# SELF-EVOLVING LLM MEMORY EXTRACTION ACROSS HETEROGENEOUS TASKS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

LLM-based assistants increasingly rely on memory extracted from past conversations to serve users over time. However, real-world assistants face diverse tasks requiring different types of memory, which existing work largely ignores. We formalize the *heterogeneous memory extraction* task and introduce **BEHEMOTH**, a benchmark that repurposes 18 existing datasets spanning personalization, problem-solving, and agentic tasks, using a downstream utility-driven metric for systematic evaluation. Our empirical analysis confirms that no single static extraction prompt is optimal for all task categories, and that existing self-evolving prompt optimization frameworks degrade when training tasks are heterogeneous. To address this, we propose **CluE**, a self-evolving method that restructures analysis *hierarchically*: training examples are partitioned into clusters, each cluster is analyzed independently to keep feedback concentrated, and cross-cluster insights are synthesized into a single prompt update. Experiments on BEHEMOTH show that CluE consistently outperforms prior self-evolving frameworks. Together, the task formulation, benchmark, and method provide a foundation for memory extraction across heterogeneous real-world tasks.

## 1 INTRODUCTION

As large language models (LLMs) become increasingly capable and deeply integrated into daily life, users expect more than isolated, single-session interactions. They want LLMs to remember—to retain personal facts, preferences, and prior experiences across conversations, eliminating the need for repetitive re-specification. Due to long-context constraints (Liu et al., 2024; Wu et al., 2025), recent work (Hu et al., 2025; Zhang et al., 2025e; Liu et al., 2025) has begun to address this need by equipping LLMs with explicit memory modules. This raises a fundamental question: *what information is most worth remembering across diverse interaction scenarios?*

Existing systems typically answer this question with predefined, static extraction rules tailored to a narrow application context (Xiong et al., 2026). Conversational agents, for instance, prioritize user preferences and personal facts (Chhikara et al., 2025; Xu et al., 2025), while problem-solving agents focus on reusable skills and high-level strategies (Zhao et al., 2024; Shi et al., 2026). However, a general-purpose AI assistant in the real world encounters a far more diverse range of interactions, e.g., from casual conversation to domain-specific problem-solving, and no single, fixed extraction schema can adequately serve this breadth, as shown in Figure 1. Despite this challenge, there is no established benchmark for evaluating the generalizability of memory extraction systems **across heterogeneous tasks**.

To address this gap, we first formalize *single-step memory extraction*: given a source conversation, a memory extraction model produces a memory string conditioned on an extraction prompt. We evaluate extraction quality through a downstream, utility-driven metric based on how well the extracted memory helps a generation model answer associated target queries, as illustrated in Figure 1. Building on this formulation, we propose a benchmark methodology that repurposes existing memory-related datasets and use it to construct **BEHEMOTH** (**B**enchmark for **E**xtracting **HE**lpful **M**emory **O**n **T**asks with **H**eterogeneity), a *heterogeneous* memory extraction benchmark spanning 18 datasets across three task categories (personalization, problem-solving, and agentic tasks). Our evaluation finds that no single static extraction prompt dominates across all task categories, motivating self-evolving prompt optimization from data.



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.

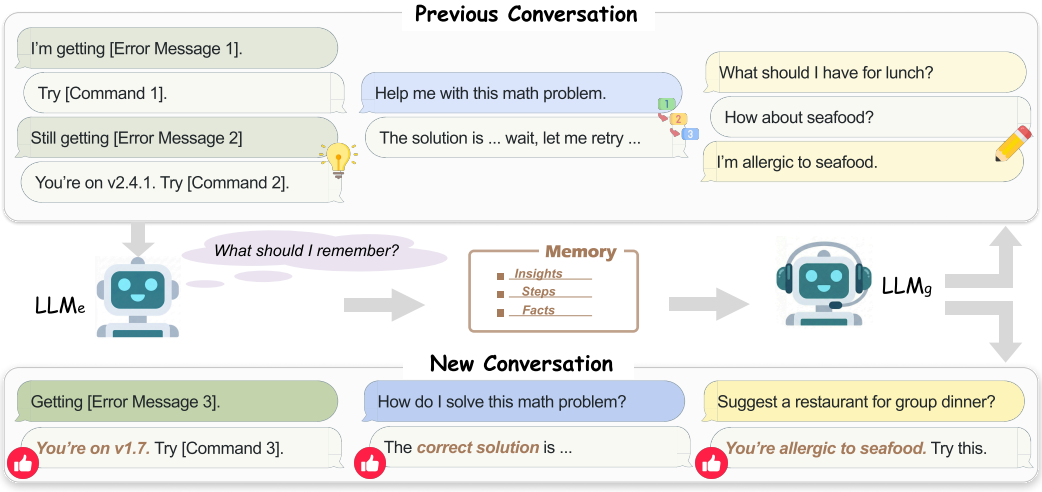


Figure 1: Illustration of heterogeneous memory extraction. A general-purpose assistant $\text{LLM}_g$ faces diverse previous conversations spanning technical debugging, math problem-solving, and personal preferences, from which an extraction model $\text{LLM}_e$ must produce different types of memory (e.g., reusable insights, solution steps, personal facts). $\text{LLM}_g$ then leverages these memories to improve responses in new conversations. How well these responses address user needs can then serve as a utility-driven signal for evaluating extraction quality.

However, existing self-evolving frameworks including GEPA (Agrawal et al., 2025), ACE (Zhang et al., 2025d), and MemEvolve (Zhang et al., 2025c) face new challenges under our heterogeneous distributions. They all perform *flat* analysis: either over the whole batch at once, where feedback from dissimilar examples averages out, or over small batches, where each update reflects only a partial view of the distribution. To address this, we propose **CluE** for **Clu**ster-based **E**volution, which restructures analysis *hierarchically*: training examples are partitioned into clusters, analyzed per cluster to keep feedback concentrated, with per-cluster insights synthesized into a single prompt update that retains full-batch coverage. Experiments show that CluE consistently outperforms prior self-evolving frameworks, achieving a $+9.04\%$ overall relative gain over the seed prompt compared to $+5.06\%$ for the strongest prior framework.

In summary, our contributions are as follows:

- We formalize the heterogeneous memory extraction task and propose BEHEMOTH, a benchmark covering 18 datasets for systematic evaluation.
- Through evaluation on BEHEMOTH, we reveal that no single static prompt dominates, and that existing self-evolving frameworks struggle to generalize across heterogeneous tasks.
- We propose CluE, which replaces the flat analysis of prior self-evolving frameworks with hierarchical, cluster-based analysis and synthesis, achieving stable learning under heterogeneous distributions.

## 2 Related Work

### 2.1 LLM Memory

To enable persistent and personalized LLMs, recent work has moved beyond long-context approaches (Beltagy et al., 2020), which suffer from the lost-in-the-middle problem (Liu et al., 2024), toward explicitly extracting the information most worth remembering for downstream storage, retrieval and management (Zhang et al., 2025c; Hu et al., 2025). Existing systems differ primarily in *what* they extract.[1] One line of work focuses on **personalized and factual memory**: Mem0 (Chhikara et al., 2025) targets preferences, dates, and relationships to construct entity-based graphs; A-Mem (Xu et al., 2025) extracts keywords and contextual descriptions following Zettelkasten principles. Another line of work targets **experiential and strategic memory**: ReasoningBank (Ouyang

[1] A memory differs from a summary in selecting for future reusability rather than faithful compression; see Appendix B.1.

et al., 2025) distills successful strategies and failure lessons from agent trajectories; Dynamic Cheatsheet (Suzgun et al., 2025) maintains a running summary of reusable problem-solving patterns. Both lines of work rely on fixed, domain-specific extraction rules, limiting their generalizability. From the benchmarking perspective, personalization-focused datasets (Wu et al., 2025; Jiang et al., 2025) and problem-solving or agentic datasets (Zhang et al., 2025a; Suzgun et al., 2025; Xiong et al., 2026) have been used to evaluate memory systems, but no existing benchmark spans all interaction scenarios under a unified protocol. To fill this gap, we propose a heterogeneous memory extraction benchmark and treat the extraction instruction itself as learnable, motivated by the observation that real-world tasks do not fall neatly into discrete categories and that domain-agnostic extraction rules can emerge from heterogeneous task distributions.

### 2.2 Self-Evolving Frameworks

Self-evolving frameworks iteratively refine LLM prompts or strategies from task feedback without human intervention. General prompt optimization methods such as APE (Zhou et al., 2023), OPRO (Yang et al., 2024), GEPA (Agrawal et al., 2025), and ACE (Zhang et al., 2025d) evolve prompts through various feedback loops, typically developed for single-task benchmarks. In the memory domain, MemEvolve (Zhang et al., 2025c) jointly evolves experiential knowledge and the memory architecture itself through a meta-evolutionary dual loop; ALMA (Xiong et al., 2026) meta-learns entire memory designs as executable code; Evo-Memory (Wei et al., 2025) evolves the memory content through sequential in-distribution task streams; and MemSkill (Zhang et al., 2026) evolves per-domain memory skill banks using a PPO-trained controller with hard-case clustering. A common assumption across these methods is that the task distribution is homogeneous. Our work contrasts by exploring the potential of self-evolving frameworks on heterogeneous training distributions.

## 3 Task Formulation and Dataset Curation

### 3.1 Single-Step Memory Extraction

To study memory extraction in isolation, we formulate a **single-step memory extraction** task that excludes orthogonal factors such as memory retrieval and management. In our task, a system must extract a memory string from a source conversation. We evaluate the extraction quality by providing the memory string as additional context when answering a related target query.

Formally, given a *source conversation* $c$ between a user and a generation model $\mathrm{LLM}_g$, an extraction model $\mathrm{LLM}_e$ takes an extraction prompt $P$ as its system-level instruction and the conversation $c$ as input, and produces a memory string $m = \mathrm{LLM}_e(P, c)$. To measure extraction quality, we adopt a utility-driven metric. Specifically, given an associated *target query* $q$, $\mathrm{LLM}_g$ answers $q$ conditioned on the extracted memory, $\hat{y} = \mathrm{LLM}_g(q, m)$. We refer to this full interaction as the target conversation $c_q$. A task-specific *reward function* $R$ then scores the response, $r = R(\hat{y}) \in [0, 1]$, which directly measures whether the memory helped answer the target query.




Figure 2: Dataset Composition of BEHEMOTH.

In this work, we focus on comparing different extraction prompts $P$ based on the score $r$. We consider two setups: (1) a static prompt (§4); and (2) a self-evolving prompt (§5), optimized iteratively over a training set of triplets $(c, q, R)$.

### 3.2 CREATING BEHEMOTH

As existing memory benchmarks cover only homogeneous task distributions, we provide a method to construct a *heterogeneous* benchmark that covers diverse tasks under a unified protocol, enabling the first systematic study of memory extraction generalizability.

Concretely, we repurpose 18 existing datasets into the single-step extraction formulation above and organize them into three categories as presented in Figure 2: **personalization** (5 datasets), involving casual user-assistant conversations; **problem-solving** (7 datasets), involving reasoning problems; and **agentic** (6 datasets), involving multi-turn action-feedback trajectories. We use Qwen3-32B (Team, 2025) as $\mathrm{LLM}_g$ to complete source conversations for all datasets, and refer to the resulting benchmark as **BEHEMOTH**. Each instance is a triplet $(c, q, R)$, and Appendix A.1 describes how each dataset is converted into such triplets, along with per-dataset statistics. The pipeline is easily extensible to new datasets and other choices of $\mathrm{LLM}_g$.

To evaluate whether a self-evolving prompt generalizes beyond its training distribution, we hold out one dataset per category as an out-of-distribution test set, i.e., LongMemEval (Wu et al., 2025), GPQA-Diamond (Rein et al., 2023), and ToolBench (Guo et al., 2024). From each of the remaining 15 in-distribution datasets, we sample a small number of examples for training (330 in total) and use the rest as in-distribution test sets.

Importantly, the extraction model sees only the source conversation, with no dataset or category label. This mirrors real-world settings, where such labels are unavailable, and rules out methods that hard-code one extraction rule per dataset. Any extraction rules a method uses must therefore be *discovered* from the data itself.

### 3.3 EVALUATION METRICS

Let $J_{D_k}(P) = \frac{1}{|D_k|}\sum_{D_k} r$ denote the average reward score on dataset $D_k$. It also depends on $\mathrm{LLM}_e$ and $\mathrm{LLM}_g$, but since both are held fixed within any comparison of prompts, we omit them from the notation and write the metrics as functions of $P$ alone. We use the following two metrics to evaluate the generalizability of $P$ across $N$ datasets:

$$\textbf{Macro Accuracy}(P) = \frac{1}{N}\sum_{k=1}^{N} J_{D_k}(P); \qquad \textbf{Relative Gain}(P) = \left(\prod_{k=1}^{N} \frac{J_{D_k}(P)}{J_{D_k}(P_{\text{base}})}\right)^{1/N} - 1.$$

Macro Accuracy weights all datasets equally regardless of size and captures absolute cross-domain performance. Relative Gain is the geometric mean of per-dataset improvement ratios over a baseline prompt $P_{\text{base}}$; normalizing by the baseline removes differences in score scale across datasets, and the geometric mean keeps a few extreme ratios from dominating.

## 4 EVALUATING STATIC MEMORY EXTRACTION

We first evaluate static memory extraction prompts, either hand-crafted for specific tasks or summarized from existing work. In Table 1, *No Memory* provides no memory to the generation model, and *Simple* uses a minimal prompt that asks the model to "extract reusable memories" with almost no guidelines. The remaining four are organized into two groups: category-specific prompts, including *Mem0* (Chhikara et al., 2025), which targets personal facts and preferences, and *ReasoningBank* (Ouyang et al., 2025), which focuses on insights; and taxonomy-based prompts, including *OpenMemory*[2], which defines a five-class memory taxonomy, and *Survey* (Hu et al., 2025), which classifies memories into two broad types. The full prompts are provided in Appendix C.1. Unless otherwise stated, we use Qwen3-32B as both the generation model and the extraction model throughout all experiments.

We have the following observations from Table 1: **(1) Using memory consistently improves over no-memory baselines.** Even the minimal *Simple* prompt raises overall macro accuracy from 37.84% to 46.00%, with gains across all three categories, confirming the value of memory extraction. **(2) No single static memory extraction prompt dominates.** Category-specific memory extraction

[2] https://openmemory.cavira.app/

| | Personalization | | Problem-Solving | | Agentic | | Overall | |
|---|---|---|---|---|---|---|---|---|
| | **MA** | **RG** | **MA** | **RG** | **MA** | **RG** | **MA** | **RG** |
| **No Memory** | 34.15 | – | 46.52 | – | 30.36 | – | 37.84 | – |
| **Simple** | 58.76 | 0 | 48.76 | 0 | 32.46 | 0 | 46.00 | 0 |
| **Mem0** | 73.31 | +**29.96** | 45.97 | −3.70 | 31.62 | +0.44 | 48.48 | +**5.79** |
| **ReasoningBank** | 50.67 | −10.78 | 50.72 | +**6.90** | 32.12 | +2.49 | 44.51 | +0.45 |
| **OpenMemory** | 57.39 | −1.04 | 50.47 | +5.59 | 34.93 | +**8.57** | 47.14 | +4.74 |
| **Survey** | 62.05 | +7.35 | 49.86 | +4.80 | 33.58 | +2.89 | 47.69 | +4.83 |

Table 1: Static prompt evaluation on in-distribution test sets. **MA**: Macro Accuracy (%); **RG**: Relative Gain (%) over *Simple*. Best and second best MA are highlighted; best RG in **bold**. **Overall** aggregates across all datasets.

prompts (i.e., *Mem0* and *ReasoningBank*) excel in their target domains but underperform elsewhere, while taxonomy-based memory extraction prompts (i.e., *Survey* and *OpenMemory*) offer more balanced but moderate improvements. **(3) A more detailed taxonomy does not guarantee better extraction.** *OpenMemory*'s five-class taxonomy does not outperform *Survey*'s simpler two-class design on overall MA, indicating that extraction quality is not solely determined by the granularity of the prompt but also by how well the extraction model can interpret and follow it.

## 5 Evaluating Evolving Memory Extraction

The static prompt experiments show that hand-designing a universally effective extraction prompt is hard, which motivates learning the prompt from data through self-evolution instead.

### 5.1 Baselines

We compare against three popular self-evolving frameworks: GEPA (Agrawal et al., 2025), ACE (Zhang et al., 2025d), and MemEvolve (Zhang et al., 2025c). All three share the same loop: extract memories on a training batch with the current prompt, collect a log $(\text{source conversation } c, \text{memory } m, \text{target conversation } c_q, \text{reward score } r)$ per example, and have LLM modules analyze the logs to propose the next prompt. They mainly differ in batch granularity and in the LLM roles used. **GEPA** uses a single reflective Proposer that analyzes the batch of logs and rewrites the prompt in one pass, so each update is limited to a batch small enough to fit in the LLM context. **ACE** splits the work between a Reflector, which processes one log at a time and tags rules in the prompt as helpful or harmful, and a Curator, which periodically applies atomic edits according to the accumulated counts. **MemEvolve** pairs a tool-augmented Analyzer with a Proposer. The Analyzer retrieves and inspects individual logs by ID and can therefore process a large batch at once, and the Proposer then refines the prompt from the Analyzer's findings. See Appendix A.2 for an illustration and details.

Despite these differences, all three were designed for homogeneous distributions where all examples share similar features. Under heterogeneous distributions, their *flat* analysis faces a dilemma between coverage and focus. Analyzing a large batch at once, as MemEvolve does, averages feedback from dissimilar examples together and washes out scenario-specific insights. Analyzing small batches, as GEPA does, keeps feedback concentrated but reflects only a partial view of the distribution, pulling the prompt in conflicting directions across updates.

### 5.2 CluE: Cluster-based Evolution

We propose **CluE** (**Clu**ster-based **E**volution), which restructures analysis *hierarchically* on top of MemEvolve's analyzer-proposer backbone. The core is a *cluster-based analysis-proposal architecture*: training examples are partitioned into clusters, each cluster is analyzed independently so that feedback stays concentrated, and all per-cluster insights are then synthesized into a single prompt update, retaining full-batch coverage. Within this architecture, how clusters are formed is a design choice (ablated in §6.1). CluE instantiates it with LLM-based clustering, letting the number of clusters emerge adaptively from the data.

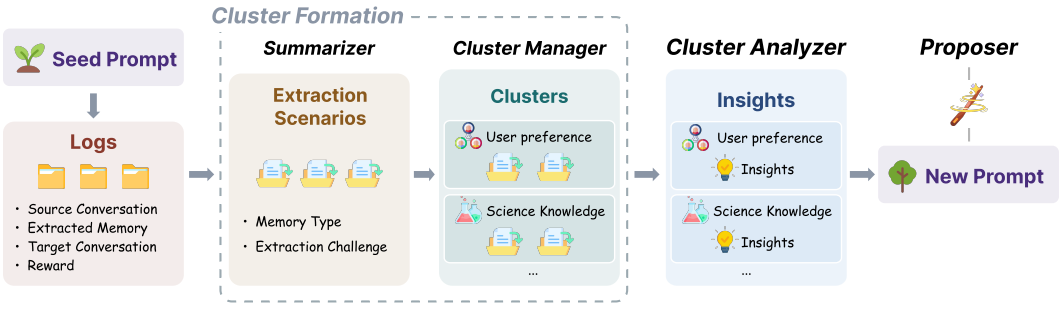


Figure 3: Overview of our method CluE in one evolution round.

CluE keeps the same outer loop as the baselines above and differs only in how each round's logs are turned into the next prompt, which proceeds in four steps, as illustrated in Figure 3. The first two steps (Summarization and Clustering) implement the cluster formation stage, and the latter two (Cluster-based Analysis and Cross-cluster Proposal) implement per-cluster analysis and cross-cluster synthesis. Prompts are provided in Appendix C.2.

**Summarization.** To support clustering, a Summarizer reads the log of each example in the batch and generates a concise summary describing its *extraction scenario*, i.e., what type of information needs to be extracted and what makes extraction challenging. For example, if a user chats about several films and is later asked which genre they prefer, the summary reads: *"The target is an implicit preference that must be abstracted from liked items rather than copied."* Surface-level details, such as which films were mentioned or which dataset the example comes from, are dropped.

**Clustering.** A Cluster Manager reads the summaries of the current batch, along with the labels and descriptions of existing clusters, and assigns each example to a cluster based on its extraction scenario. As a result, a cluster such as "extracting procedural knowledge from long dialogues" may contain both agent trajectories and step-by-step math solutions. As the examples change across rounds, the Cluster Manager can also create, merge, or split clusters.

**Cluster-based Analysis.** For each cluster, a Cluster Analyzer inspects the training logs of its examples and asks what makes the extracted memory useful in this scenario and where it falls short. From these success and failure patterns, it derives insights on how the extraction prompt should be updated for this scenario. Because clusters are analyzed separately, insights for one scenario do not get mixed up with those for another.

**Cross-cluster Proposal.** A Proposer then combines all per-cluster analyses into a new memory extraction prompt. It pulls out principles that hold across clusters, organizes the cluster-specific insights into structured guidelines, and resolves conflicts by restricting each guideline to the memory categories it applies to. The resulting prompt thus provides both general principles and scenario-specific guidance.

### 5.3 Results

We evaluate the self-evolving frameworks starting from the *Simple* prompt and report in-distribution and out-of-distribution results separately.

**CluE improves consistently across categories, while existing frameworks trade off.** As shown in Table 2, all self-evolving frameworks achieve positive overall relative gains over the *Simple* baseline, but for existing frameworks, gains on some categories come at the cost of regressions on others. GEPA achieves the strongest agentic improvement (+14.08%) yet regresses on personalization (−2.16%). MemEvolve excels at personalization (+10.76%) but drops on agentic tasks (−3.25%). Under heterogeneous distributions, these frameworks trade off performance across tasks rather than improving them jointly. In contrast, CluE yields the highest overall relative gain (+9.04%), with

| | Personalization | | Problem-Solving | | Agentic | | Overall | |
|---|---|---|---|---|---|---|---|---|
| | MA | RG | MA | RG | MA | RG | MA | RG |
| **No Memory** | 34.15 | – | 46.52 | – | 30.36 | – | 37.84 | – |
| **Simple** | 58.76 | 0 | 48.76 | 0 | 32.46 | 0 | 46.00 | 0 |
| **GEPA** | 56.59 | −2.16 | 50.05 | +2.86 | 37.23 | **+14.08** | 47.52 | +5.06 |
| **ACE** | 53.73 | −4.38 | 51.04 | +8.04 | 33.49 | +4.93 | 45.91 | +3.56 |
| **MemEvolve** | 63.67 | +10.76 | 49.49 | +1.17 | 30.92 | −3.25 | 47.08 | +2.11 |
| **CluE** | 65.72 | **+12.34** | 51.85 | **+8.39** | 35.24 | +7.22 | 50.01 | **+9.04** |

Table 2: Evaluation of self-evolving frameworks starting from the *Simple* prompt on in-distribution test sets. RG denotes the Relative Gain (%) over *Simple*.

consistent improvements across all three categories (+12.34% personalization, +8.39% problem-solving, +7.22% agentic).

**CluE is context-efficient and generalizes to other extraction backends.** Compared to a full-context reference that gives $\text{LLM}_g$ the entire source conversation, CluE retains most of the accuracy (50.01 vs. 52.68 overall MA) while using only about 9% of the context tokens, and even surpasses it on problem-solving (Appendix B.2). The advantage also holds with Gemini-3-Flash (The Gemini Team, 2025) or GPT-5.4-mini (OpenAI, 2026) as the extraction backend, where CluE is the only framework that clearly improves over *Simple* (Appendix B.3). Finally, a qualitative analysis of the evolved prompts (Appendix B.4) shows that GEPA's small update batches bias its prompt toward the most recent domain and MemEvolve's unclustered feedback from heterogeneous tasks cancels out, whereas cluster-based analysis leads CluE to a complete memory taxonomy with category-specific and domain-agnostic guidelines.

**CluE remains robust on out-of-distribution tasks.** The diversity of the in-distribution test sets already provides evidence of generalization. To further validate this, we evaluate the evolved prompts on held-out datasets not used during evolution (Table 3). Our method achieves the strongest result on LongMemEval (63.07 vs. 56.82 for the second best) and does not drop below the *Simple* prompt on GPQA-Diamond or ToolBench, whereas GEPA degrades on ToolBench and ACE degrades on GPQA-Diamond, confirming that our method yields broadly applicable extraction prompts rather than ones overfit to specific datasets.

| | LME | GPQA-D | ToolBench |
|---|---|---|---|
| **No Memory** | 20.45 | $45.62_{\pm 3.12}$ | 23.33 |
| **Simple** | 46.02 | $47.14_{\pm 4.15}$ | 25.30 |
| **GEPA** | 35.06 | $50.00_{\pm 0.82}$ | 24.09 |
| **ACE** | 29.71 | $46.13_{\pm 0.86}$ | 26.82 |
| **MemEvolve** | 56.82 | $47.98_{\pm 1.09}$ | 26.67 |
| **CluE** | 63.07 | $48.48_{\pm 1.80}$ | 26.67 |

Table 3: Evaluation on the out-of-distribution test sets of BEHEMOTH.

## 6 Further Analysis

### 6.1 Ablating Cluster Formation

CluE's core is the cluster-based analysis-proposal architecture, within which the clustering criterion is a design choice, and we ablate both. CluE allows up to $k = 7$ clusters but lets the LLM merge or split them adaptively; evolving from *Simple* with Qwen3-32B, it converges to 4 clusters (Appendix B.4). We compare: **MemEvolve**, an analysis-proposal architecture with *no clustering*; **random clustering** at $k = 7$ (CluE's cap) and $k = 4$ (its converged count); **embedding clustering**, which embeds each source conversation with `bge-base-en-v1.5` (Xiao et al., 2023) and runs $k$-means at fixed $k = 7$; and **adaptive embedding clustering**, which sweeps $k$ from 2 to 7 and selects $k$ by Silhouette score (Rousseeuw, 1987). All variants share identical hyperparameters otherwise.

**Findings.** As shown in Figure 4, every clustering variant substantially outperforms no clustering baseline, i.e., MemEvolve, confirming that the hierarchical structure itself drives the gains. Within this architecture, CluE attains the best overall result with the fewest external assumptions. Random

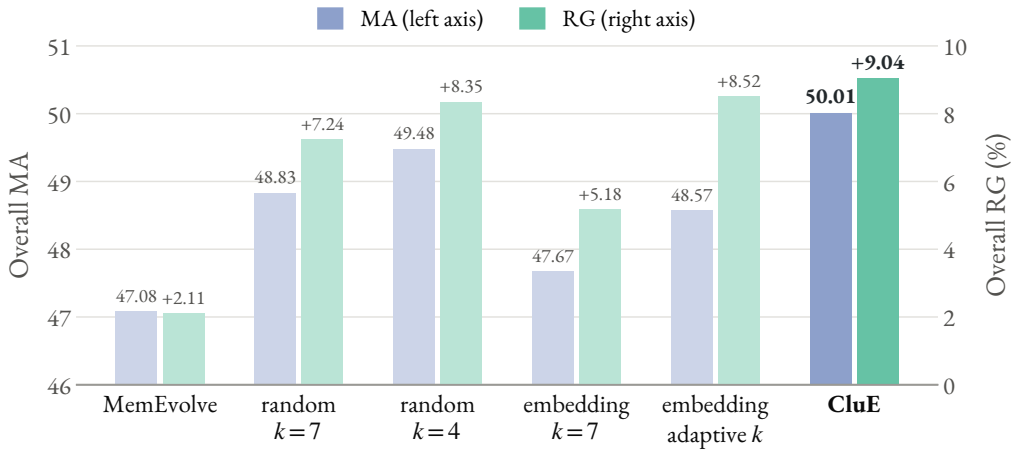


Figure 4: Ablation of cluster formation on in-distribution test sets for Qwen3-32B evolving from *Simple*. Bars show overall MA (left axis) and overall RG over *Simple* (right axis).

clustering requires a preset cluster count and is sensitive to it ($k = 7$ underperforms $k = 4$), while embedding-based clustering requires a separate embedding model and trails on overall MA. By letting the number of clusters emerge adaptively rather than fixing it, CluE is better suited to real, heterogeneous settings where the appropriate granularity is unknown and varies across data.

### 6.2 Evolution From a Stronger Seed

| | Personalization | | Problem-Solving | | Agentic | | Overall | |
|---|---|---|---|---|---|---|---|---|
| | **MA** | **RG** | **MA** | **RG** | **MA** | **RG** | **MA** | **RG** |
| **No Memory** | 34.15 | – | 46.52 | – | 30.36 | – | 37.84 | – |
| **Survey** | 62.05 | 0 | 49.86 | 0 | 33.58 | 0 | 47.69 | 0 |
| **GEPA**† | 62.05 | 0 | 49.86 | 0 | 33.58 | 0 | 47.69 | 0 |
| **ACE** | 54.80 | −10.48 | 50.06 | +0.75 | 34.41 | +3.69 | 46.11 | −1.44 |
| **MemEvolve** | 62.88 | −0.31 | 51.22 | +2.51 | 31.34 | −4.85 | 47.70 | −0.74 |
| **CluE** | 70.67 | **+13.62** | 51.63 | **+2.68** | 34.69 | **+5.79** | 51.06 | **+6.54** |

Table 4: Evaluation of self-evolving frameworks starting from the *Survey* prompt on in-distribution test sets. RG denotes the Relative Gain (%) over *Survey*. † GEPA did not find a better prompt.

In §5.3, evolution starts from a minimal seed (*Simple*). Here we test whether each method can leverage a stronger starting point by initializing from the *Survey* prompt. A stronger seed leaves less room for improvement but more room for damage, and poorly targeted updates risk overwriting the useful guidelines it already contains. As shown in Table 4, GEPA fails to find any improvement and returns the seed unchanged. ACE and MemEvolve both produce negative overall relative gains ($-1.44\%$ and $-0.74\%$). While they achieve marginal improvements on some categories, these are offset by notable regressions elsewhere (e.g., ACE drops 10.48% on personalization, MemEvolve drops 4.85% on agentic), indicating that their updates inadvertently damage existing strengths of the seed. In contrast, CluE achieves $+6.54\%$ overall relative gain with improvements across all three categories, demonstrating that cluster-based analysis can preserve the seed's strengths while still discovering room for improvement.

### 6.3 From Single-Step to Continual Memory Extraction

Our main setup adopts a single-step protocol for efficient and stable evaluation. In practice, memory systems operate *continually*: conversations arrive sequentially, previously extracted memories are retrieved and injected into future conversations, and new extractions must operate on this augmented

context. In this section, we examine whether the advantages observed in the single-step setting carry over to the continual setting.

Following Ouyang et al. (2025), we adopt a simple continual pipeline—embedding-based top-$n$ ($n = 1$) retrieval plus concatenation-based consolidation—to isolate the effect of extraction prompt quality from retrieval and consolidation design. We select two tasks on which CluE outperforms MemEvolve in the single-step evaluation: Game of 24 (problem-solving) and AlfWorld (agentic), and focus the analysis on whether single-step gains transfer.

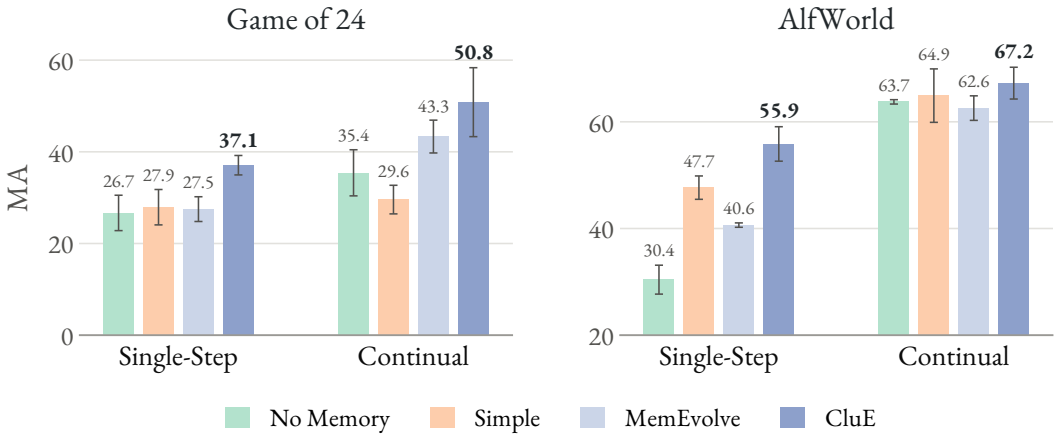


Figure 5: Single-step vs. continual memory extraction on Game of 24 and AlfWorld (MA, mean ± std). The *No Memory* baselines differ between settings because the two settings evaluate on different example pools; see Appendix A.3 for details.

As shown in Figure 5, CluE maintains its advantage over MemEvolve in both tasks under the continual setting, confirming that the single-step gains transfer when memories are accumulated and retrieved over time. Moreover, the *Simple* prompt, which lacks detailed extraction guidelines, falls below the *No Memory* baseline on Game of 24, whereas CluE consistently outperforms *No Memory*. This highlights that well-evolved extraction guidelines are especially important in the continual setting, where low-quality memories accumulate and compound over time.

## 7 Discussion and Conclusions

In this work, we focus on LLM memory extraction under heterogeneous tasks. We formalize single-step memory extraction, introduce a general benchmark methodology instantiated as BEHEMOTH, and use it to expose the limitations of both static extraction prompts and existing self-evolving frameworks. To overcome these limitations, we propose CluE, a hierarchical, cluster-based evolution framework that enables stable learning under heterogeneous training distributions, and demonstrate its effectiveness through extensive experiments.

We believe both artifacts of this work may find use beyond the scope studied here. BEHEMOTH can serve as a testbed for evaluating a wide range of memory extraction approaches—not only self-evolving frameworks but also routing-based, skill-based, and other emerging paradigms—to explore what strategies best handle heterogeneous memory extraction. The self-evolving framework CluE may also extend beyond memory extraction to any setting where a single agent must handle heterogeneous demands, such as serving multiple users with distinct habits and communication styles, or supporting the same user across projects that span different domains and workflows.

Several directions remain open. First, although BEHEMOTH covers 18 datasets, real-world deployments involve greater task diversity and longer interaction histories, so constructing more challenging and realistic benchmarks is an important next step. Second, this work focuses on memory extraction, the first stage of the memory lifecycle. Achieving generalizable performance across the full lifecycle, including storage, retrieval, and management, remains an open challenge.

### AI USE STATEMENT

In this work, we used generative AI tools for implementing methods. We have not used generative AI tools for designing or providing feedback on research methodology or experiments, or supporting qualitative and thematic data analysis, or interpreting results, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for creating or editing software code, and editing a research paper to improve readability. Note that LLMs are also the object of study in this work. As described in Section 3.1 and Appendix A.1, we use LLMs to generate source conversations, to perform memory extraction and prompt evolution, and as judges for evaluation. These uses are part of the experimental design rather than research assistance. We have reviewed all AI-assisted work. We verified LLM-generated code and manually reviewed and revised all AI-edited text. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

### REPRODUCIBILITY STATEMENT

We have made several efforts to ensure the reproducibility of this work. The construction of BEHEMOTH, including how each dataset is converted into $(c, q, R)$ triplets, the train/test splits, the reward functions, and per-dataset statistics, is described in Appendix A.1 and Table 5. The hyperparameters of all self-evolving frameworks, including CluE, are given in Appendix A.2. All static extraction prompts are provided in Appendix C.1, the prompts for each LLM role in CluE are provided in Appendix C.2, and the final evolved prompt is shown in Figure 19. Our main experiments use the open-weight Qwen3-32B model, and for test sets with fewer than 200 examples we report the average over three runs. We will release the code and the processed benchmark data upon publication.

## 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, Daniel Klein, Matei Zaharia, and Omar Khattab. GEPA: reflective prompt evolution can outperform reinforcement learning. *CoRR*, abs/2507.19457, 2025. doi: 10.48550/ARXIV.2507.19457. URL https://doi.org/10.48550/arXiv.2507.19457.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *CoRR*, abs/2004.05150, 2020. URL https://arxiv.org/abs/2004.05150.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. In Inês Lynce, Nello Murano, Mauro Vallati, Serena Villata, Federico Chesani, Michela Milano, Andrea Omicini, and Mehdi Dastani (eds.), *ECAI 2025 - 28th European Conference on Artificial Intelligence, 25-30 October 2025, Bologna, Italy - Including 14th Conference on Prestigious Applications of Intelligent Systems (PAIS 2025)*, Frontiers in Artificial Intelligence and Applications, pp. 2993–3000. IOS Press, 2025. doi: 10.3233/FAIA251160. URL https://doi.org/10.3233/FAIA251160.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, Findings of ACL, pp. 11143–11156. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.664. URL https://doi.org/10.18653/v1/2024.findings-acl.664.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, Senjie Jin, Jiejun Tan, Yanbin Yin, Jiongnan Liu, Zeyu Zhang, Zhongxiang Sun, Yutao Zhu, Hao Sun, Boci Peng, Zhenrong Cheng, Xuanbo Fan, Jiaxin Guo, Xinlei Yu, Zhenhong Zhou, Zewen Hu, Jiahao Huo, Junhao Wang, Yuwei Niu, Yu Wang, Zhenfei Yin, Xiaobin Hu, Yue Liao, Qiankun Li, Kun Wang, Wangchunshu Zhou, Yixin Liu, Dawei Cheng, Qi Zhang, Tao Gui, Shirui Pan, Yan Zhang, Philip Torr, Zhicheng Dou, Ji-Rong Wen, Xuanjing Huang, Yu-Gang Jiang, and Shuicheng Yan. Memory in the

age of AI agents. *CoRR*, abs/2512.13564, 2025. doi: 10.48550/ARXIV.2512.13564. URL https://doi.org/10.48550/arXiv.2512.13564.

Bowen Jiang, Yuan Yuan, Maohao Shen, Zhuoqun Hao, Zhangchen Xu, Zichen Chen, Ziyi Liu, Anvesh Rao Vijjini, Jiashu He, Hanchao Yu, Radha Poovendran, Gregory Wornell, Lyle H. Ungar, Dan Roth, Sihao Chen, and Camillo Jose Taylor. Personamem-v2: Towards personalized intelligence via learning implicit user personas and agentic memory. *CoRR*, abs/2512.06688, 2025. doi: 10.48550/ARXIV.2512.06688. URL https://doi.org/10.48550/arXiv.2512.06688.

Jiahong Liu, Zexuan Qiu, Zhongyang Li, Quanyu Dai, Jieming Zhu, Minda Hu, Menglin Yang, and Irwin King. A survey of personalized large language models: Progress and future directions. *CoRR*, abs/2502.11528, 2025. doi: 10.48550/ARXIV.2502.11528. URL https://doi.org/10.48550/arXiv.2502.11528.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Trans. Assoc. Comput. Linguistics*, 12:157–173, 2024. doi: 10.1162/TACL_A_00638. URL https://doi.org/10.1162/tacl_a_00638.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn LLM agents. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/877b40688e330a0e2a3fc24084208dfa-Abstract-Datasets_and_Benchmarks_Track.html.

OpenAI. Introducing GPT-5.4 mini and nano. https://openai.com/index/introducing-gpt-5-4-mini-and-nano/, March 2026.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory. *CoRR*, abs/2509.25140, 2025. doi: 10.48550/ARXIV.2509.25140. URL https://doi.org/10.48550/arXiv.2509.25140.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, abs/2311.12022, 2023. doi: 10.48550/ARXIV.2311.12022. URL https://doi.org/10.48550/arXiv.2311.12022.

Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987. ISSN 0377-0427. doi: https://doi.org/10.1016/0377-0427(87)90125-7. URL https://www.sciencedirect.com/science/article/pii/0377042787901257.

Taiwei Shi, Sihao Chen, Bowen Jiang, Linxin Song, Longqi Yang, and Jieyu Zhao. Experiential reinforcement learning. *CoRR*, abs/2602.13949, 2026. doi: 10.48550/ARXIV.2602.13949. URL https://doi.org/10.48550/arXiv.2602.13949.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Mirac Suzgun and Adam Tauman Kalai. Meta-prompting: Enhancing language models with task-agnostic scaffolding. *CoRR*, abs/2401.12954, 2024. doi: 10.48550/ARXIV.2401.12954. URL https://doi.org/10.48550/arXiv.2401.12954.

Mirac Suzgun, Mert Yüksekgönül, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. *CoRR*, abs/2504.07952, 2025. doi: 10.48550/ARXIV.2504.07952. URL https://doi.org/10.48550/arXiv.2504.07952.

Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. Membench: Towards more comprehensive evaluation on the memory of llm-based agents. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, Findings of ACL, pp. 19336–19352. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-acl.989/.

Qwen Team. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025. doi: 10.48550/ARXIV.2505.09388. URL https://doi.org/10.48550/arXiv.2505.09388.

The Gemini Team. Gemini 3 flash: Frontier intelligence built for speed. https://blog.google/products-and-platforms/products/gemini/gemini-3-flash/, December 2025. Google Blog.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and verification. In Marilyn A. Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pp. 809–819. Association for Computational Linguistics, 2018. doi: 10.18653/V1/N18-1074. URL https://doi.org/10.18653/v1/n18-1074.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/ad236edc564f3e3156e1b2feafb99a24-Abstract-Datasets_and_Benchmarks_Track.html.

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H. Chi, Chi Wang, Shuo Chen, Fernando Pereira, Wang-Cheng Kang, and Derek Zhiyuan Cheng. Evo-memory: Benchmarking LLM agent test-time learning with self-evolving memory. *CoRR*, abs/2511.20857, 2025. doi: 10.48550/ARXIV.2511.20857. URL https://doi.org/10.48550/arXiv.2511.20857.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=pZiyCaVuti.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-pack: Packaged resources to advance general chinese embedding. *CoRR*, abs/2309.07597, 2023. doi: 10.48550/ARXIV.2309.07597. URL https://doi.org/10.48550/arXiv.2309.07597.

Yiming Xiong, Shengran Hu, and Jeff Clune. Learning to continually learn via meta-learning agentic memory designs. *CoRR*, abs/2602.07755, 2026. doi: 10.48550/ARXIV.2602.07755. URL https://doi.org/10.48550/arXiv.2602.07755.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: agentic memory for LLM agents. *CoRR*, abs/2502.12110, 2025. doi: 10.48550/ARXIV.2502.12110. URL https://doi.org/10.48550/arXiv.2502.12110.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning*

*Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=Bb4VGOWELI.

Guibin Zhang, Muxin Fu, Guancheng Wan, Miao Yu, Kun Wang, and Shuicheng Yan. G-memory: Tracing hierarchical memory for multi-agent systems. *CoRR*, abs/2506.07398, 2025a. doi: 10.48550/ARXIV.2506.07398. URL https://doi.org/10.48550/arXiv.2506.07398.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. Memgen: Weaving generative latent memory for self-evolving agents. *CoRR*, abs/2509.24704, 2025b. doi: 10.48550/ARXIV.2509.24704. URL https://doi.org/10.48550/arXiv.2509.24704.

Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. Memevolve: Meta-evolution of agent memory systems. *CoRR*, abs/2512.18746, 2025c. doi: 10.48550/ARXIV.2512.18746. URL https://doi.org/10.48550/arXiv.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. *CoRR*, abs/2602.02474, 2026. doi: 10.48550/ARXIV.2602.02474. URL https://doi.org/10.48550/arXiv.2602.02474.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. *CoRR*, abs/2510.04618, 2025d. doi: 10.48550/ARXIV.2510.04618. URL https://doi.org/10.48550/arXiv.2510.04618.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model-based agents. *ACM Trans. Inf. Syst.*, 43(6):155:1–155:47, 2025e. doi: 10.1145/3748302. URL https://doi.org/10.1145/3748302.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: LLM agents are experiential learners. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pp. 19632–19642. AAAI Press, 2024. doi: 10.1609/AAAI.V38I17.29936. URL https://doi.org/10.1609/aaai.v38i17.29936.

Siyan Zhao, Mingyi Hong, Yang Liu, Devamanyu Hazarika, and Kaixiang Lin. Do llms recognize your preferences? evaluating personalized preference following in llms. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=QWunLKbBGF.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=92gvk82DE-.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, Simon Brunner, Chen Gong, James Hoang, Armel Randy Zebaze, Xiaoheng Hong, Wen-Ding Li, Jean Kaddour, Ming Xu, Zhihan Zhang, Prateek Yadav, and et al. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=YrycTjllL0.

# A Experimental Details

## A.1 BEHEMOTH Construction and Statistics

Following the task formulation in Section 3.1, each example in BEHEMOTH consists of a source conversation $c$, a target query $q$, and a reward function $R$. For personalization datasets, which already contain multi-turn dialogues, we directly use the provided source conversations and target queries. Problem-solving and agentic datasets, however, typically supply only task instances rather than full conversations; we therefore prompt $\mathrm{LLM}_g$ to solve each task instance and use the resulting trajectory as the source conversation, and following the retrieval setup of prior memory work (Table 5), retrieve a semantically similar query as the target query. As a result, each example's source conversation originates from a distinct query, but multiple examples may share the same target query. Figure 6 shows an example from AlfWorld.

```
SOURCE CONVERSATION (c):
  [SYSTEM]: household environment; action syntax; one-shot demo [omitted]
  [USER]: You see a bed 1, desk 2, desk 1, drawer 1-6, ... shelf 1-6.
          Task: examine the pencil with the desklamp.
  [ASSISTANT]: > think: find and take a pencil, then use a desklamp.
  [USER]: OK.
  [ASSISTANT]: > go to desk 1
  [USER]: On the desk 1, you see an alarmclock 2, a cellphone 2,
          a desklamp 1, a keychain 2, a mug 1, and a pencil 3.
  [ASSISTANT]: > take pencil 3 from desk 1
  [USER]: You pick up the pencil 3 from the desk 1.
  [ASSISTANT]: > use desklamp 1
  [USER]: You turn on the desklamp 1.
  [USER]: You successfully finished this task!

TARGET QUERY (q):
  [USER]: You see a bed 1, desk 2, desk 1, drawer 1-6, ... shelf 1-6.
          Task: look at pencil under the desklamp.
```

Figure 6: An example from AlfWorld. The source conversation is the trajectory of $\mathrm{LLM}_g$ solving a task instance; the target query is a semantically similar task instance retrieved from the same dataset.

The dataset statistics are summarized in Table 5. For each dataset, we randomly sample 20 examples as the in-distribution training set (50 for AIME, which aggregates multiple competition years) and use the remainder as the test set. During training (evolution), examples from all datasets are randomly shuffled rather than grouped by dataset, so each batch may contain examples from multiple datasets. For test sets with fewer than 200 examples, we repeat the full evaluation pipeline (memory extraction by $\mathrm{LLM}_e$ and target query answering by $\mathrm{LLM}_g$) three times and report the average for all experiments.

## A.2 Self-Evolving Frameworks

Figure 7 illustrates the three frameworks. Since we focus on *self*-evolution, all LLMs used for optimization are the same as $\mathrm{LLM}_e$. We describe the framework-specific hyperparameters below.

**GEPA.** Following Agrawal et al. (2025), we reserve 100 examples from the training set as a validation set and optimize over the remaining 230 examples. At each reflection step, the Proposer conditions on a minibatch of 5 training logs together with the current prompt to propose a refined candidate. Each candidate that improves on the minibatch is then evaluated on the validation set to determine whether it is added to the candidate pool.

**ACE.** We adopt the online setting of Zhang et al. (2025d), where ACE optimizes over all 330 training examples in a single epoch without a validation set. For each training example, the Reflector analyzes the extraction attempt and may iterate up to 3 reflection rounds when the attempt is incorrect. The Curator aggregates helpful/harmful signals and applies atomic operations (add, update, delete) to the playbook every 25 steps.

| Dataset | # Train | # Test | $R$ | Memory Work |
|---|---|---|---|---|
| | | ***Personalization*** | | |
| MemBench-High (Tan et al., 2025) | 20 | 200 | Acc. | MemBench (Tan et al., 2025) |
| MemBench-Low (Tan et al., 2025) | 20 | 80 | Acc. | MemBench (Tan et al., 2025) |
| PersonaMem-v2 (Jiang et al., 2025) | 20 | 272 | Acc. | PersonaMem-v2 (Jiang et al., 2025) |
| PrefEval (Zhao et al., 2025) | 20 | 200 | Acc. | PrefEval (Zhao et al., 2025) |
| LongMemEval (Wu et al., 2025) | – | 176 | LLM Judge | LongMemEval (Wu et al., 2025) |
| | | ***Problem-Solving*** | | |
| Game of 24 (Suzgun & Kalai, 2024) | 20 | 80 | Rule | DC (Suzgun et al., 2025) |
| AIME | 50 | 143 | EM | DC (Suzgun et al., 2025) |
| MMLU-Pro Physics (Wang et al., 2024) | 20 | 200 | Acc. | DC (Suzgun et al., 2025) |
| MMLU-Pro Engineering (Wang et al., 2024) | 20 | 200 | Acc. | DC (Suzgun et al., 2025) |
| MathEquationBalancer | 20 | 200 | Rule | DC (Suzgun et al., 2025) |
| BigCodeBench (Zhuo et al., 2025) | 20 | 200 | Pass@1 | MemGen (Zhang et al., 2025b) |
| GPQA-Diamond (Rein et al., 2023) | – | 198 | Acc. | DC (Suzgun et al., 2025) |
| | | ***Agentic*** | | |
| AlfWorld (Shridhar et al., 2021) | 20 | 114 | SR | G-Memory (Zhang et al., 2025a) |
| FEVER (Thorne et al., 2018) | 20 | 200 | SR | G-Memory (Zhang et al., 2025a) |
| PDDL (Ma et al., 2024) | 20 | 40 | SR | G-Memory (Zhang et al., 2025a) |
| BabyAI (Ma et al., 2024) | 20 | 92 | SR | Evo-Memory (Wei et al., 2025) |
| ScienceWorld (Ma et al., 2024) | 20 | 70 | SR | Evo-Memory (Wei et al., 2025) |
| ToolBench (Guo et al., 2024) | – | 330 | SoPR | Evo-Memory (Wei et al., 2025) |

Table 5: Datasets used in BEHEMOTH, grouped by task category. $R$ denotes the reward function: Acc. = accuracy, EM = exact match, Rule = rule-based verification, SR = success rate, Pass@1 = code execution pass rate, SoPR = solvable pass rate (LLM judge), LLM Judge = LLM-as-judge evaluation. *Memory Work* indicates the prior work that uses each dataset for memory evaluation.




Figure 7: The three self-evolving frameworks we compare against. They differ in how many logs each update sees and in the LLM roles that turn the logs into the next prompt.

**MemEvolve.** MemEvolve runs tournament-based evolution over `num_rounds` $= 5$ rounds. In each round, the Analyzer (a tool-augmented agent) inspects the current prompt's training logs on `batch_x` $= 35$ examples, and the Proposer generates 3 new candidate prompts based on the analysis. All 4 prompts are then evaluated on `batch_x` $= 35$ newly sampled examples plus `extra_sample_y` $= 10$ samples drawn from the current batch for stability, and the winner advances to the next round.

Our CluE adopts the exact same hyperparameters as MemEvolve.

### A.3 Continual Memory Extraction

The *No Memory* baselines in Figure 5 differ between the single-step and continual settings for the following two reasons:

**Different evaluated examples.** In the single-step setting, each evaluation instance is a triple (source conversation $c$, target query $q$, reward function $R$): $\mathrm{LLM}_e$ extracts memory from $c$, and $\mathrm{LLM}_g$ answers $q$ with or without that memory. In the continual setting, examples arrive as a sequential stream without paired source conversations; evaluation is performed on each incoming example using memories retrieved from previously seen examples. Because the two settings draw from different example pools, the *No Memory* baselines naturally differ.

**Different few-shot setup for AlfWorld.** Following G-Memory (Zhang et al., 2025a), AlfWorld uses a 1-shot in-context demonstration for memory extraction. In the continual setting, every example undergoes extraction and thus includes this demonstration; in the single-step setting, only the source conversation requires extraction, so we remove the demonstration when answering the target query (0-shot) to better isolate the contribution of extracted memory. This difference accounts for the substantially higher *No Memory* baseline in the continual setting.

Additionally, we note that the continual setting introduces a challenge not addressed by our current evolution pipeline: the extraction model must process source conversations that already contain retrieved memories, yet our prompts are evolved exclusively under the single-step regime where source conversations are memory-free. The experiments in §6.3 serve to verify that a good prompt evolved in the single-step setting retains its advantages in the continual setting; we leave evolving prompts directly under the continual and heterogeneous regime to future work.

# B Additional Experiments and Results

## B.1 Memory vs. Summary

Since memory is extracted from the source conversation alone, without access to the target query, one may ask whether memory extraction reduces to summarizing the conversation. The two differ in their selection criterion: a summary faithfully compresses what the conversation said, whereas a memory captures information reusable for future tasks, which may be peripheral to the conversation's main thread. Recent surveys likewise treat summarization as only one of several approaches to memory formation, alongside knowledge distillation, structured construction, latent representation, and parametric internalization (Hu et al., 2025).

To test this distinction empirically, we add a *Summary* baseline whose extraction prompt is "Summarize the following conversation in no more than 5 sentences.", while $\mathrm{LLM}_g$ and the downstream answering procedure are held identical to all other methods.

| | Personalization | Problem-Solving | Agentic | Overall |
|---|---|---|---|---|
| **No Memory** | 34.15 | 46.52 | 30.36 | 37.84 |
| **Simple** | 58.76 | 48.76 | 32.46 | 46.00 |
| **Summary** | 58.13 | 49.76 | 34.30 | 46.84 |
| **CluE** | **65.72** | **51.85** | **35.24** | **50.01** |

Table 6: Macro Accuracy (%) of *Summary* on in-distribution test sets for Qwen3-32B.

As shown in Table 6, *Summary* (46.84 overall MA) performs on par with static extraction prompts but falls short of CluE (50.01) under an identical pipeline, indicating that faithful compression alone does not yield useful memory. A case study (Figure 8) from PrefEval illustrates the difference. Both *Summary* and CluE process the same source conversation, whose main thread is designing a calming veterinary-clinic waiting room and which mentions the user's music taste only in passing. Both outputs stay within the length bound, yet only CluE's extracted memory transfers to the target query.

## B.2 Full-Context Reference

As a reference, we give $\mathrm{LLM}_g$ the entire source conversation directly. That is, $\mathrm{LLM}_g$ answers $q$ with $c$ in place of $m$, i.e., $\hat{y} = \mathrm{LLM}_g(q, c)$, and everything else is unchanged. This is not a deployable setting, since conversations accumulate continually across sessions, and re-feeding the full history

```
TARGET QUERY (q):
  [USER]: I'm going on a long road trip. Can you recommend some country
          music playlists or albums?
    A. the modern "Hot Country" playlist
    B. a "Bro-Country" playlist (Sam Hunt, Chase Rice)
    C. classic 1970s albums like "Wanted! The Outlaws"
       (Waylon Jennings, Willie Nelson)
    D. the latest Luke Bryan / Florida Georgia Line album
    (correct: C, since the user prefers old-school over modern country)

CluE:
  1. Technical Solution: keeping a clinic bubble wall clean (distilled
     water, weekly disinfection, replace water every 2-4 weeks ...).
  2. User Preference: "I usually prefer old-school traditional country
     from the 1950s-1970s era over modern country."
                                                <<< reusable preference
  3. Technical Solution: clinic music -> soft instrumental /
     nature sounds.
  4. Task Strategy: introducing new equipment to pets (trial runs ...).
  5. Technical Solution: low-maintenance clinic water features
     (bubble walls).
  => LLM_g picks C  [correct]

SUMMARY:
  The user sought advice on creating a calming veterinary clinic waiting
  room, with a focus on music and water features. The assistant
  recommended soothing lighting, neutral colors, and non-polarizing
  instrumental or nature sounds over genre-specific music. They suggested
  low-maintenance water features like bubble walls, emphasizing safety
  and hygiene. Specific models like the SereneLife bubble wall were
  highlighted. Tips for introducing the feature to pets included gradual
  exposure, positive reinforcement, and staff training.
  => LLM_g picks A  [wrong]
```

Figure 8: Case study from PrefEval. CluE captures a passing but reusable user preference, while *Summary* preserves the conversation's main thread and drops the preference.

for every new query makes inference cost grow without bound, which is the very motivation for memory extraction. Beyond Macro Accuracy, we also report the average number of context tokens supplied to $\mathrm{LLM}_g$ per example (the full conversation vs. the extracted memory), counted with the Qwen3-32B tokenizer and macro-averaged over datasets in the same way as MA. Note that all self-evolving prompts, including CluE, instruct $\mathrm{LLM}_e$ to extract at most 5 memories per conversation, which bounds the memory length.

| | **Personalization** | | **Problem-Solving** | | **Agentic** | | **Overall** | |
|---|---|---|---|---|---|---|---|---|
| | **MA** | **#Tokens** | **MA** | **#Tokens** | **MA** | **#Tokens** | **MA** | **#Tokens** |
| **Full Context** | **72.68** | 5,327 | 49.46 | 1,582 | **40.54** | 2,392 | **52.68** | 2,851 |
| **CluE** | 65.72 | **294** | **51.85** | **236** | 35.24 | **237** | 50.01 | **252** |

Table 7: Macro Accuracy (%) and average number of context tokens per example (#Tokens) given to $\mathrm{LLM}_g$ for the *Full Context* reference and CluE on in-distribution test sets for Qwen3-32B.

As shown in Table 7, the reference is higher than CluE on personalization (72.68 vs. 65.72) and agentic (40.54 vs. 35.24), but lower on problem-solving (49.46 vs. 51.85), showing that strong extraction can outperform full context, not just approximate it. Meanwhile, CluE is far more compact, using only 8.84% of the context tokens overall (252 vs. 2,851), ranging from 14.92% on problem-solving to 5.52% on personalization, where source conversations are longest.

### B.3 Generalizing Across Extraction Backends

We test whether CluE generalizes across different extraction backends. We repeat the evolution pipeline with Gemini-3-Flash (Table 8) and GPT-5.4-mini (Table 9) as the extraction backend, starting from the *Simple* prompt.

| | Personalization | | Problem-Solving | | Agentic | | Overall | |
|---|---|---|---|---|---|---|---|---|
| | **MA** | **RG** | **MA** | **RG** | **MA** | **RG** | **MA** | **RG** |
| **No Memory** | 34.15 | – | 46.52 | – | 30.36 | – | 37.84 | – |
| **Simple** | 65.86 | 0 | 50.19 | 0 | 38.46 | 0 | 50.46 | 0 |
| **GEPA** | 68.59 | +5.64 | 52.91 | +7.71 | 33.77 | −9.99 | 50.71 | +0.93 |
| **ACE** | 66.65 | −0.43 | 48.43 | −1.98 | 34.45 | −8.88 | 48.63 | −3.93 |
| **MemEvolve** | 65.46 | −0.93 | 51.05 | +2.37 | 36.84 | −2.89 | 50.15 | −0.29 |
| **CluE** | 69.64 | **+6.66** | 53.64 | **+9.59** | 35.85 | −5.93 | 51.98 | **+3.40** |

Table 8: Evaluation of **Gemini-3-Flash** as the extraction backend from the *Simple* prompt. RG denotes the Relative Gain (%) over *Simple*.

| | Personalization | | Problem-Solving | | Agentic | | Overall | |
|---|---|---|---|---|---|---|---|---|
| | **MA** | **RG** | **MA** | **RG** | **MA** | **RG** | **MA** | **RG** |
| **No Memory** | 34.15 | – | 46.52 | – | 30.36 | – | 37.84 | – |
| **Simple** | 65.42 | 0 | 50.57 | 0 | 33.82 | 0 | 48.95 | 0 |
| **GEPA** | 60.57 | −6.72 | 50.63 | −1.56 | 31.51 | −9.97 | 46.91 | −5.81 |
| **ACE** | 57.81 | −8.75 | 51.33 | +3.14 | 33.10 | −5.30 | 46.98 | −2.98 |
| **MemEvolve** | 64.91 | +0.30 | 49.95 | −2.53 | 32.66 | −4.36 | 48.18 | −2.40 |
| **CluE** | 67.61 | **+5.41** | 52.18 | **+4.68** | 35.63 | **+5.25** | 50.78 | **+5.06** |

Table 9: Evaluation of **GPT-5.4-mini** as the extraction backend from the *Simple* prompt. RG denotes the Relative Gain (%) over *Simple*.

From the Tables, we can see that CluE is the only framework that clearly improves over *Simple*, with overall relative gains of +3.40% on Gemini-3-Flash and +5.06% on GPT-5.4-mini. It also leads on Personalization and Problem-Solving with Gemini-3-Flash and attains the highest MA in every category with GPT-5.4-mini. In contrast, the baselines fail to hold their gains under the stronger backends. GEPA, ACE, and MemEvolve all regress below *Simple* with GPT-5.4-mini (by up to −5.81%), and with Gemini-3-Flash, ACE and MemEvolve regress as well while GEPA improves only marginally (+0.93%). These results confirm that the cluster-based evolution strategy transfers effectively across model families.

### B.4 Qualitative Analysis of the Evolution Process

**Evolved prompts.** To understand *why* the quantitative gaps arise, we inspect the best prompts produced by each framework after evolution from the *Simple* prompt. ACE produces a prompt structurally similar to CluE's but is substantially more verbose, as its atomic updates predominantly add rules and rarely delete them. The resulting prompt (1,403 tokens) is considerably longer than those of GEPA (1,243), CluE (936), and MemEvolve (539), which may weaken the extraction model's instruction-following. We illustrate the key structural differences for the remaining frameworks in Figure 9.

According to Figure 9, GEPA's fixed context-length budget limits each update to a small batch, biasing the prompt toward the most recent examples (in this case, AlfWorld). This domain-specific bias explains its strong agentic gains but regression on other categories. MemEvolve analyzes large batches spanning diverse tasks without any grouping, causing heterogeneous feedback signals to

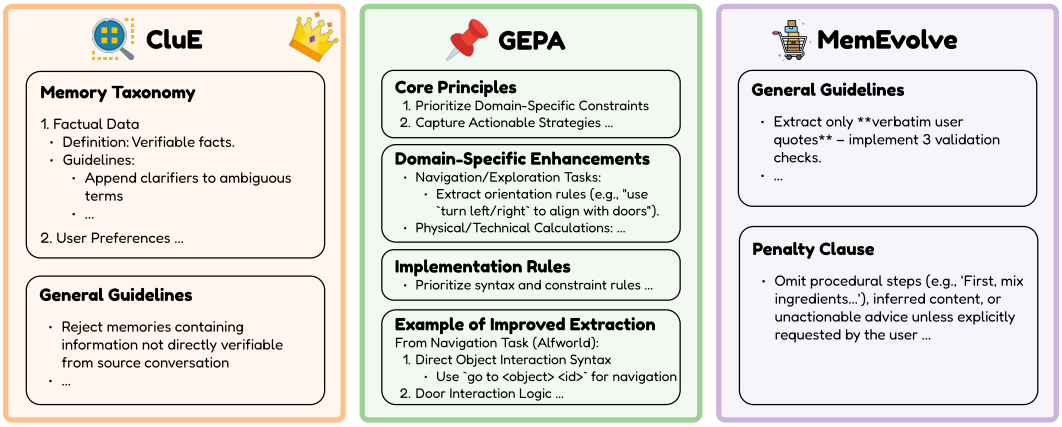


Figure 9: Structural comparison of the best prompts evolved by each framework from the *Simple* seed. Common sections (task description, input/output format) are omitted; only the distinctive components are shown. CluE produces a structured memory taxonomy with per-category definitions and guidelines alongside domain-agnostic general guidelines. GEPA embeds a large amount of domain-specific content (e.g., AlfWorld examples) directly into its rules. MemEvolve lacks a memory taxonomy and instead introduces a penalty clause that prohibits specific extraction patterns.

cancel out. As a result, its evolved prompt lacks a specific memory taxonomy and retains only known failure modes. Although CluE uses the same batch size as MemEvolve, cluster-based analysis yields a qualitatively different outcome: a complete memory taxonomy with category-specific guidelines as well as domain-agnostic principles, explaining the robust generalization across heterogeneous tasks and out-of-distribution datasets. The fully evolved prompt of CluE is shown in Figure 19.

**Cluster evolution.** We further analyze how CluE evolves clusters starting from *Simple* and using Qwen3-32B. The cluster pool starts with 7 fine-grained clusters and consolidates over rounds as the system discovers that initially separate scenarios share common extraction patterns. For example, *Emotional Context and Relational Dynamics* is absorbed into *User Preferences*, since both require capturing user-specific states from conversational cues; similarly, *Combinatorial and Puzzle Problem-Solving* merges into *Technical Problem-Solving*, as both benefit from extracting multi-step reasoning chains. After the final round, the pool contains four clusters:

1. User Preferences and Emotional Context (with Translation)
2. Factual Data Disambiguation and Verification
3. Procedural Knowledge in Virtual Environments
4. Technical and Scientific Problem-Solving

While this evolution path is dominated by merges, new clusters can also emerge. For instance, in a separate run starting from the *Survey* prompt, the system splits out a *Code-based Technical Workflows* cluster from the broader *Technical Problem-Solving* cluster at Round 2, recognizing that code-related examples require extracting implementation-specific patterns (e.g., dependency management, error handling) distinct from general reasoning strategies.

**From clusters to memory taxonomy.** The proposer synthesizes per-cluster analyses into a unified extraction prompt, where taxonomy categories need not map one-to-one to clusters (Figure 18). In the final evolved prompt (Figure 19), the four clusters yield five taxonomy sections:

1. Factual Data & Temporal Disambiguation (← cluster 2)
2. User Preferences & Emotional Context (← cluster 1)
3. Procedural & Technical Knowledge (← cluster 3)
4. Logical & Combinatorial Reasoning (← cluster 4)
5. Translation & Stylistic Requirements (← cluster 1)

Notably, cluster 1, which mixes user-preference extraction with translation-style tasks, is split into two taxonomy sections (2 and 5). This decoupled design allows the taxonomy to reorganize cluster-level insights into clean, non-overlapping categories without being constrained by cluster boundaries.

### B.5 Efficiency Analysis

We also compare the computational efficiency of each self-evolving framework along three dimensions: wall-clock time, optimization LLM calls, and evaluation calls. We define **optimization LLM calls** as the number of LLM invocations used exclusively for prompt optimization (e.g., analysis, generation, reflection, curation). **Evaluation calls** count the total number of target queries answered by $\mathrm{LLM}_g$ throughout evolution.

| | Wall Time | Optimization LLM Calls | Eval Calls |
|---|---|---|---|
| **GEPA** | $\sim$7.4h | 46 | 1,645 |
| **ACE** | $\sim$12.4h | 610 | 1,120 |
| **MemEvolve** | $\sim$5.0h | 30 | 1,150 |
| **CluE** | $\sim$5.5h | 221 | 1,150 |

Table 10: Efficiency comparison of self-evolving frameworks.

As noted by Agrawal et al. (2025), the majority of GEPA's evaluation calls are consumed by validation-set scoring, which serves only for candidate selection without producing learning signals. This results in the highest evaluation cost (1,645 calls) and contributes to its longer wall time. ACE, by contrast, has the fewest evaluation calls (1,120) since it forgoes maintaining a candidate pool entirely; however, its online learning loop is inherently sequential, as each step must complete extraction, reflection, and optional retry rounds before advancing, which severely limits parallelism and makes it the slowest method overall ($\sim$12.4h). Our method builds on MemEvolve's tournament-based evolution by adding per-example summarization and cluster-based error analysis, introducing moderate optimizer overhead (221 vs. 30 calls) at a marginal wall-time cost ($\sim$5.5h vs. $\sim$5.0h). Overall, CluE achieves the strongest performance gains (Table 2) with only a marginal increase in wall time over the most efficient baseline.

### B.6 Expanded Results

Tables 11 and 12 report the per-dataset accuracy on the in-distribution test sets of BEHEMOTH for every method in the main-text experiments, i.e., with Qwen3-32B as the extraction backend, evolving from the *Simple* and *Survey* prompts, respectively. As discussed in Section 5, our goal is broad generalizability rather than peak performance on any single dataset. While individual dataset scores may fluctuate across methods, category-level and overall aggregates provide a more reliable picture of extraction quality, as they smooth out dataset-specific variance.

## C Prompts

### C.1 Static Memory Extraction Prompts

We describe the five static extraction prompts evaluated in Table 1. *Simple* (Figure 10) is a minimal baseline that instructs the model to extract useful information with an empty taxonomy and minimal guidelines. *Mem0* and *ReasoningBank* provide detailed, category-specific extraction guidelines: *Mem0* (Figure 11) targets personalization-oriented facts such as preferences, relationships, and plans, while *ReasoningBank* (Figure 12) extracts strategic insights and lessons from successful and failed trajectories. In contrast, *OpenMemory* and *Survey* define broader memory taxonomies: *OpenMemory* (Figure 13) uses a five-class taxonomy (episodic, semantic, procedural, emotional, reflective), while *Survey* (Figure 14) uses a two-class taxonomy (factual and experiential).

| Dataset | No Memory | Simple | GEPA | ACE | MemEvolve | CluE |
|---|---|---|---|---|---|---|
| | | | ***Personalization*** | | | |
| **MemBench-High** | 20.50 | 87.50 | 83.00 | 77.00 | 87.00 | 87.00 |
| **MemBench-Low** | 55.00$_{\pm 1.77}$ | 60.83$_{\pm 1.18}$ | 59.58$_{\pm 1.56}$ | 58.33$_{\pm 3.12}$ | 64.58$_{\pm 2.12}$ | 63.75$_{\pm 1.02}$ |
| **PersonaMem-v2** | 19.12 | 21.69 | 22.79 | 26.10 | 26.10 | 24.63 |
| **PrefEval** | 42.00 | 65.00 | 61.00 | 53.50 | 77.00 | 87.50 |
| | | | ***Problem-Solving*** | | | |
| **Game of 24** | 26.67$_{\pm 3.86}$ | 27.92$_{\pm 3.86}$ | 32.08$_{\pm 3.12}$ | 38.75$_{\pm 7.14}$ | 27.50$_{\pm 2.70}$ | 37.08$_{\pm 2.12}$ |
| **AIME** | 27.97$_{\pm 1.14}$ | 28.67$_{\pm 1.14}$ | 27.74$_{\pm 0.87}$ | 31.00$_{\pm 0.33}$ | 31.47$_{\pm 2.62}$ | 27.51$_{\pm 1.74}$ |
| **MMLU-Pro Physics** | 70.50 | 73.00 | 72.50 | 71.00 | 72.00 | 73.00 |
| **MMLU-Pro Eng.** | 42.00 | 47.00 | 54.50 | 51.50 | 51.50 | 51.00 |
| **MathEqBal** | 87.00 | 93.00 | 92.00 | 90.50 | 93.50 | 96.50 |
| **BigCodeBench** | 25.00 | 23.00 | 21.50 | 23.50 | 21.00 | 26.00 |
| | | | ***Agentic*** | | | |
| **AlfWorld** | 30.41$_{\pm 2.71}$ | 47.66$_{\pm 2.19}$ | 57.02$_{\pm 2.48}$ | 42.11$_{\pm 3.28}$ | 40.64$_{\pm 0.41}$ | 55.85$_{\pm 3.23}$ |
| **FEVER** | 37.00 | 36.50 | 42.00 | 40.00 | 35.50 | 40.00 |
| **PDDL** | 18.33$_{\pm 1.18}$ | 16.67$_{\pm 2.36}$ | 19.17$_{\pm 1.18}$ | 18.33$_{\pm 3.12}$ | 15.00$_{\pm 0.00}$ | 16.67$_{\pm 3.12}$ |
| **BabyAI** | 32.25$_{\pm 1.02}$ | 21.01$_{\pm 3.36}$ | 23.19$_{\pm 1.36}$ | 23.19$_{\pm 3.12}$ | 25.36$_{\pm 2.23}$ | 23.19$_{\pm 1.36}$ |
| **ScienceWorld** | 33.81$_{\pm 3.37}$ | 40.48$_{\pm 2.94}$ | 44.76$_{\pm 3.75}$ | 43.81$_{\pm 1.78}$ | 38.10$_{\pm 1.35}$ | 40.48$_{\pm 4.71}$ |

Table 11: Per-dataset accuracy (%) on in-distribution test sets for Qwen3-32B evolving from *Simple*. Subscripts denote standard deviations over three runs.

| Dataset | No Memory | Survey | GEPA$^{\dagger}$ | ACE | MemEvolve | CluE |
|---|---|---|---|---|---|---|
| | | | ***Personalization*** | | | |
| **MemBench-High** | 20.50 | 87.00 | 87.00 | 82.00 | 85.50 | 86.00 |
| **MemBench-Low** | 55.00$_{\pm 1.77}$ | 64.58$_{\pm 6.24}$ | 64.58$_{\pm 6.24}$ | 57.08$_{\pm 1.18}$ | 67.08$_{\pm 2.36}$ | 80.83$_{\pm 3.12}$ |
| **PersonaMem-v2** | 19.12 | 24.63 | 24.63 | 24.63 | 22.43 | 26.84 |
| **PrefEval** | 42.00 | 72.00 | 72.00 | 55.50 | 76.50 | 89.00 |
| | | | ***Problem-Solving*** | | | |
| **Game of 24** | 26.67$_{\pm 3.86}$ | 34.17$_{\pm 6.80}$ | 34.17$_{\pm 6.80}$ | 32.50$_{\pm 1.02}$ | 34.58$_{\pm 1.56}$ | 34.58$_{\pm 2.57}$ |
| **AIME** | 27.97$_{\pm 1.14}$ | 31.00$_{\pm 2.64}$ | 31.00$_{\pm 2.64}$ | 29.84$_{\pm 1.44}$ | 31.24$_{\pm 2.16}$ | 31.70$_{\pm 2.01}$ |
| **MMLU-Pro Physics** | 70.50 | 71.00 | 71.00 | 71.50 | 72.00 | 70.50 |
| **MMLU-Pro Eng.** | 42.00 | 50.00 | 50.00 | 47.00 | 55.50 | 51.00 |
| **MathEqBal** | 87.00 | 90.00 | 90.00 | 92.50 | 91.00 | 98.50 |
| **BigCodeBench** | 25.00 | 23.00 | 23.00 | 27.00 | 23.00 | 23.50 |
| | | | ***Agentic*** | | | |
| **AlfWorld** | 30.41$_{\pm 2.71}$ | 47.37$_{\pm 3.12}$ | 47.37$_{\pm 3.12}$ | 46.49$_{\pm 3.28}$ | 40.35$_{\pm 3.12}$ | 49.42$_{\pm 4.07}$ |
| **FEVER** | 37.00 | 40.00 | 40.00 | 42.00 | 36.00 | 40.00 |
| **PDDL** | 18.33$_{\pm 1.18}$ | 16.67$_{\pm 3.12}$ | 16.67$_{\pm 3.12}$ | 16.67$_{\pm 1.18}$ | 15.83$_{\pm 2.36}$ | 20.00$_{\pm 2.04}$ |
| **BabyAI** | 32.25$_{\pm 1.02}$ | 21.01$_{\pm 2.71}$ | 21.01$_{\pm 2.71}$ | 25.00$_{\pm 2.66}$ | 23.55$_{\pm 2.56}$ | 23.55$_{\pm 1.02}$ |
| **ScienceWorld** | 33.81$_{\pm 3.37}$ | 42.86$_{\pm 4.21}$ | 42.86$_{\pm 4.21}$ | 41.90$_{\pm 2.94}$ | 40.95$_{\pm 4.71}$ | 40.48$_{\pm 4.10}$ |

Table 12: Per-dataset accuracy (%) on in-distribution test sets for Qwen3-32B evolving from *Survey*.

### C.2 Prompts for CluE

The prompts for the different LLM roles in the CluE optimization pipeline are provided in Figures 15, 16, 17, and 18.

*Simple* prompt

```
# Role
```

```
You are an expert **Memory Extraction Agent**. Your goal is to
    extract reusable, high-value memories from a single conversation
    , in order to benefit the assistant in future interactions where
     this conversation is no longer accessible.

# Input Format
The conversation is wrapped inside `<conversation>` tags,
    containing:
- `<system>`: System instructions.
- `<user>`: User messages.
- `<assistant>`: Assistant responses.

# Memory Taxonomy

# General Guidelines
- Extract at most 5 memories in total.

# Output Format
Output only the memories as a numbered list.
1. Memory 1
2. Memory 2
...
```

Figure 10: The *Simple* prompt.

*Mem0* prompt

```
You are an Memory Extraction Agent, specialized in accurately
    storing facts, user memories, and preferences. Your primary role
     is to extract relevant pieces of information from conversations
     and organize them into distinct, manageable facts. This allows
    for easy retrieval and personalization in future interactions.
    Below are the types of information you need to focus on and the
    detailed instructions on how to handle the input data.

Types of Information to Remember:

1. Store Personal Preferences: Keep track of likes, dislikes, and
    specific preferences in various categories such as food,
    products, activities, and entertainment.
2. Maintain Important Personal Details: Remember significant
    personal information like names, relationships, and important
    dates.
3. Track Plans and Intentions: Note upcoming events, trips, goals,
    and any plans the user has shared.
4. Remember Activity and Service Preferences: Recall preferences
    for dining, travel, hobbies, and other services.
5. Monitor Health and Wellness Preferences: Keep a record of
    dietary restrictions, fitness routines, and other wellness-
    related information.
6. Store Professional Details: Remember job titles, work habits,
    career goals, and other professional information.
7. Miscellaneous Information Management: Keep track of favorite
    books, movies, brands, and other miscellaneous details that the
    user shares.

Here are some few shot examples:

Input: Hi.
Output: {"facts" : []}
```

```
Input: There are branches in trees.
Output: {"facts" : []}

Input: Hi, I am looking for a restaurant in San Francisco.
Output: {"facts" : ["Looking for a restaurant in San Francisco"]}

Input: Yesterday, I had a meeting with John at 3pm. We discussed
    the new project.
Output: {"facts" : ["Had a meeting with John at 3pm", "Discussed
    the new project"]}

Input: Hi, my name is John. I am a software engineer.
Output: {"facts" : ["Name is John", "Is a Software engineer"]}

Input: Me favourite movies are Inception and Interstellar.
Output: {"facts" : ["Favourite movies are Inception and
    Interstellar"]}

Return the facts and preferences in a json format as shown above.

Remember the following:
- Do not return anything from the custom few shot example prompts
    provided above.
- If you do not find anything relevant in the below conversation,
    you can return an empty list corresponding to the "facts" key.
- Create the facts based on the user and assistant messages only.
    Do not pick anything from the system messages.
- Make sure to return the response in the format mentioned in the
    examples. The response should be in json with a key as "facts"
    and corresponding value will be a list of strings.

Given a conversation between the user and the assistant, you have
    to extract the relevant facts and preferences about the user, if
     any, from the conversation and return them in the json format
    as shown above.
You should detect the language of the user input and record the
    facts in the same language.

# Input Format
The conversation is wrapped inside `<conversation>` tags,
    containing:
- `<system>`: System instructions.
- `<user>`: User messages.
- `<assistant>`: Assistant responses.

# Output Format
Output only the facts in a JSON format as shown in the examples
    above.
```

Figure 11: The *Mem0* prompt.

*ReasoningBank* prompt

```
# Role
You are an expert **Memory Extraction Agent**. Your goal is to
    extract reusable, high-value memories from a single conversation
    , in order to benefit the assistant in future interactions where
     this conversation is no longer accessible.
```

```
# Input Format
The conversation is wrapped inside `&lt;conversation&gt;` tags,
    containing:
- `&lt;system&gt;`: System instructions.
- `&lt;user&gt;`: User messages.
- `&lt;assistant&gt;`: Assistant responses.

# Guidelines
You need to extract and summarize useful insights in the format of
    memory items based on the conversation.
The goal of summarized memory items is to be helpful and
    generalizable for future similar tasks.

# Important notes
- For successful conversations, you must first think why the
    conversation was successful, and then summarize the insights.
- For failed conversations, you must first reflect and think why
    the conversation failed, and then summarize what lessons you
    have learned or strategies to prevent the failure in the future.
- You can extract at most 3 memory items from the conversation.
- You must not repeat similar or overlapping items.
- Do not mention specific websites, queries, or string contents,
    but rather focus on the generalizable insights.

# Output Format
Output only the memories as a numbered list.
1. Memory 1
2. Memory 2
3. Memory 3
```

Figure 12: The *ReasoningBank* prompt.

*OpenMemory* prompt

```
# Role
You are an expert **Memory Extraction Agent**. Your goal is to
    extract reusable, high-value memories from a single conversation
    , in order to benefit the assistant in future interactions where
     this conversation is no longer accessible.

# Input Format
The conversation is wrapped inside `&lt;conversation&gt;` tags,
    containing:
- `&lt;system&gt;`: System instructions.
- `&lt;user&gt;`: User messages.
- `&lt;assistant&gt;`: Assistant responses.

# Memory Taxonomy
## Episodic Memory
- Time-bound events & experiences, e.g., "yesterday I attended a
    conference".

## Semantic Memory
- Timeless facts & knowledge, e.g., "Paris is the capital of France
    ".

## Procedural Memory
- Skills, procedures, how-tos, e.g. "to deploy: build, test, push".

## Emotional Memory
```

```
- Feelings, sentiment, mood, e.g. "I'm excited about this project!"

## Reflective Memory
- Meta-cognition, insights, e.g. "I learn best through practice".

# General Guidelines
- Extract at most 5 memories in total.

# Output Format
Output only the memories as a numbered list.
1. Memory 1
2. Memory 2
...
```

Figure 13: The *OpenMemory* prompt.

*Survey* prompt

```
# Role
You are an expert **Memory Extraction Agent**. Your goal is to
    extract reusable, high-value memories from a single conversation
    , in order to benefit the assistant in future interactions where
     this conversation is no longer accessible.

# Input Format
The conversation is wrapped inside `<conversation>` tags,
    containing:
- `<system>`: System instructions.
- `<user>`: User messages.
- `<assistant>`: Assistant responses.

# Memory Taxonomy
## Factual Memory
Factual memory refers to the assistant's declarative knowledge base
    , established to ensure consistency, coherence, and adaptability
     by recalling explicit facts, user preferences, and
    environmental states.

## Experiential Memory
Experiential memory refers to the assistant's procedural and
    strategic knowledge, accumulated to enable continual learning
    and self-evolution by abstracting from past trajectories,
    failures, and successes.

# General Guidelines
- Extract at most 5 memories in total.

# Output Format
Output only the memories as a numbered list.
1. Memory 1
2. Memory 2
...
```

Figure 14: The *Survey* prompt.

Prompt for Summarizer

```
You are analyzing a memory extraction example to produce an
    extraction-targeted summary.

Your summary should focus on the EXTRACTION SCENARIO -- NOT on the
    surface-level topic or dataset.

Given the following extraction result, write a concise 2-3 sentence
     summary that describes:
1. What type of information needs to be extracted (e.g., factual
    knowledge, procedural steps, user preferences, causal reasoning,
     spatial/temporal relations, emotional context, etc.)
2. What makes this extraction challenging (e.g., long context,
    implicit information, multi-turn reasoning, noise/irrelevant
    content, ambiguous boundaries, need for abstraction vs verbatim
    capture, etc.)

source_conversation (first 4096 chars):
{source_preview}

extracted_memory:
{extracted_memory}

target_conversation (first 4096 chars):
{target_preview}

target_reward: {target_reward}

Write ONLY the extraction-targeted summary. No extra formatting, no
     bullet points -- just 2-3 flowing sentences.
```

Figure 15: Prompt for Summarizer.

Prompt for Cluster Manager

```
You are a clustering agent for memory extraction prompt evolution.

Your task is to group the training examples below into clusters
    based on EXTRACTION SCENARIO PATTERNS -- that is, what type of
    information needs to be extracted and what makes the extraction
    challenging. Do NOT cluster by surface-level task type, dataset
    name, or domain topic.

Good cluster labels describe extraction scenarios, for example:
- "Procedural knowledge in lengthy dialogue"
- "Implicit user preferences in casual multi-turn conversation"
- "Causal reasoning chains in problem-solving episodes"
- "Factual knowledge in noisy conversational context"
- "High-level strategies in step-by-step execution traces"

You may create up to 7 clusters. Use as many as naturally emerge
    from the data -- you do not need to reach the maximum. Each
    cluster must have at least 2 examples. If a pattern has only 1
    example, merge it into the closest cluster.

{existing_pool_section}

Here are the extraction-targeted summaries for the current batch:
<batch_summaries>
```

```
{batch_summaries}
</batch_summaries>

Instructions:
- Assign EVERY example to exactly one cluster.
- If an existing cluster fits well, assign to it. You may update
    its label or description if the new examples reveal a better
    characterization.
- Create new clusters only when existing ones do not adequately
    capture the extraction scenario.
- You may merge two existing clusters if they are too similar, or
    split one if it covers clearly distinct patterns.
- Prefer stable clusters -- only change cluster definitions when
    there is strong evidence from the current batch.

Output format -- return ONLY a JSON object (no markdown, no
    explanation outside JSON):
{
  "clusters": [
    {
      "cluster_id": "cluster_XX",
      "label": "<short extraction scenario label>",
      "description": "<1-2 sentence description of what extraction
    scenario this cluster represents>",
      "example_task_ids": ["0000", "0003", ...]
    },
    ...
  ]
}
```

Figure 16: Prompt for Cluster Manager.

Prompt for Cluster Analyzer

```
You are an expert analysis agent for memory-extraction prompt
    evolution.
You are analyzing a SPECIFIC CLUSTER of examples that share a
    common extraction scenario.

Context:
- Round: {round_id}
- Cluster: {cluster_label}
- Cluster description: {cluster_description}
- Task IDs in this cluster: {task_ids}
- Total examples in cluster: {num_cluster_examples}
- Pair-level logs directory: {logs_dir}

Current base system prompt:
<base_system_prompt>
{base_prompt}
</base_system_prompt>

Your task:
1) Use tools to inspect extraction pair logs for the task IDs
    listed above ONLY.
   Focus exclusively on these tasks -- do not inspect tasks outside
     this cluster.
2) Analyze success and failure patterns SPECIFIC to this cluster's
    extraction scenario.
```

```
3) Propose prompt-level improvements targeted at this cluster's
    extraction scenario.

Hard constraints:
- Do not propose architecture changes.
- Do not propose online memory update loops.
- Focus only on improving extraction system prompt quality.
- Your analysis must be specific to this cluster's extraction
    scenario, not generic.

Required output:
Return a concise but structured report with sections:

**Success Analysis** (for high-reward examples in this cluster):
- Does the extracted memory assist the target request, and how does
     it help?
- What characteristics make this memory useful (e.g., level of
    abstraction, specificity, type of information, transferability)?
- What generalizable insights can be drawn for improving future
    memory extraction guidelines?

**Failure Analysis** (for low-reward examples in this cluster):
- What kind of memory extracted from the input could have better
    supported completing the target request?
- What was missing or ineffective in the extracted memory (e.g.,
    missing key information, too specific, too vague, not
    transferable)?
- Based on this, how should the memory extraction guidelines be
    revised?

**Targeted Prompt Recommendations** (3-5 bullets specific to this
    cluster's scenario)
```

Figure 17: Prompt for Cluster Analyzer.

Prompt for Proposer

```
You are generating an improved memory extraction system prompt
    based on per-cluster analysis reports.

You have received analysis reports from {num_clusters} distinct
    extraction scenario clusters.
Your goal is to synthesize these cluster-specific insights into ONE
     improved system prompt that can handle diverse extraction
    scenarios effectively.

Cluster analyses:
<cluster_analyses>
{cluster_analyses_text}
</cluster_analyses>

Parent prompt id: {parent_prompt_id}
Base system prompt:
<base_system_prompt>
{base_system_prompt}
</base_system_prompt>

Synthesis instructions:
```

```
1. General guidelines: Identify recommendations that appear across
    multiple clusters. Distill these into clear, actionable general
    extraction guidelines that apply regardless of memory type.
2. Memory taxonomy: Group cluster-specific insights into memory
    categories. For each category, provide a concise definition and
    category-specific extraction guidelines. Categories do not need
    to map 1:1 to clusters -- merge or reorganize as needed to
    create a clean, non-overlapping taxonomy.
3. Conflict resolution: When cluster recommendations conflict,
    organize them under the appropriate memory category rather than
    forcing a single uniform rule. Different memory types may
    require different extraction approaches.
4. Stability: Make targeted, high-impact changes. Do not rewrite
    the entire prompt if focused additions or revisions suffice.

Prompt quality requirements -- the generated system prompt must be:
- Written in clear, direct language that another language model can
     follow without ambiguity.
- Structured with explicit sections and formatting so the
    extraction model can quickly locate relevant instructions.
- Actionable: use concrete action verbs (e.g., "extract", "
    summarize", "preserve", "omit") instead of vague descriptions (e
    .g., "be aware of", "consider").
- Include brief examples where they help clarify expected behavior,
     but keep them concise.

Constraints:
- Return JSON only (no markdown, no explanations outside JSON).
- Output must be a JSON array with exactly 1 object.
- Each object schema:
  {
    "candidate_id": "cand_XX",
    "system_prompt": "<full prompt text>",
    "rationale": "<short rationale explaining which cross-cluster
    insights drove the changes>"
  }
- Do not return empty prompts.
```

Figure 18: Prompt for Proposer.

Evolved prompt

```
# Role
You are an expert **Memory Extraction Agent**. Your goal is to
    extract reusable, high-value memories from a single conversation
     to benefit future interactions.

# Input Format
The conversation is wrapped in `<conversation>` tags containing:
- `<system>`: System instructions
- `<user>`: User messages
- `<assistant>`: Assistant responses

# Memory Taxonomy

## 1. Factual Data & Temporal Disambiguation
- **Definition**: Memories resolving ambiguous entities,
    documenting verification attempts, or capturing verifiable facts
     with temporal/event anchors.
- **Guidelines**:
```

```
  - Append clarifiers like [film], [person] to ambiguous terms
  - Document all disambiguation attempts (e.g., search queries, NOT
    ENOUGH INFO conclusions)
  - For temporal conflicts, document both timestamps with message
    references and resolve contradictions using context
  - For verification claims, **explicitly document final
    classification (SUPPORTS/REFUTES/NOT ENOUGH INFO)**
  - Cross-verify all extracted factual claims against source
    conversation with 2+ message references
  - When primary entity search fails, document at least two
    alternative search paths with full result citations

## 2. User Preferences & Emotional Context
- **Definition**: Memories capturing categorical preferences,
    emotional states, or relational dynamics with temporal/event
    anchors.
- **Guidelines**:
  - Map each preference/emotion to **2+ distinct conversation
    threads** (not just messages)
  - Reject memories lacking 2+ distinct temporal anchors
  - Filter out transient emotional states mentioned <2 times
  - For translation/refinement tasks, extract only linguistic
    markers *explicitly* demonstrated in source examples
  - When encountering 'forget' commands, document both original and
    new states with timestamps
  - Include cross-topic consistency check: verify extracted
    preferences/emotions appear in at least 2 conversation threads

## 3. Procedural & Technical Knowledge
- **Definition**: Memories encoding command syntax, formulas, or
    implementation constraints.
- **Guidelines**:
  - Quote command syntax verbatim with at least 2 example usages
    from different contexts
  - For command extractions, **cross-check syntax against system-
    provided examples** and reject non-existent actions
  - Document sequential state transitions and error-handling
    patterns
  - For object manipulation tasks, explicitly list all
    preconditions (e.g., 'must be in same room')
  - For multi-step workflows, include recovery steps if
    intermediate actions fail
  - For technical components (e.g., skateboard rig), document only
    what is explicitly described in the source

## 4. Logical & Combinatorial Reasoning
- **Definition**: Memories abstracting logical rules,
    dimensionality-reduction strategies, or combinatorial formulas
    with validation.
- **Guidelines**:
  - **Cross-verify all extracted formulas/code against source
    conversation line-by-line**
  - Include at least 3 explicit constraints (input validation, edge
    cases, output requirements)
  - For physics problems, include sign convention validation and
    unit consistency checks
  - For multi-step calculations, include at least 2 distinct
    validation methods
  - Distinguish valid reasoning pathways from dead-ends using
    markers like [Valid] and [Exploratory]

## 5. Translation & Stylistic Requirements
```

```
- **Definition**: Memories capturing linguistic style preferences
    and translation constraints.
- **Guidelines**:
  - Extract only stylistic requirements (tone, formality)
    explicitly mentioned in source conversation
  - When requested stylistic adaptations, extract specific
    linguistic markers from source examples
  - For dynamic style changes (e.g., 'forget' commands), document
    explicit before/after state comparisons

# General Guidelines
- Extract at most 5 memories in total
- **Reject memories containing information not directly verifiable
    from source conversation**
- For multi-turn dialogues, abstract recurring patterns before
    specific examples
- For technical problems, distinguish rounding/truncation and
    handle multimodal edge cases
- Always cite conversation turns supporting each memory (e.g., '
    User message 3: ...')
- **Cross-verify all extracted formulas, search result references,
    and message citations against source conversation**
- When extracting code logic, explicitly note deterministic
    requirements (e.g., random seed constraints)
- Require 2+ distinct temporal/event anchors for all preference/
    emotion memories
- For physics problems, include sign convention validation and unit
     consistency checks
- Map extracted memories to the 3-phase problem-solving framework
    when applicable

# Output Format
Output only the memories as a numbered list.
1. Memory 1
2. Memory 2
...
```

Figure 19: Prompt evolved by CluE from *Simple*, using Qwen3-32B.