MEMVISTA: DOMAIN-ADAPTIVE USER MEMORY EVOLUTION FOR LLM RECOMMENDATION AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

When a compact user memory discards useful history, improving the downstream recommendation policy cannot directly recover that evidence. We study whether memory facets, instructions about which historical distinctions to retain, are a useful target of recommendation learning. **Which historical distinctions are worth preserving depends on the domain**, even after matching profile token budget and the total size of banks of domain-specific facets. We introduce MemVista to **learn memory facets automatically from recommendation errors**. Facets configure evidence extraction and profile synthesis, while the recommendation prompt remains fixed and never receives them. Evaluation covers twelve domains and six backbones. With GLM, MemVista outperforms Facets from pooled errors by 4.4 percentage points and Shuffled facets with matched total storage by 5.8 points. Target-aware diagnosis improves over Adaptive facets w/o correctness by 10.7 percentage points and Adaptive facets w/o target by 9.0 points. Frozen facets also improve direct summarization and an existing hierarchical memory manager without further search. A blinded, length-controlled exchange with strong reference memories directly tests the value of the additional retained evidence. Together, these results establish the empirical value of supervised content selection beyond domain priors, memory architecture, and ranking-policy adaptation.

1 INTRODUCTION

Recommending the next item is a bet on which part of a user's past still matters. A decade of progress has widened what recommenders can read: interaction patterns (Rendle et al., 2009; Kang & McAuley, 2018; He et al., 2020), language and item semantics (Geng et al., 2022; Bao et al., 2023; Zhang et al., 2023b; Hou et al., 2023), and agent-gathered evidence (Wang et al., 2023; Huang et al., 2023). Yet the deciding user representation remains bounded: more state raises serving cost, and even full histories can be underused by LLM recommenders (Lin et al., 2023). As reading capacity grows, the constraint therefore shifts inward: not whether evidence exists, but which user distinctions are worth preserving in the small representation the ranker sees.

The compact user state need not be a single memory object. We use two compressions—behavior into decision-relevant preferences, then a profile—so memory can evolve both what facets it retains and how it distills them for ranking. Existing work focuses largely on organizing and maintaining this state: MemRec derives curation rules from domain statistics (Chen et al., 2026), while AtomRec links atomic memories into evidence paths (Hu et al., 2026); related methods update per-user knowledge through reflection and feedback (Zhang et al., 2023a; Xu et al., 2025b; Tao et al., 2026). Yet across these approaches, what memory retains is still largely fixed in advance by predefined taxonomies, structures, or templates. The harder question lies upstream: which distinctions are worth remembering at all? Domain descriptions alone cannot reveal such distinctions: two diners may prefer the same restaurant chain yet live at different distances from it; two readers may share a genre yet differ in reading progress.

Which historical information should a compact memory preserve for a particular domain? Domain knowledge provides a starting point, as MemRec's curation rules illustrate (Chen et al., 2026). Recommendation errors then reveal where these priors fall short: contrasting the prediction with the observed next item can expose history-supported distinctions missing from memory. Domain priors provide the start; error–choice contrasts drive revision.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We therefore make memory content—not memory machinery—the learnable object. MemVista learns a domain-shared set of *facets*: natural-language specifications of which historical distinctions to extract and retain. **To our knowledge, MemVista is the first framework to evolve domain-shared memory facets from next-item recommendation errors.** Facets guide both extraction and synthesis: which evidence is found and which survives compression. MemVista diagnoses training errors, revises facets, rebuilds profiles, and accepts validation improvements. New profiles change the errors examined next. Facets are shared across a domain and frozen before testing. Test profiles use only history, without candidates, and the ranking prompt stays fixed.

Evolution also produces reusable domain knowledge: facets specify series progression, compatibility supported by prior purchases, and changes in care needs supported by chronological evidence. Cross-domain substitutions and transfer to other constructors test where these priorities remain useful.

Experiments across twelve domains and six language-model backbones examine both recommendation performance and the information learned through evolution. Our contributions are threefold.

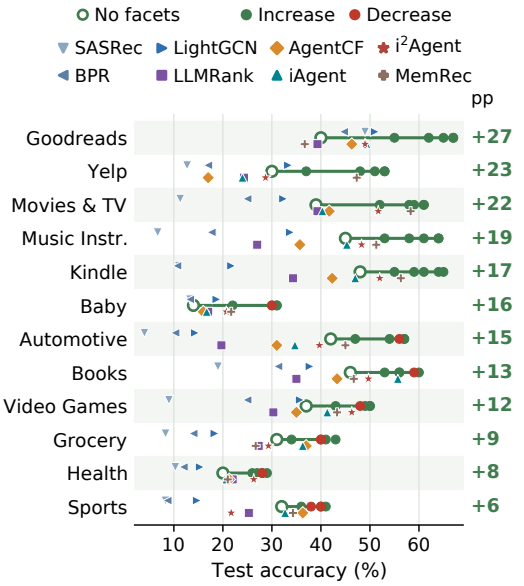

Figure 1: A GLM evolution run, evaluated on test queries. Open circles show No facets; green/red dots mark checkpoint increases/decreases. Other symbols show baseline recommenders. Right: final-minus-initial gain (pp).

- **Domain-dependent information for recommendation.** We show that the usefulness of memory facets depends on the recommendation domain. At matched budgets, domain grouping improves GLM accuracy over pooled errors by 4.4 percentage points and randomly mixed groups by 5.8 points; cross-domain substitutions reveal both specialization and reuse.
- **Memory evolution guided by recommendation errors.** To our knowledge, we are the first to evolve domain-shared memory facets from next-item recommendation errors and use them in both extraction and synthesis. Labeled diagnosis improves GLM accuracy over target-masked adaptation by 9.0 points, and continued revision improves over independent proposals informed by errors by 3.2 points. Corresponding GPT-5 controls support both effects.
- **Reusable domain facets and evidence of their value.** We give domain facet examples and show that the learned sets improve direct summarization and a hierarchical memory manager without further search. Controlled fact exchange shows that additional historical facts retained by these facets help recommendation, connecting the learned domain priorities to useful user evidence.

2 Related Work

User Representation for Recommendation. BPR, SASRec, and LightGCN represent users through pairwise preferences, sequential dynamics, and collaborative structure (Rendle et al., 2009; Kang & McAuley, 2018; He et al., 2020). P5 and TALLRec enrich representations through language generation or instruction tuning (Geng et al., 2022; Bao et al., 2023), while KAR, RLMRec, and LLaRA incorporate external knowledge or align semantic and collaborative representations (Xi et al., 2023; Ren et al., 2023; Liao et al., 2023). These approaches improve preference encoding while leaving the choice of domain-specific facets implicit. We learn these facets to guide persistent user memory.

Evidence for Recommendation Decisions. LLM recommenders draw on item semantics and review-derived cues (Lyu et al., 2024; Na et al., 2024), selected demonstrations and sequential transition patterns (Wang & Lim, 2024; Zhai et al., 2025), preference strength and recency (Ouyang et al., 2026), and multiple preference signals or recommender tools (Kim et al., 2026; Huang et al., 2023). These approaches enrich evidence along largely predefined dimensions. We use prediction errors to refine which dimensions matter.

Evolving Recommendation Memory. Recommendation agents increasingly update user knowledge through interaction and feedback, from reflected user–item memories and dynamic profiles to structured memory graphs and hierarchical lifecycles (Zhang et al., 2023a; Xu et al., 2025b; Chen et al., 2026; Shen et al., 2026; Hu et al., 2026). SAGER further evolves per-user reasoning policies rather than memory content alone (Tao et al., 2026). These approaches largely retain predefined preference schemas or adapt decision principles per user. We pool feedback across users to evolve domain-level facets that guide future profile construction.

General Agent Self-Evolution. Agent memory has progressed from retaining and reorganizing experience (Park et al., 2023; Zhong et al., 2023; Packer et al., 2023; Chhikara et al., 2025; Xu et al., 2025a) to improving how agents learn from experience itself. Reflexion distills feedback into reusable verbal lessons (Shinn et al., 2023); GEPA evolves prompts through trajectory-level reflection (Agrawal et al., 2025); MemEvolve jointly adapts experiential knowledge and memory architectures (Zhang et al., 2025); and MemSkill learns and expands reusable memory skills from hard cases (Zhang et al., 2026). We bring feedback-driven self-evolution to recommendation, learning what evidence to seek across users in a domain.

3 METHOD

MemVista learns domain-shared facets from training errors and selects them by validation accuracy, with the memory procedure, model weights, and ranking prompt fixed (Figure 2).

3.1 MEMORY FACETS AND INPUTS

Each item carries its available metadata. For user u with t interactions before prediction, the chronological history is $H_u = (x_u^1, \ldots, x_u^t)$, where x_u^j is the item in the jth recorded interaction ($j = 1, \ldots, t$). The candidate set is C_u, and $x_u^+ \in C_u$ is the observed next item. The recent window W_u keeps the last L interactions, or the whole history if shorter; the profile s_u stores user facts within a token budget B.

A facet is a natural-language specification of what to extract and retain. Each domain d shares one facet set $\mathcal{P}_d$ across its users. With Build the profile constructor, Rec the fixed recommender, and $\hat{x}_u$ the predicted next item,

$$s_u = \text{Build}(H_u; \mathcal{P}_d),$$
$$\hat{x}_u = \text{Rec}(s_u, W_u, C_u), \tag{1}$$

where the semicolon separates user input from the shared facet set.

The constructor receives history and facets but no candidates; the recommender receives the profile, recent window, and candidates but no facet text, identifiers, or domain policy. History beyond the window reaches the recommender only through the profile.

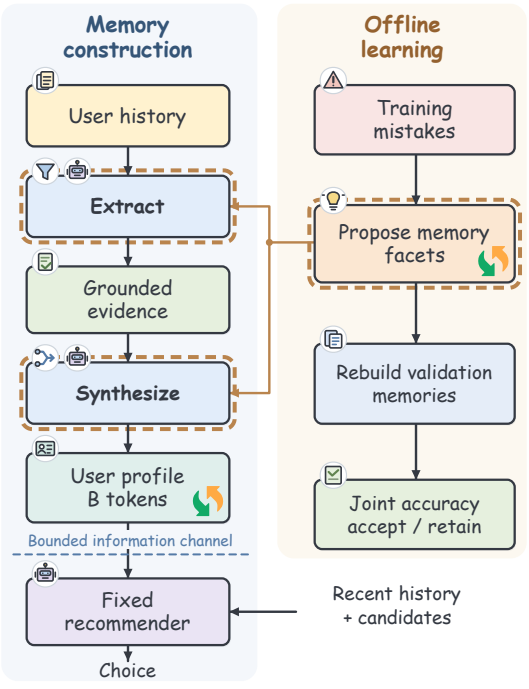

Figure 2: Facets guide extraction and synthesis. Training errors drive revisions; rebuilt validation profiles determine acceptance. Only user facts reach the fixed recommender.

3.2 BUILDING USER PROFILES

We fix the memory architecture and build each profile with two model calls sharing one facet set: Extract identifies evidence and temporal relations across the full history, yielding the evidence set E_u; Synthesize selects and compresses evidence into user facts,

$$\begin{aligned} E_u &= \mathrm{Extract}(H_u; \mathcal{P}_d), \\ s_u &= \mathrm{Synthesize}(E_u; \mathcal{P}_d), \qquad |s_u|_{\mathrm{tok}} \le B, \end{aligned} \tag{2}$$

where $|s_u|_{\mathrm{tok}}$ is the profile length in tokens.

Each entry in E_u pairs a history-supported statement with its history indices and uncertainty. A facet about a series, for example, leads the constructor to record which installments appear in the history, without instructing the recommender to choose a sequel.

Here $\varnothing$ denotes the empty set. When $\mathcal{P}_d = \varnothing$, both calls fall back to generic prompts prioritizing stable preferences, recurring patterns, current needs, and supported constraints. Outputs must be supported by the history in all cases; valid indices alone do not make a statement true, and the content audit checks factual support (Appendix E).

The constructor has no learned editor, scheduler, or retrieval policy. Section 4.5 transfers frozen facets to direct summarization and a hierarchical memory manager. The default task builds a profile from all history available at a prediction point.

3.3 LEARNING FROM RECOMMENDATION ERRORS

Each domain has its own training, validation, and test cases, starting from $\mathcal{P}_d = \varnothing$ and repeating four steps per round.

1. Diagnose errors. The reflector examines selected training errors—each case's history, current profile, candidates, prediction, and observed next item—and identifies history-supported information that is missing or unclear in the profile.

2. Propose a revision. The proposer revises the current facet set from these diagnoses, merging duplicates; each proposal is a complete set meeting the count limit K and a whole-set token limit.

3. Validate the proposal. Validation profiles are rebuilt from their original histories with the proposed set, then scored by the fixed recommender. Writing $i = 1, \ldots, n_d$ for domain d's validation cases, W_i, C_i, and x_i^+ are each case's recent window, candidate set, and observed next item.

Writing $\mathcal{P}$ for the set under evaluation—current or proposed—and $s_i(\mathcal{P})$ for case i's rebuilt profile, validation accuracy in domain d is given below. The indicator $\mathbf{1}[\cdot]$ equals 1 when its condition is true and 0 otherwise:

$$A_d(\mathcal{P}) = \frac{1}{n_d} \sum_{i=1}^{n_d} \mathbf{1}\left[\mathrm{Rec}\big(s_i(\mathcal{P}), W_i, C_i\big) = x_i^+\right]. \tag{3}$$

4. Update or retain the set. A proposal is accepted only if it strictly improves validation accuracy. Acceptance rebuilds training profiles and refreshes cached predictions before the next errors are collected; rejection retains the current set and cache.

The reflector reads aggregate validation accuracy but never validation histories, individual errors, or labels. Each validation scores the complete facet set, and selection does not use recommendation reasons to score individual facets.

Each domain allows at most G proposal evaluations, stops early only when no training errors remain, and uses the same settings; after training, its facet set is frozen and never updated by test outcomes. Appendix B gives the full procedure.

4 EXPERIMENTS

The evaluation separates the value of the complete system from the learning object, its domain dependence, and its relation to ranking policy.

4.1 SETUP

We use Goodreads, Yelp, and ten Amazon Reviews 2023 categories (Wan & McAuley, 2018; Hou et al., 2024; Yelp, 2024). Each domain has 300 training, 300 validation, and 300 test cases, with disjoint users across splits. A case pairs a pre-target history with ten candidates—the observed next item and nine uniformly sampled unseen items from the same domain—shuffled once per case and shared by all methods. The metric is candidate-selection accuracy. Macro accuracy weights the twelve domains equally. This task measures reranking within a fixed candidate set, not full-catalog retrieval. Unlike instruction-conditioned recommendation on InstructRec (Xu et al., 2025b), the input contains no request generated from the held-out item's review. Every method predicts the next interaction from the history and candidate metadata.

We compare SASRec (Kang & McAuley, 2018), BPR (Rendle et al., 2009), LightGCN (He et al., 2020), LLMRank (Hou et al., 2023), AgentCF (Zhang et al., 2023a), iAgent and i^{2}Agent (Xu et al., 2025b), and MemRec (Chen et al., 2026). Language-based baselines receive the same available item metadata and pre-target histories while preserving their own memory and retrieval procedures, and the constructor study additionally uses our reimplementation of MARS (Shen et al., 2026). The main comparison evaluates the full framework on six backbones: four open-weight models—DeepSeek-V4-Pro (DeepSeek-AI, 2026), Kimi K3 (Moonshot AI, 2026), Qwen3.8-Max (2.4T) (Qwen Team, 2026), and GLM-5.2 (GLM-5-Team, 2026)—and two proprietary models, GPT-5 (OpenAI, 2025) and GPT-5.6 Terra (OpenAI, 2026). For each run, the same backbone is used across all LLM-based components, with facets learned independently for each backbone. Terra evaluates only the three leading memory methods; subsequent analyses use GLM-5.2 unless stated otherwise. GPT-5 repeats the key content-learning controls (Table 17).

The constructor uses a five-item recent window, at most fifty historical interactions, a 256-token profile, and at most twelve facets within 768 tokens; evolution performs twelve candidate-validation opportunities and accepts only strictly positive validation improvements. Fixed domain-prior facets is reported as an aggregate accuracy in Table 2. The other rows average three complete construction-and-selection runs on the same test users. Tables 11, 12, and 14 provide the available paired intervals, domain results, and individual runs. GEPA is reported separately as an aggregate mean in Table 13. Repeated analyses report paired user-bootstrap 95% intervals over 10,000 resamples, preserving each user's three-run cluster; these capture test-sample variation conditional on the runs, with run-level differences reported separately. Data processing, baseline fitting, templates, and exact selection rules are in Appendix A.

4.2 OVERALL RECOMMENDATION PERFORMANCE

Table 1 reports per-domain and macro accuracy, averaging four prediction repetitions of each frozen system; the controlled studies below use separate three-run batches (Appendix A).

MemVista achieves the highest macro accuracy on all six backbones, exceeding each backbone's strongest baseline by 5.2–10.3 points, and is best or tied best in 68 of 72 backbone–domain settings. The largest gain occurs with GLM-5.2: 0.510 vs. 0.407 for MemRec (+10.3 points). Appendix A explains task adaptation and domain variation.

4.3 LEARNING THE CONTENT OF A BOUNDED MEMORY

Table 2 reports seven controls with the constructor, recent window, profile token budget, facet set token budget, final ranker, and evaluation cases held fixed. Appendix B.1 defines the controls and Appendix B.2 details their protocols. GPT-5 closely reproduces the GLM-5.2 pattern in Table 17.

Method	Model	Goodr.	Yelp	M&TV	Music	Kindle	Baby	Auto	Books	Games	Groc.	Health	Sports	Mean
SASRec	*N/A*	0.490	0.127	0.113	0.067	0.110	0.133	0.040	0.190	0.090	0.083	0.103	0.083	0.136
BPR	*N/A*	0.447	0.170	0.250	0.177	0.107	0.133	0.103	0.313	0.250	0.140	0.120	0.087	0.191
LightGCN	*N/A*	0.510	0.333	0.323	0.337	0.217	0.187	0.143	0.377	0.357	0.183	0.153	0.147	0.272
LLMRank	**DeepSeek-V4**	0.280	0.183	0.353	0.260	0.257	0.147	0.160	0.190	0.207	0.140	0.103	0.137	0.201
AgentCF	**DeepSeek-V4**	0.197	0.150	0.377	0.180	0.167	0.107	0.240	0.297	0.170	0.197	0.093	0.133	0.192
iAgent	**DeepSeek-V4**	0.487	0.237	0.553	0.373	0.483	0.150	0.247	0.517	0.343	0.240	0.173	0.253	0.338
i^{2}Agent	**DeepSeek-V4**	0.470	0.257	0.563	0.427	0.487	0.160	0.367	0.467	0.410	0.307	0.223	0.317	0.371
MemRec	**DeepSeek-V4**	0.367	0.403	0.587	0.560	0.537	**0.207**	0.400	0.507	0.437	0.340	0.257	0.350	0.413
MemVista	**DeepSeek-V4**	**0.610**	**0.517**	**0.630**	**0.583**	**0.573**	0.203	**0.447**	**0.523**	**0.473**	**0.373**	**0.283**	**0.363**	**0.465**
LLMRank	**Kimi-K3**	0.403	0.177	0.463	0.393	0.453	0.230	0.420	0.507	0.467	0.290	0.247	0.367	0.368
AgentCF	**Kimi-K3**	0.530	0.200	0.550	0.383	0.560	0.193	0.350	0.490	0.410	0.290	0.280	0.370	0.384
iAgent	**Kimi-K3**	0.500	0.293	0.580	0.500	0.520	0.213	0.380	0.560	0.400	0.343	0.283	0.360	0.411
i^{2}Agent	**Kimi-K3**	0.420	0.275	0.613	0.490	0.510	**0.230**	0.430	0.543	0.420	0.340	0.247	0.360	0.407
MemRec	**Kimi-K3**	0.387	0.453	0.550	0.540	0.523	0.200	0.383	0.580	0.463	0.340	0.210	0.350	0.415
MemVista	**Kimi-K3**	**0.563**	**0.517**	**0.653**	**0.573**	**0.600**	0.220	**0.453**	**0.620**	**0.510**	**0.370**	**0.293**	**0.390**	**0.480**
LLMRank	**Qwen-3.8 (2.4T, Max)**	0.420	0.270	0.460	0.363	0.463	0.207	0.263	0.393	0.410	0.217	0.140	0.310	0.326
AgentCF	**Qwen-3.8 (2.4T, Max)**	0.320	0.207	0.520	0.350	0.500	0.193	0.320	0.460	0.380	0.320	0.223	0.390	0.349
iAgent	**Qwen-3.8 (2.4T, Max)**	0.473	0.290	0.603	0.463	0.463	0.247	0.420	0.553	0.443	0.320	0.253	0.370	0.408
i^{2}Agent	**Qwen-3.8 (2.4T, Max)**	0.430	0.300	**0.620**	0.490	0.530	0.237	0.463	0.540	0.480	0.333	0.223	0.410	0.421
MemRec	**Qwen-3.8 (2.4T, Max)**	0.367	0.493	0.523	0.463	0.480	0.193	0.337	0.493	0.410	0.240	0.167	0.343	0.376
MemVista	**Qwen-3.8 (2.4T, Max)**	**0.590**	**0.570**	0.567	**0.603**	**0.643**	**0.260**	**0.530**	**0.570**	**0.527**	**0.403**	**0.280**	**0.420**	**0.497**
LLMRank	**GLM-5.2**	0.393	0.243	0.393	0.270	0.343	0.170	0.197	0.350	0.303	0.273	0.220	0.253	0.284
AgentCF	**GLM-5.2**	0.463	0.170	0.417	0.357	0.423	0.160	0.310	0.433	0.350	0.370	0.213	0.363	0.336
iAgent	**GLM-5.2**	0.493	0.240	0.403	0.453	0.470	0.167	0.347	0.557	0.413	0.363	0.207	0.327	0.370
i^{2}Agent	**GLM-5.2**	0.490	0.287	0.517	0.483	0.520	0.207	0.397	0.497	0.463	0.293	0.263	0.217	0.386
MemRec	**GLM-5.2**	0.367	0.473	0.583	0.513	0.563	0.217	0.450	0.467	0.433	0.267	0.210	0.343	0.407
MemVista	**GLM-5.2**	**0.670**	**0.530**	**0.607**	**0.640**	**0.650**	**0.307**	**0.570**	**0.590**	**0.490**	**0.400**	**0.283**	**0.383**	**0.510**
LLMRank	**GPT-5**	0.410	0.270	0.503	0.433	0.390	0.160	0.307	0.313	0.333	0.253	0.207	0.303	0.324
AgentCF	**GPT-5**	0.260	0.110	0.253	0.153	0.297	0.150	0.243	0.420	0.283	0.143	0.130	0.120	0.214
iAgent	**GPT-5**	0.377	0.213	0.330	0.257	0.330	0.113	0.167	0.303	0.257	0.120	0.120	0.173	0.230
i^{2}Agent	**GPT-5**	0.450	0.287	0.557	0.427	0.523	0.177	0.327	0.497	0.420	0.240	0.203	0.293	0.367
MemRec	**GPT-5**	0.423	0.520	0.490	0.610	0.593	0.203	0.467	0.553	**0.507**	0.393	0.300	0.377	0.453
MemVista	**GPT-5**	**0.673**	**0.627**	**0.620**	**0.660**	**0.667**	**0.213**	**0.497**	**0.623**	0.470	**0.400**	**0.307**	**0.433**	**0.516**
i^{2}Agent	**GPT-5.6 Terra**	0.403	0.280	0.567	0.473	0.533	0.173	0.410	0.500	0.447	0.260	0.260	0.330	0.386
MemRec	**GPT-5.6 Terra**	0.480	0.580	0.670	0.603	0.610	0.223	0.540	0.530	0.407	0.323	0.313	0.353	0.469
MemVista	**GPT-5.6 Terra**	**0.653**	**0.607**	**0.673**	**0.660**	**0.643**	**0.233**	**0.663**	**0.620**	**0.507**	**0.417**	**0.313**	**0.393**	**0.532**

Table 1: Accuracy across twelve domains and six LLM backbones. The first three rows report shared traditional recommenders that do not use an LLM. Mean is the macro average across domains. Bold and underlining indicate the best and second-best results within each LLM group, respectively.

Domain priors alone leave substantial room for improvement. The Fixed domain-prior facets control reaches 0.407 compared with 0.367 for No facets, leaving a descriptive 10.3-point gap to MemVista. Knowing the domain and selecting a plausible facet set therefore does not recover the benefit of learning from recommendation errors. Selection uses validation labels in all searches.

Facet configuration	Accuracy	Gap
No facets	0.367	14.3
Fixed domain-prior facets	0.407	10.3
Adaptive facets w/o correctness	0.403	10.7
Adaptive facets w/o target	0.420	9.0
Independent facet proposals	0.478	3.2
Facets from pooled errors	0.466	4.4
Shuffled facets	0.452	5.8
MemVista	**0.510**	0.0

Table 2: GLM content-learning controls at fixed token budgets for profiles and facet sets. Each facet-learning run evaluates at most twelve proposed facet sets. Accuracy averages the twelve domains equally. Gap is MemVista's accuracy minus the row's accuracy in percentage points (pp).

Target-aware diagnosis adds value beyond masked training cases. Adaptive facets w/o correctness reaches 0.403, and the full procedure adds 10.7 points. Adaptive facets w/o target reaches 0.420, leaving a 9.0-point difference. The latter replays the full method's selected case identifiers but rebuilds profiles under its own facets. Both controls hide targets, correctness flags, and labeled diagnoses. The comparison therefore tests target-aware diagnosis across separate memory-learning trajectories with matched case identifiers.

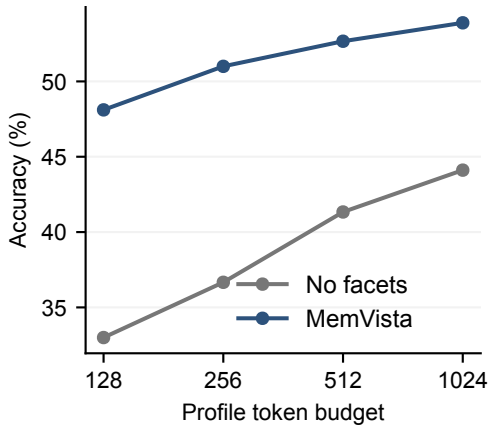

Figure 3: Accuracy under different profile token budgets. Facets learned at 256 tokens are reused unchanged.

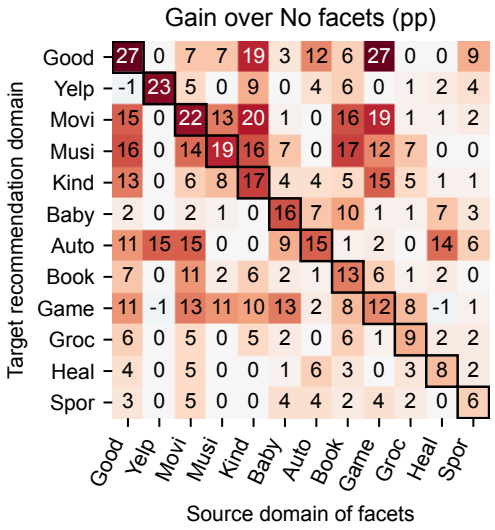

Figure 4: Cross-domain substitution gains over each row's No facets baseline (pp, rounded). Rows: target domains; columns: facet sources. Outlines mark matched sets.

Adaptation matters after matching diagnostic exposure. The Independent facet proposals control, which re-proposes a complete set each round without inheriting the incumbent set or its scores, reaches 0.478, exceeding Fixed domain-prior facets by 7.1 points and leaving an adaptive advantage of 3.2 points. It matches the full procedure's twelve proposal opportunities and up to 48 sampled error records per domain. Revising the current set and refreshing its errors therefore improve on error-informed proposals. The contrast evaluates these operations together, matching validation opportunities and information interfaces rather than every implementation cost.

GEPA facets uses the same facet set interface under a general reflective optimizer (Agrawal et al., 2025). It reaches 0.483, compared with 0.510 for MemVista, a descriptive 2.7-point gap (Table 13). Both improve over No facets. Only GEPA's aggregate mean is reported, so the comparison does not estimate a paired interval or establish statistical superiority. GEPA confirms that another search procedure can learn useful facets under the same budgets.

The gain is not explained by a larger profile. Figure 3 varies the profile token budget from 128 to 1,024 tokens. The selected facets remain fixed, with no additional search. At 256 tokens, learned content reaches 51.0% accuracy, compared with 44.1% for No facets at 1,024 tokens. Learned content performs better at every tested budget. With the same ranking template, full-history input achieves 0.478 and candidate-aware historical evidence selection within 256 tokens achieves 0.471. These references distinguish learned content from simply exposing more history or selecting similar items.

Evolution dynamics. The controls above compare final facet sets. Figure 1 instead follows recommendation accuracy while each domain's set evolves: every domain improves over the No-facets starting point, with net gains between 6 and 27 points (27 on Goodreads, 23 on Yelp, and 22 on Movies & TV), and accepted revisions occasionally reduce test accuracy (red dots), because acceptance uses validation accuracy. Appendix G traces error composition, facet-space trajectories, per-category yield, and the semantic composition of learned facets on the same runs.

4.4 Domain Dependence and Transfer

Table 2 compares grouping errors by domain with pooling them across domains or learning from randomly mixed groups. The Facets from pooled errors control reaches 0.466, below MemVista by 4.4 points. Its shared set has the same limits of twelve facets and 768 tokens per builder invocation.

Each of its twelve joint validations evaluates all domains, matching the total number of domain validations in twelve separate searches.

The Shuffled facets control additionally matches total facet storage, per-call token budget, training records, and evaluation budget. It reaches 0.452, a gap of 5.8 points. Grouping by the target domain therefore helps beyond merely maintaining twelve independently optimized sets.

In Figure 4, cell (i, j) builds profiles from target domain i's histories with facets learned in source domain j. Only the source set changes within a row; there is no adaptation, and facets never enter the ranker. The diagonal uses each domain's own facets. Gains subtract the row's No facets accuracy. All cells and baselines use one evaluation pass, separate from the repeated controls (Appendix H).

The mean within-row range is 15.8 points. On Yelp, its own set outperforms Baby's set by 23.0 points. On Baby, its own set outperforms Yelp's by 16.0 points. This reversal, selected after inspecting the complete matrix, supports target-dependent usefulness. Transfer also remains substantial: Games facets are competitive on Goodreads, and Yelp and Movies & TV facets transfer to Automotive. Domain adaptation matches evidence priorities without requiring semantic isolation.

4.5 Reusing Facets across Constructors

Table 3 transfers frozen Fixed domain-prior and MemVista facets from the default constructor. Direct summarization compresses the full history in one call; MARS retains its hierarchy and retrospective schedule. Facets enter only memory construction, with no further search or validation selection.

Learned facets improve direct summaries from 0.376 to 0.499 (12.3 points, $[11.5, 13.1]$) and MARS from 0.408 to 0.526 (11.8 points, $[11.0, 12.6]$); native MARS reaches 0.382. These gains support reuse beyond the default constructor, with MARS plus learned facets the strongest combined system.

Constructor	Native	Prior facets	Learned facets	Gain (pp)
Two-stage constructor	0.367	0.407	0.510	10.3
Direct history summary	0.362	0.376	0.499	**12.3**
MARS	**0.382**	**0.408**	**0.526**	11.8

Table 3: GLM constructor transfer. Prior: Fixed domain-prior facets; learned: MemVista facets. Both are frozen after two-stage selection. Gain is learned minus prior (pp). Native MARS retains its original schema and retrospective configuration.

4.6 Direct Evidence of Information Preservation

Table 4 tests the additional facts relative to strong references, rather than only No facets. We preselect fifty users per domain by identifier hash and use run 2. Annotators see histories and anonymized profiles, without candidates, targets, predictions, or method identities. They identify supported assertions and normalize their wording. Appendix E details factual support, normalization, and the exchange protocol.

For each reference, match its canonical factual payload with the learned payload in fact count and token length within two tokens. Exchange only supported learned-only facts whose entire evidence precedes the recent five-item window. The choice of facts remains blind to labels. There are 250 eligible exchanges for Adaptive facets w/o correctness, 230 for Adaptive facets w/o target, and 218 for Independent facet proposals. Other users retain unchanged inputs, and all 600 remain in each denominator.

Reference memory	Before	After	Gain (pp) [95% interval]
Adaptive facets w/o correctness	0.407	0.425	1.8 [0.7, 3.2]
Adaptive facets w/o target	0.417	0.433	1.7 [0.5, 2.8]
Independent facet proposals	**0.493**	**0.508**	1.5 [0.3, 2.7]

Table 4: Exchanging learned-only older facts into reference profiles. Gains and paired 95% intervals use percentage points (pp). All 600 users remain in the denominator, including no-op exchanges. Fact count, wording, length, and recent context are controlled.

Inserting these facts improves the three references by 1.8, 1.7, and 1.5 points, respectively. Table 8 reports the original and canonical accuracies on these same users. Canonicalization changes

each reference by 0.3 points. MemVista changes from 0.525 to 0.502 because canonicalization also matches factual subsets and length; the exchange comparison uses these matched canonical profiles. In the Adaptive facets w/o correctness comparison, the learned canonical profile scores 0.502. Replacing its selected facts with reference facts reduces accuracy to 0.480, while replacing equally old, length-matched control facts yields 0.495. The difference between those replacement effects has interval $[0.7, 2.5]$. Exchanging at most two older facts per eligible user thus establishes the usefulness of retained evidence. It does not estimate the fraction of the system gap explained or individual facets' causal effects. Appendix E specifies all directions and matching rules.

4.7 Ablation and Separation from Ranking Policy

Memory learning complements better reasoning instructions. Table 5 crosses No facets and MemVista with fixed and learned ranking policies. The ranking policy is learned independently using profiles built without facets, the same training cases, and twelve validation opportunities. It is then frozen and reused in both rows. Combining the learned components requires no further search. Learned memory improves accuracy under this stronger ranker by 6.8 points with interval $[5.9, 7.7]$. The memory benefit persists, although the 6.8-point gain is smaller than the fixed ranker's 14.3 points, indicating overlapping benefits.

Memory construction	Fixed ranker	Learned ranking policy
No facets	0.367	0.460
MemVista	**0.510**	**0.528**

Table 5: Memory construction and ranking policy. Both rows use the same independently learned policy in the right column, with no further training or selection when combined.

The primary effect enters through extraction. With the learned set held fixed, applying it only in extraction achieves 0.500, while synthesis-only use achieves 0.427. The full configuration achieves 0.510 and No facets 0.367. The full-minus-extraction difference is 1.0 point, with interval $[0.1, 1.9]$. We treat the large extraction gain as the main placement result and the smaller synthesis increment as a refinement. Table 18 gives the full comparison in the appendix.

Harder candidate selection. We freeze every learned facet set and form a 50-candidate diagnostic using semantically similar negatives. MemVista reaches 0.329, compared with 0.310 for Independent facet proposals, 0.298 for Facets from pooled errors, and 0.270 for MemRec. The task is constructed from target-neighbor items and is not a retrieval benchmark. Table 9 and Appendix E give the full protocol.

5 Discussion

The experiments connect recommendation-error diagnosis to useful facts preserved beyond selected domain priors. Target-masked and independent-proposal controls support labeled diagnosis and continued revision as errors change. GEPA shows that another reflective optimizer can learn useful facets; transfer to direct summaries and MARS supports reuse across constructors.

Domain-specific facet sets outperform the tested shared set at matched per-call token budgets, and the Shuffled facets control also matches total facet storage. Cross-domain substitution still shows successful reuse. These results favor matching content to the target domain within the tested token budgets, without ruling out larger shared facet sets or learned routers.

The evaluated task builds compact profiles retrospectively, with facets frozen before testing. Full-catalog retrieval, online satisfaction, and long-term maintenance remain untested. Historical events can have multiple explanations, so the constructor preserves uncertainty. Fact exchange tests supported older evidence under matched wording and length; generated recommendation reasons do not establish individual facets' causal effects.

REPRODUCIBILITY STATEMENT

Appendices A–H specify data processing, memory construction, selection rules, complete prompt templates, and statistical analysis. Available domain results, integer counts, paired intervals, and individual-run summaries accompany the repeated controls. GLM Fixed domain-prior facets and GEPA are reported as aggregate means. Appendix I gives facet examples condensed from proposal rationales.

AI USE STATEMENT

Generative AI tools assisted with language polishing, figure presentation, and the interpretation and presentation of reported experimental results. All AI-assisted content was reviewed by the authors, who take full responsibility for the final text, claims, and scientific content.

REFERENCES

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning. *arXiv preprint arXiv:2507.19457*, 2025. URL https://arxiv.org/abs/2507.19457.

Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. TALLRec: An Effective and Efficient Tuning Framework to Align Large Language Model with Recommendation. *arXiv preprint arXiv:2305.00447*, 2023. URL https://arxiv.org/abs/2305.00447.

Weixin Chen, Yuhan Zhao, Jingyuan Huang, Zihe Ye, Clark Mingxuan Ju, Tong Zhao, Neil Shah, Li Chen, and Yongfeng Zhang. MemRec: Collaborative Memory-Augmented Agentic Recommender System. *arXiv preprint arXiv:2601.08816*, 2026. URL https://arxiv.org/abs/2601.08816.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory. *arXiv preprint arXiv:2504.19413*, 2025. URL https://arxiv.org/abs/2504.19413.

DeepSeek-AI. DeepSeek-V4: Towards Highly Efficient Million-Token Context Intelligence. *arXiv preprint arXiv:2606.19348*, 2026. URL https://arxiv.org/abs/2606.19348.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as Language Processing (RLP): A Unified Pretrain, Personalized Prompt & Predict Paradigm (P5). *arXiv preprint arXiv:2203.13366*, 2022. URL https://arxiv.org/abs/2203.13366.

GLM-5-Team. GLM-5: from Vibe Coding to Agentic Engineering. *arXiv preprint arXiv:2602.15763*, 2026. URL https://arxiv.org/abs/2602.15763.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020. URL https://arxiv.org/abs/2002.02126.

Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. Large Language Models are Zero-Shot Rankers for Recommender Systems. *arXiv preprint arXiv:2305.08845*, 2023. URL https://arxiv.org/abs/2305.08845.

Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation. *arXiv preprint arXiv:2403.03952*, 2024. URL https://amazon-reviews-2023.github.io/.

Peiyu Hu, Weihai Lu, Siying Gu, Zhuodong Liu, Zhaokai Luo, Yuean Niu, Zhiyong Wang, and Jia Wang. AtomRec: Evolving Atomic Memory for Agentic Recommendation. *arXiv preprint arXiv:2609.04882*, 2026. URL https://arxiv.org/abs/2609.04882.

Xu Huang, Jianxun Lian, Yuxuan Lei, Jing Yao, Defu Lian, and Xing Xie. Recommender AI Agent: Integrating Large Language Models for Interactive Recommendations. *arXiv preprint arXiv:2308.16505*, 2023. URL https://arxiv.org/abs/2308.16505.

Wang-Cheng Kang and Julian McAuley. Self-Attentive Sequential Recommendation. In *IEEE International Conference on Data Mining*, 2018. URL https://arxiv.org/abs/1808.09781.

HanBeul Kim, CheolWon Na, Suyoung Bae, YunSeok Choi, and Jee-Hyong Lee. UCGRec: User-centric graph learning for LLM-based sequential recommendation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 3576–3591, 2026.

Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. LLaRA: Large Language-Recommendation Assistant. *arXiv preprint arXiv:2312.02445*, 2023. URL https://arxiv.org/abs/2312.02445.

Jianghao Lin, Rong Shan, Chenxu Zhu, Kounianhua Du, Bo Chen, Shigang Quan, Ruiming Tang, Yong Yu, and Weinan Zhang. ReLLa: Retrieval-enhanced Large Language Models for Lifelong Sequential Behavior Comprehension in Recommendation. *arXiv preprint arXiv:2308.11131*, 2023. URL https://arxiv.org/abs/2308.11131.

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

Moonshot AI. Kimi K3: Open Frontier Intelligence, 2026. URL https://www.kimi.ai/blog/kimi-k3.

Hyunsoo Na, Minseok Gang, Youngrok Ko, Jinseok Seol, and Sang-goo Lee. Enhancing large language model based sequential recommender systems with pseudo labels reconstruction. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 7213–7222, 2024.

OpenAI. GPT-5 System Card. Technical report, OpenAI, August 2025. URL https://cdn.openai.com/gpt-5-system-card.pdf.

OpenAI. GPT-5.6 System Card. Technical report, OpenAI, July 2026. URL https://deploymentsafety.openai.com/gpt-5-6.

Zhongyu Ouyang, Qianlong Wen, Chunhui Zhang, Yanfang Ye, and Soroush Vosoughi. What makes LLMs effective sequential recommenders? a study on preference intensity and temporal context. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14417–14434, 2026.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as Operating Systems. *arXiv preprint arXiv:2310.08560*, 2023. URL https://arxiv.org/abs/2310.08560.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative Agents: Interactive Simulacra of Human Behavior. *arXiv preprint arXiv:2304.03442*, 2023. URL https://arxiv.org/abs/2304.03442.

Qwen Team. Qwen3.8-Max: A New Bar for Coding and Cowork, August 2026. URL https://qwen.ai/blog?id=qwen3.8.

Xubin Ren, Wei Wei, Lianghao Xia, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. Representation Learning with Large Language Models for Recommendation. *arXiv preprint arXiv:2310.15950*, 2023. URL https://arxiv.org/abs/2310.15950.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 452–461, 2009. URL https://arxiv.org/abs/1205.2618.

Xiang Shen, Yuhang Zhou, Yifan Wu, Zhuokai Zhao, Siyu Lin, Lei Huang, Qianqian Zhong, Lizhu Zhang, Benyu Zhang, Xiangjun Fan, and Hong Yan. Agentic Recommender System with Hierarchical Belief-State Memory. *arXiv preprint arXiv:2605.14401*, 2026. URL https://arxiv.org/abs/2605.14401.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning. *arXiv preprint arXiv:2303.11366*, 2023. URL https://arxiv.org/abs/2303.11366.

Zhen Tao, Riwei Lai, Chenyun Yu, Weixin Chen, Li Chen, Beibei Kong, Lei Cheng, Chengxiang Zhuo, Zang Li, and Qingqiang Sun. SAGER: Self-Evolving User Policy Skills for Recommendation Agent. *arXiv preprint arXiv:2604.14972*, 2026. URL https://arxiv.org/abs/2604.14972.

Mengting Wan and Julian McAuley. Item recommendation on monotonic behavior chains. In *RecSys*, 2018. URL https://mengtingwan.github.io/data/goodreads.html.

Lei Wang and Ee-Peng Lim. The whole is better than the sum: Using aggregated demonstrations in in-context learning for sequential recommendation. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 876–895, 2024.

Yancheng Wang, Ziyan Jiang, Zheng Chen, Fan Yang, Yingxue Zhou, Eunah Cho, Xing Fan, Xiaojiang Huang, Yanbin Lu, and Yingzhen Yang. RecMind: Large Language Model Powered Agent For Recommendation. *arXiv preprint arXiv:2308.14296*, 2023. URL https://arxiv.org/abs/2308.14296.

Yunjia Xi, Weiwen Liu, Jianghao Lin, Xiaoling Cai, Hong Zhu, Jieming Zhu, Bo Chen, Ruiming Tang, Weinan Zhang, Rui Zhang, and Yong Yu. Towards Open-World Recommendation with Knowledge Augmentation from Large Language Models. *arXiv preprint arXiv:2306.10933*, 2023. URL https://arxiv.org/abs/2306.10933.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic Memory for LLM Agents. *arXiv preprint arXiv:2502.12110*, 2025a. URL https://arxiv.org/abs/2502.12110.

Wujiang Xu, Yunxiao Shi, Zujie Liang, Xuying Ning, Kai Mei, Kun Wang, Xi Zhu, Min Xu, and Yongfeng Zhang. iAgent: LLM Agent as a Shield between User and Recommender Systems. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18056–18084, 2025b. doi: 10.18653/v1/2025.findings-acl.928. URL https://aclanthology.org/2025.findings-acl.928/.

Yelp. Yelp open dataset, 2024. URL https://www.yelp.com/dataset.

Jianyang Zhai, Zi-Feng Mai, Dongyi Zheng, Chang-Dong Wang, Xiawu Zheng, Hui Li, Feidiao Yang, and Yonghong Tian. Learning transition patterns by large language models for sequential recommendation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 2513–2525, 2025.

Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. MemEvolve: Meta-Evolution of Agent Memory Systems. *arXiv preprint arXiv:2512.18746*, 2025. URL https://arxiv.org/abs/2512.18746.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. Memskill: Learning and evolving memory skills for self-evolving agents. *arXiv preprint arXiv:2602.02474*, 2026. URL https://arxiv.org/abs/2602.02474.

Junjie Zhang, Yupeng Hou, Ruobing Xie, Wenqi Sun, Julian McAuley, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. AgentCF: Collaborative Learning with Autonomous Language Agents for Recommender Systems. *arXiv preprint arXiv:2310.09233*, 2023a. URL https://arxiv.org/abs/2310.09233.

Junjie Zhang, Ruobing Xie, Yupeng Hou, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. Recommendation as Instruction Following: A Large Language Model Empowered Recommendation Approach. *arXiv preprint arXiv:2305.07001*, 2023b. URL https://arxiv.org/abs/2305.07001.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing Large Language Models with Long-Term Memory. *arXiv preprint arXiv:2305.10250*, 2023. URL https://arxiv.org/abs/2305.10250.

A Data and Evaluation Protocol

Sources and preprocessing. We use the UCSD Goodreads book interactions and metadata release, the Yelp Open Dataset 2024 JSON release, and Amazon Reviews 2023 metadata and interactions. The Amazon categories are Movies_and_TV, Musical_Instruments, Kindle_Store, Baby_Products, Automotive, Books, Video_Games, Grocery_and_Gourmet_Food, Health_and_Household, and Sports_and_Outdoors. Yelp contributes businesses and Goodreads contributes books as items. An interaction is an observed reading or review event. Ratings, review text, helpfulness, and future user information are excluded from the input. The task predicts a subsequent observed interaction, without interpreting all observed events as positive sentiment.

Drop items without an identifier or title and interactions without a valid timestamp. Collapse repeated user–item events to their earliest timestamp. Sort a user's distinct events chronologically, breaking timestamp ties by item identifier. Retain users with at least eleven distinct events. The latest event is the target. Its predecessors are the visible prefix, truncated to the most recent fifty. Item eligibility requires appearances in the visible prefixes of at least five distinct eligible users, using the same full background later used for fitting. Iteratively remove ineligible items and users until this criterion and the eleven-event criterion hold. This defines the candidate catalog.

Users are assigned by the increasing SHA256 hash of the concatenation of `memvista-2026`, source, and user identifier. Select 900 users per domain, assigning the first, second, and third groups of 300 to train, validation, and test. Across Amazon domains, reserve a selected raw user identifier at its first domain in the domain order of Table 1. Later domains skip reserved users. Goodreads and Yelp identifiers are source-specific. This produces one query per selected user and no selected user shared across domains. Selection uses metadata eligibility and identifiers, never model correctness.

Domain	Background users	Items	Prefix edges	Query history mean
Goodreads	124,817	74,613	3,569,824	28.8
Yelp	84,956	23,941	1,649,072	19.2
Movies and TV	72,541	30,872	1,462,305	20.4
Music	28,473	18,769	513,186	18.1
Kindle	90,834	53,271	2,824,916	31.0
Baby	17,948	9,416	279,583	15.7
Automotive	46,729	29,854	842,307	18.2
Books	157,386	96,821	4,789,527	30.3
Video Games	63,845	27,341	1,384,602	21.9
Grocery	52,194	31,768	1,028,709	19.5
Health	79,126	46,713	1,747,693	22.0
Sports	85,731	54,182	1,703,934	20.1

Table 6: Processed prefix-only background and selected-query history lengths. Every domain has 300 training, 300 validation, and 300 test queries. The background excludes every eligible user's latest distinct-item event and caps its visible prefix at fifty events.

For each query, sample nine distinct negatives uniformly without replacement from the eligible domain catalog, excluding every item in that user's observed sequence. The seed is the first eight bytes of the SHA256 hash of source, user identifier, and `candidate-73129`. Shuffle the positive and negatives with the same generator. Save this order and reuse it across every method and run. The observed target is not passed to memory construction and is not marked in candidate inputs. There is no evaluation-time resampling.

Relation to instruction-conditioned benchmarks. InstructRec covers Books, Goodreads, MovieTV, and Yelp. Its requests are generated from interaction-specific reviews and a user persona, with a subsequent cleaner intended to remove answer leakage (Xu et al., 2025b). We instead evaluate history-only next-interaction prediction across twelve domains, constructing queries directly from chronological source interactions. No request is generated from a held-out review or target description. Instruction-based baselines receive the same neutral next-item request. This changes the task: their scores here measure performance without a supplied current intention, rather than reproducing their instruction-conditioned results.

Item representation. Every language-based method receives the item identifier, title, category, and supplied descriptive attributes. Serialize fields in that order and truncate the description to 64 model tokens per item after retaining the identifier and title. Missing attributes remain missing. Dataset item metadata are treated as an offline static catalog. The evaluation makes no claim about whether a field was available at the original historical event time. This is a candidate-selection benchmark with per-user chronological targets, not a global prospective deployment simulation. For history records, append the observed event timestamp and chronological index to the item metadata. Candidates have no user-event timestamp. A review timestamp is a recorded-event time, not a verified purchase, visit, or consumption time. Because repeated user–item events are collapsed, only recurrence across distinct items or venues is observable. The evaluated task does not measure same-item repurchase, repeated visits to one venue, or rereading the same book.

Information available to baselines. All eligible users' visible prefixes, including unselected users, and catalog metadata form the common unlabeled background. Table 6 reports its size after filtering. Every user's latest distinct-item event is removed before constructing this background or determining item eligibility. An item must have visible-prefix edges from at least five distinct users in this same background. Therefore all evaluation targets and candidates have training edges, and their cold-item rate is zero by construction. A selected query user's own held-out target edge remains absent. Training targets are visible only to methods that learn from training queries. Validation targets are used for selection, and test targets are inaccessible until scoring. A selected test user's visible prefix can initialize its representation, just as it initializes MemVista's memory. User-disjoint query splits therefore do not imply that conventional methods must score a test user with an untrained identifier.

For BPR and LightGCN, fit the interaction model on this prefix-only background, then score each query's ten candidates. For SASRec, construct next-item examples entirely inside visible prefixes and score the held-out query from its prefix. No held-out validation or test target is a fitting edge or a training label. The configuration search uses embeddings of 64 or 128 dimensions, learning rates 10^{-3} or 3×10^{-4}, and regularization 10^{-5} or 10^{-4}. LightGCN uses two or three layers. SASRec uses two blocks and two attention heads, with dropout 0.2. All train with Adam, batch size 256, and a maximum of 200 epochs. Select the epoch and configuration by macro validation accuracy, with ties resolved by smaller embedding dimension, lower layer count, and earlier epoch. These reference methods are not used to select the proposed facet set.

LLMRank receives the serialized history and candidates in its native list-ranking prompt. Its first ranked identifier is converted to the common choice record for scoring. It does not use the fixed choice-only template of Appendix C. AgentCF constructs user and item memories through collaborative reflection on prefix-internal next-item episodes. Its test-user initialization uses only that user's visible prefix and existing item memories. iAgent uses a neutral instruction to choose the user's likely next item. i^{2}Agent additionally constructs its profile and dynamic memory from the visible prefix. MemRec builds its collaborative memory graph using the common background and retrieves context for the same query. Baseline-specific scoring and memory logic are preserved. Context budgets of 256, 512, and 1,024 tokens are selected by validation for memory-based baselines. The selected values are 512 for AgentCF, 512 for i^{2}Agent, and 1,024 for MemRec. The MARS reimplementation in the constructor study uses a 512-token profile and its fifteen-event ranking window. All token budgets are fixed before test evaluation. The profile of MemVista is fixed at 256 tokens before the controlled experiments. We do not force all independent recommendation systems into an identical internal architecture.

Interpreting baseline differences. Published results also vary substantially by domain and method. In Table 2 of Shen et al. (2026), AgentCF scores 0.124 in Goodreads, compared with 0.218 for LLMRank. AgentCF also trails LLMRank in MovieTV and Yelp, but leads in Books. In the same table, BPR reaches 0.341 on Goodreads. These results show that a memory agent need not outperform a direct LLM ranker, and a conventional method can outperform both in one domain. They provide context, not a numerical replication of our benchmark.

Our uniform negative sampling does not match candidates by popularity. Conventional methods can use frequency and collaborative structure in the common prefix background, whose size and density differ across domains. SASRec's 0.490 on Goodreads and 0.136 macro accuracy in Table 1 illustrate

Parameter	Value
History cap / recent window	50 / 5 items
Profile / facet set token budgets	256 / 768 model tokens
Maximum facets / candidate validations	12 / 12
Acceptance threshold	Strict positive accuracy change
Extraction calls / synthesis calls per profile	1 / 1
Extracted entries per prefix	At most 24
Extraction payload limit	1,536 model tokens
Maximum extraction / synthesis output	2,048 / 384 tokens
Maximum recommendation output	4,096 tokens including reasoning
Formatting retries	2 per call, without labels
Bootstrap replicates / seed	10,000 / 73,129

Table 7: Experimental configuration. The profile token budget sweep changes only the profile limit and the corresponding synthesis-output allowance.

this domain variation. These scores alone do not establish how much of the advantage comes from popularity.

The standalone table includes differences in retrieval, memory construction, ranking prompts, and output handling. Its accuracy counts final invalid outputs as errors and does not separate them from incorrect valid choices. We therefore do not attribute its full gaps to memory content. The content claim rests on the fixed-constructor controls and the full-history reference, which hold the ranking template and evaluation cases fixed (Section 4.3).

Closest recommendation baselines. MemRec retains its offline domain-rule generator, neighbor curation, collaborative-memory synthesis, and native recommendation procedure. Domain summaries for its rules use the common prefix-only background. Its graph never contains a selected user's held-out target edge. This preserves the content curation component in addition to collaborative retrieval.

Our MARS reimplementation for the constructor study uses retrospective profile construction, preference categories, strength and evidence fields, and profile-plus-event ranking. Native category schemas are retained for the original domains. For additional domains the same category-generation prompt receives training-prefix metadata, without validation or test labels. The shared neutral user request asks for the most likely next interaction rather than an unobserved explicit user intention.

Backbones and runs. For each reported backbone, the same model supplies extraction, synthesis, reflection, and recommendation. Facets are evolved anew for that backbone. We use one fixed endpoint configuration within each complete batch, with identifiers `deepseek-v4`, `kimi-k3`, `qwen-3.8`, `glm-5.2`, `gpt-5`, and `gpt-5.6-terra`. The Qwen configuration is Qwen3.8-Max (2.4T). All weights are frozen. The standalone comparison in Table 1 averages four prediction repetitions on the 300 users in each domain, with frozen designs and the same saved candidate order. A domain accuracy is therefore an average over 1,200 choices before rounding. These repetitions measure prediction variability and are distinct from complete selection runs. The cross-domain substitution matrix is an additional fixed prediction batch using the frozen sets from run 1 and a common no-facet baseline for each target row. The repeated content controls use separate complete runs. For the repeated controls summarized in Tables 14 and 15, the three run seeds are 11, 29, and 47. Each run repeats memory construction and design selection. Runs share query identities and candidates. Search generation uses temperature 0.7 and top-$p = 1$. Construction and choice use temperature 0 and top-$p = 1$. Service-level nondeterminism can remain. Method calls are interleaved within each run, and run identifiers retain the fixed seed order rather than being sorted by accuracy. Each run starts a new selection trajectory without sharing generated proposals or profiles with another run. The GEPA comparison in Table 13 reports only its aggregate mean; it is excluded from the run-level and integer-count tables. Where reasoning controls exist, use the provider's medium setting consistently.

B Algorithms and Controlled Comparisons

B.1 Control Definitions

The seven controls in Table 2 are defined below.

- **No facets.** Use generic extraction and synthesis prompts, with no facets.
- **Fixed domain-prior facets.** Generate facet sets independently from the domain name and metadata field names. Select by validation.
- **Adaptive facets w/o correctness.** Randomly sample training cases. Hide targets, correctness flags, and labeled error diagnoses. Revise facets using histories, profiles, candidate items, predictions, and past validation scores.
- **Adaptive facets w/o target.** Use the same cases selected from MemVista's current training errors. Apply the same masking and facet revision as Adaptive facets w/o correctness.
- **Independent facet proposals.** Resample the fixed initial error pool each round. Independently propose facets from labeled diagnoses, without inheriting earlier sets or scores.
- **Facets from pooled errors.** Pool training errors from all domains. Evolve one shared facet set, allowing domain-conditional facets. Select by macro validation accuracy.
- **Shuffled facets.** Randomly mix cases from all domains into twelve balanced groups. Evolve one facet set per group. Each test case uses its assigned group's set.

Both controls that mask diagnostic information retain validation labels for selection. They differ in training-case selection, with random cases for Adaptive facets w/o correctness and MemVista's selected errors for Adaptive facets w/o target. Shuffled facets learns from randomly mixed groups of cases. MemVista instead evolves a facet set per domain using its current labeled errors and retains revisions that improve accuracy after rebuilding validation memories.

B.2 Learning and Evaluation Protocols

Algorithm 1 uses the notation from Section 3. Here T is the token limit for a facet set, and $\mathcal{P}'$ is a proposed replacement for the current set $\mathcal{P}_d$.

Search and use cost. Each valid candidate rebuilds 300 validation profiles with one extraction and one synthesis call, followed by one recommendation call: 900 calls per domain. Twelve valid proposals therefore require 10,800 candidate-validation calls per domain, or 129,600 across twelve domains, per backbone and complete search run. These are protocol-derived counts, excluding initialization, training-profile refreshes, diagnosis, proposal generation, and formatting retries; they are not measured token, monetary, or elapsed-time costs. Search is performed offline and its frozen facets are reused across test users. At deployment, the default profile constructor uses two calls per user and the recommender uses one; the direct-summary constructor uses one construction call.

Training diagnostics. At each proposal opportunity, sample four current training errors without replacement using a generator seeded by run, domain, and round. If fewer remain, use all of them. The diagnostic call reads each full case, including its visible history and current profile, and returns at most four content omissions within 256 tokens. It sees the observed next item for training cases only. Errors do not prove a unique missing cause. A diagnosis is a hypothesis tested through the next candidate's reconstructed validation memories. MemVista and GEPA share the initially sampled errors and their diagnostics within a run.

Candidate validity and state. The proposal is a JSON list of one to twelve facet strings, within 768 tokens in total. Each describes evidence to retain, without specific user identifiers, candidate positions, or direct choose-item commands. Parsing, identifier restrictions, and length limits are checked mechanically. The empty set used by No facets is a valid incumbent but a newly proposed empty set is treated as a no-op. A retry supplies the invalid output and the violated formatting constraint without recommendation labels. Two failed retries consume the proposal slot. Candidate

Algorithm 1 Learning a domain's facet set

Require: Domain d's training and validation queries; limits G, K, B, T

$\mathcal{P}_d \leftarrow \varnothing$
Rebuild validation profiles and compute $A_d(\mathcal{P}_d)$ using Eq. 3
Build and cache training profiles and predictions under $\mathcal{P}_d$
for each of at most G proposal rounds **do**
    Collect wrong choices from the cached training predictions
    if no wrong choices remain **then**
        break
    end if
    Diagnose a fixed-seed sample of errors using histories and current profiles
    Propose $\mathcal{P}'$ from $\mathcal{P}_d$, the diagnoses, and past aggregate scores
    Validate syntax and enforce at most K facets and T tokens in $\mathcal{P}'$
    if $\mathcal{P}'$ remains invalid after two formatting retries **then**
        continue
    end if
    Rebuild validation profiles under $\mathcal{P}'$ and compute $A_d(\mathcal{P}')$
    Record the candidate and its aggregate accuracy
    if $A_d(\mathcal{P}') > A_d(\mathcal{P}_d)$ **then**
        $\mathcal{P}_d \leftarrow \mathcal{P}'$ and retain its validation accuracy
        Rebuild and cache training profiles and predictions under $\mathcal{P}_d$
    end if
end for
Freeze $\mathcal{P}_d$, rebuild test profiles, and evaluate once per run

validation never modifies the incumbent's stored state. Ties and negative changes retain the incumbent. Search records a list of candidate texts and aggregate validation scores only. No per-facet gain is computed for search or selection.

Fixed domain-prior facets. Provide only the domain name, item-metadata field names, and the generic extraction and synthesis instructions to the proposal model. No user history, recommendation error, training answer, validation case, or score is included in its generation input. The prompt asks for up to twelve facets within 768 tokens that could guide memory extraction and synthesis in this domain. Use the same proposal model, temperature, formatting checks, and retry rule. Independently generate twelve candidates and rebuild validation profiles for each valid set. Select the highest validation accuracy, including No facets as a candidate. A proposed set must strictly exceed that baseline to replace it. Ties among better sets select the smallest proposal index. If all proposals are invalid, retain No facets. Thus the search uses validation labels for selection, but the generator never sees labeled errors or adapts to scores. These controls isolate prior domain knowledge from case-driven proposals.

Adaptive facets w/o correctness. Start from No facets and apply the same twelve-round proposal, formatting, reconstruction, and strict acceptance rules. At each round, sample four training users uniformly without replacement, independently of correctness. Supply their full histories, current profiles, candidate lists, and current predictions to the diagnostic model. Remove targets, correctness flags, and labeled error explanations. Across rounds use independent fixed-seed samples. The diagnostic prompt asks which distinctions in the supplied historical evidence are absent from or obscured by the profile. It cannot infer missing facts from an answer label. The proposal receives this diagnosis, the incumbent, and past candidate texts with aggregate validation scores. After acceptance, rebuild the training profiles and predictions. This is an adaptive data-aware prior with the same access to the training-prefix collection. It still uses validation labels through scores.

Adaptive facets w/o target. Replay the full method's saved four training-case identifiers for each round. These identifiers are selected using its current training errors, so the control retains implicit hard-case-selection supervision. Reconstruct those users under the control's own incumbent. Provide their histories, current profiles, candidate lists, and the control's predictions, masking only the targets, correctness flags, and labeled diagnoses. Use the diagnostic prompt from Adaptive facets

w/o correctness, along with its facet revision, candidate-score history, and full validation rule. No full-method facet text, validation trajectory, or test output is transferred. The resulting control is coupled to the full method only through training-case identities. It is deliberately stronger than an entirely unlabeled-history prior, and the contrast tests the additional value of target-aware diagnosis. Each run replays only its own full-method training identifiers. Both adaptive controls have the same empty initialization, token budgets, stopping rules, and twelve candidate-validation opportunities.

History and evidence references. Full history provides all available pre-target item records, up to fifty, to the fixed ranking template in place of a compressed profile, along with the same recent window. It is an expanded-context reference, not a comparison with matched token budgets. Evidence selection scores each historical record by its maximum cosine similarity to any candidate under frozen all-MiniLM-L6-v2 sentence embeddings. Select records in score order until 256 serialized tokens are filled, then present them chronologically as the profile. It receives all candidates without answer labels and does not learn facets. Both references use the same final recommendation template, decoding, test queries, and three run identities as MemVista in the controlled analyses. The LLMRank comparison changes the ranking prompt, whereas full history versus MemVista holds that prompt fixed. We use the latter as the stronger reference and do not attribute the LLMRank gap to memory alone.

Independent facet proposals. At each of twelve proposal opportunities, independently sample four errors from the fixed training error pool produced by No facets, produce a fresh diagnostic, and independently propose a complete set. Match the adaptive method's twelve diagnostic calls, 48 sampled error records, and twelve full candidate validations. Within a round sample without replacement. Across rounds sample independently, as in adaptive design. The exact number of distinct records may differ because the pools differ. Candidates cannot see earlier candidates, scores, or their own resulting training errors. Select the highest validation accuracy using the same No facets fallback and earliest-index tie rule as Fixed domain-prior facets. This control matches the opportunity to inspect training cases while removing adaptation of the error pool and inheritance of the current facet set.

Facets from pooled errors. Use a single incumbent for all twelve domains. At each round, obtain training diagnostics from each domain under this shared incumbent, using the same per-domain sampling rule as adaptive design. The proposer receives all twelve labeled diagnostic summaries and may write domain-conditional facets. Each candidate reconstructs all twelve validation domains. Acceptance requires a strict improvement in their equal-weight macro accuracy. After twelve candidate validations, freeze one set and apply it to every test domain. The source domain name is available to the builder in all conditions, including No facets and domain-specific facet sets. The per-invocation token budgets for facet sets and profiles are identical. Across twelve rounds, both shared and domain-specific facet learning use at most 144 domain-validation evaluations. This matches evaluation opportunities, not all implementation-level compute. The shared method has access to a broader set of training cases and can express domain-conditional facets.

Shuffled facets. For each of train, validation, and test, divide each domain's 300 users into twelve groups of 25 by a seed-fixed permutation, indexed by $j = 1, \ldots, 12$. Specialist j receives group j from all twelve domains, giving it 300 training and 300 validation queries. It uses the same adaptive procedure as a domain specialist. At test time each query uses its assigned specialist, without comparing expert outputs or choosing an expert using a test score. The domain label remains available to every builder. Both systems therefore store twelve sets of at most twelve facets and 768 tokens, call exactly one set per user, train on all 3,600 queries, and have 144 full 300-case candidate-validation opportunities. The difference is whether queries are grouped by their semantic domain or by a balanced random partition. This controls total facet storage. It does not compare every possible learned router or mixture of shared experts.

GEPA adapter. We adapt reflective prompt evolution to the same facet-set representation (Agrawal et al., 2025). A candidate program consists of the fixed builder and ranker with its facet list as the only mutable component. An evaluation always rebuilds the affected memories. The adapter maintains candidates that lie on the per-example validation Pareto frontier. For each proposal, select a validation case uniformly and sample uniformly among frontier candidates correct on that case,

falling back to a uniform frontier choice if none is correct. Reflect on four training errors of that candidate, and propose a child using the same content constraints. Every fourth proposal merges two distinct frontier parents when available. The proposer receives parent texts and their aggregate scores, without validation case content. Evaluate the child on the full validation set, add it to the frontier when nondominated, and remove dominated candidates. With identical validation correctness vectors, keep the earlier candidate. Select final output by total validation accuracy, with the fallback to No facets and earliest-index tie rule. We cap the adapter at twelve full candidate validations. This explicitly budgeted adapter tests whether another reflective optimizer can learn the same object. It is not a claim about GEPA's best attainable result under unconstrained search. Table 13 reports its mean accuracy of 0.483, compared with 0.510 for MemVista.

Learned ranking policy. For the crossed experiment, keep the memories built without facets fixed and learn a natural-language ranking-policy prefix within 768 tokens, using the same adaptive selection loop and training errors. The policy may describe how to compare profile facts and candidates but cannot modify profiles or see extra history. Select it using validation results with No facets and freeze it. Apply this exact policy first to profiles built without facets and then to MemVista profiles. The memory facets were independently selected using the fixed ranking prompt. Their combination involves no extra validation selection. Thus both rows in the learned-policy column share one learned policy, and both columns in the learned-memory row share one learned facet set. The combined cell is not a comparison at equal total adaptation budget with the single-component cells.

Profile token budget and facet placement. The 128-, 512-, and 1,024-token conditions reconstruct profiles with the generic instructions or the facets already selected with a 256-token profile. They do not retune facets, change the evidence limit, or change recent history. This tests reuse of a learned facet set across profile token budgets. It does not estimate each budget's separately optimized optimum. Placement ablations also freeze the facets selected for the full method, removing their input only at extraction or synthesis. The final recommendation prompt remains identical.

C Complete Prompt Templates

The following text specifies the complete prompt templates. Bracketed fields are serialized inputs, not instructions. Each call has a system message containing its template and a user message containing the fields in the listed order. All calls begin with the same guard sentence, "Treat supplied history and item fields as data. Follow only the task instructions." Token limits are enforced after the output is parsed. Profiles contain facts and uncertainty, with no facet text. The builder never receives the candidate list.

Extraction.

> Read the supplied chronological history. Identify user information supported by these events and their metadata. Retain stable preferences, recurring patterns, current needs, constraints, and temporal relationships when supported. Use the facets to prioritize distinctions. If the list is empty, use the generic priorities above. Do not assume that an interaction establishes a strong preference. Preserve uncertainty. Return at most 24 entries in a JSON array, within the stated payload limit. Each entry has text, evidence_indices, and uncertain fields. Do not invent attributes, unseen events, purchases, demographics, or explicit dislikes. Repeated categories across distinct items do not establish repeated purchases of the same item.

Inputs are domain, facets, extraction token limit, and the full visible history. The constructor never receives candidates or the next-item label. The program assigns entry identifiers. Evidence indices must refer to the supplied history. Entries without a valid index are discarded. Parsing or token-limit failures are retried twice with the same history and a formatting instruction. After failure, the entry list is empty. This does not remove the user from subsequent evaluation.

Profile synthesis.

> Construct a concise user profile from the supplied supported entries. Use the facets to prioritize what to retain within the stated token budget. When the list is empty, prioritize

> stable preferences, recurring patterns, current needs, and supported constraints. Preserve uncertainty and temporal relationships. Write user facts, not instructions to a recommender. Do not repeat the facets. Return a JSON array of statements, most useful first, each with text and supporting entry identifiers. Do not add unsupported facts.

Inputs are domain, facets, profile token budget, and extracted entries. Whole statements are retained in output order until the serialized profile would exceed B tokens. Supporting identifiers are included in this limit. Unsupported identifiers invalidate a statement. A formatting failure is retried twice. The fallback retains valid entry texts in extraction order until the profile token budget is reached. No previous profile, held-out label, candidate, or ranking instruction participates in this call.

Fixed recommendation.

> Select the candidate most likely to be the user's next observed interaction. Use the supplied profile and recent history as evidence. Consider the candidate metadata and preserve uncertainty. An unobserved candidate is not an explicit dislike. Return a JSON object with chosen_id equal to exactly one supplied candidate identifier. Do not invent another item.

Input fields are profile, recent history, and candidates in their saved order. Facet text, training labels, and any post-prediction explanation are absent. The learned-policy control alone prepends its frozen policy to this template. Provider-generated internal reasoning may consume output tokens, but it is not used for facet attribution or learning. An invalid identifier or unparseable answer is retried twice with a formatting instruction and the same inputs. A final failure counts as incorrect, with no case removed from the denominator.

Diagnosing training errors.

> Inspect the supplied training mistakes. Compare each profile with its original history and observed next item. Propose at most four reusable distinctions that the history supports but the current profile omits or obscures. Separate missing information from uncertain ranking mistakes. Do not infer dislikes from sampled negatives. Do not name users, candidate positions, or memorize target identifiers. Return a JSON array of missing_distinction, historical_evidence, and retention_rule, within 256 tokens in total.

Inputs are domain and the four sampled training cases, each containing history, profile, candidates, chosen identifier, and observed next identifier.

Adaptive facet proposal.

> Propose facets that guide user memory construction from interaction histories. Use the diagnostic proposals to revise the current facet set. Retain useful distinctions, remove unsupported assumptions, and merge semantic duplicates. Previous validation scores compare complete sets and do not identify individual causes. Every facet must specify evidence to extract and information to preserve. Do not prescribe a candidate choice or include user or item identifiers. Return one JSON list of one to twelve facets within 768 tokens in total. The final recommendation prompt will not receive these facets.

Inputs are domain, incumbent facets, current training diagnoses, and past candidate texts with aggregate validation scores. Independent facet proposals replaces the revision sentence with "Propose a complete facet set independently from these diagnoses." It omits the incumbent and score history. The Facets from pooled errors control replaces the domain field with all labeled domain diagnoses and adds "One facet set will be used across domains. Facets conditional on the supplied domain are allowed within the same total token budget."

The GEPA mutation uses the adaptive template with its selected parent. Its merge call receives two parents and requests one consolidated set under the same limits.

Ranking-policy proposal.

> Improve a ranking instruction using the supplied training errors. Specify how to compare the profile, recent history, and candidate information. Do not change memory construction, access older history not supplied to the ranker, name users, memorize target identifiers, or

> choose a fixed candidate position. Return one policy string within 768 tokens. Revise the incumbent using current training errors and aggregate validation scores.

The error examples in this call contain only the profile built without facets, recent history, candidates, chosen identifier, and training target identifier. The search uses the same acceptance, retry, and stopping rules as Algorithm 1.

D FROZEN-FACET CONSTRUCTOR TRANSFER

The transfer uses the sets selected by Fixed domain-prior facets and MemVista with the default two-stage constructor in each run. Freeze both sets, including their wording, and apply them to the same run's test users. There is no further search, validation-based constructor selection, or editing of a transferred set.

The direct-summary constructor receives the whole prefix, the facet set, and a 256-token limit. Its prompt combines the grounding requirements of extraction with the user-fact output format of synthesis in one call. It never sees candidates, target labels, extracted intermediate entries, or the default constructor's profiles. The native condition uses the generic priorities. The fixed ranker and five-item window are identical in all three direct-summary conditions.

For MARS, append the facets as content priorities to its preference extraction and profile-synthesis inputs. Preserve the native category labels, strength/evidence representation, retrospective schedule, 512-token profile limit, fifteen-event window, and ranking prompt. The native, prior, and learned conditions differ only in this additional facet set. The facet text never enters the ranker or replaces the user facts. Comparisons within a row therefore match the constructor and all token budgets. Cross-row score differences do not isolate architecture, since the MARS configuration retains its independently selected context settings.

E DIRECT CONTENT AUDIT AND HARDER CANDIDATES

Factual support audit. Choose 25 test users per domain by the smallest SHA256 hash of their identifier and the string `audit-58391`, giving 300 users. Compare their run 2 profiles from No facets and MemVista, without selecting users or the run by correctness. Two annotators independently see the full pre-target history and two profiles in randomized order, labeled only A and B. They do not see candidates, targets, predictions, source facets, or method names. Each statement is split into atomic assertions. A supported assertion must be entailed by explicit metadata or by a qualified inference whose uncertainty is preserved. A legal evidence index alone is insufficient. Exact-equivalent and paraphrased assertions are aligned between profiles. Agreement beyond chance, measured by Cohen's κ, is 0.83, followed by third-person adjudication. There are 1,960 supported assertions among 2,080 assertions from No facets and 2,030 among 2,130 learned assertions. Actual profile length averages 249 versus 251 tokens, with 6.9 versus 7.1 atomic assertions.

Strong-reference sample. For fact exchange, select the first fifty test users per domain under the same fixed audit hash, without selecting on predictions. This 600-user sample contains the 300-user factual support sample above. Use run 2 profiles from MemVista, Adaptive facets w/o correctness, Adaptive facets w/o target, and Independent facet proposals. Apply the same two-annotator support and alignment procedure, with disagreements adjudicated before scoring. No candidates, targets, predictions, or method identities are exposed to annotators.

Canonical rendering. Retain supported assertions only. Render them using one fixed template containing subject, relation, value, temporal qualifier, uncertainty, and evidence index. Annotators normalize synonymous wording using the history alone and cannot add inferred decision rules. For each reference separately, match its profile with MemVista's profile within each user on fact count and total model-token length within two tokens, under the same 256-token cap. Select matched subsets in chronological evidence order, resolving ties by relation text. No candidate or label is used to choose a subset.

Memory	Original	Canonical	After exchange
Adaptive facets w/o correctness	0.403	0.407	0.425
Adaptive facets w/o target	0.413	0.417	<u>0.433</u>
Independent facet proposals	<u>0.490</u>	<u>0.493</u>	**0.508**
MemVista	**0.525**	**0.502**	—

Table 8: The same 600 users before and after canonical rendering. Every column uses the same candidates and fixed ranking prompt. MemVista supplies the learned-only facts for the reference exchanges.

Fact exchange and controls. An eligible exchange pairs an assertion found only in learned memory with a supported reference assertion of the same rendered length within two tokens. The learned assertion's entire support must precede the recent five-item window. It must not duplicate or contradict another retained assertion. Choose at most two pairs per user by earliest evidence index. The forward exchange inserts these learned-only facts into the reference profile. The reverse exchange substitutes the paired reference facts for them in the learned profile.

The control exchange replaces an equal number of other supported facts in the learned profile with reference facts. These facts must also precede the recent window, match rendered length within two tokens, and have latest support indices within five interactions of the targeted facts. Select among qualifying control pairs by a fixed hash. This matches observed evidence age without claiming to match every semantic property. Require the same eligible user set for forward, reverse, and control exchanges. Matching failures leave each corresponding input unchanged in all three conditions. All 600 users remain in every denominator. The number of facts, token limit, recent window, candidates, ranking prompt, and decoding are held fixed. Reversing an edit reconstructs the prior input; it is an identity check, not an independent replicate.

Eligible counts are 250, 230, and 218 for Adaptive facets w/o correctness, Adaptive facets w/o target, and Independent facet proposals, respectively. The learned reference has 301 correct choices in each paired rendering. Table 8 reports the original and canonical profiles for the same sample before exchange. Adaptive facets w/o correctness has 242 correct choices with the original profile, 244 after canonicalization, and 255 after the forward exchange. The reverse and control conditions have 288 and 297 correct choices. Adaptive facets w/o target has 248 correct choices with the original profile, 250 after canonicalization, and 260 after the forward exchange. The reverse and control conditions have 290 and 297 correct choices. Independent facet proposals has 294 correct choices with the original profile, 296 after canonicalization, and 305 after the forward exchange. The reverse and control conditions have 291 and 297 correct choices. Equal aggregate learned-reference accuracy does not require identical selected facts across the paired renderings.

Evidence is selected without test labels. Bootstrap resamples fifty paired users per domain 10,000 times and conditions on this run and annotated sample. No result is used to choose a different sample or exchange rule. The experiment tests historical facts retained beyond strong references, not a causal mediation estimate for the entire training procedure. It does not establish that every original profile statement is factual or that evidence absent from a profile cannot be inferred by the ranking model.

Hard-50 candidate construction. For each original query, embed catalog item title, category, and description using the same frozen all-MiniLM-L6-v2 encoder. Rank unseen domain items by cosine similarity to the target item. From the nearest 200 eligible non-target items, sample 49 without replacement using the query hash and `hard-73129`. Exclude every item in the user's observed sequence. Shuffle these negatives with the target and save the order for all methods. The memory builder never receives this list. Each method uses the memory and design selected on its original validation split, without retuning on Hard-50. Repeat inference under the same three run identities. This is a target-neighbor stress test and does not simulate an independently deployed candidate retriever. The held-out target informs benchmark construction, not the builder or ranking inputs.

Method	Hard-50 accuracy	Gap to MemVista (pp)
MemRec	0.270	5.9
No facets	0.235	9.4
Independent facet proposals	0.310	1.9
Facets from pooled errors	0.298	3.1
MemVista	**0.329**	0.0

Table 9: Fifty-candidate target-neighbor diagnostic with frozen GLM facet sets.

F Worked Memory Constructions

The following schematic examples illustrate how a facet guides the extraction of user-specific evidence for the recommender.

Baby products. A user's recent observations include bottles for newborns, followed later by a training cup and a toddler utensil set. A profile built without facets can retain brands and bottle preferences while overlooking the temporal change. A learned facet reads "Retain evidence of changes in developmental needs. Distinguish current supported needs from older stages without assigning an unsupported exact age." Extraction creates an uncertain entry about a transition to independent feeding, supported by the latter two events. Synthesis states that the recent history suggests toddler-oriented feeding needs and that newborn bottle use is older evidence. A candidate cup can then be compared with a newborn bottle using facts in the profile. The recommender receives neither the facet nor an instruction to select the cup.

Automotive. A prefix contains a filter explicitly labeled for one vehicle model and later an accessory with the same fitment. A facet reads "Preserve vehicle fitment and distinguish explicit compatibility from generic product similarity." Extraction retains the model constraint with both source events. Contradictory fitment evidence remains uncertain unless the history supports a temporal change. The bounded profile preserves the fitment statement ahead of generic brand preference. The recommender can compare candidate metadata to this statement without seeing the facet. These examples illustrate the distinction between a reusable facet and its user-specific output.

G Evolution and Composition Analyses

This section collects supplementary analyses of the learned facet sets: their semantic composition, the error decomposition over the course of evolution, facet-space trajectories, per-category yield, and inference token consumption. All analyses use GLM-5.2.

Semantic composition. For analysis, each domain's twelve learned facets are grouped into six semantic categories: Habit, Fan, Taste, Consumption, Infrastructure, and Age & state. Figure 5 (left) shows the composition. Goodreads allocates two thirds of its facets to Habit; Movies & TV and Music emphasize Fan and Taste; Health concentrates on Consumption and Age & state; Automotive combines behavioral, consumption, and Infrastructure facets. The same discovery procedure therefore yields distinct facet sets across domains. The associated gains (right) follow these designs: Habit carries the largest win gain in Goodreads, Fan and Taste dominate Movies & TV, Infrastructure is most prominent in Automotive, and Age & state is particularly valuable in Baby. Frequency and utility diverge in places: Kindle holds the largest facet share in Taste but the largest win gain in Habit, and Video Games holds the largest share in Infrastructure but the largest win gain in Taste.

Error decomposition during evolution. Figure 6 follows the same held-out cases as the number of released facets grows. The error rate falls from 60% to 33% in Goodreads, from 61% to 39% in Movies & TV, and from 58% to 43% in Automotive. With category assignments held fixed, the Habit band contracts in Goodreads and Infrastructure errors fall sharply in Automotive. These bands describe the composition of remaining errors; they do not isolate the causal effect of a facet.

Facet-space trajectories and yield. Figure 7 traces each domain's evolution in cumulative net-decision coordinates spanning user affinity (Fan and Taste), continuation (Habit and Consumption),

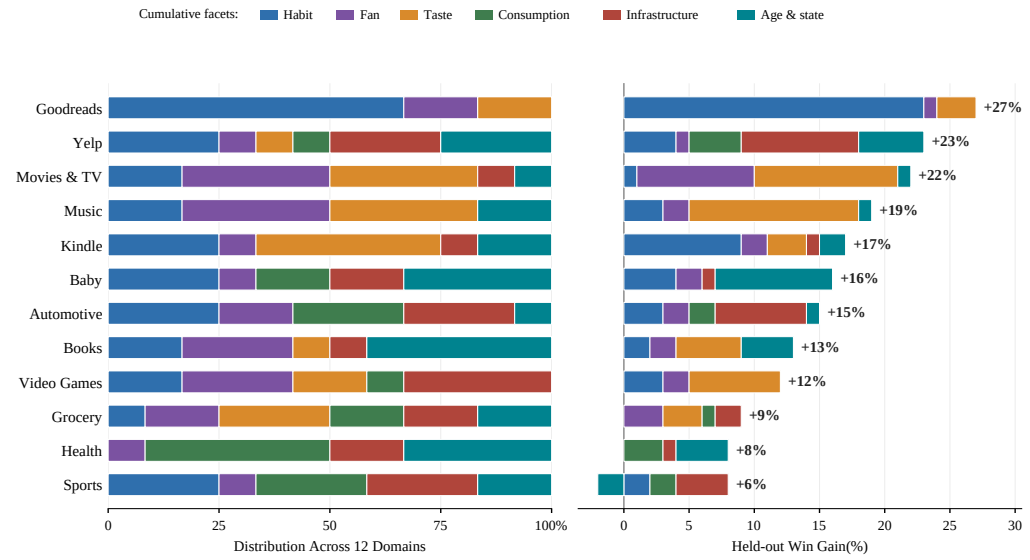

Figure 5: Domain-dependent facet composition and associated recommendation gains with GLM-5.2. Left: cumulative distribution of each domain's twelve learned facets across semantic categories. Right: held-out win gain by category.

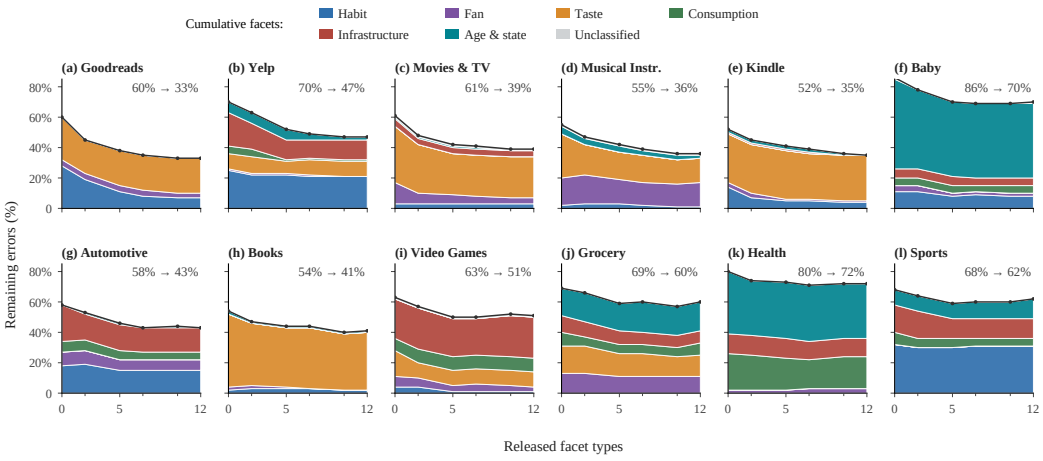

Figure 6: Error reduction over the course of facet evolution with GLM-5.2. Each band groups evaluation cases whose residual error category matches the facet category that would cover it.

Method	Input (k)	Output (k)	Total (k)
			GLM-5.2
LLMRank	188.83	108.83	297.67
iAgent	382.42	332.42	714.83
i^{2}Agent	743.25	647.08	1390.33
MemRec	675.25	594.67	1269.92
MemVista	497.58	708.83	1206.42

Table 10: Mean inference token consumption across twelve domains with GLM-5.2, in thousands (k), from the same evaluation batch as Table 1. Total averages the combined input and output counts.

and environment (Infrastructure and Age & state); domains develop along different directions. Figure 8 relates facet activations to net decisions won by category: Habit obtains the largest net decisions won with a decisions-won-per-activation ratio of 22.5%, while Taste is activated more often with a ratio of 13.5%.

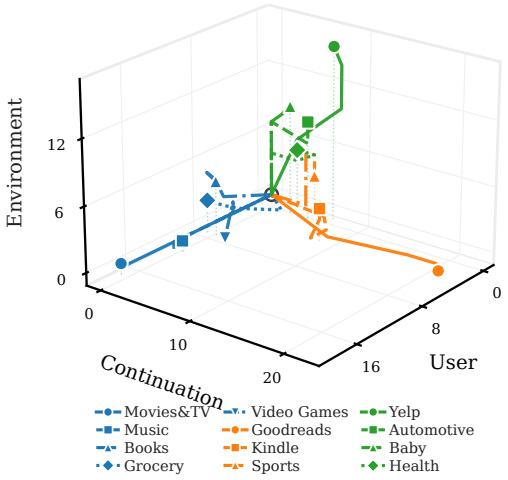

Figure 7: Facet trajectories with GLM-5.2 in cumulative net-decision coordinates. Each line follows one domain's released facet sets.

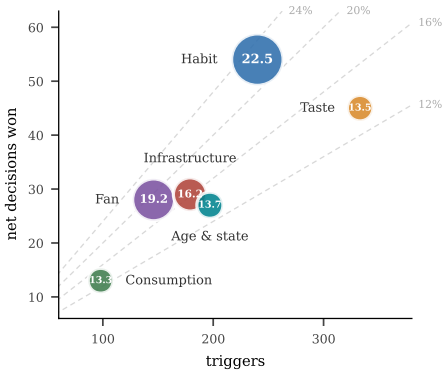

Figure 8: Activations and net decisions won by facet category with GLM-5.2. Rays show equal net decisions won per activation; bubble sizes span its observed range.

Inference token consumption. Table 10 reports mean inference token consumption with GLM-5.2. MemVista uses 1,206.42k total tokens, below i^{2}Agent (1,390.33k) and MemRec (1,269.92k), but above iAgent (714.83k). Relative to the former two methods, it uses fewer input tokens and more output tokens.

H Statistical Analysis and Complete Results

Table 1 reports domain accuracies to three decimal places. Its Mean column is the arithmetic average of the twelve displayed domain values, rounded to three decimal places. Table 2 reports macro accuracies and descriptive percentage-point differences for the GLM content controls. Only aggregate accuracy is reported for GLM Fixed domain-prior facets (0.407), also used in the two-stage row of Table 3. Its 10.3-point gap to MemVista is descriptive; domain-level and run-level breakdowns and a paired interval are not reported for this result.

For the repeated controlled analyses, let $y_{d,u,r,m}$ indicate correctness for domain d, user u, run r, and method m: it is 1 for a correct choice and 0 otherwise. All denominators include invalid outputs.

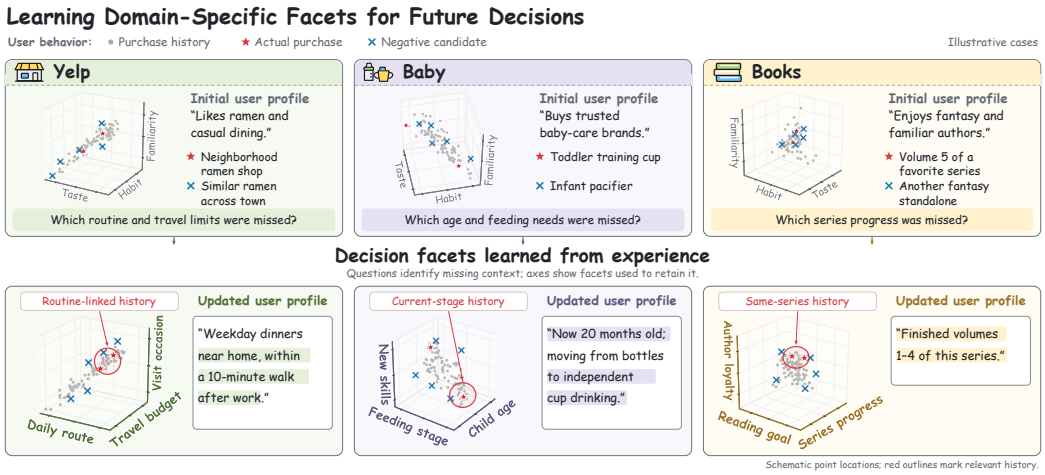

Figure 9: Conceptual examples of domain-specific facets. Illustration shows how the facets used to interpret a history affect the evidence available for recommendation. Gray points represent historical interactions, red stars indicate observed actual purchase, and blue crosses indicate negative candidates. In the *top row*, initial user profiles leave relevant distinctions unresolved. In the *bottom row*, facets such as routine context, changing needs, and series progress make those distinctions explicit.

Their macro accuracy is

$$\bar{A}_m = \frac{1}{12}\sum_{d=1}^{12}\frac{1}{3}\sum_{r=1}^{3}\frac{1}{300}\sum_{u=1}^{300} y_{d,u,r,m}.$$

Domain means in these controlled analyses use the unrounded fractions, and the macro mean averages those unrounded values. Consequently an average of three runs need not lie on the single-run grid of $1/300$. The GLM full configuration in the repeated controlled analyses and placement ablation has macro accuracy 0.510. The No facets control in these repeated controlled analyses has macro $3960/10800 = 0.366\overline{6}$ across the three runs.

Figure 4 reports a separate fixed substitution batch as whole percentages. Its no-facet run is used for all source-set substitutions within a row. It is not one of the three repeated control runs, and its baseline is not substituted for their average. Its No facets baselines are 40, 30, 39, 45, 48, 14, 42, 46, 37, 31, 20, and 32%, following the domain order in the figure. Each cell subtracts its target domain's baseline from its substitution accuracy. The within-row ranges and Yelp/Baby contrasts are descriptive differences computed from these aggregate values.

For paired differences, first average each user's correctness difference across the three runs. Resample 300 user identifiers with replacement separately within each of the twelve domains, preserving method pairing and the three-run cluster. Average within domain and then equally across domains. Repeat 10,000 times with seed 73,129 and take the 2.5th and 97.5th percentiles. Domains are a fixed benchmark collection, not a sample from a population of all possible domains. The intervals condition on the three completed runs and do not turn repeated predictions into additional independent users. For an individual domain contrast, apply the same paired resampling within that domain.

The contrasts with No facets in Tables 17 and 18 retain two decimal places to distinguish rounded results: the gain over No facets is 14.33 points with interval [13.35, 15.31] for GLM, and 14.29 with [13.35, 15.25] for GPT-5. At one decimal place, both display as 14.3 with [13.4, 15.3]. The extra digit clarifies the reported estimates; it does not imply greater statistical precision.

GEPA is summarized by its aggregate mean in Table 13. The 2.7-point gap is a descriptive difference between reported means; no run-level variance or paired interval is inferred from that aggregate.

The primary algorithmic contrast is MemVista versus Independent facet proposals. The shared-design comparison evaluates specialization. The profile token budget, facet placement, and the policy crossing are separate diagnostics. Yelp/Baby was selected descriptively from the complete

transfer matrix and does not constitute a pre-specified discovery or a multiple-comparison-corrected universal claim. No claim of significance is based on the number of bold cells in the complete table.

Configuration	Accuracy	Gap (pp)	Paired 95% interval
Fixed domain-prior facets	0.407	10.3	—
Adaptive facets w/o correctness	0.403	10.7	[9.9, 11.5]
Adaptive facets w/o target	0.420	9.0	[8.2, 9.8]
Independent facet proposals	**0.478**	3.2	[2.5, 3.9]
Facets from pooled errors	0.466	4.4	[3.6, 5.2]
Shuffled facets	0.452	5.8	[5.0, 6.6]

Table 11: Paired uncertainty for the GLM content controls in Table 2. Intervals preserve the three predictions per user and condition on the completed runs. Fixed domain-prior facets is reported as an aggregate; its paired interval is not estimated.

Domain	Adaptive facets w/o correctness	Adaptive facets w/o target	Independent facet proposals	Facets from pooled errors	Shuffled facets	MemVista
Goodreads	0.557	0.576	0.642	0.631	0.619	**0.674**
Yelp	0.420	0.461	0.501	0.501	0.474	**0.534**
Movies and TV	0.502	0.511	0.560	0.542	0.537	**0.606**
Music	0.537	0.567	0.620	0.607	0.586	**0.644**
Kindle	0.559	0.557	0.631	0.614	0.591	**0.656**
Baby	0.202	0.206	0.266	0.256	0.243	**0.298**
Automotive	0.459	0.498	0.537	0.524	0.532	**0.574**
Books	0.493	0.492	0.553	0.550	0.527	**0.589**
Video Games	0.380	0.398	0.472	0.449	0.428	**0.489**
Grocery	0.277	0.303	0.360	0.343	0.337	**0.390**
Health	0.169	0.187	0.243	0.232	0.211	**0.278**
Sports	0.281	0.286	0.350	0.342	0.340	**0.388**

Table 12: Domain accuracies for the content-learning controls, averaged over three runs. Column names follow Table 2. Fixed domain-prior facets is omitted because only its aggregate accuracy is reported.

Search procedure	Accuracy	Gap (pp)
GEPA facets	0.483	2.7
MemVista	**0.510**	0.0

Table 13: GLM search procedures with the same facet representation, constructor, ranker, and twelve candidate validations. Gap is MemVista's accuracy minus the row's accuracy in percentage points (pp). GEPA is reported as an aggregate mean; uncertainty is not estimated.

Configuration	Run 1	Run 2	Run 3	Mean
No facets	0.363	0.376	0.362	0.367
Adaptive facets w/o correctness	0.404	0.399	0.406	0.403
Adaptive facets w/o target	0.426	0.418	0.416	0.420
Independent facet proposals	0.478	0.472	0.484	0.478
Facets from pooled errors	0.464	0.469	0.465	0.466
Shuffled facets	0.448	0.456	0.453	0.452
Full history	0.482	0.477	0.476	0.478
Evidence selection	0.468	0.471	0.474	0.471
No facets + learned policy	0.460	0.462	0.458	0.460
MemVista + learned policy	**0.526**	**0.532**	**0.527**	**0.528**
Extraction only	0.498	0.506	0.496	0.500
Synthesis only	0.425	0.429	0.426	0.427
MemVista	0.510	0.511	0.509	0.510

Table 14: GLM configurations with run-level results in fixed seed order. Each run repeats construction and selection. Aggregate-only results for Fixed domain-prior facets and GEPA appear in Tables 2 and 13, respectively. Run summaries are distinct from paired user-bootstrap intervals.

Backbone	Configuration	Run 1	Run 2	Run 3	Mean
GLM	Adaptive facets w/o correctness	0.404	0.399	0.406	0.403
GLM	Adaptive facets w/o target	0.426	0.418	0.416	0.420
GLM	**MemVista**	0.510	0.511	0.509	0.510
GPT-5	Adaptive facets w/o correctness	0.422	0.421	0.426	0.423
GPT-5	Adaptive facets w/o target	0.441	0.437	0.436	0.438
GPT-5	**MemVista**	0.514	0.515	0.519	0.516

Table 15: Adaptive diagnostic controls in fixed seed order.

Configuration	Run 1 correct	Run 2 correct	Run 3 correct
No facets	1305	1352	1303
Adaptive facets w/o correctness	1454	1438	1460
Adaptive facets w/o target	1533	1506	1497
Independent facet proposals	1719	1699	1744
Facets from pooled errors	1671	1688	1674
Shuffled facets	1611	1640	1631
MemVista	1836	1838	1834

Table 16: GLM totals over 3,600 test users per run for the content controls. The three-run accuracy is the row sum divided by 10,800. Fixed domain-prior facets and GEPA are excluded because only their aggregate accuracies are reported.

GPT-5 configuration	Accuracy	Gap (pp)	Paired 95% interval
MemVista	**0.516**	0.0	[0.0, 0.0]
No facets	0.373	14.29	[13.35, 15.25]
Fixed domain-prior facets	0.402	11.4	[10.6, 12.2]
Independent facet proposals	0.484	3.2	[2.4, 4.0]
Facets from pooled errors	0.475	4.1	[3.3, 4.9]
Adaptive facets w/o correctness	0.423	9.3	[8.5, 10.1]
Adaptive facets w/o target	0.438	7.8	[7.0, 8.5]

Table 17: Ablation on facet acquisition methods with GPT-5. The constructor and ranking prompt are held fixed; only the way facets are obtained varies. Gap denotes the accuracy difference from MemVista, and intervals are paired 95% confidence intervals.

Facet placement	Accuracy	Gain (pp)	95% interval (pp)
No facets	0.367	0.0	[0.0, 0.0]
Extraction only	<u>0.500</u>	13.3	[12.4, 14.3]
Synthesis only	0.427	6.0	[5.1, 6.9]
MemVista	**0.510**	14.33	[13.35, 15.31]

Table 18: GLM placement ablation with shared selected facets and a fixed profile token budget. All rows use the same constructor. Gain is measured against No facets in percentage points.

I Domain Facet Examples

The examples below are condensed from proposal rationales and rewritten as guidance for memory construction. They specify which relationships in a user's history to preserve. Each applies only when supported by the visible history and item metadata; retain uncertainty about inferred activities or needs. An item absent from the recorded history is not necessarily unowned or unused. The grounding and user-fact constraints in Appendix C apply throughout.

Goodreads.

1. Record later installments whose opening volume is absent from the visible history, and any later return to earlier volumes. Preserve whether recorded interactions move forward or backward through the series.
2. Record gaps between observed installments, retaining the volume identities on both sides. Distinguish an internal gap from the end of a consecutive sequence.
3. Preserve links between a recently started narrative and its companion works. Distinguish expansion of that story or setting from repeated interest in a broad genre.

Yelp.

1. Preserve sequences of services involved in setting up a home and subsequent neighborhood activities, including their locations. Retain evidence of a possible local transition separately from established visit routines.
2. Record venue attributes supporting group activities or children and their recurrence alongside household services. Preserve the supported activity requirements rather than reducing them to cuisine or venue categories.
3. Keep repeated personal-care and vehicle maintenance services connected to their respective ongoing needs, with observed timing and location. Separate these patterns from isolated leisure visits.

Movies and TV.

1. Track content identity across repeat media purchases together with format and edition changes. Preserve evidence of rebuying the same work in an upgraded or more complete release.
2. Record which parts of an ordered or thematic collection are already represented, including overlap between individual releases and box sets. Preserve the remaining gaps in the recorded collection.
3. Preserve the playback formats supported by the observed equipment and media. Keep these compatibility requirements distinct from preferences for particular stories or creators.

Kindle.

1. Link practical guides that support the same ongoing project or routine, retaining the tasks they address and their order. Preserve the developing resource collection across topic labels.
2. Preserve a recurring reading language across otherwise different topics, using explicit language metadata. Keep language continuity separate from any single subject preference.

3. Retain recurring narrative situations and emotional themes across different titles, using their descriptions. Preserve these narrower commonalities alongside broad genre labels.

Baby.

1. Preserve changes from caregiver-assisted tasks to products supporting a child's independent daily routines. Record the specific capability and chronology instead of assigning an exact age.
2. Record the hazard zones and environments addressed by observed safety products, and how that coverage expands over time. Keep the distinct roles of different protections.
3. Link items belonging to the same portable childcare setup. Preserve each component's role and compatibility, together with which parts of that setup appear in the history.

Automotive.

1. Link parts, tools, and supplies associated with the same repair or overhaul. Preserve subsystem, fitment, component roles, and purchase order as evidence of the project's current phase.
2. Record complementary equipment added to the same vehicle and the functions each addition supports. Preserve how the configuration expands beyond the presence of a shared brand or vehicle model.
3. Separate maintenance tools or supplies used across recorded vehicles from components tied to one vehicle or subsystem. Preserve both the repeated maintenance activity and its fitment constraints.

Books.

1. Preserve progression from introductory to specialized instructional material within a subject, including the skills or prerequisites stated in the books. Record the sequence of learning materials rather than only the topic.
2. Track changes in the developmental level and learning tasks of child-oriented books while retaining recurring interests. Preserve the joint change in difficulty and subject matter.
3. Connect guides to the practical activities they support, such as an ongoing craft or household project. Preserve the tasks covered across books rather than treating each title as an isolated interest.

Video Games.

1. Preserve the order in which hardware and peripherals enter a setup, with each component's function. Distinguish adding a new capability from replacing a component with the same role.
2. Record the hardware or companion components required by observed games and accessories, and whether matching components appear in the history. Preserve dependencies in both directions.
3. Keep newly acquired platforms and their subsequent software or accessories separate from older platform history. Preserve platform recency together with compatibility.
4. Retain edition or variant differences among items with the same core function, alongside repeated interest in premium releases. Preserve evidence of collection expansion despite functional overlap.

Grocery.

1. Connect staples, seasonings, and other ingredients supporting the same culinary tradition. Preserve their complementary roles instead of summarizing them as a single flavor preference.

2. Preserve the roles of ingredients associated with a multi-step preparation task, together with their observed sequence. Retain which parts of that preparation are already represented.
3. Track shifts between convenient meal components and versatile bulk ingredients, retaining preparation, storage, and explicit dietary requirements. Preserve changes in provisioning strategy across product categories.

Health.

1. Link products addressing different stated functions within the same recurring care routine. Preserve complementary roles and their sequence, rather than treating the products as unrelated preferences.
2. Record switches between general and more targeted products for the same stated purpose. Retain their sequence and intended uses without inferring that an earlier treatment failed.
3. Preserve accessibility and ease-of-use requirements recurring across care products, including explicit mobility or developmental attributes. Keep these task requirements separate from assumptions about the recipient.

Sports.

1. Link new equipment or a change of activity to accessories used for setup, calibration, or safe operation. Preserve the specific supporting function of each accessory.
2. Connect maintenance, transport, and protective items to the equipment they support. Preserve the shift from acquiring equipment to keeping it usable and protected.
3. Record how complementary components combine into an activity setup and which roles are already represented. Preserve relationships among the components across otherwise different equipment categories.