information about users and items, but this "memory" remains fixed as a static context during inference and does not evolve.

- **iAgent** (Xu et al., 2025b): An LLM-based autonomous agent designed for recommendation. It employs a static profile for each user, constructed from their historical interactions and available descriptive data. This fixed profile is fed into the LLM as context to generate recommendations. The key characteristic is that its understanding of the user does not adapt over time after initial construction.

(3) Dynamic Memory Agents (Isolated Updates) These agents possess a dynamic memory mechanism, allowing them to reflect on interactions and update their understanding. However, these updates are isolated to the individual agent and do not propagate collaboratively.

- **i^{2}Agent** (Xu et al., 2025b): An extension of iAgent that introduces a "reflection" mechanism. After recommendations, the agent can reflect on user feedback to refine its internal state or strategy for future interactions. While dynamic, these reflections are confined to the individual agent's experience with a specific user.

- **AgentCF** (Zhang et al., 2024b): An agent-based collaborative filtering framework that simulates user-item interactions. Agents representing users and items can autonomously interact, learn from these interactions, and update their own preferences or characteristics. The memory update is dynamic but remains localized to the individual agents involved in the direct interaction.

- **RecBot** (Tang et al., 2025): A conversational recommender system that uses an LLM to engage with users. It maintains a dynamic dialogue history and can update its understanding of user preferences based on the ongoing conversation. This dynamic memory allows for multi-turn interactions but is limited to the context of the current user session.

A.3 Implementation Details

Model Deployment We utilize a diverse set of models across different configurations. For proprietary models, we access `gpt-4o-mini` (Standard config.) and `gpt-4o` (Ceiling config.) via the Microsoft Azure OpenAI Service, using API version `2024-08-01-preview`. For the Cloud-OSS configuration, we employ the large-scale open-source **`gpt-oss-120b`** model via Azure Serverless APIs. For local open-source ablations (`Qwen-2.5-7B-Instruct`, `Meta-Llama-3-8B-Instruct`), we deploy them locally using the **vLLM** library (Kwon et al., 2023) for optimized high-throughput inference in **FP16 (half-precision)** mode. For the Vector configuration, we utilize the `all-MiniLM-L6-v2` Sentence Transformer (Reimers and Gurevych, 2019). We used Gemini to polish sentences and improve language flow only. The core ideas, research, and results are fully our own work.

Hardware Environment All local experiments (specifically for Local-Qwen and Local-Llama) were conducted on a workstation equipped with a single **NVIDIA RTX A5000 GPU (24GB VRAM)**. While our local setup exhibits higher latency compared to optimized cloud APIs (as detailed in Table 5), deploying 7B models on enterprise-grade inference hardware would likely yield competitive speeds.

Hyperparameters We set the neighbor count $k = 16$ and the number of synthesis facets $N_f = 7$. The max token budget for context retrieval is set to $\tau = 1800$. We use a temperature of 0.0 for all LLM calls to ensure reproducibility.

Neighbor Representation Strategy To efficiently manage the strict context token budget ($\tau = 1800$) while maintaining broad neighbor coverage ($k = 16$) during Stage-R, we implemented a practical tiered representation strategy. While item

neighbors utilize their truncated semantic memory (initialized by metadata descriptions), user neighbors are represented by their sequence of recent interactions (e.g., titles of the last three acted items). This acts as a dense, token-efficient proxy for immediate interests, enabling the inclusion of diverse collaborative signals within limited context windows without incurring prohibitive latency.

A.4 LLM-Generated Curation Rules

To efficiently curate the collaborative subgraph in Stage-R, LM_{Mem} generates domain-specific heuristic rules in a zero-shot manner based on domain statistics.

Figure 16 presents the generated rules for the **Books** and **Goodreads** datasets, which emphasize content similarity (genre/theme) and social signals, respectively. Figure 17 shows the rules for the **MovieTV** and **Yelp** datasets, where recency and categorical constraints (e.g., cuisine/price) play a more dominant role. These interpretable rules act as a fast, domain-adaptive filter before memory synthesis. In our experiments, to ensure efficiency, we relied on statistical signals (e.g., recency, co-interactions) and used constant similarity scores instead of performing computationally expensive semantic calculations.

A.5 Cost Estimation Methodology

The cost estimates presented in Section 3.4 and Figure 7 are based on public cloud pricing for the Azure OpenAI Service Standard tier as of December 2025.[§]

- **High-Tier Model (gpt-4o):** $2.50 per 1M input / $10.00 per 1M output.
- **Low-Tier Models (gpt-4o-mini & gpt-oss-120b):** $0.15 per 1M input / $0.60 per 1M output.
- **Local Deployment:** Negligible marginal cost.

B Detailed Related Works

B.1 Memory Architectures for LLM Agents

Building autonomous agents capable of long-horizon tasks requires overcoming the inherent constraints of LLM context windows (Liu et al., 2024) and ensuring long-term knowledge retention. Early solutions combined LLMs with external vector databases (Johnson et al., 2019) to create Retrieval-Augmented Generation (RAG) pipelines (Lewis et al., 2020). Recent advances like Graph RAG (Edge et al., 2024) further demonstrate the value of structuring retrieved context into knowledge graphs for complex reasoning. Building on this, dedicated memory systems emerged. *MemGPT* (Packer et al., 2023) introduced an OS-inspired virtual context management system, while *Zep* (Rasmussen et al., 2025) structures memory into temporal knowledge graphs. Seminal works like *Generative Agents* (Park et al., 2023) demonstrated how synthesizing high-level reflections from memory streams could drive believable agent behavior.

Complementarily, research explores learning-based memory policies (Xu et al., 2025a; Yan et al., 2025), where a dedicated manager learns to optimize storage and retrieval for multi-hop reasoning. General agent frameworks like LangChain (Chase, 2022) and AutoGPT (Significant Gravitas, 2023) have integrated modular components, such as tool use capabilities (Zhao et al., 2024b) and memory systems, to support complex workflows. Recent advancements also explore multi-dimensional memory management to handle long-context conversations (Ye et al., 2026). While sophisticated, these systems are designed for general factual or conversational contexts, fundamentally neglecting the specialized, high-order connectivity required for graph-based collaborative domains. MemRec adopts the core principle of a decoupled memory manager (LM_{Mem}) but augments it with explicit graph context structure.

B.2 Memory in Agentic RS

The integration of memory into recommender systems has evolved from latent states in sequential models (Chen et al., 2022; Hidasi et al., 2016; Kang and McAuley, 2018) to explicit, dynamic structures managed by LLM agents. Early applications of LLMs in recommendation explored stateless approaches, leveraging prompting or efficient architectures for ranking without maintaining persistent user states (Liu et al., 2023; Lyu et al., 2024; Geng et al., 2022; Ren and Huang, 2025; Bao et al., 2023). Subsequent works, such as *Chat-REC* (Gao et al., 2023) and *iAgent* (Xu et al., 2025b), introduced explicit memory in the form of static user profiles or retrieved historical summaries, similar to standard RAG approaches. While enabling natural language interaction, these systems cannot adapt to evolving user interests based on real-time feedback.

[§]Official pricing source: https://azure.microsoft.com/en-us/pricing/details/cognitive-services/openai-service/. Pricing is subject to change.

To address plasticity, recent works introduce dynamic memory mechanisms, often incorporating planning or tool-using capabilities (Wang et al., 2024b; Huang et al., 2025; Wang et al., 2024c; Shu et al., 2024). Systems like *i*^{2}*Agent* (Xu et al., 2025b) and *RecBot* (Tang et al., 2025) employ a "self-reflection" mechanism, where the agent updates its own memory after an interaction. Similarly, simulation frameworks like *AgentCF* (Zhang et al., 2024b), *Agent4Rec* (Zhang et al., 2024a), and *RecAgent* (Wang et al., 2025) model users and items as agents with evolving memories to study emergent behaviors and feedback loops.

Crucially, these dynamic approaches remain bound to isolated, self-reflective memory. The memory update is confined to the interacting user or item. While some recent works attempt to combine LLMs with graph structures for recommendation, they typically use LLMs for feature enhancement (Wei et al., 2024), structure refinement (Wang et al., 2024a), graph vocabulary learning (Zhu et al., 2025), or task-aware retrieval augmentation on dynamic graphs (Tao et al., 2026), rather than for managing collaborative memory propagation in an agentic manner.

Furthermore, our robustness analysis on niche users (Section 3.6) connects agentic recommender systems to the broader challenge of user-side fairness and long-tail recommendation (Chen et al., 2025b,c). While prior pre-LLM works typically address data sparsity for non-overlapping or long-tail users through cross-domain knowledge transfer (Zhao et al., 2026), fairness-aware representation learning (Chen et al., 2025a), or mining neutral signals and uncertainty from unlabeled data (Zhao et al., 2024a, 2025), `MemRec` offers a novel agentic perspective. By using LLM-curated collaborative graphs, it democratizes recommendation quality, allowing users with sparse histories to fully benefit from community-level insights.

C Extended Efficiency and Modularity Analysis

This appendix provides the detailed quantitative data supporting the analysis in Section 3.4 of the main text. Table 5 presents a comprehensive breakdown of different architectural configurations of `MemRec`. It reports performance metrics (H@1, N@5), alongside efficiency metrics including average experimental latency per user session, average total token consumption, and a qualitative cost estimate.

Table 5 reports latencies measured in a sequential execution pipeline designed for rigorous benchmarking. In real-world deployments, user-perceived latency can be significantly reduced through standard engineered optimizations, such as aggressively *caching* synthesized collaborative contexts (Stage-R outputs) for popular items to bypass redundant computations, and employing *token streaming* for the final LLM output (Stage-ReRank) to drastically reduce the time-to-first-byte (TTFB) and improve perceived responsiveness.

The reported latencies require careful interpretation due to the nature of cloud API-based experimentation. Firstly, our experiments utilized standard, non-real-time API endpoints for all LLMs. Secondly, we observe a counter-intuitive result where the latency of the **Ceiling** configuration (`gpt-4o`) is lower than the **Standard** configuration (`4o-mini`), despite the former being a significantly larger model. We attribute this discrepancy to opaque operational factors related to the cloud provider's infrastructure, such as differential load balancing, resource allocation priorities, or transient network conditions at the time of measurement, rather than intrinsic differences in model inference speed. This highlights the variability inherent in benchmarking against black-box APIs.

D Additional Experimental Results

D.1 Results for Larger Candidate Set

To demonstrate robustness, we present results for a larger candidate set ($N = 20$) in Table 6 and Table 7.

D.2 Rationale Quality Analysis

To assess the qualitative impact of collaborative memory on reasoning output, we conducted a human-aligned evaluation using GPT-4o as an automated judge on the `books` subset. The detailed evaluation protocol, including the model mapping and the exact prompts used for the GPT-4o judge, is described in Appendix F.6.

We compared rationales generated by three model configurations: **Base LLM** (Vanilla, no memory), **MemRec w/o Collab** (static user history only), and **MemRec (Full)** (collaborative memory). GPT-4o rated each rationale on a Likert scale (1-5) across three distinct dimensions: *Specificity* (richness of item details), *Relevance* (connection to user interests), and *Factuality* (accuracy of claims).

Table 5: Comprehensive architectural analysis across different model configurations. **Latency**: average online time measured in our experimental setup (sequential execution). **Tokens/U (On / Total)**: average tokens consumed per user during online inference versus the complete session (including asynchronous offline memory maintenance). **Cost**: qualitative estimate of online serving efficiency.

Configuration	Model Selection		Performance		Efficiency Metrics			Key Takeaway
	LLM_{Rec}	LM_{Mem}	H@1	N@5	Latency	Tokens/U (On / Total)	Cost (Online)	
Vanilla LLM	`4o-mini`	-	0.330	0.524	**∼5.1s**	**∼1.2k / ∼1.2k**	**Lowest**	*No memory, fast but poor precision*
Single-User Mem	`4o-mini`	`4o-mini (Iso.)`	0.475	0.631	∼10.0s	∼1.4k / ∼5.5k	Low	*Memory overhead without collaboration*
Standard	`4o-mini`	`4o-mini`	0.524	0.663	∼16.5s	**∼1.5k / ∼6.8k**	Low	*Collaborative gains over single-user*
Vector	*Vector*	`4o-mini`	0.209	0.387	∼5.3s	- / ∼6.3k	Lowest	**Ultra-fast, pluggable reranker**
Local-Qwen	`4o-mini`	`Qwen-2.5-7B`	0.470	0.627	∼34.0s‡	∼1.6k / ∼7.0k	Low⋆	*Best on-premise performance*
Local-Llama	`4o-mini`	`Llama-3-8B`	0.360	0.550	∼34.4s‡	∼1.4k / ∼6.2k	Low⋆	*Alternative local option*
Ceiling	**`gpt-4o`**	`4o-mini`	**0.580**	**0.722**	∼10.4s	∼1.5k / ∼6.9k	High	*Peak performance, high serving cost*
Cloud-OSS	`4o-mini`	**`OSS-120B`**	0.561	0.699	∼11.8s	∼1.8k / ∼10.0k	Low	**Near-ceiling results w/ moderate cost**

⋆ The offline memory maintenance incurs zero API cost due to local hosting, leaving only the minimal online LLM_{Rec} cost. ‡ Local latency is hardware-dependent (measured sequentially on single NVIDIA A5000 GPU) and not directly comparable to highly optimized cloud APIs.

Table 6: Main results for **Books** and **Goodreads** (N=20). Notation follows Table 2; all improvements are significant ($p < 0.05$).

Model	Books					Goodreads				
	H@1	H@5	N@5	H@10	N@10	H@1	H@5	N@5	H@10	N@10
LightGCN	0.1276	0.2622	0.1947	0.5512	0.2854	0.1617	0.5566	0.3588	**0.8177**	0.4434
SASRec	0.0453	0.2353	0.1378	0.4896	0.2188	0.0699	0.3053	0.1859	0.5435	0.2621
P5	0.1648	0.3051	0.2331	0.5216	0.3022	0.1038	0.2611	0.1798	0.5041	0.2572
Vanilla LLM	0.1730	0.4155	0.2955	0.6129	0.3599	0.0999	0.3245	0.2211	0.6712	0.3291
iAgent	0.3258	0.5069	0.4173	0.6209	0.4537	0.1621	0.4107	0.2871	0.6035	0.3490
RecBot	0.2471	0.4030	0.3247	0.5768	0.3801	0.1234	0.3364	0.2289	0.5583	0.2999
AgentCF	0.2470	0.5481	0.4026	0.7250	0.4594	0.1875	0.5427	0.3692	0.7805	0.4462
i^2Agent	0.3712	0.5947	0.4874	0.7387	0.5336	0.2065	0.5350	0.3767	0.7428	0.4435
MemRec (Ours)	**0.4236**	**0.6351**	**0.5332**	**0.7667**	**0.5756**	**0.2657**	**0.6062**	**0.4434**	0.7948	**0.5042**
Improv.	*+14.12%*	*+6.79%*	*+9.40%*	*+3.79%*	*+7.87%*	*+28.67%*	*+8.91%*	*+17.71%*	-	*+13.69%*

Figure 10 presents the average scores with 95% confidence intervals and significance annotations (paired t-test). The results reveal distinct trends regarding the role of different memory types:

- **Specificity exhibits a clear step-wise improvement.** Adding user history (w/o Collab) significantly improves specificity over the Base LLM ($p < 0.001$), likely by grounding the generation in the user's genre. Crucially, adding collaborative memory (Full) yields a further significant boost ($p < 0.001$). This confirms that neighbor signals provide the rich, specific item details needed for high-quality justification that user history alone cannot provide.

- **Collaborative signals are key to perceived Relevance.** Surprisingly, user history alone (*w/o Collab*) did not yield a statistically significant improvement over the Base LLM in perceived relevance ($p > 0.05$). However, the full collaborative context (*MemRec*) achieved a substantial and significant increase ($p < 0.001$). This suggests that simply mentioning user history is insufficient; grounding recommendations in peer experiences makes them feel significantly more relevant and convincing to the judge.

- **MemRec maintains high Factuality.** While all models maintain high factuality scores (>4.0), `MemRec` achieves a slight but statistically significant improvement over the others ($p < 0.001$ vs w/o Collab), indicating that grounding generation based on real collaborative memories helps reduce hallucinations compared to ungrounded generation.

D.3 Latency and Token Breakdown Analysis

A critical aspect of `MemRec`'s cost-efficiency lies in how it utilizes tokens relative to standard commercial pricing structures.

Most commercial LLM providers adopt an **asymmetric pricing model**, where output (generated) tokens are significantly more expensive than input (context) tokens (typically a 3x to 4x ratio, see Appendix A.5). `MemRec`'s architecture is inherently

Table 7: Main results for **MovieTV** and **Yelp** (N=20). Notation follows Table 2; all improvements are significant ($p < 0.05$).

	MovieTV					Yelp				
Model	**H@1**	**H@5**	**N@5**	**H@10**	**N@10**	**H@1**	**H@5**	**N@5**	**H@10**	**N@10**
LightGCN	0.2657	0.5330	0.4064	0.6815	0.4537	0.2549	0.5437	0.4046	0.7481	0.4692
SASRec	0.2923	0.5128	0.4092	0.6311	0.4470	0.1678	0.3993	0.2879	0.5590	0.3389
P5	0.1113	0.2769	0.1902	0.5137	0.2657	0.0634	0.2492	0.1537	0.5051	0.2354
Vanilla LLM	0.2379	0.5003	0.3648	0.7261	0.4406	0.0254	0.1461	0.0831	0.5128	0.2010
iAgent	0.3236	0.5362	0.4331	0.6762	0.4778	0.3236	0.5658	0.4499	0.6597	0.4799
RecBot	0.2420	0.4201	0.3316	0.6015	0.3895	0.1949	0.3742	0.2851	0.5519	0.3414
AgentCF	0.2870	0.6288	0.4648	0.7616	0.5077	0.1115	0.3897	0.2512	0.6372	0.3309
i^{2}Agent	0.3822	0.6367	0.5178	0.7735	0.5617	0.3287	0.6083	0.4744	0.7562	0.5216
MemRec (Ours)	**0.4750**	**0.7543**	**0.6212**	**0.8752**	**0.6606**	**0.3620**	**0.6329**	**0.5035**	**0.7708**	**0.5478**
Improv.	*+24.28%*	*+18.47%*	*+19.97%*	*+13.15%*	*+17.61%*	*+10.13%*	*+4.04%*	*+6.13%*	*+1.93%*	*+5.02%*

designed to exploit this structure.

As shown in Table 8, our key memory operations, Stage-R (Synthesis) and Stage-W (Propagation), are heavily *input-biased*. They digest large volumes of raw collaborative context (cheap input) to produce highly condensed, structured insights (expensive output). For example, in the Standard configuration, input tokens account for **nearly 80%** of the total usage. This makes the effective cost of running `MemRec` significantly lower than a naive estimation based on total token counts would suggest.

D.4 Hyperparameter Analysis

We analyze the sensitivity of `MemRec` to its two main hyperparameters on the `books-1k` subset: the number of neighbors k (Stage-R Curation) and the number of facets N_f (Stage-R Synthesis). While the primary metric H@1 is shown in Figure 11 in the main text, Figure 12 presents the heatmaps for additional metrics (H@3, H@5, NDCG@3, NDCG@5). The trends across these metrics are consistent with H@1, confirming the robustness of the optimal hyperparameter region.

D.5 Full Metrics for Architectural Analysis

Table 9 presents the comprehensive performance metrics covering Hit Rate (H@K) and NDCG (N@K) at varying cutoff points ($K = \{1, 3, 5\}$) across three datasets. These detailed results substantiate the findings discussed in Section 3.3, demonstrating the consistent superiority of MemRec over both the Vanilla baseline and the Naive Agent across diverse ranking depths.

E Qualitative Case Study: A Complete Collaborative Journey

This case study illustrates the complete workflow of `MemRec` for **User 2057, a fan of Young Adult (YA) fantasy and graphic novels.** Figure 13 demonstrates how collaborative signals are synthesized (Stage-R), leveraged for reasoning (Stage-ReRank), and propagated back into the memory graph (Stage-W). For brevity and clarity, we display only representative subsets of retrieved neighbors (Stage-R) and propagated updates (Stage-W) to illustrate the process. We use **blue text** to highlight collaborative signals and **orange text** to indicate user-specific signals.

F Prompt Templates and Contexts

This appendix provides the complete prompt templates governing the memory management (LM_{Mem}) and reasoning (LLM_{Rec}) agents, ensuring full transparency and reproducibility of our framework.

F.1 Meta-Prompt Template

To enable zero-shot domain adaptation for Stage-R neighbor pruning, we employ a Meta-Prompt Template (Figure 14). This high-level instruction inputs structured domain descriptions (metadata, statistics) to guide LM_{Mem} in synthesizing interpretable, domain-specific heuristic rules offline.

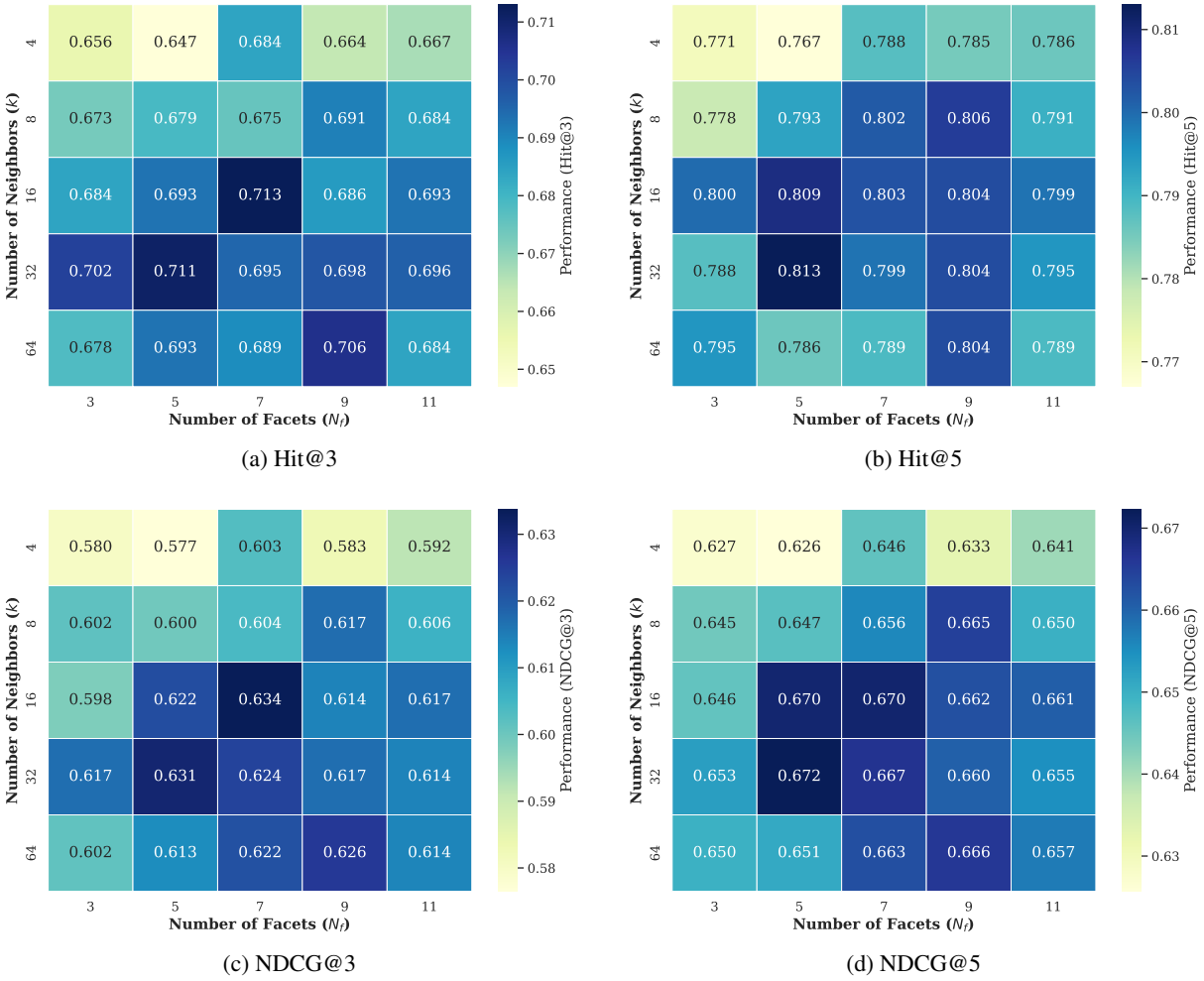
F.2 Domain-Specific Prompt Contexts

Figure 15 presents the specific Domain Context Blocks injected into the Meta-Prompt for our four evaluated datasets. These blocks outline key metadata fields, interaction modes, and unique domain characteristics to generate effective, tailored rules offline.

Table 8: Detailed breakdown of average Input vs. Output token consumption per stage per user (measured on books for the Standard configuration). The high Input/Output ratio in memory stages exploits the asymmetric pricing of commercial LLMs.

Pipeline Stage	Primary Role	Avg Input	Avg Output	Total	I/O Ratio
Stage-R (Synthesis)	LM_{Mem}	∼1,400	∼460	∼1,860	3.0 : 1
Stage-ReRank	LLM_{Rec}	∼1,100	∼450†	∼1,550	2.4 : 1
Stage-W (Async)	LM_{Mem}	∼1,600	∼315	∼1,915	5.1 : 1
Total per User	-	**∼5,100**	**∼1,300**	**∼6,400**	**3.9 : 1**

† A significant portion of Stage-ReRank output is dedicated to generating interpretable rationales, adding user value.



(a) Hit@3 (b) Hit@5

(c) NDCG@3 (d) NDCG@5

Figure 12: Hyperparameter sensitivity analysis for additional metrics on the books subset. Trends are consistent with H@1 shown in the main text.

F.3 Stage-R Memory Synthesis Prompt

The **Stage-R Synthesis Prompt** (Figure 18) enables LM_{Mem} to distill raw neighbor memories into high-signal "memory facets." It instructs the LLM to identify common themes from the user's and neighbors' memories, synthesizing them into structured facets with confidence scores to form the collaborative context (M_{collab}) for downstream reasoning.

F.4 Stage-ReRank Scoring Prompt

The final ranking decision in Stage-ReRank is performed by the reasoning agent (LLM_{Rec}) using the **Stage-ReRank Scoring Prompt**. As displayed in Figure 19, this prompt is designed to ground the LLM's reasoning in the provided context. It integrates three key inputs: the user's natural language instruction, specific details of the candidate item being evaluated, and, crucially, the synthesized col-

Table 9: Comprehensive performance comparison for the architectural analysis across datasets. **Vanilla LLM** serves as the non-memory baseline. **Naive Agent** utilizes raw, uncurated collaborative context. **MemRec** employs the proposed decoupled architecture. The best performance in each metric per dataset is marked in **bold**.

		Hit Rate @ K			NDCG @ K	
Dataset	**Model**	**H@1**	**H@3**	**H@5**	**N@3**	**N@5**
Books	Vanilla LLM	0.330	0.549	0.724	0.450	0.524
	Naive Agent	0.390	0.638	0.760	0.533	0.583
	MemRec	**0.524**	**0.700**	**0.795**	**0.625**	**0.663**
Yelp	Vanilla LLM	0.176	0.522	0.684	0.370	0.437
	Naive Agent	0.242	0.486	0.639	0.383	0.446
	MemRec	**0.489**	**0.686**	**0.792**	**0.604**	**0.648**
MovieTV	Vanilla LLM	0.407	0.774	0.861	0.610	0.646
	Naive Agent	0.418	0.605	0.795	0.523	0.602
	MemRec	**0.563**	**0.791**	**0.894**	**0.696**	**0.738**

laborative memory facets (M_{collab}). The prompt instructs the LLM to synthesize these signals to generate a calibrated relevance score (between 0 and 1) along with a concise, natural language rationale justifying the score based on collaborative evidence.

F.5 Stage-W Propagation Prompts

Following a user interaction (e.g., clicking an item), the **Stage-W Propagation Prompts** are employed asynchronously by LM_{Mem} to evolve the semantic graph. While efficiently implemented as a single LLM call to minimize latency, we conceptually decompose this process into two distinct logical operations, as illustrated in Figure 20:

1. **User/Item Memory Update**: Updating the narrative memories of the interacting user and item nodes to reflect the new interaction.
2. **Collaborative Propagation**: Identifying relevant neighboring nodes and propagating insights from the interaction to update their memories, thereby enriching the graph's collaborative signals for future retrievals.

Figure 20 shows the comprehensive prompt that handles these updates concurrently.

F.6 Rationale Quality Evaluation Protocol

We evaluate the quality of the generated explanations (rationales) using GPT-4o as an automated judge. The judge is instructed to score rationales from three different models independently based on Specificity, Relevance, and Factuality on a 1-5 Likert scale.

The three models evaluated are mapped as follows:

- **Model A**: Base LLM (Vanilla LLM)
- **Model B**: MemRec w/o Collab (Isolated Memory)
- **Model C**: `MemRec` (Full)

The system prompt providing the evaluation criteria and scoring rubric, along with the user prompt template used for the GPT-4o judge, are presented in Figure 21. We use 'gpt-4o' with a temperature of 0 to ensure consistent evaluation.

G Methodology Analysis

G.1 Comparison of Curation Approaches

To justify our design choice for the neighbor pruning step in Stage-R, Table 10 presents a comparative analysis of three distinct approaches: traditional rule-based heuristics (e.g., Random Walk (Perozzi et al., 2014)), our proposed LLM-generated rules, and a fully learned neural scorer (f_θ, e.g., GNN-based attention weights (Veličković et al., 2018)). Our **LLM-generated rules** approach occupies a sweet spot. Unlike traditional heuristics, it is *domain-adaptive* (rules are tailored to specific dataset statistics) and offers better estimated performance. Unlike a learned scorer, it requires *no training data*, maintains *high interpretability* (rules are human-readable), and ensures extremely low online inference cost. This balance makes it a practical and robust solution for efficiently curating high-signal subgraphs.

Table 10: Design comparison of neighbor curation approaches. Our approach uniquely balances the interpretability and efficiency of rule-based methods with the domain-adaptivity of learning-based methods, in a zero-shot manner.

	Rule-based (Traditional heuristic)	**LLM-generated rules** (**Our approach**)	**Learned** f_θ (Neural scorer)
Training required?	No	**No**	Yes
Domain-adaptive?	Limited	**Yes**	**Yes**
Interpretable?	**High**	**High**	Low
Inference cost	**<1ms**	**<1ms**[a]	**<1ms**
Performance (est.)	Moderate	Good	**Best**
Generalization	**High**	**High**	Low
Implementation	**Simple**	Moderate	Complex

[a] The cost reflects applying pre-generated rules online. The one-time offline rule generation by the LLM is negligible per inference.

Table 11: Neighbor Filtering Precision (evaluated on a 1,000-user test subset with $k = 16$). The proposed LLM curation drastically reduces noisy item neighbors compared to a generic heuristic rule.

Metric	**Generic Rule (w/o LLM)**	**LLM Curation (Ours)**	Δ
Total Neighbors Processed	15,782	15,782	-
Irrelevant Items Retained	1,188	311	**-73.8%** $\downarrow$
Irrelevant Users Retained	1,754	1,866	+6.4% $\uparrow$
Overall Filtering Precision	0.813	**0.861**	**+5.9%** $\uparrow$

G.2 Quantitative Analysis of LLM Curation

To further demonstrate the superiority of the LLM-guided context curation over generic heuristics, we conducted a quantitative precision analysis. Recognizing the absence of an absolute "gold standard" for neighbor relevance in this unsupervised context, we established an intuitive statistical proxy to identify highly probable irrelevant neighbors across a 1,000-user test subset (processing 15,782 neighbors in total). Specifically, we flagged:

- **Irrelevant Items (Outdated):** Items with a recency score < 0.3 (corresponding to interactions older than approximately 255 days). An item a user interacted with nearly a year ago typically no longer reflects their current, actively evolving preferences.
- **Irrelevant Users (Low Similarity):** Users with an interaction overlap ratio < 0.1. For instance, a peer who shares almost zero common interaction history with the target user provides exceptionally weak traditional collaborative signals.

We compared our zero-shot LLM curation against a "Generic Rule" variant (w/o LLM Curation), which utilizes a standard, non-adaptive heuristic applying equal weights to all available normalized features (e.g., recency, co-interaction count, metadata overlap) without domain-specific semantic adjustments.

As shown in Table 11, our zero-shot LLM curation provides a significant advantage, particularly in filtering noisy items. The LLM curation effectively reduced irrelevant item neighbors by 73.8%, acting as a critical noise buffer for the downstream reasoning agent. This capability to identify and filter noisy interactions shares conceptual goals with hardness-adaptive negative sampling in traditional representation learning (Lai et al., 2026). However, rather than relying on complex training objectives, `MemRec` achieves robust noise reduction through zero-shot semantic curation. Interestingly, the LLM curation retained slightly more users (+6.4%) with low ID-overlap. We view this as a feature rather than a flaw: the LLM identifies semantic similarity from memory narratives (e.g., two users who both enjoy "dystopian YA novels" even if they have not clicked the exact same items), thereby prioritizing diverse but conceptually relevant peers over rigid statistical overlap.

Stage-R: Collaborative Synthesis (LM_{Mem})

Task: Synthesize collaborative context for User 2057 based on retrieved neighbors (showing 3 of $k = 16$).
Selected Neighbor Memories (Input Noise):
- **Item Neighbor (*Pastworld*):** "...In 2050, civilization has become sterile, controlled... boring. A dystopian setting..."
- **Item Neighbor (*The Lighthouse Land*):** "...Thirteen-year-old Jamie... leaves New York... for an island off the coast... magical adventure."
- **User Neighbor (User-4023):** "Recent: *PrettyTOUGH*, *Head Games*, *Heartbreak River*." (Interactions implying YA contemporary themes).

... (processing a total of 16 neighbors' contextual representations)

Synthesized Collaborative Context (High-Signal Output): The community around User 2057 engages deeply with speculative narratives. Key themes include **dystopian and speculative fiction**, often featuring young protagonists in **adventurous themes**. There is a strong appreciation for **fantasy novels with strong world-building** and complex, often morally ambiguous character dynamics.
Extracted Collaborative Facets:
- **Facet 1: Interest in dystopian and speculative fiction.**
- **Facet 2: Preference for young adult literature with adventurous themes.**
- **Facet 3: Liking for fantasy novels with strong world-building.**

*Rationale: "LM_{Mem} successfully distills common themes from YA fantasy neighbors into compact facets (highlighted in **blue**), capturing the user's affinity for expansive, speculative worlds."*

Stage-ReRank: Grounded Reasoning (LLM_{Rec})

User Instruction: "**I seek a book that can provide a visually immersive experience... stunning visuals and a captivating storyline... as detailed and engaging as a well-crafted piece of code...**" (User explicitly requests a **graphic novel**).
Input Context: User 2057's Profile (YA Fantasy history) + **Synthesized Collaborative Context** (from Stage-R).

Recommended Item: *Attack on Titan: No Regrets Complete Color Edition*
Generated Rationale: "*'Attack on Titan: No Regrets'* is a **graphic novel** known for its **stunning visuals** and **complex storyline**, aligning perfectly with your request for an immersive narrative. It strongly resonates with the **collaborative insight** that users similar to you enjoy **dystopian and speculative fiction** with **strong world-building** and emotionally layered storytelling."

Stage-W: Asynchronous Propagation (LM_{Mem})

Event: User 2057 interacts positively with *Attack on Titan*.

---

1. User Memory Update (User 2057):
- **[OLD]**: "Fan of YA fantasy, adventurous themes, and strong world-building."
- **[NEW]**: "Fan of YA fantasy and **dystopian fiction**. Confirmed strong interest in the **'graphic novel format'** combined with **complex, emotional storytelling**."

2. Target Item Memory Update (*Attack on Titan*):
- **[OLD]**: "...graphic novel, spin-off series..."
- **[NEW]**: "...graphic novel known for stunning visuals. **Validated appeal for YA fantasy readers seeking immersive dystopian worlds and complex narratives.**"

Rationale: "The item's memory is updated to reflect its appeal to the specific user profile that engaged with it."

3. Neighbor Propagation Update (User-4023):
- **[OLD]**: "Interested in humorous young adult novels... explores teenage experiences..."
- **[NEW]**: "...appreciates **strong character development and emotional depth**. May enjoy narratives that blend humor with serious, complex themes."

Rationale: "Insights on 'strong character development' from the interaction are propagated to relevant neighbors, enriching the collaborative graph."

Figure 13: **Complete Collaborative Journey (User 2057).** The figure illustrates the data flow across MemRec's three stages. **Stage-R:** LM_{Mem} synthesizes **collaborative signals (blue)** from noisy neighbors (e.g., dystopian, YA fantasy themes). **Stage-ReRank:** LLM_{Rec} combines these signals with the user's explicit intent for a **graphic novel with stunning visuals (orange)** to recommend *Attack on Titan*. **Stage-W:** Following interaction, the validated insights are propagated back, updating the user, the item, and relevant neighbors like User-4023. Note that only representative subsets shown for brevity.

Meta-Prompt Template for Rule Generation

You are an expert AI engineer specializing in recommender systems and graph-based memory networks. Your task is to generate a set of domain-specific heuristic rules for a **collaborative neighbor pruning** algorithm. The goal of this algorithm is to select the top-k most relevant neighbors (users or items) from a candidate graph to build a compact, high-signal context for a downstream LLM recommender (MemRec).
Here is the context for the specific recommendation domain:

DOMAIN CONTEXT

- **Domain Name:** {Domain Name}
- **Primary Interaction:** {Primary Interaction with example}
- **Key Metadata:** {Key Metadata}
- **Available Features:**
  - `edge_weight`: {Domain-specific explanation}
  - `recency_days`: {Domain-specific explanation}
  - `co_interaction_count`: {Domain-specific explanation}
  - `metadata_overlap_score`: {Domain-specific explanation}
  - `memory_similarity_score`: {Domain-specific explanation}

INSTRUCTIONS:

1. Based *only* on the domain context provided, generate 3-5 high-priority, interpretable ranking rules.
2. The rules should explain how to *combine* or *prioritize* the available features to find the best neighbors for *this specific domain*.
3. Be specific about thresholds and weights. For example:
   - Good: "Prioritize users with ‘co_interaction_count‘ > 3 AND apply a 2.0x multiplier to ‘metadata_overlap_score‘"
   - Bad: "Use metadata when relevant"
4. Consider that book recommendations are highly content-driven (genre, author, themes) and users often have stable long-term preferences.

OUTPUT FORMAT:

Rule 1: [Your rule here]
Rule 2: [Your rule here]
Rule 3: [Your rule here]
...

Figure 14: The generic meta-prompt template used by LM_{Mem} to generate domain-specific curation rules.

Context: InstructRec-Books

Domain Name: InstructRec-Books
Primary Interaction: Explicit ratings with text-based preference instructions. Example: "I love fantasy novels with strong female protagonists"
Key Metadata: `title`, `description` (genre hints, author info)
Domain Characteristics: Content-driven, stable preferences, sparse interactions.

Context: InstructRec-GoodReads

Domain Name: InstructRec-GoodReads
Primary Interaction: Explicit ratings in a social reading context.
Key Metadata: `title` (series info), `description`.
Domain Characteristics: Very dense graph (avg 52.7 books/user), strong community effects, series-aware reading.

Context: InstructRec-MovieTV

Domain Name: InstructRec-MovieTV
Primary Interaction: Explicit ratings with viewing preferences.
Key Metadata: `title`, `description` (Plot, Cast).
Domain Characteristics: Sparse graph, recency matters (trending content), volatile preferences.

Context: InstructRec-Yelp

Domain Name: InstructRec-Yelp
Key Metadata: `categories` (Cuisine), `attributes` (Price, WiFi).
Domain Characteristics: Context-rich but sparse. Strong categorical constraints (cuisine/price/location). Recency is critical.

Figure 15: The specific ‘DOMAIN CONTEXT‘ blocks injected into the meta-prompt for each dataset.

Generated Rules: Books (Content-Driven)

Rule 1: Content Similarity Boost
- If `metadata_overlap_score` > 0.6 (strong genre/theme match): **Apply 2.5x multiplier** to base score.
- *Rationale: Books are highly content-driven; genre/author similarity is the strongest signal.*

Rule 2: Collaborative Filtering with Threshold
- If `co_interaction_count` > 3: **Apply 1.8x multiplier** to `edge_weight`.
- Additional 1.5x boost if `memory_similarity_score` > 0.5.
- *Rationale: Meaningful CF signal requires sufficient overlap (>3 books).*

Rule 3: Mild Recency Decay
- If `recency_days` > 180: Apply decay factor: $\exp(-0.004 \times$ `recency_days`$)$.
- Else: No decay.
- *Rationale: Books have long-term appeal; old interactions remain relevant.*

Rule 4: Memory-Enhanced Ranking
- For item neighbors: Boost by $1.0 + 1.2 \times$ `memory_similarity_score`.
- For user neighbors: Boost by $1.0 + 0.8 \times$ `memory_similarity_score`.
- *Rationale: User memory captures learned preferences; stronger for items.*

Generated Rules: GoodReads (Social/Dense)

Rule 1: High Co-interaction Boost
- If `co_interaction_count` > 10: **Apply 2.0x multiplier** to `edge_weight`.
- Additional 1.5x boost to `memory_similarity_score`.
- *Rationale: Dense graph enables strong CF; >10 overlaps indicate similar taste.*

Rule 2: Series Detection
- If `metadata_overlap_score` > 0.8 (likely series match): **Apply 3.0x multiplier**.
- *Rationale: GoodReads users follow series religiously.*

Rule 3: Social Signal Priority
- If `co_interaction_count` > 15:
  - Weight `edge_weight` by 0.7x (downweight pure CF).
  - Weight `co_interaction_count` contribution by 1.5x.
- *Rationale: Explicit social overlap > implicit CF in dense graphs.*

Rule 4: Minimal Recency Decay
- If `recency_days` > 365: Apply decay factor: $\exp(-0.002 \times$ `recency_days`$)$.
- *Rationale: Book preferences are stable; old data remains valuable.*

Rule 5: Memory Amplification
- Boost by $1.0 + 1.8 \times$ `memory_similarity_score` when `co_interaction_count` > 10.
- *Rationale: Memory + social signal = very strong preference indicator.*

Figure 16: LLM-generated curation rules for Books and GoodReads datasets.

Generated Rules: MovieTV (Recency-Critical)

Rule 1: Strong Recency Decay
- If `recency_days` > 60: Apply decay factor: $\exp(-0.018 \times$ `recency_days`$)$.
- If `recency_days` > 180: Apply stronger decay: $\exp(-0.025 \times$ `recency_days`$)$.
- *Rationale: Movie/TV preferences are volatile; old data loses relevance quickly.*

Rule 2: Metadata Compensation
- If `co_interaction_count` < 3 (sparse signal): **Apply 2.8x multiplier** to `metadata_overlap_score`.
- *Rationale: Compensate for sparse CF with genre/cast similarity.*

Rule 3: Rare CF Signal Boost
- If `co_interaction_count` ≥ 3 (rare but strong): **Apply 2.5x multiplier** to `edge_weight`.
- Additional 1.8x boost to `memory_similarity_score`.
- *Rationale: Any overlap is meaningful in sparse graphs.*

Rule 4: Memory-Guided Ranking
- Boost by $1.0 + 1.5 \times$ `memory_similarity_score`.
- Additional 0.5x if `metadata_overlap_score` > 0.6.
- *Rationale: Memory captures evolving tastes better than old interactions.*

Rule 5: Recency Threshold Filter
- If `recency_days` > 365: Apply 0.3x penalty (very outdated).
- *Rationale: Movies >1 year old rarely relevant unless classics.*

Generated Rules: Yelp (Category-Driven)

Rule 1: Categorical Dominance
- If `metadata_overlap_score` > 0.7 (same cuisine + price range): **Apply 3.5x multiplier**.
- If `metadata_overlap_score` > 0.85 (+ attribute match): **Apply 4.5x multiplier**.
- *Rationale: Category match is essential; CF secondary.*

Rule 2: Very Strong Recency Decay
- If `recency_days` > 90: Apply decay factor: $\exp(-0.028 \times$ `recency_days`$)$.
- If `recency_days` > 180: Apply penalty: additional 0.5x multiplier.
- *Rationale: Restaurants change rapidly; old reviews unreliable.*

Rule 3: Attribute-Aware Memory
- If `metadata_overlap_score` includes attribute match: Boost `memory_similarity_score` by 2.2x.
- *Rationale: User memory + context (outdoor seating, etc.) = strong signal.*

Rule 4: Sparse CF Handling
- If `co_interaction_count` ≥ 2 (rare): **Apply 2.0x multiplier** to `edge_weight`.
- Else: Downweight CF by 0.5x, prioritize metadata.
- *Rationale: Very sparse; any overlap is meaningful but metadata dominates.*

Rule 5: Location/Price Filter
- If `metadata_overlap_score` < 0.4 (different cuisine/price): Apply 0.2x penalty (strong filter).
- *Rationale: Cross-category recommendations rarely work for restaurants.*

Figure 17: LLM-generated curation rules for MovieTV and Yelp datasets.

Stage-R Prompt: Collaborative Memory Synthesis

You are an intelligent memory retrieval system for personalized recommendation. Your task is to analyze the user's personal memory and collaborative memories from their neighbors to extract preference facets.
Target User: User {user_id}
User's Personal Memory: User Memory Summary: {user_memory_summary}
Collaborative Neighbor Memories: The following neighboring users and items provide collaborative signals for understanding this user's preferences:
Collaborative Neighbors: {formatted_neighbor_list}
Context (Candidate Items): Candidates to Rank: {formatted_candidate_list} (Note: These candidates are for context only, do not score them)
Your Task: Analyze the user's personal memory and the collaborative memories from neighboring users and items to identify {n_facets} distinct preference facets that characterize this user's interests and tastes. For each preference facet, provide:
1. A concise natural language description of the preference (e.g., "interest in mystery novels with strong female protagonists")
2. A confidence score between 0 and 1 indicating how strongly this facet is supported by the evidence
3. A list of supporting neighbors (user IDs or item IDs) that provide evidence for this facet
Additionally, identify the collaborative edges between neighboring users/items and the target user, with edge weights (0-1) indicating the strength of collaborative signal.
Expected Output Format: Your response should be a JSON object with two fields:
- "facets": An array of facet objects, each containing:
  * "facet": A string describing the preference
  * "confidence": A number between 0 and 1
  * "supporting_neighbors": An array of neighbor IDs (e.g., ["User-123", "Item-456"])
- "support_edges": An array of edge objects, each containing:
  * "from": The source neighbor ID (string)
  * "to": The target user ID (string)
  * "w": The edge weight between 0 and 1 (number)

Figure 18: The prompt used by LM_{Mem} to synthesize high-level memory facets from retrieved collaborative neighbors in Stage-R. **Candidate items act as context to guide task-relevant synthesis.**

Stage-ReRank Prompt (MemRec Mode)

You are an intelligent recommendation scoring system. Your task is to evaluate how well each candidate item matches the target user's preferences based on their personal memory and collaborative signals.
Target User: User {user_id}
User's Current Request: {instruction}
User Preferences (Extracted from Collaborative Memories): Based on collaborative signals from neighboring users and items, we have identified the following preference patterns: {formatted_facets}
Candidate Item Memories: {formatted_item_memories}
Your Task: For each of the candidate items listed above, provide a relevance score between 0 and 1 that indicates how well the item matches the user's preferences:
• 1.0 = Excellent match, highly aligned with user's facets and memory
• 0.5 = Moderate match, partially relevant
• 0.0 = Poor match, not aligned with user's interests
For each item, provide a brief rationale explaining your scoring decision based on the user's preference facets and personal memory.
Expected Output Format: Your response should be a JSON object with a single field:
- "scores": An array of scoring objects, each containing:
  * "item_id": The item's ID (integer)
  * "score": Your relevance score between 0 and 1 (number)
  * "rationale": A brief explanation of your scoring (string)

Figure 19: The prompts used by LLM_{Rec} for candidate scoring in Stage-ReRank.

Stage-W Prompt: Asynchronous Collaborative Propagation

You are an intelligent memory management system for collaborative recommendation. Your task is to update the personal memories of the user, the clicked item, and relevant collaborative neighbors based on this new interaction.
Interaction Context: User {user_id} has just interacted with (clicked) Item {item_id} ({clicked_item_info}).
User Preferences (Extracted from Collaborative Memories): The following preference patterns were identified for this user: {formatted_facets}
Current Personal Memory of User {user_id}: {current_user_memory}
Current Memory of Item {item_id} ({clicked_item_info}): {current_item_memory}
Collaborative Neighbors Available for Memory Propagation: The following {n_neighbors} collaborative neighbors are available for potential memory updates: {formatted_neighbors}
Your Task: Generate UPDATED memories for:
1. **The current user** (synthesize current memory + facets + clicked item)
2. **The clicked item** (describe what it is and who might enjoy it)
3. **Selected neighbors** (IMPORTANT: collaborative propagation is key!)
   * Analyze the available neighbors and their current memories
   * Select neighbors that are RELEVANT to this interaction (e.g., similar themes, related topics)
   * Update their memories to reflect new insights from this interaction
   * This helps the system learn collaboratively!
Output Requirements:
- "user_memory": Concise natural language description of user's interests and preferences
   * Synthesize themes (e.g., "holistic health", "children's education")
   * Be specific (e.g., "interested in Reiki and aromatherapy")
   * DON'T just list item titles
   * Keep it focused (typically a few sentences)
- "item_memory": Concise description of the clicked item
   * What it's about and who might enjoy it
   * Keep it brief but informative
- "neighbor_updates": Array of neighbor memory updates (OPTIONAL but recommended)
   * Select neighbors that are MOST relevant to this interaction
   * Choose as many as needed (typically a few, but flexible)
   * For each neighbor, provide updated memory content (NOT just appending text)
   * Rationale explains why this neighbor is relevant
Expected Output Format: Your response should be a JSON object with three fields:
- "user_memory": The updated personal memory for the user (string)
- "item_memory": The updated memory for the clicked item (string)
- "neighbor_updates": An array of neighbor update objects (may be empty), each containing:
   * "neighbor_id": The neighbor's ID, e.g., "User-123" or "Item-456" (string)
   * "memory_update": The updated memory content for this neighbor (string)
   * "rationale": A brief explanation of why this neighbor should be updated (string)

Figure 20: The prompt used by LM_{Mem} to asynchronously update user and neighbor memories in Stage-W.

GPT-4o Judge Prompt for Rationale Quality Evaluation

SYSTEM PROMPT:
You are an expert evaluator for recommender systems. Your task is to assess the quality of explanations (rationales) generated by three different AI agents designed to recommend items to users.
You will be provided with:
1. A summary of the target User's interests.
2. The recommended Item name.
3. Rationale A (generated by Model A).
4. Rationale B (generated by Model B).
5. Rationale C (generated by Model C).
You must evaluate each rationale independently on three distinct criteria using a 1-5 Likert scale.
Evaluation Criteria & Scoring Rubric:
1. Specificity (1-5 Points) Measure how concrete and detailed the rationale is regarding the recommended item.
- 1 (Vague): Very generic; could apply to many items in the category (e.g., "It's a good book").
- 3 (Moderate): Mentions general themes or genre traits but lacks specific details.
- 5 (Highly Specific): Richly detailed; mentions specific plot elements, character traits, writing style, or unique features of the item.
2. Relevance (1-5 Points) Measure how well the rationale explains *why* this item suits this specific user based on their profile.
- 1 (Irrelevant): A generic recommendation unrelated to the user's known interests.
- 3 (Acceptable): Makes a basic connection to user genre preferences.
- 5 (Highly Personalized): explicitly ties specific item features to specific aspects of the user's history or tastes.
3. Factuality (1-5 Points) Measure the accuracy of the claims made about the item.
- 1 (Hallucinated): Contains major factual errors or describes a different item entirely.
- 5 (Accurate): All claims about the item's content and characteristics are factually correct.
Output Format: You must output strictly valid JSON immediately, without any additional text. The format should be: {
"model_a": {"specificity": <int>, "relevance": <int>, "factuality": <int>},
"model_b": {"specificity": <int>, "relevance": <int>, "factuality": <int>},
"model_c": {"specificity": <int>, "relevance": <int>, "factuality": <int>}
}

USER PROMPT TEMPLATE:
User Interests Summary: {user_history_summary}
Recommended Item: {item_title}
Rationale A: {rationale_model_a}
Rationale B: {rationale_model_b}
Rationale C: {rationale_model_c}
Please evaluate Rationale A, Rationale B, and Rationale C based on the system instructions and provide the JSON output.

Figure 21: The system prompt and user input template used for the GPT-4o based rationale quality evaluation. The judge evaluates three models simultaneously across Specificity, Relevance, and Factuality domains.