IMPROVE LARGE LANGUAGE MODEL SYSTEMS WITH USER LOGS

Anonymous authors
Paper under double-blind review

ABSTRACT

Scaling training data and model parameters has long driven progress in large language models (LLMs), but this paradigm is increasingly constrained by the scarcity of high-quality data and diminishing returns from rising computational costs. As a result, recent work increasingly focuses on continual learning from real-world deployment, where user interaction logs provide a rich source of authentic human feedback and procedural knowledge. However, learning from user logs is challenging due to their unstructured and noisy nature. Vanilla LLMs often struggle to distinguish useful feedback signals from noisy user behavior, and the disparity between user log collection and model optimization (e.g., the off-policy optimization problem) further strengthens the problem. To this end, we propose UNO (User log-driveN Optimization), a unified framework for improving LLM systems (LLMsys) with user logs. UNO first distills logs into semi-structured rules and preference pairs, then employs query-and-feedback-driven clustering to manage data heterogeneity, and finally quantifies the cognitive gap between the model's prior knowledge and the log data. This assessment guides the LLMsys to adaptively filter out noisy feedback and construct different modules for primary and reflective experiences extracted from user logs, thereby improving future responses. Extensive experiments show that UNO achieves state-of-the-art effectiveness and efficiency, significantly outperforming Retrieval Augmented Generation (RAG) and memory-based baselines. [1]

1 INTRODUCTION

In recent years, the development of large language models (LLMs) has largely followed scaling laws, in which scaling training data and model parameters yields predictable capability gains (Kaplan et al., 2020; Fang et al., 2024). However, this progress is increasingly constrained by the scarcity of high-quality training data and diminishing marginal returns relative to rapidly rising computational costs (Villalobos et al., 2024). These challenges motivate a growing interest in moving beyond static, pre-trained models toward dynamic systems that continually adapt through real-world interaction (Shi et al., 2025a; Zheng et al., 2025).

In information retrieval (IR), exploiting user logs (e.g., click behavior) to construct self-improving loops is a well-established paradigm for continuous optimization without human supervision (Zhang et al., 2019; Croft et al., 2009). Similarly, deployed LLMs accumulate extensive user logs (e.g., prompts, model outputs, and user feedback) offering large-scale, authentic feedback and valuable procedural memories. These logs represent a rich but underexplored resource for optimizing LLM-based systems (LLMsys) (Ai et al., 2025).

Despite these successes in IR, log-driven continual learning for LLMsys faces fundamental challenges. First, unlike fine-grained click behaviors in IR (Joachims, 2002; Kelly & Teevan, 2003), user feedback for LLMs is sparse and unstructured, making it difficult to extract useful signals. For example, LLMsys provides responses in free-form text, and users either directly exit without interacting with it or only provide coarse-grained verbal or action feedback (e.g., copy) to the response as a whole. Our preliminary study reveals LLMs struggle to distinguish useful knowledge and feedback from noisy signals in user logs, presenting a *Signal-or-Noise Dilemma*.[2] Second, in contrast to IR systems, where

[1] Code is available at https://anonymous.4open.science/r/UNO .

[2] We describe this in detail in Section 3.3.3.

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.

knowledge is explicitly stored outside the retrieval model, LLMs entangle knowledge and reasoning implicitly within their parameters, making autonomous learning highly susceptible to catastrophic forgetting in the parameter space (Gao et al., 2026). Finally, logs collected from product systems may mismatch the target model's distribution. Such off-policy optimization is particularly challenging for LLMs due to their massive parameter size and black-box nature (Levine et al., 2020).

To address these challenges, we introduce **UNO** (User log-driveN Optimization), a framework for continual LLM evolution from user logs. UNO has two stages: learning and inference. In the learning stage, UNO constructs multiple experience modules from collected user logs, each implemented as a specialized adapter functioning either as a generation expert or a critic for iterative refinement. Specifically, for each user session, UNO extracts a semi-structured rule set[3] from the feedback signal (e.g., user corrections), which tells the models how to improve the response. Following these rules, it generates a revised response, forming a preference pair. Then it conducts agglomerative clustering (Müllner, 2011) based on queries and rule sets to divide the logs into multiple clusters. Inspired by Dewey's philosophy on Experience and Nature (Dewey, 2012), UNO constructs two types of specialized parameter-efficient adapters (e.g., LoRA (Hu et al., 2022)) for these clusters: *primary experience modules* and *reflective experience modules*. The primary experience module acts as an expert for direct generation. The reflective experience module learns a critic adapter that critiques a draft response and provides actionable revision suggestions, enabling an iterative refine-and-regenerate loop, without directly changing the base model's weights at inference time. During inference stage, UNO retrieves experience modules relevant to the current context and composes them with the base LLM to generate enhanced responses.

To determine which type of experience module to use for each cluster, we define **cognitive gap** to measure how well the experience extracted from user logs matches the original LLM's understanding of the cluster's queries. If the gap is small, we train a primary experience module and evaluate it (i.e., the LoRA learned directly from user logs) using a simulated verifier based on the LLM-as-Judge paradigm (Li et al., 2025a), using the extracted rules as context. If the verifier determines the performance is satisfactory, the cluster remains a primary experience cluster. Otherwise, or if the cognitive gap is large, the cluster is treated as a reflective experience cluster and user logs are used to construct the critic models that provide suggestions instead of directly changing the parameters. Since user logs may be collected under historical policies or contain noises, this module-level adaptation with verification or critique mitigates off-policy risks and prevents harmful direct updates to the base model. We evaluate UNO on MemoryBench (Ai et al., 2025), a comprehensive continual learning benchmark covering multiple datasets, domains, tasks, and languages, and further evaluate performance on wild and off-policy user logs (WildFB (Peng et al., 2026)). Compared to conventional training, Retrieval-Augmented-Generation (RAG) (Lewis et al., 2020), and memory-based methods, UNO demonstrates state-of-the-art performance in both effectiveness and efficiency.

To summarize, our contributions are as follows:

- We conduct, as far as we know, the first study on the optimization of LLMsys capabilities using user logs and characterize several challenges, including the *Signal–or-Noise Dilemma*.
- We propose UNO, a novel framework for LLMsys continual learning from user logs. It features cluster-based multiple experience management driven by model-log cognitive gaps.
- Extensive experiments show UNO achieves state-of-the-art results across tasks and languages, outperforming RAG and memory baselines.

2 Related Work

2.1 Memory for LLM Systems

Research on LLM memory has evolved into complex systems (e.g., MemoryOS (Kang et al., 2025)) and corresponding benchmarks (e.g., LoCoMo (Maharana et al., 2024)). A key distinction between these and ours is data granularity. While existing systems focus on user-level personalization evaluated on individual logs, we leverage system-wide logs. We aggregate diverse user feedback to drive continual learning and evaluate performance on unseen online requests. Furthermore, they focus on declarative memory while in our scenarios, procedural memory is more critical.

[3] Actionable editing guidelines distilled from unstructured logs. See Section 3.3.1 for details.

2.2 Evolutionary Agent Frameworks

Continual learning (Shi et al., 2025b) often employs evolutionary frameworks (Zhai et al., 2025) (e.g., AlphaEvolve (Novikov et al., 2025)) and memory integration (e.g., MemRL (Zhang et al., 2026)) for self-evolution. However, these systems rely on explicit, repeated reward signals from environments (e.g., code executors). In contrast, we optimize LLMs using real-world user logs. This presents the unique challenge of driving continual learning from implicit, non-repeated feedback, as regular users can not act as dedicated annotators.

2.3 User Log-driven Optimization Paradigms

User log-based optimization is well-established in search and recommendation, but remains relatively nascent in the LLM domain. Recent works explore specific behavioral signals (e.g., clicks) for query suggestions (Yin et al., 2025) or action feedback for emotional support (Han et al., 2025), but they are limited to narrow tasks. While datasets like WildChat (Zhao et al., 2024) and WildFeedback (Shi et al., 2026) provide real-world logs for preference alignment, and SDPO (Buening et al., 2026) proposes self-distillation from user follow-up messages, these approaches focus on aligning preferences rather than enhancing core capabilities. Their off-policy nature, originating from various, often closed-source models, makes it difficult to distinguish feedback-driven capability gains from mere knowledge distillation. Moreover, they lack downstream tasks to evaluate capability improvements. MemoryBench (Ai et al., 2025) resolves evaluation bottlenecks by offering system-specific dialogues and using future queries for evaluation, enabling our paradigm to enhance core LLMsys capabilities via raw user logs.

3 Methodology

In this section, we detail the UNO (**U**ser log-drive**N** **O**ptimization) framework, which enables autonomous, continual LLM learning via raw user logs. We first formulate the task and notations, then outline the framework and its complete workflow (preprocessing, training, and inference).

3.1 Problem Formulation and Preliminaries

Let π_θ denote a base LLM policy with parameters θ. User logs comprise sessions $\mathcal{D} = \{S_i\}_{i=1}^{N}$, where each $S_i = (q_i, y_i, \mathcal{F}_i)$ comprises a user query q_i, an initial response $y_i \sim \pi_\theta(\cdot \mid q_i)$, and a subsequent interaction trajectory $\mathcal{F}_i = \{(u_{i,t}, r_{i,t})\}_{t=1}^{T_i}$ of $T_i \geq 0$ dialogue turns ($u_{i,t}$ is user input, $r_{i,t}$ is LLM response). These trajectories may reflect explicit corrections or implicit preferences from users, but may also contain irrelevant noise or be empty (i.e., $T_i = 0$). The goal is to exploit $\mathcal{D}$ to optimize π_θ into an improved policy π'. Successful optimization requires that for a new test query q_{test}, the expected response quality $\mathcal{V}$ improves:

$$\mathbb{E}_{q_{test}}[\mathcal{V}(\pi'(\cdot \mid q_{test}))] \geq \mathbb{E}_{q_{test}}[\mathcal{V}(\pi_\theta(\cdot \mid q_{test}))],$$

Since user logs may contain both high-value signals and noise, the framework requires: 1) **Positive Adaptivity** to convert valuable signals into capability improvements; and 2) **Noise Robustness** to maintain stable performance against low-quality logs. Although not originally designed for this task, several existing approaches can be adapted. Traditional full fine-tuning updates model parameters to θ' to directly fit the log distribution, yielding $\pi_{FT}(\cdot \mid q_i) = \pi_{\theta'}(\cdot \mid q_i)$. Memory-based systems retrieve external memory entries to construct a context M_i, yielding $\pi_{Mem}(\cdot \mid q_i) = \pi_\theta(\cdot \mid q_i, M_i)$.

3.2 Overview of UNO

UNO is the first optimization framework enabling LLMsys to continuously and adaptively evolve on user logs while maintaining noise robustness. UNO consists of four stages: 1) **Preprocessing**: it filters and distills noisy user logs into a preference dataset, partitions the data, and assesses how well the LLM accepts user feedback to estimate optimization difficulty. This guides constructing either a 2) **Primary Experience Module** or a 3) **Reflective Experience Module** per cluster, inspired by John Dewey's philosophy on Experience and Nature (Dewey, 2012). The former trains an Expert LoRA to directly answer queries, while the latter trains a Critic LoRA that offers guidance on the base LLM's initial response during inference. For the Primary Experience Module, we further build a

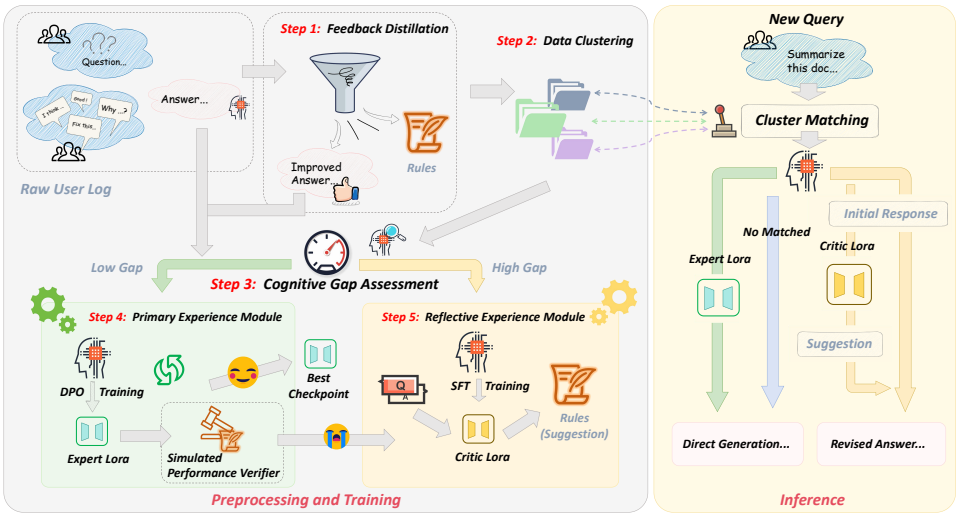

Figure 1: The workflow of UNO. UNO first distills and filters raw user logs, then performs clustering and a cognitive gap assessment to select the type of experience module (primary or reflective). At inference time, UNO identifies the appropriate cluster and applies an inference strategy aligned with the type of that cluster.

simulated performance verifier using distilled rules to evaluate LoRAs before deployment. Finally, at 4) **Inference Workflow**, for a new query, UNO first matches it to the nearest cluster. Primary clusters load the Expert LoRA to answer. Reflective clusters have the base LLM generate an initial answer, and then the Critic LoRA provides feedback, and the LLM revises it accordingly. Figure 1 and Algorithm 1 outline the workflow and training pseudocode, and Appendix D shows case studies.

3.3 Preprocessing

3.3.1 Raw User Log Filtering and Distillation

Raw user logs consist of unstructured text and are potentially noisy (Ai et al., 2025). We utilize the LLM to filter meaningless content and convert the rest into semi-structured rule sets. Specifically, for each dialogue session $(q_i, y_i, \mathcal{F}_i)$ with non-empty $\mathcal{F}_i$, the base LLM performs *feedback filtering and distillation*, transforming unstructured feedback into an explicit rule set $\mathcal{R}_i = \{r_{i,n}\}_{n=1}^{k}$. For instance, if a user complains that a generated journalistic report is too technical and lacks social impact, the distilled rules would be: *1) Use accessible, non-technical news style; 2) Emphasize real-world impact.* We remove items with empty rule sets, as these correspond to uninformative dialogues. Subsequently, the LLM is guided to revise the original response y_i under $\mathcal{R}_i$'s constraints, yielding a better response $y_i^w \sim \pi_\theta(\cdot \mid q_i, \mathcal{R}_i, y_i)$. This constructs a preference pair (q_i, y_i^w, y_i^l), where $y_i^l = y_i$.

3.3.2 Dual-feature Data Clustering

To facilitate training and more precise assessment of noise risk, we cluster the data by jointly leveraging information from both the query and the rule set. We construct a vector $v_i = \text{Norm}[E(q_i) \oplus E(\mathcal{R}_i)]$ where $E(\cdot)$ represents a text encoder and $\oplus$ denotes concatenation. We employ the standard hierarchical agglomerative clustering implementation from *scikit-learn* (Pedregosa et al., 2011), using Ward's linkage (Müllner, 2011) to iteratively merge cluster pairs minimizing the increase in intra-cluster variance, ultimately yielding a set of clusters $\mathcal{C}$ (details in Appendix A.2). This aligns samples within each cluster in semantic intent and applicable rules, thereby lowering the complexity of model training.

3.3.3 COGNITIVE GAP ASSESSMENT

Signal-or-Noise Dilemma. The main challenge in exploiting user logs for model optimization is determining whether they truly contribute to improvement. While LLMs filter noise during distillation, this self-filtering fails when incoming data exceeds their capability boundaries. When signals in logs differ substantially from the model's cognition, two interpretations emerge: (1) they encode unmastered but valuable hard knowledge; or (2) they are dominated by harmful noise or user bias. A self-improving system cannot theoretically distinguish these cases, creating the ***Signal-or-Noise Dilemma***. To address this, we introduce the ***cognitive gap*** to reflect task difficulty and identify potentially high-risk noise within clusters.

Assessment Process. We quantify the cognitive gap through three steps: **1) *Rule Prediction***: For each query q_i in cluster $\mathcal{C}_k$, the base model π_θ independently generates the rule set $\mathcal{R}_i^{LLM}$ without access to user logs. **2) *Gap Quantification***: A pre-trained reranker computes the semantic distance between predicted and distilled rules, yielding a cognitive gap score $g_i = \text{Dist}(\mathcal{R}_i^{LLM}, \mathcal{R}_i) \in [0, 1]$. **3) *Cluster-level Profiling***: We calculate the average gap $\mu_k = \frac{1}{|\mathcal{C}_k|} \sum_{i \in \mathcal{C}_k} g_i$ for each cluster. Based on the value of μ_k and a predefined threshold τ^*, clusters are classified as "Low-Gap" or "High-Gap".

Formalization of Design Principles. We formalize the intuitions behind UNO's cognitive gap assessment, which serves as both a predictor of training cost and a pre-filter for noisy clusters. Let H denote high-quality data and N noisy data, with priors $P(H) = \alpha$ and $P(N) = 1 - \alpha$. In low-gap regions, high-quality data are largely aligned with the model's cognition, whereas noisy data are inherently random, making the two highly separable. In contrast, high-gap regions contain samples deviating substantially from the model's capabilities, causing the LLM to fail at distinguishing "informative novel signals" from "uninformative random noise" (Appendix B.1). We formalize this with the following assumption:

Assumption 3.1. For a small threshold $\tau > 0$, in the low-gap region, $P_N(g_i \leq \tau) \ll P_H(g_i \leq \tau)$. Conversely, in the high-gap region, the distributions of high-quality and noisy data converge, such that $P_H(g_i > 1 - \tau) \approx P_N(g_i > 1 - \tau)$.

Empirical support for this assumption is provided in Appendix F.2.2: a Fisher exact test shows that high-gap clusters fail training at a significantly higher rate ($p = 0.018$).

Proposition 3.2 (Noise Risk Bound)**.** *For data with small cognitive gaps, the posterior probability of noise $P(N|g_i \leq \tau)$ is bounded:*

$$P(N|g_i \leq \tau) \leq \frac{1}{1 + \alpha \frac{P_H(g_i \leq \tau)}{P_N(g_i \leq \tau)}} \tag{1}$$

The proof is based on Bayes' theorem and detailed in Appendix B.2. As P_H/P_N grows sharply in the low-gap region, training risk decreases rapidly with smaller τ. In contrast, for high-gap regions ($g_i > 1 - \tau$), the posterior approaches $P(N|g_i > 1 - \tau) \approx 1 - \alpha$, indicating hard samples become indistinguishable from noise without a meaningful risk bound. Next, we illustrate that under some assumptions (e.g., L_Ψ-Lipschitz continuity), clustering concentrates the cognitive gaps within each cluster, forming distinct low- and high-gap clusters to guide our optimization strategy.

Proposition 3.3 (Variance Reduction via Clustering)**.** *Under UNO's dual-feature clustering, the intra-cluster variance of the cognitive gap Var(g_i) is bounded by the cluster diameter ϵ:*

$$Var(g_i) < [C \cdot (1 + L_\Psi) \cdot \epsilon]^2 \tag{2}$$

where C and L_Ψ are constants.

The proof bounds intra-cluster gap differences using Lipschitz continuity and the triangle inequality (Appendix B.3). According to Proposition 3.3, clustering reduces the variance of g_i by bounding diameter ϵ, enabling reliable decisions based on μ_k. Clusters with low μ_k contain absorbable knowledge; clusters with high μ_k lack reliable risk bounds, so UNO adopts *Reflective Experience Construction*. These propositions motivate the design (Appendix B.4); in practice, the cognitive gap serves as a pre-filter, while a simulated verifier provides post-filtering (Appendix F).

3.4 PRIMARY EXPERIENCE MODULE

For clusters below the cognitive gap threshold τ^*, the data serve only as a fine-tuning of the model's cognition, and the risk of noise is controlled. Thus, we directly internalize log-derived capability by training a LoRA on preprocessed preference data.

Cluster-specific Preference Learning. We partition data into training and validation sets. For low- or moderate-gap clusters, we train an *Expert LoRA* $\Delta\theta_C$ using the Direct Preference Optimization (DPO (Rafailov et al., 2023)) loss alongside a negative log-likelihood (NLL) loss on chosen responses for stability. This directly embeds new capabilities into the parameter space.

Simulation-based Validation. We construct a simulated performance verifier from the validation set to assess training efficacy. For each query q_i, using $\mathcal{R}_i$ as context, an LLM-as-Judge compares $y_i^C \sim \pi_{\theta+\Delta\theta_C(\cdot|q_i)}$ against the base $y_i \sim \pi_\theta(\cdot|q_i)$ per epoch. Crucially, by strictly grounding the judge in the rule set $\mathcal{R}_i$ extracted from actual user feedback, we effectively mitigate potential "judge-hacking" where the adapter might otherwise exploit the base model's inherent biases. We retain the expert module if its peak win rate $\text{WinRate}_{best} > \gamma$. Otherwise, we fall back to *Reflective Experience Construction*. Furthermore, to accelerate the evaluation process and further alleviate potential reward-hacking, we pre-filter responses that are essentially empty or deviate drastically from the original response (e.g., those with a BLEU-4 score (Papineni et al., 2002) below a small threshold, such as 0.05).

3.5 REFLECTIVE EXPERIENCE MODULE

For clusters with large cognitive gaps or those failing primary validation, we train a *Critic LoRA* $\Delta\omega_C$, which is designed to provide "pseudo user feedback" $\hat{\mathcal{R}}$ for the model's initial responses. The training input consists of a query q_i and a model response y_i, while the output is an extracted rule set $\mathcal{R}_i$, optimized using a negative log-likelihood loss. The checkpoint with the lowest validation loss becomes the final Critic LoRA. During inference, Critic LoRA critiques the base LLM's initial output, allowing the base LLM to refine its response. Our experiments show that, for clusters with high noise risk, this strategy effectively leverages informative signals in the logs and significantly reduces noise-induced collapse compared to directly training an Expert LoRA. Moreover, the Reflective Experience Module prevents UNO's optimization framework from becoming overly conservative, ensuring robust utilization of valuable user-log signals even for clusters filtered out by cognitive gap assessment or unsuccessful primary experience construction.

3.6 INFERENCE WORKFLOW

During inference, for a query q_{new}, the policy $\pi_{\theta'}$ computes the Euclidean distance between q_{new}'s embedding and the query embeddings of all cluster centroids, and assigns it to the closest cluster C^*. Response generation depends on C^*'s type:

- **Primary Path**: For a **primary experience cluster**, we sample directly from its *Expert LoRA*: $y \sim \pi_{\theta+\Delta\theta_{C^*}}(\cdot|q_{new})$.
- **Reflective Path**: For a **reflective experience cluster**, we use a two-stage generation: $y_{\text{init}} \sim \pi_\theta(\cdot|q_{new})$, $\hat{\mathcal{R}} \sim \pi_{\theta+\Delta\omega_{C^*}}(\cdot|q_{new}, y_{\text{init}})$, and $y \sim \pi_\theta(\cdot|q_{new}, y_{\text{init}}, \hat{\mathcal{R}})$.
- **Fallback**: If q_{new} is an outlier, i.e., having a distance to the nearest centroid exceeding threshold d, we revert to the base policy: $y \sim \pi_\theta(\cdot|q_{new})$.

4 EXPERIMENTAL SETUP

Next, we introduce the evaluation and implementation details. Further details are in Appendix C.

4.1 EVALUATION

We evaluate the methods on **MemoryBench** (Ai et al., 2025), the first benchmark for user log-driven continual learning, where a carefully validated *User Simulator* produces interaction logs conditioned on each model's responses. The benchmark includes user logs collected during the early stage (the

Table 1: Main Results in MemoryBench. "Short-Long" denotes the "Short-Input-Long-Output" task. Use MemoryBench's aggregate metrics (↑): Norm-Score (min–max normalized) and Z-Score. Scores are only meaningful for relative comparison within the same model and dataset. We bold the best, underline the second best, and shade our proposed methods in gray. "Base" denotes the initial response. "**" denotes significantly worse performance than UNO ($p < 0.05$, paired t-test). "-" means failure to generate valid output.

Model	Type	Method Name	Short-Long		Short-Short		Long-Long		Long-Short	
			Norm-Score	Z-Score	Norm-Score	Z-Score	Norm-Score	Z-Score	Norm-Score	Z-Score
Qwen3-8B	-	Base	74.43**	-9.52**	72.35**	-2.31**	63.41	14.87	46.94	6.02
	RAG	Embedding	73.22**	-17.04**	72.96**	4.21**	62.04	7.58**	48.41	12.29
		BM25	74.54**	-8.83**	72.71**	3.50**	61.49**	6.50**	48.40	10.87
	Memory	MemOS	72.29**	-23.07**	66.57**	-27.26**	57.95**	-7.71**	47.14	-12.16**
		ReMem	71.95**	-25.55**	70.80**	-7.69**	59.09**	-4.30**	43.78**	-7.70**
		A-Mem	70.36**	-35.02**	70.15**	-10.89**	60.55**	2.22**	47.94	5.91
		Mem0	68.25**	-51.93**	61.07**	-48.34**	58.29**	-5.70**	-	-
		MemoryOS	74.62**	-8.96**	70.56**	-5.68**	45.96**	-59.89**	38.51**	-24.87**
	Training	SDPO$_{w/\ UNO\ Data}$	70.35**	-35.09**	68.75**	-28.07**	50.54**	-39.93**	44.53**	-24.24**
		DPO$_{w/\ UNO\ Data}$	70.39**	-34.68**	72.89**	-8.77**	61.65**	6.87**	47.14	2.30
		SFT$_{w/\ UNO\ Data}$	69.95**	-37.85**	75.03	5.42**	62.24	8.98**	48.80	7.00
		UNO-Single	76.36**	2.99	75.82	15.89	63.77	15.66	46.81	5.34
		UNO	**77.09**	**7.16**	**76.26**	**21.54**	**64.23**	**17.74**	**49.99**	**13.49**
phi-4	-	Base	66.67**	-63.21**	72.17	-21.81	57.07	-17.36	46.46**	-1.59**
	RAG	Embedding	66.27**	-64.57**	72.82	-20.79	54.43**	-28.63**	42.91**	-9.62**
		BM25	69.33	-44.72	71.03	-30.36**	53.67**	-31.60**	43.26**	-15.07**
	Memory	MemOS	67.96**	-53.78**	67.40**	-40.49**	53.71**	-31.43**	42.83**	-26.43**
		ReMem	66.66**	-61.81**	58.51**	-92.61**	47.72**	-54.90**	38.78**	-35.81**
		A-Mem	67.27**	-57.95**	70.40**	-29.58	53.22**	-32.94**	39.69**	-34.26**
		Mem0	68.81	-48.25	65.11**	-58.86**	52.23**	-37.53**	-	-
		MemoryOS	22.44**	-350.94**	49.72**	-256.49**	24.23**	-161.25**	25.59**	-89.25**
	Training	SDPO$_{w/\ UNO\ Data}$	66.44**	-64.43**	62.55**	-103.62**	47.88**	-50.87**	34.19**	-75.07**
		DPO$_{w/\ UNO\ Data}$	67.17**	-58.71**	71.85	-16.77	52.54**	-36.95**	32.59**	-54.20**
		SFT$_{w/\ UNO\ Data}$	66.97**	-60.05**	70.43**	-34.00**	55.63**	-24.44**	38.97**	-30.81**
		UNO-Single	69.61**	-43.08**	72.91	-18.04	57.14	-17.10	47.16**	1.93**
		UNO	**70.66**	**-36.69**	**73.19**	**-15.94**	**57.84**	**-13.62**	**52.60**	**18.40**

training set) and new requests (the test set). All strategies are performed on the training set and evaluated on the test set across four task-based datasets: Short-input–Long-output, Short-input–Short-output, Long-input–Long-output, and Long-input–Short-output. Each task contains logs (dialogue sessions, including model responses and user verbal feedback) derived from queries in distinct data subsets. Since test queries span datasets, MemoryBench first computes dataset-specific metrics and applies min–max normalization or z-score for final task-level performance. For both, higher values correspond to better performance. Raw dataset-specific results are in Appendix G.

Moreover, to assess performance on real human interactions, we experiment on **WildFB** (Peng et al., 2026), a subset of WildChat (Appendix C.1). Using WildReward-8B from the original paper as the evaluator, we compute the win rate of generated versus original responses. This task mainly evaluates optimization frameworks under realistic noise and user behavior. Notably, a gap exists between this setting and our primary evaluation, as the models in dialogues differ from those being optimized.

We evaluate strategies using Qwen3-8B (Yang et al., 2025) (no-thinking mode) and phi-4 (14B) (Abdin et al., 2024). We also test a lighter variant, UNO-Single, in which the Reflective Path is removed, and responses for filtered clusters are generated by the base LLM. UNO-Single adopts more conservative optimization without additional inference. Significance is tested via paired t-test (Student, 1908).

4.2 BASELINES

Prior work lacks a pipeline specifically for user log-driven optimization. Besides the model's direct output, we compare against three related categories: 1) **RAG**: We use BM25 (Robertson & Zaragoza, 2009) or Qwen3-Embedding-0.6B (Zhang et al., 2025) as retriever. 2) **Memory for LLM Systems**: We evaluate MemOS (Li et al., 2025b), ReMem (Wei et al., 2026), A-Mem (Xu et al., 2025),

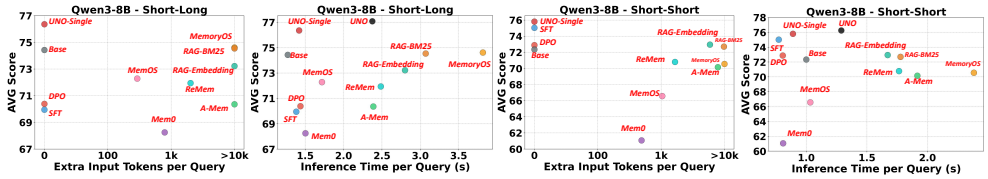

Figure 2: Performance (Norm-Score) vs. extra tokens and inference time. Better cases lie upper-left. Token metrics use UNO-Single; inference time evaluates the full UNO pipeline.

Mem0 (Chhikara et al., 2025), and MemoryOS. 3) **Training Methods**: Using UNO's preference data, we evaluate SFT, DPO, and Self-Distillation Policy Optimization (SDPO (Buening et al., 2026)).

4.3 Implementation Details of UNO

UNO uses Qwen3-Embedding-0.6B for clustering. When constructing preference data, both original logs and extracted rules provide context for candidate generation, selecting the best output. For cognitive gap, we use Qwen3-Reranker-0.6B to evaluate whether each extracted rule is independent of the base LLM's knowledge, extracting the model-generated scores to compute $\text{Dist}(\mathcal{R}_i, \mathcal{R}_i^{\text{LLM}})$ (Appendix C.3). For the simulated verifier, we compare each checkpoint against the base LLM, where the judge model is the base LLM itself but using rules extracted from user logs as context.

Batch-wise Adaptation Settings. We further evaluate UNO on phi-4 in a batch-wise adaptation setting to test if it can continue to learn using newly collected logs after deployment. We evenly split the training data into two batches. After initial optimization on the first batch, UNO interacts with the User Simulator via the second batch to collect new user logs. A key consideration in batch-wise adaptation is whether clustering should preserve the original centroids. We perform clustering over all user logs and adopt new centroids if cluster counts change, then retrain using the full logs. Otherwise, we keep the original centroids and perform continual training.

5 Experimental Results

5.1 Main Results

Main Performance. Table 1 shows UNO outperforms baselines on MemoryBench across models and subtasks. UNO's adaptive dual-path framework extracts knowledge from noisy logs more effectively than RAG or Memory systems. Applying full DPO or SFT to UNO's preference data proves unstable, as the model struggles to distinguish complex cases from noise even after LLM-based filtering. Figure 3 confirms UNO's superiority on real-world, off-policy logs (WildFB), underscoring the necessity of its **cognitive gap assessment** and **clustering** mechanisms.

Challenges of Learning from User Logs. On complex datasets like Long-Long, where user bias and noise in raw logs can easily divert the model's attention, all baselines fail to achieve positive gains. This highlights UNO's core advantage: identifying and addressing high-risk clusters via the **reflective path**, safely extracting valuable signals while preserving capabilities.

Efficiency-Performance Trade-off. Figure 2 plots efficiency against performance (detailed in Appendix C.4), measured by extra input tokens and inference time. Since full UNO's reflective path adds output tokens and LLM calls, for the token analysis, we report UNO-Single (primary path only). UNO-Single matches the zero-input-token advantage of direct training while outperforming Memory systems that struggle with noise. For inference time, full UNO selectively executes the reflective path, incurring only a 1.5-1.8$\times$ average overhead, and is faster than most Memory baselines while achieving peak performance. Together, UNO and UNO-Single offer a flexible trade-off.

Batch-wise Adaptation. Figure 4 evaluates UNO in a batch-wise adaptation setting using phi-4. UNO naturally supports batch-by-batch iteration, showing steady gains with added logs. Batch-wise and single-batch adaptation exhibit complementary strengths: the latter uses a weaker initial model to gather more feedback, while the former employs incremental log collection for more informative feedback. This indicates the potential online evolution settings of UNO.

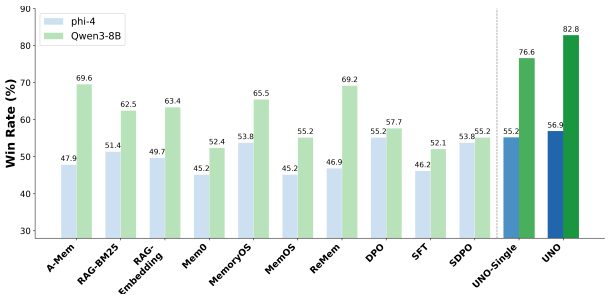

Figure 3: Main results in WildFB, measuring the win rate of generated against original base responses.

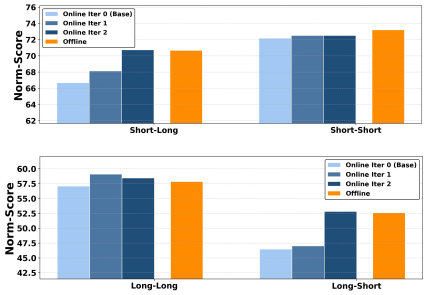

Figure 4: Results of batch-wise adaptation on phi-4. "Offline" represents the single-batch setting in the main table.

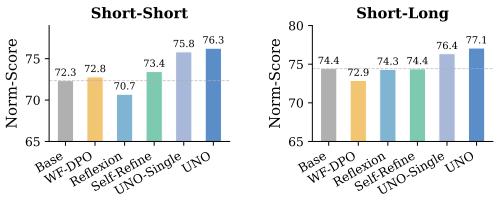

Figure 5: Comparison with out-of-setting baselines on MemoryBench (Qwen3-8B, Norm-Score). WF-DPO is WildFeedback-DPO.

Table 2: Ablation study of UNO (Qwen3-8B). UNO-Rfl/Prm use only Reflective/Primary Module. UNO-Prm is identical to UNO-Single.

	Short-Long		Short-Short	
	Norm-S	Z-Score	Norm-S	Z-Score
RAG-Emb (UNO's Rules)	71.50	-29.88	67.74	-22.14
RAG-BM25 (UNO's Rules)	72.36	-24.36	66.48	-31.72
UNO w/o Clustering	70.95	-32.82	74.73	9.84
DPO w/ UNO's Verifier	74.43	-9.52	72.35	-2.31
UNO-Rfl	75.44	-3.38	72.79	3.34
UNO-Prm (UNO-Single)	76.36	2.99	75.82	15.89
UNO	**77.09**	**7.16**	**76.26**	**21.54**

Comparison with Out-of-setting Baselines. We additionally compare UNO with baselines under different assumptions (Figure 5): Self-Refine (Madaan et al., 2023) and Reflexion (Shinn et al., 2023) perform test-time self-critique without user logs, while WildFeedback-DPO (Shi et al., 2026) trains on user logs with human-predefined rubrics. Self-Refine and Reflexion yield limited gains, confirming that self-generated critique cannot substitute for real user feedback. WildFeedback-DPO trains a single global adapter without cluster-level adaptation or post-training verification. UNO-Single, using only the Primary Path, also outperforms all these baselines, confirming that UNO's advantage stems from adaptive exploitation of user feedback. Protocols are in Appendix C.2.1.

5.2 Ablation Study

We ablate UNO on the Short-Long and Short-Short tasks for Qwen3-8B (Table 2). Removing clustering (*UNO w/o Clustering*) degrades performance, showing that decoupling tasks provides a reliable basis for cognitive gap assessment. Applying UNO's verifier to a global DPO adapter rejects it entirely, reverting to Base performance. The dual-path optimization proves complementary and superior to single-path variants: *UNO-Prm* is overly conservative in noisy settings, while *UNO-Rfl* increases latency and underperforms on low-gap tasks. Retrieving distilled rules via RAG underperforms the base model due to poor generalization and introduced noise, whereas UNO internalizes this knowledge via LoRA training. Additional ablations on per-query rule count, training objective (DPO+NLL vs. SDPO), cognitive gap threshold, no-feedback self distillation, clustering distance thresholds, simulated verifier, and clustering feature isolation are in Appendix F.

6 Conclusion

We present UNO, a continual learning framework optimizing LLMsys via user logs. To address the Signal-or-Noise Dilemma, UNO integrates dual-feature clustering and cognitive gap assessment to adaptively route logs for direct optimization or critique-based refinement. Experiments on MemoryBench and WildFB confirm state-of-the-art performance across tasks, languages, and models, demonstrating that raw user logs can effectively drive continuous improvement of deployed LLMs.

AI USAGE STATEMENT

In this work, we used generative AI tools to refine hypotheses (specifically, iterating on the formalization of the cognitive gap mechanism and its assumptions). We have not used generative AI tools for other required-disclosure tasks: generating synthetic datasets, developing theoretical models or conceptual frameworks, formulating mathematical claims, proposing hypotheses, writing of the proofs, designing research methodology or experiments, implementing methods, translation, data cleaning, qualitative analysis, or interpreting results. Additionally, we used generative AI tools to polish selected paragraphs for readability. All AI-assisted content was manually reviewed by the authors: the formulization were re-derived independently, and polished text was checked against the original intent. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, our implementation of UNO, including all preprocessing, training, and evaluation scripts, is open-sourced at `https://anonymous.4open.science/r/UNO`. The evaluation benchmarks used in this work (MemoryBench and WildFB) are publicly available with their official evaluation code. Detailed hyperparameters are reported in Appendix C.3, baseline configurations in Appendix C.2.1, and the offline processing cost and hardware setup in Appendix C.4. We hope these measures facilitate the replication of our results and foster future studies on user log-driven continual learning for LLM systems.

REFERENCES

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Xin Wang, Rachel Ward, Yue Wu, Dingli Yu, Cyril Zhang, and Yi Zhang. Phi-4 technical report, 2024. URL `https://arxiv.org/abs/2412.08905`.

Qingyao Ai, Yichen Tang, Changyue Wang, Jianming Long, Weihang Su, and Yiqun Liu. Memorybench: A benchmark for memory and continual learning in llm systems, 2025. URL `https://arxiv.org/abs/2510.17281`.

Thomas Kleine Buening, Jonas Hübotter, Barna Pásztor, Idan Shenfeld, Giorgia Ramponi, and Andreas Krause. Aligning language models from user interactions, 2026. URL `https://arxiv.org/abs/2603.12273`.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

W. Bruce Croft, Donald Metzler, and Trevor Strohman. *Search Engines - Information Retrieval in Practice*. Pearson Education, 2009. ISBN 978-0-13-136489-9. URL `http://www.search-engines-book.com/`.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, et al. Deepseek-v3 technical report, 2025. URL `https://arxiv.org/abs/2412.19437`.

John Dewey. *Experience and nature*. Courier Corporation, 2012.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. 2024.

Yan Fang, Jingtao Zhan, Qingyao Ai, Jiaxin Mao, Weihang Su, Jia Chen, and Yiqun Liu. Scaling laws for dense retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1339–1349, 2024.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, Hongru Wang, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Qihan Ren, Cheng Qian, Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence, 2026. URL https://arxiv.org/abs/2507.21046.

Eric Han, Jun Chen, Karthik Abinav Sankararaman, Xiaoliang Peng, Tengyu Xu, Eryk Helenowski, Kaiyan Peng, Mrinal Kumar, Sinong Wang, Han Fang, and Arya Talebzadeh. Reinforcement learning from user feedback, 2025. URL https://arxiv.org/abs/2505.14946.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pp. 133–142, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 158113567X. doi: 10.1145/775047.775067. URL https://doi.org/10.1145/775047.775067.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory OS of AI agent. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25961–25970, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1318. URL https://aclanthology.org/2025.emnlp-main.1318/.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Diane Kelly and Jaime Teevan. Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum*, 37(2):18–28, September 2003. ISSN 0163-5840. doi: 10.1145/959258.959260. URL https://doi.org/10.1145/959258.959260.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020. URL https://arxiv.org/abs/2005.01643.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, Kai Shu, Lu Cheng, and Huan Liu. From generation to judgment: Opportunities and challenges of llm-as-a-judge, 2025a. URL https://arxiv.org/abs/2411.16594.

Zhiyu Li, Chenyang Xi, Chunyu Li, Ding Chen, Boyu Chen, Shichao Song, Simin Niu, Hanyu Wang, Jiawei Yang, Chen Tang, Qingchen Yu, Jihao Zhao, Yezhaohui Wang, Peng Liu, Zehao Lin, Pengyuan Wang, Jiahao Huo, Tianyi Chen, Kai Chen, Kehang Li, Zhen Tao, Huayi Lai, Hao Wu, Bo Tang, Zhengren Wang, Zhaoxin Fan, Ningyu Zhang, Linfeng Zhang, Junchi Yan,

Mingchuan Yang, Tong Xu, Wei Xu, Huajun Chen, Haofen Wang, Hongkang Yang, Wentao Zhang, Zhi-Qin John Xu, Siheng Chen, and Feiyu Xiong. Memos: A memory os for ai system, 2025b. URL https://arxiv.org/abs/2507.03724.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594. Curran Associates, Inc., 2023. doi: 10.52202/075280-2019. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91edff07232fb1b55a505a9e9f6c0ff3-Paper-Conference.pdf.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms, 2011. URL https://arxiv.org/abs/1109.2378.

Curtis G. Northcutt, Lu Jiang, and Isaac L. Chuang. Confident learning: Estimating uncertainty in dataset labels, 2022. URL https://arxiv.org/abs/1911.00068.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. Alphaevolve: A coding agent for scientific and algorithmic discovery, 2025. URL https://arxiv.org/abs/2506.13131.

OpenAI. Introducing GPT-5.2, December 2025. URL https://openai.com/index/introducing-gpt-5-2/. OpenAI blog post.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin (eds.), *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040/.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Hao Peng, Yunjia Qi, Xiaozhi Wang, Zijun Yao, Lei Hou, and Juanzi Li. Wildreward: Learning reward models from in-the-wild human interactions, 2026. URL https://arxiv.org/abs/2602.08829.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, April 2009. ISSN 1554-0669. doi: 10.1561/1500000019. URL https://doi.org/10.1561/1500000019.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. Continual learning of large language models: A comprehensive survey. *ACM Comput. Surv.*, 58(5), November 2025a. ISSN 0360-0300. doi: 10.1145/3735633. URL https://doi.org/10.1145/3735633.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. Continual learning of large language models: A comprehensive survey. *ACM Comput. Surv.*, May 2025b. ISSN 0360-0300. doi: 10.1145/3735633. URL https://doi.org/10.1145/3735633. Just Accepted.

Taiwei Shi, Zhuoer Wang, Longqi Yang, Ying-Chun Lin, Zexue He, Mengting Wan, Pei Zhou, Sujay Jauhar, Sihao Chen, Shan Xia, Hongfei Zhang, Jieyu Zhao, Xiaofeng Xu, Xia Song, and Jennifer Neville. Wildfeedback: Aligning llms with in-situ user interactions and feedback, 2026. URL https://arxiv.org/abs/2408.15549.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

Student. The probable error of a mean. *Biometrika*, pp. 1–25, 1908.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9275–9293, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.746. URL https://aclanthology.org/2020.emnlp-main.746/.

Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. Position: will we run out of data? limits of llm scaling based on human-generated data. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020.

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, 2026. URL https://arxiv.org/abs/2511.20857.

Wujiang Xu, Kai Mei, Hang Gao, Juntao Tan, Zujie Liang, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Junhao Yin, Haolin Wang, Peng Bao, Ju Xu, and Yongliang Wang. From clicks to preference: A multi-stage alignment framework for generative query suggestion in conversational system, 2025. URL https://arxiv.org/abs/2508.15811.

Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, Zhaoyang Liu, Bolin Ding, and Jingren Zhou. Agentevolver: Towards efficient self-evolving agent system, 2025. URL https://arxiv.org/abs/2511.10395.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yutao Qi, Bo Tang, and Muning Wen. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory, 2026. URL https://arxiv.org/abs/2601.03192.

Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv.*, 52(1), February 2019. ISSN 0360-0300. doi: 10.1145/3285029. URL https://doi.org/10.1145/3285029.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models, 2025. URL https://arxiv.org/abs/2506.05176.

Haiquan Zhao, Chenhan Yuan, Fei Huang, Xiaomeng Hu, Yichang Zhang, An Yang, Bowen Yu, Dayiheng Liu, Jingren Zhou, Junyang Lin, Baosong Yang, Chen Cheng, Jialong Tang, Jiandong Jiang, Jianwei Zhang, Jijie Xu, Ming Yan, Minmin Sun, Pei Zhang, Pengjun Xie, Qiaoyu Tang, Qin Zhu, Rong Zhang, Shibin Wu, et al. Qwen3guard technical report, 2025. URL https://arxiv.org/abs/2510.14276.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models, 2026. URL https://arxiv.org/abs/2601.18734.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatgpt interaction logs in the wild. 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=Bl8u7ZRlbM.

Junhao Zheng, Shengjie Qiu, Chengming Shi, and Qianli Ma. Towards lifelong learning of large language models: A survey. *ACM Comput. Surv.*, 57(8), March 2025. ISSN 0360-0300. doi: 10.1145/3716629. URL https://doi.org/10.1145/3716629.

A Details of Methodology of UNO

A.1 Pseudo Code of Training Workflow of UNO

We present the training pseudocode of UNO at Algorithm 1.

A.2 Details of Dual-feature Data Clustering

In this subsection, we elaborate on the implementation details of the dual-feature clustering process introduced in Section 3.3.2. Our approach aims to construct a clustering space that captures both the user's semantic intent (query) and the domain-specific constraints (rules), while maintaining compatibility with inference scenarios where only the query is available.

A.2.1 Feature Extraction and Fusion

For a given data point containing a query q_i and a corresponding rule set $\mathcal{R}_i$, we utilize a pre-trained sentence encoder (e.g., `Qwen3-Embedding-0.6B`) to extract dense representations. To ensure that both the query and the rule set contribute equally to the distance metrics during clustering, we apply L2-normalization independently to both embeddings before concatenation. The final dual-feature vector v_i is formulated as:

$$v_i = \text{Norm}\left(E(q_i) \oplus E(\mathcal{R}_i)\right)$$

where $E(\cdot)$ is the sentence encoder which produces normalized embeddings, $\text{Norm}(\cdot)$ denotes L2-normalization, and $\oplus$ is the concatenation operator. This final normalization step ensures that v_i remains a unit vector, which is optimal for Euclidean distance-based clustering.

A.2.2 Agglomerative Clustering and Ward's Linkage

We employ Agglomerative Hierarchical Clustering to group the dual-feature vectors. A critical component of this algorithm is the linkage criterion, which determines the distance between sets of observations. We specifically adopt **Ward's Linkage** (Ward's minimum variance method) (Müllner, 2011).

Algorithm 1 Training Framework of UNO

Input: Initial policy π_θ, User logs $\mathcal{D} = \{(q_i, y_i, \mathcal{F}_i)\}_{i=1}^{N}$, Thresholds $\tau^*, \gamma, \epsilon_{var}$
Output: Expert LoRAs $\{\Delta\theta_k\}$ and Critic LoRAs $\{\Delta\omega_k\}$
/* Data Construction */
$\mathcal{D}_{pref} \leftarrow \emptyset$
for each $(q_i, y_i, \mathcal{F}_i) \in \mathcal{D}$ **do**
  $\mathcal{R}_i \leftarrow \text{Distill}(\mathcal{F}_i)$
  if $\mathcal{R}_i \neq \emptyset$ **then**
    Sample $y_i^w \sim \pi_\theta(\cdot \mid q_i, \mathcal{R}_i, y_i)$
    $\mathcal{D}_{pref} \leftarrow \mathcal{D}_{pref} \cup \{(q_i, y_i^w, y_i, \mathcal{R}_i)\}$
  end if
end for
/* Dual-feature Clustering */
$\{\mathcal{C}_k\}_{k=1}^{K} \leftarrow \text{AgglomerativeClustering}(v_i, \epsilon_{var})$, where $v_i = [\text{Norm}(E(q_i)) \oplus \text{Norm}(E(\mathcal{R}_i))]$
/* Adaptive Optimization for Different Clusters */
for each cluster $\mathcal{C}_k$ **do**
  // Cognitive Gap Assessment
  for each sample $i \in \mathcal{C}_k$ **do**
    $\mathcal{R}_i^{LLM} \sim \pi_\theta(\cdot \mid q_i)$
    $g_i = \text{Dist}(\mathcal{R}_i^{LLM}, \mathcal{R}_i)$
  end for
  $\mu_k = \frac{1}{|\mathcal{C}_k|} \sum_{i \in \mathcal{C}_k} g_i$
  status $\leftarrow$ Reflective
  if $\mu_k \leq \tau^*$ **then**
    // Primary Experience Construction
    $\Delta\theta_k \leftarrow \arg\min_\Delta \mathbb{E}_{\mathcal{C}_k}[\mathcal{L}_{DPO}(\pi_{\theta+\Delta}; \pi_\theta) + \alpha\mathcal{L}_{NLL}(y^w)]$
    $BestWinRate \leftarrow \text{SimulatedVerifier}(\pi_{\theta+\Delta\theta_k}, \pi_\theta, \mathcal{C}_k^{val})$
    if $BestWinRate > \gamma$ **then**
      status $\leftarrow$ Primary
    end if
  end if
  if *status* == Reflective **then**
    // Reflective Experience Construction
    // Training a Critic LoRA for this cluster
    $\Delta\omega_k \leftarrow \arg\min_\Delta \mathbb{E}_{\mathcal{C}_k}[-\log \pi_{\theta+\Delta}(\mathcal{R}_i \mid q_i, y_i)]$
  end if
end for

Unlike other linkage methods (e.g., single or complete linkage) that measure distances between individual points in different clusters, Ward's linkage focuses on variance. At each step of the agglomerative process, the algorithm evaluates all possible pairs of clusters and merges the pair that results in the minimum increase in the total intra-cluster variance (or error sum of squares, ESS). Mathematically, the distance $d(u, v)$ between two clusters u and v in Ward's method is defined as the increase in ESS when they are merged:

$$d(u, v) = \text{ESS}(u \cup v) - (\text{ESS}(u) + \text{ESS}(v))$$

This linkage is particularly well-suited for our Euclidean space vectors, as it tends to produce compact and evenly sized clusters, effectively aligning samples with similar semantic intents and applicable rules. In our implementation, the agglomerative merging process stops once the scaled distance $d(u, v)$ exceeds a threshold of 4.

A.2.3 Asymmetric Centroid Calculation and Inference

A key innovation in our implementation is the decoupling of the clustering structure space and the inference space. While the cluster assignments $\mathcal{C}$ are derived using the dual-feature vectors v_i, the cluster centroids are computed *exclusively* in the query embedding space.

Specifically, for a cluster c_k, its centroid μ_k is calculated as the mean of the normalized query embeddings for all samples assigned to c_k:

$$\mu_k = \frac{1}{|c_k|} \sum_{j \in c_k} \text{Norm}(E(q_j))$$

This asymmetric design bridges the gap between training and inference. During the inference phase, a new user query q_{new} typically arrives without rules, since the rule set is extracted from user feedback. We simply compute its query embedding $E(q_{new})$ and assign it to the cluster with the nearest centroid μ_k using Euclidean distance. Furthermore, we define a maximum distance threshold $d_{max} = 1.2$, and if the distance to the nearest centroid exceeds this threshold, the query is flagged as an out-of-distribution class.

B Details of Propositions

B.1 Discussion on Assumption 3.1

The foundational premise of our assumption rests on the varying capacity of LLMs to differentiate noise from high-quality data across different cognitive boundaries. When the cognitive gap is small, high-quality instances generally represent incremental adjustments that align well with the model's existing knowledge representations, making them easily distinguishable from inherently unstructured noise. Conversely, as the cognitive gap widens, the instances increasingly diverge from the model's current capabilities. In these high-gap regions, we struggle to differentiate between truly informative, complex signals and meaningless random noise, resulting in the convergence of their cognitive distributions.

Empirical findings from deep learning research in natural language processing provide strong support for this intuition. In a study on dataset cartography (Swayamdipta et al., 2020), researchers map out dataset regions based on model behavior during training and identify a distinct category of instances that models find "hard-to-learn". Upon conducting qualitative analysis on these "hard-to-learn" instances, they observe that the underlying composition of this region is notably mixed. On one hand, this region captures a significant amount of mislabeled instances, which directly correspond to uninformative random noise. On the other hand, it simultaneously includes valid instances that are inherently challenging even for human annotators, representing complex, novel signals. Crucially, because both genuine noise and valid, complex information are co-located in the exact same region from the model's perspective, the model is essentially blind to their differences. This empirical observation perfectly corroborates our assumption: in high-gap areas, the model's inability to reliably distinguish between informative novel signals and uninformative random noise causes the distributions of high-quality data and noisy data to inherently converge.

B.2 Proof of Proposition 3.2

Proposition 3.2 (Noise Risk Bound) *For data with small cognitive gaps, the posterior probability of noise $P(N|g_i \leq \tau)$ is bounded:*

$$P(N|g_i \leq \tau) \leq \frac{1}{1 + \alpha \frac{P_H(g_i \leq \tau)}{P_N(g_i \leq \tau)}} \tag{3}$$

Proof. By Bayes' theorem, the posterior probability that a sample is noise is given by

$$P(N \mid g_i \leq \tau) = \frac{P(g_i \leq \tau \mid N)\, P(N)}{P(g_i \leq \tau \mid N)\, P(N) + P(g_i \leq \tau \mid H)\, P(H)}.$$

Substituting $P(N) = 1 - \alpha$ and $P(H) = \alpha$ leads to

$$P(N \mid g_i \leq \tau) = \frac{P_{\text{noise}}(g_i \leq \tau)(1-\alpha)}{P_{\text{noise}}(g_i \leq \tau)(1-\alpha) + P_H(g_i \leq \tau)\alpha}.$$

The theorem holds trivially when $\alpha = 1$. When $\alpha < 1$, normalizing by dividing both the numerator and denominator by $P_{\text{noise}}(g_i \leq \tau)(1-\alpha)$ yields

$$P(N \mid g_i \leq \tau) = \frac{1}{1 + \frac{\alpha}{1-\alpha} \frac{P_H(g_i \leq \tau)}{P_{\text{noise}}(g_i \leq \tau)}}.$$

Since $1 - \alpha \in (0, 1]$, we have $\frac{\alpha}{1-\alpha} \geq \alpha$, which establishes Proposition 3.2. □

B.3 Proof of Proposition 3.3

Assumption B.1. The LLM rule-generation function $\Psi : \mathcal{Q} \to \mathcal{R}$ is L_Ψ-Lipschitz continuous. The semantic distance obeys the axioms of a metric (or at least the triangle inequality), and the function $Dist(\cdot, \cdot)$ used to evaluate the cognitive gap is treated as a semantic distance. Moreover, the embedding distance provides an upper bound on the semantic distance: $d_{\text{sem}}(A, B) \leq C \cdot |E(A) - E(B)|$

Proposition 3.3 (Variance Reduction via Clustering) *Under UNO's dual-feature clustering, the intra-cluster variance of the cognitive gap Var(g_i) is bounded by the cluster diameter ϵ:*

$$\text{Var}(g_i) < [C \cdot (1 + L_\Psi) \cdot \epsilon]^2 \tag{4}$$

where C and L_Ψ are constants.

Proof. Consider any two samples $i, j \in \mathcal{C}_k$ belonging to the same cluster. By the Lipschitz continuity of Ψ,

$$d_{\text{sem}}(\mathcal{R}_i^{LLM}, \mathcal{R}_j^{LLM}) \leq L_\Psi \cdot d_{\text{sem}}(q_i, q_j).$$

For a cluster with diameter ϵ,

$$|(E(q_i) \oplus E(\mathcal{R}_i)) - (E(q_j) \oplus E(\mathcal{R}_j))| < \epsilon.$$

This implies $|E(q_i) - E(q_j)| < \epsilon$ and $|E(\mathcal{R}_i) - E(\mathcal{R}_j)| < \epsilon$, and consequently

$$d_{\text{sem}}(q_i, q_j) < C \cdot \epsilon, \quad d_{\text{sem}}(\mathcal{R}_i, \mathcal{R}_j) < C \cdot \epsilon.$$

Thus,

$$\begin{aligned} |g_i - g_j| &\leq |g_i - \text{Dist}(\mathcal{R}_i^{LLM}, \mathcal{R}_j)| \\ &\quad + |\text{Dist}(\mathcal{R}_i^{LLM}, \mathcal{R}_j) - g_j| \\ &= |\text{Dist}(\mathcal{R}_i, \mathcal{R}_i^{LLM}) - \text{Dist}(\mathcal{R}_j, \mathcal{R}_i^{LLM})| \\ &\quad + |\text{Dist}(\mathcal{R}_j, \mathcal{R}_i^{LLM}) - \text{Dist}(\mathcal{R}_j, \mathcal{R}_j^{LLM})| \\ &\leq \text{Dist}(\mathcal{R}_i, \mathcal{R}_j) + \text{Dist}(\mathcal{R}_i^{LLM}, \mathcal{R}_j^{LLM}) \\ &\leq C \cdot (1 + L_\Psi) \cdot \epsilon. \end{aligned} \tag{5}$$

By the definition of variance, Proposition 3.3 is proved. □

B.4 Empirical Verification of Assumptions

Proposition 3.3 relies on three assumptions: (i) the embedding distance upper-bounds the semantic distance, i.e., $d_{\text{sem}}(A, B) \leq C \cdot \|E(A) - E(B)\|$; (ii) the rule generation function Ψ is L_Ψ-Lipschitz in the embedding space; and (iii) the semantic distance Dist approximately satisfies metric properties (symmetry and the triangle inequality). We verify each assumption on the actual rule sets from the main experiments (Short-Short: 336 queries, Short-Long: 353 queries), using the same Qwen3-Reranker-0.6B scores as in the paper. For each split, we sample 300 intra-cluster point pairs and 300 triples.

Embedding upper bound. Setting $C = 1.0$ covers 99.0% (SS) and ∼96% (SL) of intra-cluster pairs; full coverage requires $C = 1.08$ (SS) and $C = 1.24$ (SL). The Spearman correlation between d_{sem} and embedding distance is 0.77 (SS) and 0.83 (SL).

Lipschitz chain. Over 99% of intra-cluster pairs satisfy $D(\mathcal{R}_i^{\text{LLM}}, \mathcal{R}_j^{\text{LLM}}) \leq 1.1 \cdot \|E(q_i) - E(q_j)\|$ (p99 ratio: 1.02–1.09 across splits), yielding an effective $C \cdot L_\Psi \approx 1.1$.

Metric properties. Mean asymmetry of intra-cluster pairs is 0.03–0.06 (SS/SL). The triangle inequality holds for 96–99% of sampled triples; violations have mean magnitude 0.06–0.16.

These results confirm that the assumptions hold approximately in our experimental setting. We emphasize that Propositions 3.2 and 3.3 serve as design motivation for the cognitive gap mechanism, providing formal intuition for why low-gap clusters are safer to train on and why clustering helps. The actual routing decisions in UNO are governed by the empirically calibrated threshold τ^* and the simulated verifier, not by the propositions directly.

C Experimental Details

C.1 Additional Evaluation Details

Introduction of MemoryBench. MemoryBench spans multiple languages, tasks, domains, and datasets. Its log data are generated by a carefully designed and validated *User Simulator* that produces interaction logs conditioned on each model's own responses. The simulator outputs are validated via A/B testing, in which human annotators cannot distinguish simulator-generated logs from real user logs. The benchmark comprises two parts: user logs collected during the early stage (the training set) and subsequent new requests (the test set). All strategies are performed on the training set and evaluated on the test set. For dataset sampling, we follow the main experimental setting of MemoryBench, splitting the data into training and test sets at a 4:1 ratio.

Evaluation Metrics of MemoryBench. MemoryBench comprises 19 subsets from 11 public datasets, each evaluated using its original dataset-specific metric. These metrics fall into two categories. *Deterministic metrics* include F1 (Locomo), Accuracy (DialSim), Rouge-L (LexEval), and METEOR (WritingPrompts). *LLM-as-Judge metrics* are used for datasets with open-ended outputs (e.g., HelloBench, WritingBench, NF-Cats, LimitGen-Syn); for datasets with multiple evaluation criteria (e.g., IdeaBench, JuDGE, SciTechNews), an LLM-as-Judge integrates these sub-metrics into a single score. All evaluation metrics and prompts follow the official MemoryBench protocol without modification (Ai et al., 2025). The metric design of each dataset has been validated by the original authors through human evaluation. MemoryBench uses DeepSeek-V3 (DeepSeek-AI et al., 2025) as the judge for most datasets, with the exception of WritingBench subsets, which use their official critic model.

Since test queries for each task originate from multiple datasets, MemoryBench first computes these dataset-specific metrics and then applies either min–max normalization or z-score to obtain the final task-level performance. For both aggregated metrics, higher values correspond to better performance. We adopt the normalization parameters provided by the official repository.

Independence of Evaluation. The evaluation judge model (DeepSeek-V3) is independent of the model being optimized (Qwen3-8B or phi-4), the simulated performance verifier (the base LLM itself), and the rule extraction model (also the base LLM). This separation ensures that evaluation results are not confounded by any component in UNO's training pipeline.

Details of Evaluation on WildFB. We filter the dataset to include only initial user queries without prior dialogue history to ensure that user feedback remains on a single topic and to facilitate evaluation, retaining strictly English and Chinese conversations identified via language detection. We randomly sample 1,200 instances for the evolution logs (training set) and reserve the remainder for the evaluation tasks (test set), which aligns with the data scale of MemoryBench. For evaluation, we adopt WildReward-8B [4] from the original WildFB paper as the evaluator and compute the win rate of the generated responses against the original responses in the dataset.

C.2 Detailed Baseline Settings

- **Retrieval-Augmented Generation (RAG):** In our setting, we treat each dialogue session as an entry and use the test question as the search query to retrieve relevant user queries from the dialogues. We use either BM25 (Robertson & Zaragoza, 2009) or Qwen3-Embedding-0.6B (Zhang et al., 2025) as the retriever and include entries relevant to the top-5 retrieved queries as the LLM context. When the context exceeds the LLM's maximum length, we truncate the documents using a bisection strategy.
- **Memory for LLM Systems:** We evaluate MemOS (Li et al., 2025b), ReMem (Wei et al., 2026), A-Mem (Xu et al., 2025), Mem0 (Chhikara et al., 2025), and MemoryOS (Kang et al., 2025). All user logs are organized at the session level for memory storage, and the resulting systems are treated as evolved systems, uniformly using the top-5 entries. For the latter three methods, we follow all settings from the original MemoryBench paper. Although MemOS provides a complete API for the memory system, to ensure a consistent base LLM across all the methods, we use the official memory and retrieval APIs and evaluate on the test set with the same base LLM as other baselines. Note that the Long-Short dataset contains extremely long contexts, and we find that

[4] https://huggingface.co/THU-KEG/WildReward-8B

Mem0 cannot process them and cannot produce responses within a reasonable time. Thus, we do not report Mem0's performance on this dataset. This finding is consistent with the conclusions reported in the original MemoryBench paper.

- **Training Methods:** Using preference data constructed by UNO, we evaluate three offline training approaches: supervised fine-tuning (SFT), DPO, and Self-Distillation Policy Optimization (SDPO (Buening et al., 2026)). All training settings follow UNO's settings. We select the epoch that achieves the lowest cross-entropy loss on the validation set as the final evaluation epoch.

C.2.1 Out-of-setting Baselines

We reproduce the following baselines on UNO's training logs with the same train/validation split.

WildFeedback-DPO. We follow the protocol of WildFeedback (Shi et al., 2026). Each dialogue session $(q_i, y_i, \mathcal{F}_i)$ is classified with the original 9+9 SAT/DSAT(either sactisfactory or not) rubrics (per-rubric Y/N plus a 1–10 score, judging user utterances only). A preference summary is extracted, and on-policy preference pairs are generated: y^w is sampled with the preference summary plus a safety instruction as system prompt, while y^l is a zero-shot regeneration at temperature 0.1. Training proceeds in two stages: SFT (batch 128, lr 1×10^{-4}) followed by DPO (batch 32, lr 1×10^{-5}, β=0.1; SAT:DSAT ratio 10:1). The checkpoint with the lowest cross-entropy on the validation set is selected. A single global adapter is trained without clustering.

Self-Refine. We follow the core loop of Self-Refine (Madaan et al., 2023): for each test query, the same Qwen3-8B generates an initial answer, produces a self-critique (with a stop marker), and revises accordingly, for up to 3 rounds. No training dialogues are used and no cross-sample memory is accumulated.

Reflexion. We follow the core loop of Reflexion (Shinn et al., 2023): for each test query, Qwen3-8B generates an initial answer, self-evaluates whether it is satisfactory, produces a verbal reflection that accumulates within the query, and regenerates conditioned on all prior reflections, for up to 3 rounds. No training dialogues or cross-sample memory are used.

C.3 Details of UNO's Settings

Preference Data Construction. For each user log instance (query q, initial model response y, and user feedback f), UNO first extracts a set of actionable rules $\mathcal{R}$ via rule distillation. To construct candidate revised responses, we use both original logs and extracted rules to provide context for candidate generation. Each candidate is then scored on a 1–10 scale by the base LLM itself to evaluate adherence to the extracted rules. The highest-scoring candidate is selected as the chosen response y^+, while the original model response y serves as the rejected response y^-, forming the preference pair (q, y^+, y^-) for DPO training.

Clustering and Training. In agglomerative clustering, we stop merging clusters when the inter-cluster variance increment reaches 4. In the cluster matching process, distances greater than 1.2 are treated as outliers. We set the LoRA rank to 64 and the dropout rate to 0.05. Training is performed with the TRL library (von Werra et al., 2020) using AdamW (β_1=0.9, β_2=0.98, ϵ=10^{-8}, weight decay 10^{-4}) with a cosine learning rate schedule, peak learning rate 5×10^{-4}, for 8 epochs. The per-device batch size is 1 with 8 gradient accumulation steps (effective batch size 32 on 4 GPUs). The maximum sequence length is 12,000 tokens for Expert training and 8,000 for Critic training, in bf16 precision. For DPO, we set the beta parameter to 0.1 and assign equal weights (0.5) to the DPO loss and the NLL loss.

Evaluation and Thresholds. For cognitive gap assessment, we measure the novelty of each feedback-derived rule with respect to the LLM's own knowledge. Specifically, for a given instance i, let $\mathcal{R}_i = \{r_1, r_2, \ldots, r_m\}$ denote the rules extracted from user feedback, and $\mathcal{R}_i^{\text{LLM}} = \{t_1, t_2, \ldots, t_n\}$ denote the rules generated by the LLM itself. We use Qwen3-Reranker-0.6B (Zhang et al., 2025) to compute a relevance score for each rule r against $\mathcal{R}_i^{\text{LLM}}$; a high relevance score indicates that r is already covered by the LLM's knowledge. The cognitive gap score is defined as the minimum novelty (i.e., one minus relevance) across all rules:

$$s_i = \text{Dist}(\mathcal{R}_i, \mathcal{R}_i^{\text{LLM}}) = \min_{r \in \mathcal{R}_i} \left(1 - \text{Relevance}(r, \mathcal{R}_i^{\text{LLM}})\right), \tag{6}$$

where Relevance$(\cdot)$ is the reranker score. Taking the minimum provides a conservative estimate of risk: a high s_i guarantees that all rules are novel to the LLM, reliably identifying the cluster as high-gap and routing it to the Reflective Path. A low s_i indicates that at least one rule overlaps with existing knowledge, suggesting the cluster may be safe for direct Expert training; if Expert training nonetheless fails, the downstream simulated verifier catches it. This design makes the pre-filter precise about flagging high-risk clusters (high $s_i \geq \tau^*$), while delegating uncertain cases to the verifier. Furthermore, the overall cognitive gap score for a cluster is computed as the average across all instances:

$$S_{\text{cluster}} = \frac{1}{N} \sum_{i=1}^{N} s_i \tag{7}$$

We set the threshold τ^* of the cognitive gap to 0.45, which is the median of the cluster-level gap scores computed on the training set across both models and all datasets on MemoryBench. No test-set information is used in selecting τ^*. In the LLM-as-Judge verifier, we sample the judge 3 times and average the scores (from 1 to 10). We set the win-rate threshold to 0.53 (0.03 above the base performance) and the BLEU threshold to 0.05; answers with BLEU scores below this threshold relative to the original responses are directly assigned a score of 0. We use vLLM (Kwon et al., 2023) for inference, and adopt the same inference settings as MemoryBench, with a temperature of 0.1.

Batch-wise Adaptation Criteria. For unchanged clusters during batch-wise adaptation, we perform continual training of the expert LoRA using incremental data, requiring the win rate to exceed the pre-adaptation level by 0.03; otherwise, we retrain using the full dataset. For LoRA in the Reflective Experience Module, we retrain using the full data and require the best validation loss to decrease by more than 0.2 to ensure stable optimization. Due to the limited amount of distilled preference data for batch-wise training with Qwen3-8B in some datasets, we conduct the batch-wise experiments using phi-4.

Prompt Templates. The following prompt extracts raw user logs into a semi-structured set of rules.

System Prompt of Rule Set Generation from User Feedback.

Role
You are a conversation analysis expert.

Task
Your task is to analyze a conversation snippet that includes an initial user query, the model's initial response, and subsequent user feedback. Based on your analysis, you need to summarize and output the following two key pieces of information in JSON format:

1. Whether the user is satisfied with the model's initial response.
2. Extract rules or suggestions from the user's feedback that can help the model better answer the initial query.

Input Format
You will receive a text containing the following three parts:

[Initial User Query]
[Initial Model Response]
[User Feedback] (May contain feedback from one or more rounds of conversation)

Output Requirements
You must strictly follow the JSON format below for your output:

{
"rules": [
"<string>",
"<string>",
...
]

}

Field Descriptions
- rules (list of strings): A list containing all the effective suggestions or rules extracted from the user feedback.

Important Constraints

1. Only extract feedback that is directly related to the [Initial User Query]. If the user's replies deviate from the initial question (e.g., start small talk or ask a new question), ignore this irrelevant content.
2. If all feedback is unrelated to the initial question, or if the user expresses satisfaction without providing any specific suggestions, this list can be empty [].
3. Each rule should be a clear, actionable instruction. It should be in affirmative form, stating what to do rather than what not to do.

The following prompt is used for revising the initial response according to the suggestions, both for data preprocessing and the reflective path at inference time.

Prompt of Response Revision

Initial Answer
{old_answer}

Suggestion
{suggestion}

User Question
{question}

Follow the suggestion to revise the initial answer. Directly output the revised complete answer that accurately and naturally addresses the user's question. Don't repeat the question itself at the beginning of your response.

The following prompt is used for judging the response according to the extracted rule set in the simulated validation module during training of the Primary Path.

Prompt of LLM-AS-JUDGE of Simulated Validation

ROLE
You are a rigorous and objective AI Answer Quality Evaluator.

TASK
Your task is to comprehensively evaluate the quality of the provided [Answer] based on the [Question] and the [Expert Suggestion]. You must output a final integer score from 1 to 10.

CORE INSTRUCTIONS

1. **[Expert Suggestion] is the Primary Standard**: You must strictly adhere to the [Expert Suggestion]. It outlines the core direction, key points to include, or critical errors to avoid for a high-quality answer. The degree to which the [Answer] follows the [Expert Suggestion] is the most critical factor in determining its score.

2. **Comprehensive Judgment**: While the [Expert Suggestion] is paramount, it may not cover all evaluation dimensions. Therefore, you must also apply your own knowledge and judgment to assess the answer based on these supplementary criteria:
* **Accuracy**: Is the information in the answer factually correct and free of errors?

* **Completeness**: Does the answer fully address all parts of the [Question]?
* **Clarity & Logic**: Is the answer well-structured, easy to understand, and logically sound?
* **Relevance**: Does the answer stay focused on the [Question] without including irrelevant information?

SCORING RUBRIC

Use the following guidelines to assign an integer score from 1 to 10:

* **9-10 (Excellent):**
* Exemplary adherence to all key points in the [Expert Suggestion].
* The answer is highly accurate, comprehensive, and clearly articulated.
* May provide additional valuable insights beyond the direct scope of the question.

* **7-8 (Good):**
* Strongly adheres to the core points of the [Expert Suggestion].
* The answer is factually correct and effectively addresses the question.
* May have minor imperfections, such as a slight lack of detail or minor stylistic issues.

* **5-6 (Average):**
* Partially adheres to the [Expert Suggestion] but with noticeable deviations or omissions.
* Answers the main part of the [Question] but contains some inaccuracies, is incomplete, or lacks clarity.
* The answer is fundamentally acceptable but has clear room for improvement.

* **3-4 (Poor):**
* Largely disregards or contradicts the [Expert Suggestion].
* Contains significant factual errors, logical flaws, or is substantially incomplete.
* Fails to effectively answer the [Question].

* **1-2 (Very Poor):**
* Completely fails to follow the [Expert Suggestion].
* The answer is factually incorrect, irrelevant to the question, nonsensical, or potentially harmful.

OUTPUT FORMAT

Your output **MUST** be a single JSON object that strictly follows the format below. Do not add any extra text, comments, or explanations before or after the JSON code block.

```json
{
"reason": "str",
"score": "int"
}
```

- reason (str): Provide a detailed and specific explanation for the score you have given. In your reasoning, you must explicitly reference the [Expert Suggestion] and describe the extent to which the [Answer] adhered to it. Also, incorporate your assessment of other criteria (e.g., accuracy, completeness).
- score (int): Your integer score between 1 and 10.

EVALUATION CONTENT
Question

{question}

Expert Suggestion
{suggestion}

Answer
{answer}

C.4 Details of Efficiency Evaluation

For the calculation of efficiency metrics, all token counts are computed using the Qwen3-8B tokenizer. Regarding the measurement of inference time, our evaluation is conducted on GPUs with computational capabilities equivalent to the NVIDIA H100-SXM-80GB. The models are deployed using the vLLM framework distributed across 4 GPUs, with concurrency 10. We enable the batch-invariant setting during inference.

Furthermore, our evaluation deliberately excludes the time consumed by offline processing phases, such as model training and the construction of retrieval indices. Instead, we focus exclusively on the online inference latency, which encompasses the time required for LLM generation, retrieval matching, cluster matching, and so on.

Offline Processing Cost. UNO's offline stage consists of data construction (log filtering, rule distillation, cognitive gap scoring, and clustering) followed by per-cluster training and verification. The cognitive gap assessment determines which clusters train an Expert LoRA directly and which train a Critic LoRA, so not every cluster requires both modules. All generation and scoring are performed on locally deployed vLLM instances (4-GPU tensor parallelism, concurrency 64), incurring no external API cost.

Table 3: Wall-clock time of one complete offline adaptation (4 GPUs, Qwen3-8B).

Stage	Short-Short	Short-Long
Data construction	∼10 min	∼10 min
Expert training	1.4 h	1.8 h
Verification	∼6 min	∼12 min
Critic training	1.1 h	1.8 h
Test generation & aggregation	∼6 min	∼6 min
Total	**∼2.9 h**	**∼4.1 h**

Table 4: Wall-clock comparison between UNO and a single-adapter DPO baseline (4 GPUs, Qwen3-8B).

Task	DPO	UNO Expert	UNO Critic	UNO Total
Short-Short	2.9 h	1.4 h	1.1 h	**2.5 h**
Short-Long	3.0 h	1.8 h	1.8 h	3.6 h

Table 3 reports the wall-clock time for a complete offline adaptation on Qwen3-8B. Table 4 compares UNO's training wall-clock against a single-adapter DPO baseline trained on the same data. UNO is not necessarily slower than DPO: when the cognitive gap correctly predicts cluster difficulty, Expert and Critic training are each performed on smaller data subsets and finish faster than global DPO. Additional cost arises only when the prediction is wrong, i.e., an Expert fails verification and a Critic must be trained on the same cluster. UNO's overhead beyond DPO (clustering, cognitive gap scoring, and verification) takes less than 20 minutes.

Storage and Module Switching. Each LoRA adapter occupies 349 MB. At serving time, all adapters for a split are loaded once at startup via vLLM's runtime LoRA interface without restarting the service. Batch-loading all adapters from a memory-backed directory takes 1.61 s in total (median

0.12 s per adapter). At inference time, the only additional per-request cost is a single nearest-centroid distance comparison for routing.

D CASE STUDIES OF UNO

Figure 6 illustrates how UNO transforms raw user logs into training data. The "Extracted Rule Set" comprises semi-structured revision suggestions derived from user feedback. Conditioned on the initial response, the user feedback, and the rule set, the model generates a revised response. The initial and revised responses are treated as the rejected and chosen responses, respectively, for DPO training.

Figure 7 presents the inference workflow of UNO. The Reflective Path loads the Critic LoRA, produces suggestions based on the initial response, and then prompts the base LLM to generate a revision. In contrast, the Primary Path directly loads the Expert LoRA to produce the response. In this case, both approaches outperform the base LLM's initial output. The Reflective Path incorporates more detailed considerations in the legal judgment, whereas the Primary Path appends the complete legal provisions, thereby rendering the judgment document more comprehensive.

E DISCUSSION ON LIMITATIONS AND INDUSTRIAL-GRADE SCALABILITY

While UNO achieves state-of-the-art performance on existing benchmarks, our evaluations are primarily conducted on academic datasets that do not fully encapsulate the extreme scale and diversity of real-world industrial LLM services processing millions of user sessions daily. Furthermore, the development of user log-driven continual learning benchmarks is still in its infancy. We encourage the community to build more comprehensive, large-scale benchmarks to better evaluate LLMsys evolution in the wild. UNO's Reflective Path also involves a multi-stage inference process. Although we have shown that this design outperforms most retrieval-based memory methods in inference latency (Figure 2), for strictly latency-constrained applications the overhead is non-trivial. Below, we discuss the feasibility of scaling UNO to industrial scenarios and a future direction for folding the Reflective Path into single-stage generation.

In our main experiments, UNO demonstrates state-of-the-art performance on existing academic-level benchmarks. However, a critical question arises when deploying UNO in real-world, industrial LLM services, where the system might process hundreds of millions of user interactions daily: *How does the framework scale regarding cluster count, algorithmic complexity, and inference serving overhead?* To address this, we provide a theoretical discussion to explore the industrial feasibility of UNO, outlining proposed architectures and concepts for managing massive-scale deployments.

Quantitative Estimation of LoRA Modules Consider a hypothetical industrial scenario with 10^8 daily user sessions. Even after rigorous pre-filtering (e.g., retaining only 1% of sessions with explicit and actionable user feedback), the system would process approximately 10^6 training samples. To maintain the fine-grained semantic and rule alignment that UNO requires, assuming an optimal capacity of $1,000$ to $5,000$ sessions per cluster, the system would generate approximately 200 to $1,000$ distinct clusters. Consequently, this translates to maintaining and serving up to $1,000$ specialized LoRA modules (Expert or Critic).

Algorithmic Scalability of Clustering A primary bottleneck in scaling UNO directly is the clustering phase. The current implementation utilizes Agglomerative Clustering, which has a computational complexity ranging from $\mathcal{O}(N^2)$ to $\mathcal{O}(N^3)$. Applying this naively to $N = 10^6$ samples is computationally intractable. To adapt UNO for such scales, we conceptualize and propose a two-stage hierarchical clustering pipeline as a highly scalable alternative. In the proposed first stage, a highly scalable algorithm such as an approximate nearest neighbor (ANN) search library like FAISS (Douze et al., 2024) could perform coarse-grained clustering to divide the massive log data into dozens of macro-clusters. In the theoretical second stage, the computationally intensive Agglomerative Clustering could be safely executed in parallel within each macro-cluster to perform fine-grained refinement, strictly bounding the maximum N for any single agglomerative operation.

Raw User Logs:

Query

Write a report of this paper in journalistic style.
Title: Real-time decoding of question-and-answer speech dialogue using human cortical activity
Content: *Natural communication often occurs in dialogue... However, previous attempts to decode speech directly from the human brain typically consider listening or speaking tasks in isolation. Here, human participants listened to questions and responded aloud... while we used high-density electrocorticography (ECoG) recordings to detect when they heard or said an utterance... [...].*

Initial Response

Breakthrough in Brain-Computer Interfaces: Real-Time Decoding of Conversational Speech from the Human Brain
In a groundbreaking study, researchers have taken a major step forward by successfully decoding real-time question-and-answer speech dialogues directly from human cortical activity. [...]

Using high-density electrocorticography (ECoG), the team was able to detect when participants heard or produced an utterance. [...]

The results were impressive. The team achieved an accuracy rate of up to 61% for decoding produced utterances and 76% for decoding perceived ones... [...]

User Feedback

Looks good! But can you add a quick comparison to the Pancho study to give it some context? Moreover, could you also add a brief explanation of what ECoG actually is? Lay readers might not know the term.

Processed Data:

Extracted Rule Set

Rule 1: *Include a brief comparison to relevant prior studies (e.g., the Pancho study) to provide context for the research's significance.*
Rule 2: *Explain technical terms like ECoG for readers who may not be familiar with them.*

Revised Response:

Breakthrough in Brain-Computer Interfaces: Real-Time Decoding of Conversational Speech from the Human Brain
In a groundbreaking study published in Nature Neuroscience, researchers have taken a major step forward in the field of brain-computer interfaces (BCIs)... [...]

The study... was conducted by a team of neuroscientists and engineers... ***Unlike previous efforts—such as the foundational Pancho study, which focused on decoding speech in isolation—****the researchers designed an experiment that mimicked real-life dialogue, where participants listened to questions and responded aloud with answers.*

Using high-density electrocorticography (ECoG)—a clinical technique involving electrodes placed directly on the brain's surface to record neural activity with high spatial and temporal resolution*—the team was able to detect when participants heard or produced an utterance. The key innovation lay in the integration of context... [...]*

The results were impressive. The team achieved an accuracy rate of up to 61%... [...]

Figure 6: Case of Training Data Preprocessing. This item is from SciTechNews dataset, which belongs to the Short-Input-Short-Output subset of MemoryBench.

Query

Role: *You are a legal assistant. Please draft a complete, legally valid Chinese criminal judgment based on the provided facts.*

Case Facts: *Since March 2015, Defendant Wang conspired with accomplices to set up a fraudulent webpage, deceiving victims into transferring money. The total defrauded amount was 10,400 RMB, with Wang taking a 70% share. Wang later voluntarily surrendered to the police. The prosecutor charged Wang with fraud and recommended a sentence of 6 to 12 months imprisonment along with a fine. The accomplices' families have fully compensated the victims.*

Task: *Generate a formal judicial document that is logically structured and rigorously adheres to official writing standards.*

Reflective Path

Step 1: Initial Response

[...Standard court headers, defendant info, and factual findings omitted for brevity...]
Court's Opinion: *The court holds that Defendant Wang's actions constitute fraud. Since Wang surrendered voluntarily and the accomplices have compensated the victims, a lighter punishment may be given.*

Judgment:
I. Defendant Wang is guilty of fraud, sentenced to six months of imprisonment, and fined 5,000 RMB.
II. Defendant Wang is ordered to refund the illegal gains of 7,280 RMB, which shall be recovered.
[...Standard closing formats omitted...]

Step 2: Suggestions from Critic LoRA

1. Explicitly state whether the public prosecutor's sentencing recommendation is adopted, and elaborate on the reasoning in the court's opinion section.
2 . Explicitly state the recovery of illegal gains and reflect this decision in the main body of the judgment.

Step 3: Final Revised Response

[...Standard court headers, defendant info, and factual findings omitted for brevity...]
Court's Opinion: *The court holds that Defendant Wang's actions constitute fraud. [...] The court adopts the sentencing recommendation proposed by the public prosecutor. Although Defendant Wang played a primary role in the case, he voluntarily surrendered and truthfully confessed, and the accomplices have compensated the victims; therefore, a lighter punishment is imposed.*

Judgment:
I. Defendant Wang is guilty of fraud, sentenced to six months of imprisonment, and fined 5,000 RMB.
II. Defendant Wang is ordered to refund the illegal gains of 7,280 RMB, which shall be recovered.

[...Standard closing formats omitted...]

Primary Path

Direct Response from Expert LoRA

[...Standard court headers, defendant info omitted...]
Court's Opinion:
The court holds that Defendant Wang's actions constitute fraud. [...] He played a primary role in the joint crime and is considered an accessory, which legally warrants a lighter punishment. Since the accomplices' families have compensated the victims, a further lighter punishment is applicable.

Judgment:
I. Defendant Wang is guilty of fraud, sentenced to six months of imprisonment, and fined 2,000 RMB.
II. Defendant Wang is ordered to refund the victim 10,400 RMB.
[...Appeals information omitted...]

Appendix: Relevant Legal Articles
Article 266 of the Criminal Law of the PRC: Whoever defrauds public or private property...
[...Articles 67, 52, 53, 64 omitted...]

Figure 7: Case of Inference Workflow. This item is from JuDGE dataset, which belongs to the Short-Input-Long-Output subset of MemoryBench.

Table 5: Pre-filtering performance of the cognitive gap assessment for Qwen3-8B. The first row shows the proportion of reflective experience clusters, including those tagged during both the pre- and post-filtering stages. The second row quantifies the reduction in Primary Experience Module training attributable to pre-filtering. The final row reports the proportion of clusters identified early by pre-filtering; higher values indicate greater accuracy of pre-filtering in predicting failures of Primary Experience Module Construction.

	Short-Long	**Short-Short**	**Long-Long**	**Long-Short**
#Reflective Clus. / #Clus.	2 / 3	1 / 2	1 / 4	2 / 3
DPO Cost Savings	$\sim 53\%$	$\sim 57\%$	0	$\sim 78\%$
Pre-Filter Recall	1 / 2	1 / 1	0 / 1	2 / 2

Scalable Multi-LoRA Serving To address the memory and routing overhead of matching incoming queries to up to $1,000$ potential LoRAs at inference time, naive sequential loading is inadequate. It is worth noting that the routing step itself is fundamentally a vector similarity matching process; since the total number of clusters is orders of magnitude smaller than the document corpora handled by standard text retrieval systems, identifying the correct cluster introduces virtually zero computational overhead. For LoRA loading, the subsequent memory and operational overhead of loading these weights can be effectively mitigated by leveraging the advanced multi-LoRA serving capabilities natively supported by modern inference frameworks like vLLM. Current vLLM implementations incorporate highly optimized multi-LoRA serving features by utilizing unified memory management and custom CUDA kernels to dynamically page active LoRA weights from CPU host memory to GPU VRAM. By fully exploiting the multi-LoRA infrastructure of modern inference frameworks, we hypothesize that UNO's multi-module architecture can efficiently route queries to the correct cluster and transition the corresponding experience module on the fly with negligible latency overhead, ensuring the feasibility and cost-effectiveness of UNO in production environments. In the small-scale experiment presented in Figure 2, we also observe that this overhead is negligible.

Folding the Reflective Path As noted above, the Reflective Path in UNO relies on a multi-stage inference process (initial generation, critique, and refinement). While we provide the single-stage *UNO-Single* variant for strictly latency-constrained applications or industrial scenarios, an exciting avenue for future work is to fold the Reflective Path into a single-stage generation process via on-policy self-distillation (Zhao et al., 2026). Specifically, the high-quality critique-and-refine trajectories generated offline could be used as distillation data to train a dedicated Expert LoRA. This approach may completely eliminate the multi-stage inference overhead at test time while preserving the robust performance and noise-resistance of the reflective experience.

Discussion on Noisy and Malicious Feedback UNO incorporates multiple layers of defense against low-quality user feedback. The cognitive gap pre-filter identifies clusters whose log content deviates substantially from the model's knowledge, routing them away from direct fine-tuning. The simulated performance verifier then rejects any adapter that fails to improve over the base model. Together, these mechanisms prevent harmful data from degrading the deployed model, as demonstrated by the noise injection experiments in Appendix F (Table 11), where UNO maintains stable performance under up to 10% irrelevant noise. For production deployments with fully adversarial user populations, a lightweight safety detection model such as Qwen3Guard (Zhao et al., 2025) can preprocess the raw user logs before they enter the UNO pipeline, filtering out overtly harmful or toxic content at the earliest stage.

F Further Ablation Study

F.1 Cognitive Gap Assessment as Pre-Filtering

Cognitive gap assessment and simulated performance verification serve as pre-filtering and post-filtering mechanisms for noisy user logs, respectively. Our analysis shows that even without pre-filtering, the verifier can also reliably filter out clusters with failed optimization, meaning cognitive gap assessment does not directly affect final model performance. Nevertheless, it significantly reduces the computational cost of Primary Experience Construction and simulated performance verification

Table 6: Ablation study for simulated performance verifier. Perplexity and the DPO Chosen Reward are computed on the positive examples of the validation set.

Verification Criteria	Short-Long		Short-Short	
	Norm-S	Z-S	Norm-S	Z-S
Win-Rate in Simulated Verifier	**77.09**	**7.16**	**76.26**	**21.54**
Perplexity	71.13	-29.49	72.97	-6.49
DPO Chosen Reward	76.30	2.07	72.97	-6.49
Last Epoch	69.15	-43.87	71.76	-17.03

by accurately estimating log quality at the cluster level before training and guiding optimization decisions. Table 5 reports the number of clusters filtered at each stage, the total number of clusters produced, the reduction in DPO training volume due to cognitive gap assessment (measured by reduced training data), and the recall of pre-filtering (i.e., the proportion of clusters that should be filtered and are identified before training). These results demonstrate that cognitive gap assessment effectively evaluates log quality in advance and substantially reduces resource consumption.

On the Long-Long task, the pre-filter recall is 0% because all clusters have cognitive gap scores below τ^*, so none are flagged during pre-filtering. The single cluster whose Expert training fails is caught by the simulated verifier during post-filtering, confirming the complementary roles of the two-stage design.

F.2 Discussion on Cognitive Gap

F.2.1 Cognitive Gap vs. Model Confidence

Estimating data difficulty via model confidence (e.g., Confident Learning (Northcutt et al., 2022)) is well-established in noisy label learning. However, directly applying token-level confidence (such as perplexity) to open-ended LLM generation often encounters calibration issues, as probabilities are confounded by generation formats and task openness rather than just data difficulty. Cognitive Gap addresses this by shifting the estimation from the probability space to the semantic reasoning space. Measuring the semantic distance between model-generated rules and those distilled from user logs is significantly more robust and computationally efficient, as it utilizes a smaller proxy model.

Figure 8 compares the classification performance for identifying Primary and Reflective Paths using Cognitive Gap versus negative log-likelihood (Qwen3-8B on MemoryBench). Since this task acts as a pre-filter, achieving 100% recall for Primary Path clusters is crucial. At this decision boundary, model confidence misclassifies substantially more Reflective Path clusters into Primary Experience Module training, wasting computational resources. Cognitive Gap, however, yields significantly fewer errors. Furthermore, standard model confidence achieves an Area Under the Receiver Operating Characteristic curve (AUROC) of 0.8611, lower than the 0.8889 achieved by Cognitive Gap. This confirms that semantic-level metrics more reliably separate valuable feedback from noise in log-driven LLM optimization.

F.2.2 Cognitive Gap as a Predictor of Training Outcome

We provide empirical evidence that cognitive gap predicts whether training an expert adapter on a given cluster improves or degrades downstream performance, using data from all 12 clusters in the main experiments.

High-gap clusters are more likely to fail training. Table 7 shows a Fisher exact test on whether the expert adapter's validation win rate falls below 0.5 (i.e., loses to the base model). Among all the low-gap clusters, none fail; among all the high-gap clusters, 75% fail ($p = 0.018$, one-sided).

Rank correlations. Table 8 reports Spearman correlations across all 12 clusters. Validation win rate strongly predicts downstream gain ($\rho = +0.713$, $p = 0.009$), and the 5 clusters gated by the 0.53 threshold all show negative downstream gain (mean $\Delta = -0.037$), confirming that the verifier threshold effectively prevents harmful deployments. Cognitive gap is negatively correlated with

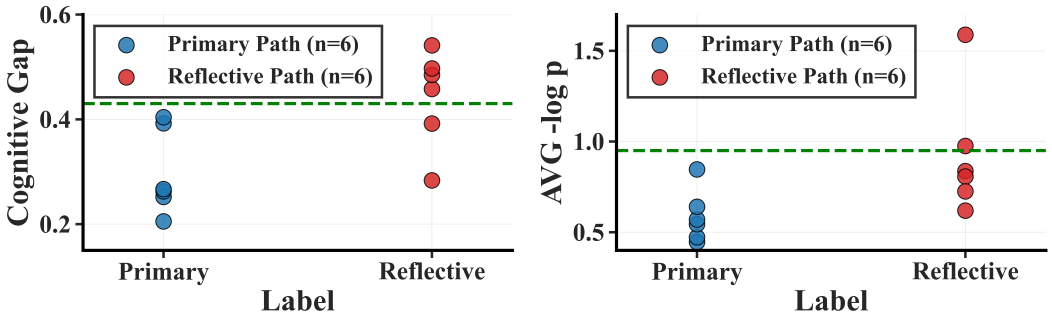

Figure 8: Comparison of the pre-filtering capabilities of Cognitive Gap and model confidence (mean negative log-probability). Scatter points represent individual clusters, and the dashed line indicates the decision boundary for completely filtering out clusters belonging to the Primary Path.

Table 7: Fisher exact test: cognitive gap vs. training failure (win rate < 0.5).

Group	**Fails**	**Succ.**	p
Low gap (n=8)	0	8	**.018**
High gap (n=4)	3	1	

Table 8: Rank correlations across 12 clusters (Qwen3-8B).

Variable Pair	**Spearman ρ**	**p-value**
Gap vs. downstream gain	-0.584	**0.046**
Gap vs. validation win rate	-0.483	0.112
Win rate vs. downstream gain	$+0.713$	**0.009**

downstream gain ($\rho = -0.584$, $p = 0.046$), supporting the two-stage design: gap-based pre-filtering reduces training cost, while verifier-based post-filtering ensures downstream safety.

F.3 Simulated Verifier as Post-Filtering

As a post-filter, the simulated performance verifier ensures quality control before deployment. We compare it with several naive validation strategies, including perplexity on positive validation samples, DPO chosen reward ($\beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$), and selecting the final training epoch. As shown in Table 6, verifier-based evaluation is more effective at identifying superior checkpoints or filtering failed clusters. Implicit metrics derived from positive samples perform poorly, likely because the verifier-based evaluation relies on generation-and-scoring, which is more robust and better aligned with generative performance.

F.4 Ablation Studies on Key Hyperparameters

Figure 9 presents a sensitivity analysis of the cognitive gap threshold, τ^*. In the main experiments, we set τ^* to the median of the training-set gap scores computed over all data from the two models, namely 0.45. With other hyperparameters fixed, we evaluate different threshold values on the MemoryBench dataset using Qwen3-8B. The results indicate that this hyperparameter is generally robust. Only on a few datasets (e.g., Long-Long) does an excessively small threshold slightly degrade performance. This robustness stems from the post-filtering mechanism: as long as the threshold is chosen conservatively (i.e., relatively large), performance remains unaffected, although the potential savings in training resources decrease. When the threshold is set too small, clusters that should be assigned to the primary path may instead be routed to the reflective path. Such misclassification has minimal impact on performance but substantially increases inference overhead.

Figure 10 shows the impact of varying the win rate threshold in the Simulated Verifier. This hyperparameter likewise exhibits strong robustness. When the threshold is set exactly to 0.5, performance drops on certain datasets, such as Short-Long and Long-Long. This decline likely arises from the model's inherent judgment capability: for samples with only marginal differences, decisions are more sensitive to noise and random perturbations, causing clusters formed at a 0.5 threshold to potentially reflect ineffective training. Conversely, setting the threshold too high may also misroute clusters that

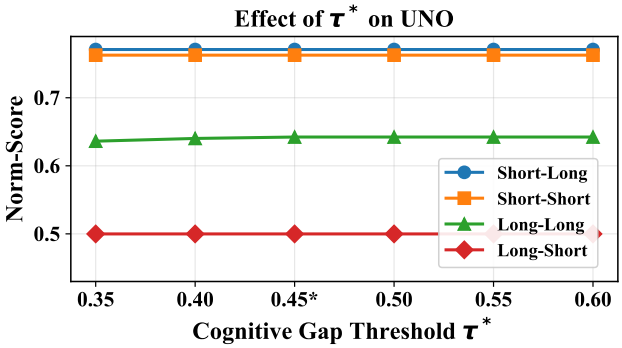

Figure 9: Ablation Study of Cognitive Gap Threshold τ^* on Qwen3-8B.

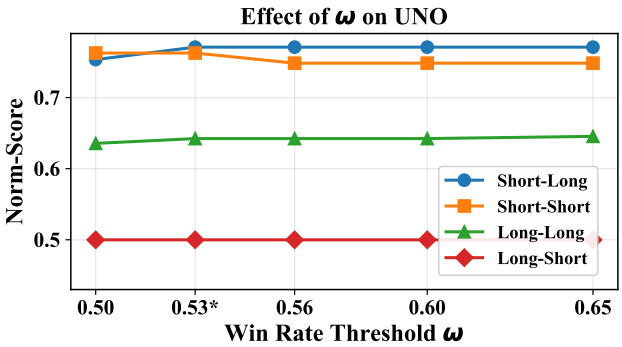

Figure 10: Ablation Study of Win Rate Threshold w on Qwen3-8B.

should belong to the primary path into the reflective path, thereby significantly increasing inference overhead.

F.5 Clustering Distance Threshold

We sweep the distance threshold that controls cluster granularity in agglomerative clustering (Figure 11). Performance follows an inverted-U pattern on both tasks: overly fine-grained clustering (threshold $= 2$) degrades performance since each cluster contains too few training samples for reliable LoRA training, while removing clustering entirely (threshold$\geq$10) causes a large drop, particularly on Short-Long where UNO falls by 6.1 Norm-Score points. The default threshold of 4 lies in the stable plateau region. UNO-Single shows greater sensitivity to over-segmentation, since it lacks the Reflective Path to absorb training failures on small clusters.

F.6 Per-query Rule Count

We force each query to have exactly N rules and compare against the default variable-count extraction (i.e., the LLM decide how many rules to extract by itself) in Figure 12. UNO is moderately sensitive to N: on Short-Short the peak is at N=5 (77.08), while on Short-Long N=7 performs best (75.73). UNO-Single is largely insensitive to N (range 74.0–74.3 on Short-Short). On Short-Long, the default variable-N strategy (77.09) outperforms all fixed-N configurations. On Short-Short, the default (76.26) is slightly below N=5 (77.08), a difference within seed variance (std $= 0.64$, Appendix F.12). Since the optimal fixed N differs across tasks, the variable-N strategy provides a robust default without per-task tuning.

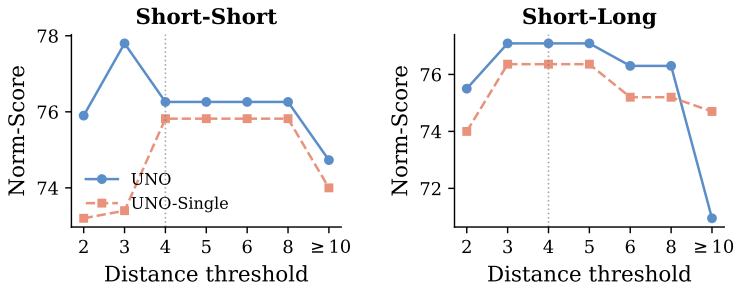

Figure 11: Performance (Norm-Score) vs. clustering distance threshold on Qwen3-8B. The dotted line marks the default threshold (4).

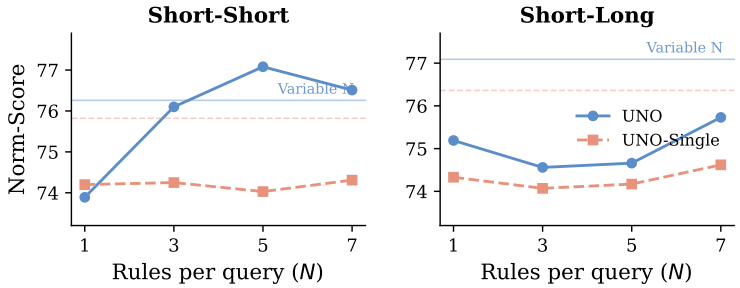

Figure 12: Performance (Norm-Score) vs. forced per-query rule count N on Qwen3-8B. Horizontal lines indicate the default variable-N performance.

F.7 Training Objective: DPO+NLL vs. SDPO

Table 9 examines the training objective for UNO's Expert LoRA. The upper section sweeps the balance between DPO and NLL losses with their combined weight fixed at 1. Neither pure SFT (ratio=0) nor pure DPO (ratio=1) is optimal. Pure SFT tends to overfit noisy logs, whereas pure DPO can induce distribution shift without sufficient alignment guidance. The mixed objective at ratio=0.5 achieves the best overall performance. The lower section compares DPO+NLL with SDPO. UNO-Single trained with DPO+NLL consistently outperforms the SDPO variant. A plausible explanation is that the DPO+NLL term provides direct supervision on high-quality data modified directly based on user feedback, whereas SDPO's self-distillation signal is noisier.

F.8 No-Feedback Self-Distillation

To verify that UNO's gains stem from real user feedback rather than self-distillation, we run a No-Feedback Self-Distillation (NFSD) ablation: the identical UNO-Single pipeline is executed, but user-feedback-derived rules are replaced with rules the base model generates autonomously (i.e., self-generated improvement suggestions without access to any user log). Table 10 shows that NFSD yields negligible improvement over the base model on Short-Long and zero improvement on Short-Short (output collapses to the base model's response). The substantial gap between UNO-Single and NFSD confirms that real user feedback is the essential driver of UNO's gains.

F.9 Failure Mode Analysis

We analyze two failure modes of learning from user logs and how UNO addresses them.

Failure Mode 1: Noisy user logs produce poor rules. UNO provides multiple protection layers: cognitive gap pre-filtering routes high-risk clusters away from direct optimization (Table 5), and the simulated verifier gates adapter deployment by rejecting those failing to surpass the base model (Table 6: without it, Z-Score drops from 7.16 to -43.87). Furthermore, UNO achieves the highest win rates on WildFB (Figure 3), which contains real noisy logs, empirically confirming noise robustness.

Table 9: Training objective ablation for Expert LoRA (Qwen3-8B, UNO-Single, Norm-Score). Upper: DPO loss ratio sweep. Lower: comparison with SDPO.

Configuration	Short-Long	Short-Short
DPO ratio = 0 (pure SFT)	74.24	72.35
DPO ratio = 0.25	73.86	75.93
DPO ratio = 0.5 (default)	**76.36**	**75.82**
DPO ratio = 0.75	74.84	75.83
DPO ratio = 1.0 (pure DPO)	73.45	72.35
SDPO	74.05	74.02

Table 10: No-Feedback Self-Distillation ablation (Qwen3-8B).

Method	Short-Long		Short-Short	
	Norm-S	Z-S	Norm-S	Z-S
Base	74.43	-9.52	72.35	-2.31
NFSD (no real feedback)	74.98	-6.40	72.35	-2.31
UNO-Single	**76.36**	**2.99**	**75.82**	**15.89**

Failure Mode 2: Valid feedback but incorrect rule extraction. We manually sampled 50 instances from Short-Short and verified rule fidelity. The infidelity rate is 0%: all extracted rules accurately capture user intent. Rule extraction from explicit corrections is straightforward for capable LLMs when feedback is informative.

Noise Injection Experiment. To quantify sensitivity to rule quality, we randomly replace extracted rules with irrelevant ones from other instances at varying ratios (Table 11). At 10% noise, performance is virtually unaffected (Norm-Score drops by only 0.15). At 25–40%, performance degrades substantially, but such noise rates are unrealistic given the 0% infidelity confirmed above.

F.10 Verifier Self-Validation Reliability

UNO uses the base Qwen3-8B as the simulated verifier. This self-validation design avoids introducing a stronger external model, which would make comparison unfair and obscure whether the base model itself can reliably self-improve from user logs. To validate that the resulting scoring bias is acceptable, we replace the verifier with GPT-5.2 (OpenAI, 2025) and compare scoring consistency, checkpoint selection, and downstream performance.

Scoring consistency. Table 12 shows that Qwen3-8B scores are systematically higher (mean $\approx$8) than GPT-5.2 (mean $\approx$5), but the relative ranking is moderately correlated (Spearman $\rho \approx 0.58$–0.59).

Checkpoint selection. Despite different absolute scores, the coarse-grained decision of whether to use the expert adapter or fall back to the base model agrees in 60% of the clusters. In the two disagreeing clusters, both verifiers select an expert adapter but at different epochs. This stability arises because the judge is grounded in extracted user-feedback rules and actual generation quality, which constrains scoring variance across verifiers.

Downstream performance. Switching to GPT-5.2 produces nearly similar downstream performance (Table 13). The complete UNO framework further absorbs verifier discrepancies through its Reflective Path fallback.

F.11 Ablation on Clustering Features

To isolate the contribution of each feature component in UNO's dual-feature clustering, we replace the default query$\oplus$rule concatenation with three alternatives: query-only, rule-only, and random partition. For all variants, the remaining pipeline (encoding model, Ward linkage, distance threshold, per-cluster training, and routing) is kept identical. The random partition preserves the same cluster count and size distribution as the default.

The default dual-feature clustering achieves the highest performance on both tasks. Random partition degrades substantially, falling to a level comparable to the no-clustering baseline (Table 2), confirming

Table 11: Noise injection: replacing rules with irrelevant ones (Qwen3-8B, Short-Long).

Noise Ratio	Norm-S	Z-S
0% (original)	77.09	7.16
10%	76.94	6.42
25%	72.62	-20.07
40%	70.34	-34.22

Table 12: Per-answer scoring consistency between Qwen3-8B and GPT-5.2 verifiers.

Split	n	Qwen3 M/Med	GPT-5.2 M/Med	r	ρ	± 1
SS	535	7.96 / 9	5.37 / 6	.616	.594	27.9%
SL	664	8.26 / 9	5.14 / 6	.676	.578	10.4%

Table 13: Downstream Norm-Score with different verifiers.

Split	System	Qwen3	GPT-5.2
SS	UNO	76.26	76.36
	UNO-Single	75.82	76.32
SL	UNO	77.09	77.20
	UNO-Single	76.36	75.25

that the clustering structure itself is essential. Both query-only and rule-only partitions perform reasonably, indicating that each feature carries useful information; combining them yields the best results.

F.12 Seed Robustness Analysis

To verify the statistical reliability of UNO's gains, we run UNO and three representative baselines (Base, RAG-BM25, DPO w/ UNO Data) with three random seeds on Qwen3-8B for the Short-Short and Short-Long tasks. Table 15 reports the mean, standard deviation, and 95% confidence interval of Norm-Score across seeds.

G Detailed evaluation results on MemoryBench

MemoryBench evaluates the overall performance of the LLMsys after optimization on the complete system log and therefore adopts aggregated relative metrics. Our main experiments follow this setting. In addition, we report fine-grained metrics for each sub-dataset, as presented in Table 16 and Table 17.

H Licensing

Qwen3-8B, Qwen3-Embedding-0.6B, and Qwen3-Reranker-0.6B are released under the Apache License 2.0. Phi-4 is released under the MIT license. Both evaluation datasets, MemoryBench and WildFB, are released under the MIT license. This paper's research objective is academic exploration, which aligns with the terms of this license.

WildFB involves real-world user dialogues. As stated in the original study, these data originate from WildChat and have undergone thorough anonymization. Furthermore, our filtering pipeline does not introduce any additional sensitive information. Since MemoryBench utilizes a user simulator, it does not involve any private content.

Table 14: Norm-Score under different clustering features (Qwen3-8B).

Feature	Short-Short	Short-Long
Query $\oplus$ Rule (default)	**76.26**	**77.09**
Query-only	76.03	76.65
Rule-only	75.93	76.92
Random	74.72	71.04

Table 15: Mean Norm-Score, standard deviation, and 95% confidence interval over three random seeds (Qwen3-8B).

System	Short-Short			Short-Long		
	Mean	Std	95% CI	Mean	Std	95% CI
Base	72.28	0.34	71.44–73.12	74.51	0.49	73.29–75.73
RAG-BM25	72.77	0.42	71.73–73.81	74.49	0.44	73.40–75.58
DPO w/ UNO Data	72.78	0.58	71.34–74.22	70.51	0.72	68.72–72.30
UNO	**76.35**	0.64	74.76–77.94	**76.99**	0.67	75.33–78.65

Table 16: Detailed evaluation results of LLMsys with Qwen3-8B backbone on MemoryBench. For each dataset, we report raw performances using their dataset-specific evaluation metrics. Except for metrics marked with↓, higher values indicate better performance.

Type	Dataset	Metric	Base	Embedding	BM25	MemOS	ReMem	A-Mem	Mem0	MemoryOS	SDPO	DPO	SFT	UNO-Single	UNO
Short-Short	SciTechNews	BERTScore-F1	0.8179	0.8154	0.8151	0.8192	0.8165	0.8171	0.8192	0.8127	0.8239	0.8242	0.8256	0.8179	0.8170
		CLI↓	16.4672	16.0973	15.9736	16.3992	16.3043	16.9455	16.4013	15.6473	16.2969	16.5301	17.2997	16.4672	16.1465
		DCRS↓	12.6027	12.5169	12.4130	12.6677	12.5773	12.7851	12.6697	12.3047	12.4812	12.7092	13.1793	12.6027	12.4879
		FKGL↓	16.0443	15.8845	15.8840	16.2397	16.0830	16.4449	16.0793	15.2932	15.7381	16.2405	17.1901	16.0443	15.5792
		Rouge-L	0.1212	0.1230	0.1215	0.1225	0.1163	0.1205	0.1242	0.1072	0.1331	0.1361	0.1418	0.1212	0.1192
	LexEval-QA	Rouge-L	0.1111	0.1128	0.1155	0.0840	0.1112	0.1312	0.1050	0.1206	0.0497	0.1350	0.1649	0.1451	0.1451
	NFCats	Score	4.5600	4.5600	4.5000	4.1400	4.3600	4.1000	3.2600	4.2000	4.8600	4.5200	4.4200	4.6000	4.6000
Short-Long	HelloBench -Academic &Knowledge -QA	Avg. Score	0.8332	0.8483	0.8488	0.8290	0.8272	0.8458	0.8610	0.8375	0.7958	0.8625	0.8693	0.8768	0.8768
	JuDGE	Amount Score	0.5650	0.5481	0.5993	0.5870	0.5059	0.5520	0.2975	0.5554	0.5286	0.4302	0.4992	0.5650	0.5585
		Crime Precision	0.9700	0.9700	0.9500	0.9600	0.9600	0.9100	0.9100	0.9600	0.9500	0.9200	0.9400	0.9700	0.9600
		Crime Recall	1.0000	1.0000	0.9600	0.9800	0.9800	0.9200	0.9200	0.9800	0.9800	0.9400	0.9600	1.0000	0.9800
		Judge Bert Score	0.7901	0.7720	0.7683	0.7806	0.7588	0.7646	0.4544	0.7486	0.7429	0.8671	0.8079	0.7901	0.7907
		Judge Meteor	0.4009	0.3807	0.3755	0.4003	0.3663	0.4084	0.2900	0.3257	0.3779	0.5987	0.4617	0.4009	0.4052
		Penalcode Index Precision	0.7294	0.7200	0.7065	0.7189	0.7010	0.7592	0.6011	0.6721	0.6636	0.7970	0.6572	0.7294	0.7413
		Penalcode Index Recall	0.7476	0.7385	0.7989	0.7399	0.7316	0.7748	0.3382	0.7390	0.7234	0.6215	0.6581	0.7476	0.7516
		Reasoning Bert Score	0.8140	0.8147	0.8160	0.7723	0.8147	0.7844	0.7049	0.8138	0.7932	0.8154	0.8069	0.8140	0.8081
		Reasoning Meteor	0.5101	0.4861	0.5133	0.4580	0.5007	0.4974	0.3621	0.4895	0.5006	0.4940	0.4946	0.5101	0.5186
		Time Score	0.7224	0.7237	0.7108	0.7057	0.6834	0.6910	0.5071	0.7139	0.7058	0.7049	0.6627	0.7224	0.7168
	WritingPrompts	Meteor	0.2367	0.2245	0.2342	0.2238	0.2251	0.2056	0.2209	0.2425	0.2170	0.2008	0.1998	0.2367	0.2451
Long-Long	HelloBench -Academic &Knowledge -Writing	Avg. Score	0.8515	0.8319	0.8250	0.8333	0.8353	0.8240	0.8206	0.7951	0.6510	0.8377	0.8319	0.8211	0.8294
	HelloBench -Creative&Design	Avg. Score	0.8291	0.7624	0.7888	0.7142	0.7633	0.7895	0.7791	0.7578	0.7066	0.7711	0.7894	0.8246	0.8250
	LexEval-Judge	Rouge-L	0.0753	0.0895	0.0628	0.0238	0.0695	0.0676	0.0235	0.0082	0.0086	0.0700	0.0843	0.0901	0.0901
	WritingBench -Academic &Engineering	Score	6.9706	6.7353	7.0882	6.5294	6.0588	6.8235	6.6471	6.0294	6.3235	6.7647	6.7647	7.1176	7.1176
	WritingBench -Creative&Design	Score	6.6744	6.6512	6.5698	6.6977	6.4535	6.4651	6.4884	4.8953	5.8721	6.6744	6.6512	6.6512	6.7326
	WritingBench -Politics&Law	Score	6.8780	6.8537	6.8537	6.6341	6.6585	6.6829	6.6829	4.6829	6.5854	6.8780	6.8049	6.9756	6.9756
Long-Short	IdeaBench	Bert Score	0.5605	0.5676	0.5675	0.5418	0.5632	0.5576	-	0.5553	0.5633	0.5681	0.5672	0.5605	0.5655
		LLM Feasibility Ranking Score	0.1267	0.1867	0.1733	0.0133	0.1267	0.1400	-	0.0867	0.0667	0.1000	0.0867	0.1267	0.1533
		LLM Novelty Ranking Score	0.7600	0.7667	0.7400	0.5067	0.7133	0.6400	-	0.4533	0.6333	0.7067	0.6600	0.7600	0.7400
		LLM Rating Score	4.8800	4.8800	4.7000	3.0800	4.8800	4.6000	-	4.8800	4.5600	4.2400	4.7000	4.8800	4.7600
	LexEval -Summarization	Rouge-L	0.2237	0.2306	0.2256	0.2191	0.2205	0.2351	-	0.2400	0.2229	0.2191	0.2295	0.2218	0.2218
	LimitGen-Syn	Accuracy	0.4400	0.4600	0.5000	0.6000	0.3800	0.5000	-	0.1400	0.6200	0.5000	0.5600	0.4400	0.5600
		Rating	1.1400	1.2200	1.2800	1.8000	0.9400	1.3200	-	0.3400	1.7800	1.3200	1.4800	1.1400	1.5400

Table 17: Detailed evaluation results of LLMsys with phi-4 backbone on MemoryBench. For each dataset, we report raw performances using their dataset-specific evaluation metrics. Except for metrics marked with↓, higher values indicate better performance.

Type	Dataset	Metric	Base	Embedding	BM25	MemOS	ReMem	A-Mem	Mem0	MemoryOS	SDPO	DPO	SFT	UNO-Single	UNO
Short-Short	SciTechNews	BERTScore-F1	0.8280	0.8261	0.8268	0.8239	0.8257	0.8244	0.8248	0.8081	0.8226	0.8271	0.8304	0.8280	0.8220
		CLI↓	18.2423	18.2172	18.4444	17.2607	18.5251	17.9994	17.5355	14.7873	15.8190	17.5655	18.4226	18.2423	17.6598
		DCRS↓	13.4198	13.3969	13.4585	13.0143	13.4976	13.2664	13.0972	14.4874	12.3708	13.1547	13.5274	13.4198	13.1462
		FKGL↓	17.3397	17.4866	17.7378	16.7457	17.8619	17.0598	16.5825	14.3187	15.1072	16.6808	17.4200	17.3397	16.6477
		Rouge-L	0.1419	0.1377	0.1411	0.1307	0.1378	0.1350	0.1301	0.0715	0.0787	0.1412	0.1475	0.1419	0.1285
	LexEval-QA	Rouge-L	0.1471	0.1494	0.1468	0.1466	0.0375	0.1508	0.1434	0.1398	0.0470	0.1442	0.1450	0.1533	0.1533
	NFCats	Score	4.4600	4.5400	4.3800	3.8800	4.0600	4.2200	3.7600	3.8200	4.8400	4.3600	4.3400	4.4800	4.5000
Short-Long	HelloBench -Academic &Knowledge -QA	Avg. Score	0.8479	0.8354	0.8508	0.8616	0.8155	0.8475	0.8743	0.6752	0.8336	0.8524	0.8679	0.8700	0.8700
	JuDGE	Amount Score	0.2725	0.2823	0.3266	0.3192	0.3198	0.3069	0.3256	0.0000	0.2829	0.2765	0.3187	0.3095	0.3095
		Crime Precision	0.8500	0.8500	0.8400	0.8800	0.8500	0.8800	0.9000	0.0000	0.8900	0.8900	0.8800	0.8300	0.8300
		Crime Recall	0.8600	0.8500	0.8500	0.9000	0.8600	0.9000	0.9200	0.0000	0.9000	0.9000	0.8900	0.8600	0.8600
		Judge Bert Score	0.4613	0.4909	0.5412	0.4561	0.5276	0.5823	0.5357	0.0000	0.5284	0.6531	0.4135	0.5941	0.5941
		Judge Meteor	0.2767	0.2529	0.2748	0.3023	0.2633	0.2626	0.2819	0.0000	0.2927	0.3128	0.2836	0.2994	0.2994
		Penalcode Index Precision	0.6094	0.6297	0.6617	0.5656	0.5883	0.6373	0.5579	0.0000	0.6513	0.5940	0.4660	0.6160	0.6160
		Penalcode Index Recall	0.3502	0.3355	0.4085	0.3742	0.3025	0.3140	0.4000	0.0000	0.3518	0.3678	0.3894	0.3752	0.3752
		Reasoning Bert Score	0.6964	0.6930	0.7142	0.7110	0.6901	0.7289	0.7180	0.4035	0.7180	0.7324	0.6979	0.7269	0.7269
		Reasoning Meteor	0.3206	0.3493	0.3735	0.3552	0.3422	0.3532	0.3576	0.0000	0.3400	0.3246	0.3840	0.3577	0.3577
		Time Score	0.4384	0.5031	0.5436	0.5142	0.5296	0.5209	0.4900	0.0000	0.4691	0.5177	0.4729	0.5333	0.5333
	WritingPrompts	Meteor	0.2254	0.2150	0.2267	0.2188	0.2192	0.2159	0.2200	0.0535	0.2246	0.2157	0.2111	0.2254	0.2342
Long-Long	HelloBench -Academic &Knowledge -Writing	Avg. Score	0.8186	0.8289	0.7760	0.8642	0.8000	0.8431	0.7799	0.3147	0.8294	0.8196	0.8314	0.8191	0.8260
	HelloBench -Creative&Design	Avg. Score	0.8027	0.8016	0.8289	0.7767	0.7709	0.8030	0.7941	0.4096	0.7938	0.7576	0.8129	0.7956	0.8278
	LexEval-Judge	Rouge-L	0.1629	0.1601	0.1582	0.1402	0.0834	0.1391	0.1396	0.0725	0.0142	0.1474	0.1855	0.1629	0.1626
	WritingBench -Academic &Engineering	Score	6.0588	5.5294	5.4412	5.5294	5.0000	5.0294	5.2353	3.6765	5.3529	5.3824	5.2647	6.1471	6.1471
	WritingBench -Creative&Design	Score	5.5581	5.4302	5.3140	5.2209	5.0581	5.3372	5.2209	3.6163	5.1512	5.2326	5.3488	5.6163	5.6744
	WritingBench -Politics&Law	Score	5.1463	4.4146	4.5854	4.8293	4.2927	4.7073	4.8780	3.6585	4.9750	4.7073	4.7561	5.0488	4.9512
Long-Short	IdeaBench	Bert Score	0.5640	0.5628	0.5654	0.5476	0.5621	0.5496	-	0.5314	0.5510	0.5620	0.5685	0.5640	0.5660
		LLM Feasibility Ranking Score	0.1200	0.1400	0.0800	0.0000	0.1400	0.0733	-	0.0000	0.0000	0.0267	0.0667	0.1200	0.0967
		LLM Novelty Ranking Score	0.6667	0.6600	0.5733	0.0467	0.6400	0.1133	-	0.0067	0.0733	0.2067	0.3467	0.6667	0.6800
		LLM Rating Score	4.5200	4.3400	4.6200	4.3600	4.7200	4.4200	-	4.4000	4.5000	4.5600	4.7200	4.5200	4.5400
	LexEval -Summarization	Rouge-L	0.2212	0.2139	0.2124	0.2277	0.1304	0.2211	-	0.1612	0.0841	0.2164	0.2185	0.2307	0.2307
	LimitGen-Syn	Accuracy	0.5000	0.3400	0.4200	0.5000	0.4200	0.3800	-	0.0000	0.5200	0.0600	0.2600	0.5000	0.6600
		Rating	1.3000	0.8400	1.1000	1.3400	1.1000	0.9600	-	0.0000	1.5800	0.1400	0.7000	1.3000	1.9000