# EVOLVING AGENT EXPERTISE FROM CONVERSATIONAL FEEDBACK

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

## ABSTRACT

Deployed Large Language Model (LLM) agents accumulate extensive interaction logs in which user feedback implicitly signals where and how the model should improve. Learning from such feedback is difficult because the signal is both weak and mixed. It is weak because it arrives as free-form text with no success oracle, and because each remark covers only a fragment of what a task requires, so a reusable lesson emerges only after many remarks are aggregated. It is mixed because a single remark may point to a flawed procedure, a missing fact, or a skill applied where it does not belong. Existing methods each maintain a single kind of store and therefore absorb only part of this signal. Memory systems retain facts but never distill them into procedures, while skill-evolution methods distill procedures but discard the facts and never learn when a skill should not be applied. To address this, we present SkillWeave, which learns from feedback logs alone, without a success oracle or parameter updates, by jointly evolving a skill library, an evidence-grounded knowledge base, and the applicability conditions of each skill. Logs are processed in batches. Each batch first extracts missing facts, then groups logs into task topics, and finally creates or revises one skill per topic together with the conditions that govern its use. On user-feedback benchmarks covering legal, academic, and open-domain tasks, SkillWeave outperforms eight memory and skill-evolution baselines across three backbone LLMs, and it stays competitive on long-document tasks. [1]

## 1 INTRODUCTION

Large Language Model (LLM) agents deployed in production accumulate extensive interaction logs containing user queries, model responses, and follow-up feedback. These logs represent a rich source of learning signals: the feedback implicitly encodes where the model falls short and how its behavior should change. Since the backbone model's parameters cannot always be updated in practice, a growing line of work explores how to distill these signals into external memory structures that augment the model at inference time (Ai et al., 2025; Xu et al., 2025a; Liu et al., 2026).

However, conversational feedback is a difficult signal to learn from. Its central difficulty is that the signal is mixed. A single feedback turn may point to a flawed procedure, a missing fact, or a previously learned skill applied where it does not belong, and these lessons must be written to different kinds of accumulated state. The signal is also sparse. Users rarely comment on every aspect of a response, so any one remark covers only a fragment of what a task requires, and a reusable lesson emerges only when remarks from many interactions are aggregated. Finally, there is no success oracle. Users react in natural language, and no reference answer or automated judge grades the model's output.

Existing approaches each maintain a single kind of store and therefore absorb only one part of this mixed signal. Memory systems (Fang et al., 2026; Liu et al., 2026; Xu et al., 2025a) organize past interactions into retrievable records, enabling the model to recall relevant episodes or facts, but they never distill those records into procedures. Skill-evolution methods (Yang et al., 2026b; Zhang et al., 2026; Yang et al., 2026a) distill interaction patterns into reusable procedural documents that guide how the model should handle recurring task types, but they discard the facts that feedback reveals and never learn when a skill should or should not be applied.

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



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

We propose **SkillWeave**, which maintains two evolving stores from the same log stream: a *skill library* of reusable task procedures and a *knowledge base* of evidence-grounded facts. Each skill carries its own applicability conditions, so that the lessons about when a skill does not apply are stored alongside the procedure itself. Logs are processed in batches. Each batch first extracts missing facts, then groups logs into task topics so that sparse remarks are aggregated before any skill is written, and finally creates or revises one skill per topic. At inference time, retrieved skills and knowledge augment the frozen backbone.

We evaluate on benchmarks built from user logs, across three LLMs. SkillWeave outperforms eight baselines spanning memory systems and skill-evolution methods on MemoryBench and stays competitive on long-document tasks. Our contributions are as follows:

- We propose a framework that absorbs the procedural, factual, and applicability lessons carried by mixed conversational feedback into a skill library and a knowledge base, without a success oracle or model parameter updates.
- We adopt a batch-by-batch update pipeline that aggregates sparse feedback into task topics before writing skills, reducing computation cost and supporting online deployment where logs arrive incrementally.
- We demonstrate that SkillWeave outperforms baselines on legal, academic, and open-domain tasks and remains competitive on long-document tasks.

## 2 Related Work

### 2.1 Memory Systems for LLM Agents

Research on LLM memory has progressed from simple retrieval pipelines to structured management systems (Kang et al., 2025). LightMem (Fang et al., 2026) maintains a lightweight memory store with efficient indexing for long-horizon interactions. SimpleMem (Liu et al., 2026) organizes memories as structured summaries updated incrementally from conversation logs. A-Mem (Xu et al., 2025a) introduces an agentic memory architecture that decides when and what to remember. These systems focus on retaining and retrieving factual or episodic records. SkillWeave complements this direction by accumulating procedural skills alongside factual knowledge, and by co-evolving the retrieval interface that governs how both are surfaced.

### 2.2 Skill Evolution

A growing paradigm packages procedural knowledge for LLM agents as *skills*: structured documents that specify how to handle a particular task type (Wang et al., 2026). During inference, relevant skills are selected and loaded into the LLM context to guide response generation. Several recent methods learn such skills from agent interactions. AutoSkill (Yang et al., 2026b) extracts and maintains skill documents from experiences. MemSkill (Zhang et al., 2026) learns memory-operation skills and selection policies through downstream task rewards. SkillOpt (Yang et al., 2026a) optimizes skill documents via scored execution trajectories and validation-set acceptance. MemSkill and SkillOpt are designed for settings where a task-success signal (reward or validation score) is available, which limits their applicability to open-ended conversational scenarios where no such oracle exists. These methods advance procedural knowledge extraction, but each addresses a single aspect of the accumulation problem: they do not jointly maintain a factual knowledge base or evolve the conditions under which skills should be retrieved. SkillWeave extends skill evolution by coordinating it with evidence-grounded knowledge accumulation and by embedding applicability boundaries directly into the skill representation.

### 2.3 Learning from Conversational Feedback

The use of implicit user signals for model improvement has a long history in information retrieval (Robertson & Zaragoza, 2009). For LLMs, recent work has explored learning from user interactions at scale (Zhao et al., 2024). MemoryBench (Ai et al., 2025) introduces a standardized evaluation framework for measuring how well agents leverage accumulated experience, enabling

systematic comparison of memory and skill methods under a common protocol. SkillWeave operates within this evaluation paradigm: it uses on-policy interaction logs with feedback as the sole training signal and does not require task-success labels or preference annotations.

# 3 Problem Formulation

A Large Language Model (LLM) agent interacts with users over a sequence of sessions. Each session produces an interaction log $l = (h, q, a, f)$, where $h$ is the conversation history (possibly empty for single-turn tasks), $q$ is the current user query, $a$ is the model response, and $f$ is the user's follow-up feedback. Feedback $f$ may correct factual errors, request changes in task execution, or flag inappropriate application of prior conventions. We assume no reference answer and no task-success oracle. The sole supervision signal is the natural-language feedback embedded in the logs.

Logs arrive as a stream that is processed in batches $B_1, B_2, \ldots$ of fixed size $b$. The system maintains two persistent stores that evolve across batches. The **skill library** $\mathcal{S} = \{s_1, \ldots, s_n\}$ contains reusable task procedures. Each skill $s$ comprises a textual description $d_s$, a methodology body $m_s$, a set of negative triggers $\bar{t}_s$ specifying when the skill should *not* be applied, and optional knowledge-retrieval directives $r_s$. The **knowledge base** $\mathcal{K}$ stores factual entries that the backbone model may lack. Each entry $k \in \mathcal{K}$ is a short text passage grounded in external evidence.

The backbone model $M$ is frozen throughout, whose parameters are never updated. All behavioral change comes from the evolving contents of $\mathcal{S}$ and $\mathcal{K}$, which are loaded into $M$'s context at inference time via retrieval. Processing is divided into an **accumulation phase**, during which each batch updates $\mathcal{S}$ and $\mathcal{K}$, and a **deployment phase**, during which $\mathcal{S}$ and $\mathcal{K}$ are frozen and used to augment $M$'s responses.

The goal of accumulation is to produce a skill library and knowledge base such that, when retrieved and presented to $M$ at deployment time, the resulting responses improve over the no-memory baseline across the tasks represented in the log stream. Since no task-score oracle is available during accumulation, the system must extract learning signals from the feedback text itself.

# 4 Methodology

## 4.1 Overview

SkillWeave organizes experience accumulation as a batch-wise pipeline on the log stream. Figure 1 illustrates the full architecture. For each batch $B_t$, the system executes four steps in a fixed order:

1. **Knowledge accumulation**: each log in $B_t$ is examined for factual gaps. When the model identifies a knowledge deficit and retrieved evidence supports the missing fact, the fact is rewritten into a self-contained entry and added to $\mathcal{K}_t$.
2. **Incremental clustering**: a single interaction log rarely suffices to define a reusable skill, so logs must first be grouped by task similarity. The first batch is clustered from scratch. Each log in a later batch joins its nearest existing cluster, and a new cluster is created only when no existing cluster is close enough.
3. **Topic extraction**: a cluster may still contain logs that differ in genre or intent, so each cluster that received new logs is further decomposed into one or more fine-grained task topics.
4. **Skill generation**: for each topic with enough supporting cases, the model either creates a new skill or rewrites the existing one using all cases assigned to that topic.

The four steps execute in a fixed order within each batch: knowledge accumulation first, followed by clustering, topic extraction, and skill generation. States from all four steps persist across batches: cluster assignments, topic memberships, the skill library, and the knowledge base carry forward and are incrementally updated as new batches arrive.

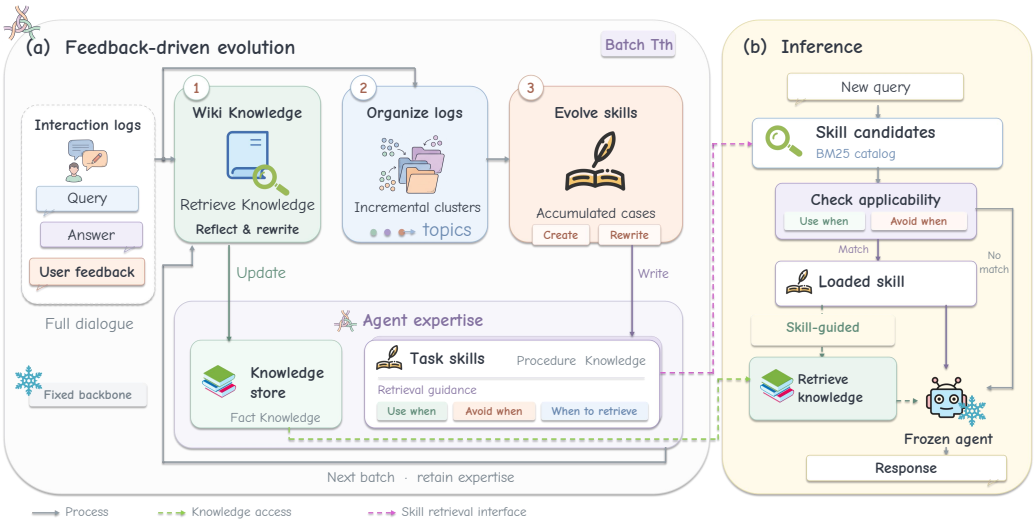



Figure 1: Overview of SkillWeave. Interaction logs are processed in batches. Each batch first accumulates knowledge entries, then clusters logs and extracts topics, and finally generates or updates skills. At inference, skills and knowledge are retrieved on demand to augment the frozen LLM.

### 4.2 Knowledge Accumulation

Conversational feedback often reveals gaps in the model's factual knowledge. A natural question is why these gaps should be addressed during accumulation rather than by searching a general corpus at inference time. The difficulty is that a general corpus is vast: retrieving from it with a short user query returns passages that may be irrelevant to the model's actual deficits or require further interpretation. By reflecting on full interaction logs during accumulation, the system can identify which facts the model specifically lacks for its task distribution and rewrite them into self-contained entries. The resulting private knowledge base $\mathcal{K}$ is compact and targeted, making inference-time retrieval both faster and more precise. To prevent unsupported claims from entering $\mathcal{K}$, every entry must be backed by evidence from a reference corpus (e.g., Wikipedia). For each log $l$ in the current batch, knowledge accumulation proceeds through three steps.

**Retrieval.** The full dialog text, including the feedback turn, is used as a query against a BM25 (Robertson & Zaragoza, 2009) index over the reference corpus. The top $k_{\text{kb}}$ passages are returned. If no passage is retrieved, the log is skipped and no model call is made.

**Reflection and judgment.** A single language-model call reads the interaction log together with the retrieved passages. The model determines whether the interaction exposes a factual gap in its own knowledge and whether the retrieved passages provide supporting evidence. If both conditions hold, the model rewrites the supported fact into a self-contained entry. The output is a structured record containing an acceptance flag, a title, and the rewritten text. Only entries with a positive acceptance flag are written to $\mathcal{K}$.

**Storage.** Accepted entries are appended to $\mathcal{K}$. Each log is processed independently within a batch. For benchmarks whose tasks require recalling prior conversation sessions (e.g., multi-turn dialogue memory), the session transcripts are written directly into $\mathcal{K}$ without any model call, since these transcripts are the source material.

### 4.3 Feedback-Conditioned Skill Evolution

The system must convert accumulated logs into reusable skills at the right granularity. A skill that is too specific overfits to individual examples, while a skill that is too broad loses the task-relevant

structure that makes it useful. SkillWeave addresses this through a two-level grouping (clusters, then topics) followed by generalization.

**Incremental clustering.** Each log $l$ is encoded into a dense vector $\mathbf{v}_l$ using a sentence embedding model. For the first batch, Ward's hierarchical clustering (Müllner, 2011) groups vectors into clusters, with the number of clusters determined by a distance threshold $\tau = \sqrt{2(1-\theta)}$, where $\theta$ is a cosine similarity parameter. For subsequent batches, each new vector $\mathbf{v}_l$ is assigned to the cluster whose centroid is nearest, provided the distance is below $\tau$; otherwise a new cluster is opened. The embedding is computed over the concatenated raw log text.

**Adaptive topic extraction.** A cluster may group logs that share a broad domain but differ in genre or intent. To determine the appropriate number of topics per cluster, the system computes the average pairwise cosine similarity $\bar{\sigma}$ among the cluster's member vectors. When $\bar{\sigma}$ exceeds a threshold $\sigma_{\text{high}}$, the cluster is treated as homogeneous and assigned a single topic. When $\bar{\sigma}$ falls below $\sigma_{\text{high}}$, an LLM reads the dialog summaries in the cluster and partitions them into up to $k_{\max}$ finer topics, assigning each case to exactly one topic.

**Skill generation and update.** Each topic $p$ that has accumulated at least $n_{\min}$ cases receives a skill. The LLM reads all cases assigned to $p$, including queries, responses, and feedback, and generalizes them into a skill document containing: a task description $d_s$, a methodology body $m_s$ with recommended procedures, negative triggers $\bar{t}_s$ specifying when the skill should be avoided, and knowledge-retrieval directives $r_s$ indicating whether the agent should query $\mathcal{K}$ when executing this skill.

When new cases arrive for an existing topic in later batches, the skill is *rewritten* from all accumulated cases for that topic, incorporating both old and new evidence. This ensures that skills improve as more interactions become available. Since all skill components ($d_s$, $m_s$, $\bar{t}_s$, $r_s$) are generated together, the retrieval interface co-evolves with the skill content: each rewrite updates the description and negative triggers that govern how the skill is retrieved and selected at inference time.

**Length calibration.** We observe that once skills are loaded, the model's output length often drifts away from that of the base model, typically producing redundant responses. Since appropriate length differs by task type, each skill stores length percentiles $\ell_s = (p_{75}, p_{90})$ over the training responses of its topic, recomputed whenever the skill is rewritten. At inference, a response longer than the matched skill's $p_{90}$ is compressed to about $p_{75}$ characters by a follow-up LLM call, unless the query explicitly asks for a longer output.

### 4.4 Inference with Skills and Knowledge

At deployment time, $\mathcal{S}$ and $\mathcal{K}$ are frozen. SkillWeave follows the standard way agents read skills, known as progressive disclosure (Anthropic, 2025). The model first sees only a short catalog entry for each candidate skill, reads the descriptions, and decides which skills to expand; only the expanded skills load their full methodology body $m_s$ into context. Our addition is to put the negative triggers into this catalog.

Given a new query, the system retrieves the top-$k$ ($k$=5) catalog entries via BM25 and presents them in the agent's system prompt. Each entry contains the skill name, the description $d_s$, and an "Avoid when" summary derived from $\bar{t}_s$. Retrieval by similarity alone can surface skills that share vocabulary with the query but require different treatment, so the negative triggers let the model reject such skills before expanding them. They act as soft guidance: the model itself judges whether the current task falls under an exclusion condition. To keep negative-trigger vocabulary from pulling in irrelevant skills, BM25 scoring ignores the negative-trigger text.

The agent selects which skills to expand through a tool call, and each expanded skill's methodology guides how the agent structures its response. If $r_s$ directs knowledge retrieval, the agent queries $\mathcal{K}$ with task-relevant terms, and the returned entries provide factual support. After the response is generated, the length calibration mechanism (§4.3) checks whether the response exceeds the matched skill's $p_{90}$ threshold and compresses it if necessary. Skills and knowledge play complementary roles: skills provide procedural guidance on how to execute a task type, while knowledge entries supply the domain facts that the procedure may require.

# 5 EXPERIMENTAL SETUP

**Benchmarks and metrics.** We evaluate on MemoryBench (Ai et al., 2025), a benchmark for continual learning from user feedback logs, in which a user simulator provides feedback on the model's own responses to training queries and the evolved system is evaluated on held-out test queries. It covers three domains. **Legal** contains 5 sub-benchmarks covering judicial opinion generation, legal QA, summarization, and legal writing (241 test items). **Academic & Knowledge** contains 6 sub-benchmarks spanning academic QA, idea generation, constrained generation, and academic writing (244 test items). **Open-Domain** contains 8 sub-benchmarks including multi-turn dialogue memory (e.g., Locomo (Maharana et al., 2024)), creative writing, and factual recall (334 test items). Each sub-benchmark keeps its original metric, and we report the official domain score obtained by min-max normalization (Appendix C). Avg. is the unweighted mean of the three domains. We additionally evaluate on ExpertLongBench (Ruan et al., 2025), 120 long-document tasks scored by raw rubric F1 under the CLEAR protocol; evaluation details for both benchmarks are in Appendix C. Significance is assessed by a two-sided paired $t$-test on per-item scores ($^{**}$: $p < 0.05$ relative to SkillWeave).

**Backbone models.** The methods are tested on three LLMs: DeepSeek-V4-Flash (DeepSeek-AI et al., 2026), Qwen3.5-27B (Qwen Team, 2026), and Gemini-3.7-Flash (Google, 2026). They are used both for accumulation-phase LLM calls and for inference-phase response generation. All evaluation uses DeepSeek-V4-Pro as the judge, which is distinct from all tested backbones.

**Baselines.** We compare against baselines spanning four categories. *No memory*: **Base** answers each query with the frozen backbone alone, serving as the lower bound. *Retrieval-augmented*: **RAG-BM25** (Robertson & Zaragoza, 2009) retrieves user logs via BM25, testing whether raw log retrieval suffices. *Memory methods*: **LightMem** (Fang et al., 2026) maintains a lightweight indexed memory store; **SimpleMem** (Liu et al., 2026) incrementally updates structured summaries from conversation logs; **A-Mem** (Xu et al., 2025a) uses an agentic architecture that autonomously decides what to memorize. *Skill-evolution methods*: **AutoSkill** (Yang et al., 2026b) extracts and maintains skill documents from experiences; **MemSkill** (Zhang et al., 2026) learns memory-operation skills via downstream task rewards; **SkillOpt** (Yang et al., 2026a) optimizes skill documents through scored trajectories and validation-set acceptance. The last three methods are not evaluated on Gemini-3.7-Flash since their build cost or wall-clock time are prohibitive (Table 3, Figure 3).

**Training protocol.** We strictly follow the MemoryBench setting, including its training/test splits and on-policy log generation. The backbone answers each training query, and Qwen3-32B (Yang et al., 2025) acts as the user simulator, giving feedback based on the response and benchmark-provided reference material (e.g., rubric checklists or reference answers). This material is never exposed to any method, and every method learns only from the dialog text (query, response, and feedback). SkillWeave additionally grounds its knowledge entries in English[2] and Chinese[3] Wikipedia dumps. All methods share the same backbone, test set, judge, and token budget (max_tokens = 4096), and SkillWeave uses batches of $b$=50 with one hyperparameter set for all domains (Appendix B).

**Baseline adaptation.** MemSkill and SkillOpt are originally designed for settings with task-success rewards or validation scores. In our evaluation, MemSkill uses a frozen pretrained controller to ingest dialog text into its memory bank; no reward or gradient signal is used at adaptation time. SkillOpt iteratively revises its skill document, using an LLM judge that scores candidate documents against the recorded user feedback, and accepts a revision only if it improves on a held-out validation split of the training logs. Neither method accesses reference answers or external reward signals.

**Retrieval budget.** All retrieval-based baselines inject the top-5 retrieved items per query (training-log passages for RAG-BM25, memory entries for LightMem, SimpleMem, A-Mem, and MemSkill), following the MemoryBench default. AutoSkill injects its top-1 skill, and SkillOpt injects its single optimized skill document, following their original designs. SkillWeave retrieves the top-5 skill catalog entries and, when a skill requests knowledge, the top-5 knowledge entries.

[2] https://huggingface.co/datasets/DragonLLM/Clean-Wikipedia-English-Articles

[3] https://huggingface.co/datasets/fjcanyue/wikipedia-zh-cn

# 6 RESULTS

## 6.1 MAIN RESULTS

Table 1 reports results on all three backbones. SkillWeave achieves the highest MemoryBench average on every backbone, with significant gains on most domains, while SimpleMem is the strongest baseline and ranks second in average throughout. Raw log retrieval (RAG-BM25) trails all memory and skill methods on most domains, most clearly on Academic & Knowledge, indicating that unprocessed logs add noise that offsets the benefit of relevant examples. Skill-evolution baselines do not consistently beat memory methods, and AutoSkill falls well below Base on Qwen3.5-27B, suggesting that extracting skills without accumulating knowledge does not fully exploit feedback logs and can be sensitive to the backbone.

The gains of SkillWeave are largest on Legal and Open-Domain. Legal tasks need both procedural conventions (e.g., judicial opinion structure) and facts (e.g., statutory provisions), which the joint skill and knowledge stores supply, and the diverse Open-Domain sub-tasks (dialogue memory, creative writing, factual recall) benefit from adaptive clustering producing skills tailored to each task type. The margin is smaller on Academic & Knowledge, likely because its more uniform tasks let simpler memory mechanisms capture much of the useful pattern.

On ExpertLongBench, SkillWeave improves over Base on every backbone ($+2.1$, $+12.9$, and $+1.3$ points on DS-V4-Flash, Qwen3.5-27B, and Gemini-Flash), ranks first on DS-V4-Flash, and stays within 0.5 points of the best method (LightMem) on the other two. The margins over baselines are smaller than on MemoryBench because these tasks are limited mainly by reading and extracting from a single long document, where reusable procedures and accumulated facts help less than on tasks that depend on procedural conventions or domain knowledge. The large gain on Qwen3.5-27B comes mostly from fixing its weak base behavior on this format, which the learned skills correct.

## 6.2 ABLATION AND ONLINE ACCUMULATION

Table 2 isolates component contributions on Legal. To test the sparsity challenge (Section 1), we disable clustering (*w/o clustering*): every training log is processed on its own and may create, update, or delete a skill based on that single log. Such a skill reflects only what one user remark happened to mention, so this variant keeps 36 skills from the 960 logs yet scores 2.2 points lower than 8 skills written from clusters of related logs. Forcing each cluster into a single topic (*Single topic*) keeps 8 skills but blurs their task boundaries and costs 5.1, the largest training-stage drop. Skill granularity thus matters more than quantity. Removing the knowledge base (*w/o KB*, $-2.9$) shows that skills alone cannot supply domain facts such as statutory provisions. At inference, with the same skill library, disabling length calibration (*w/o length stats*, $-3.2$) lets the model over-generate on tasks that need concise answers, and hiding negative triggers from the catalog (*w/o neg. triggers*, $-2.3$) leads the model to misapply skills to neighboring task types. The hard variant (*Hard neg. filter*) instead applies negative triggers as a retrieval rule: a skill is dropped from the catalog whenever any of its trigger clauses shares a word with the query, before the model sees it. This is the most harmful change ($-6.9$), because word overlap alone discards many applicable skills, whereas soft guidance lets the model judge whether an exclusion condition really applies.

The remaining variants change the sources of feedback and knowledge. In *Qwen3-32B feedback*, Qwen3-32B generates feedback without the benchmark's reference material, yielding pseudo-feedback. SkillWeave still improves over Base (47.7 vs. 39.5) and over every DS-V4-Flash baseline on Legal in Table 1, so the framework itself is beneficial even with weaker feedback. It remains below training on the official MemoryBench simulator (50.1), whose feedback was validated by human annotators to resemble real users (Appendix C), showing that realistic user feedback brings further gains. Retrieving raw Wikipedia at inference instead of the private knowledge base (*w/o KB, Wiki at inf.*) recovers only part of its gain (48.6 vs. 47.1 without knowledge and 50.1 with it), and BM25 over both logs and Wikipedia (*BM25 (logs + Wiki)*) reaches only 44.9 (vs. 38.6 for log-only RAG-BM25) despite having the same raw material as SkillWeave. Organizing feedback into skills and gap-targeted knowledge entries therefore matters beyond the information itself.

In the online setting (Figure 2), each domain's training logs are randomly split into four chunks. After each chunk, the backbone re-answers its queries with the current skill library, the simulated user gives feedback, and the resulting on-policy logs update the skill library and knowledge base

Table 1: Main results on MemoryBench and ExpertLongBench ($\times 100$). MemoryBench domains use the official min-max normalized score (Appendix C), and Avg. is the unweighted mean of the three MemoryBench domains; ExpertLongBench is raw rubric F1 (not normalized). $^{**}$: $p < 0.05$ vs. SkillWeave (paired $t$-test). **Bold**: best (ties both bolded); underline: second best per column within each model. Judge: DeepSeek-V4-Pro for all evaluations.

| Model | Type | Method | MemoryBench | | | | ExpertLongBench |
|---|---|---|---|---|---|---|---|
| | | | Legal | Acad. & Know. | Open-Domain | Avg. | |
| DS-V4-Flash | | Base | 39.5 $^{**}$ | 66.8 $^{**}$ | 64.7 $^{**}$ | 57.0 $^{**}$ | 46.2 |
| | | RAG-BM25 | 38.6 $^{**}$ | 47.3 $^{**}$ | 53.8 $^{**}$ | 46.6 $^{**}$ | 43.9 $^{**}$ |
| | Memory | LightMem | 44.7 $^{**}$ | 66.1 $^{**}$ | 59.6 $^{**}$ | 56.8 $^{**}$ | 48.0 |
| | | SimpleMem | 42.7 $^{**}$ | 68.7 | 68.5 $^{**}$ | 60.0 $^{**}$ | 45.9 |
| | | A-Mem | 45.5 $^{**}$ | 67.1 $^{**}$ | 58.7 $^{**}$ | 57.1 $^{**}$ | 47.6 |
| | Skill-Evo. | AutoSkill | 41.5 $^{**}$ | 67.1 | 63.0 $^{**}$ | 57.2 $^{**}$ | 45.3 $^{**}$ |
| | | MemSkill | 41.0 $^{**}$ | 64.6 $^{**}$ | 62.9 $^{**}$ | 56.2 $^{**}$ | 47.1 |
| | | SkillOpt | 41.9 $^{**}$ | 67.9 | 58.8 $^{**}$ | 56.2 $^{**}$ | 46.0 |
| | Co-Evo. | SkillWeave | **50.1** | **68.8** | **74.6** | **64.5** | **48.3** |
| Qwen3.5-27B | | Base | 40.1 $^{**}$ | 65.1 | 64.2 $^{**}$ | 56.5 $^{**}$ | 33.4 $^{**}$ |
| | | RAG-BM25 | 39.4 $^{**}$ | 55.6 $^{**}$ | 58.2 $^{**}$ | 51.1 $^{**}$ | 44.5 |
| | Memory | LightMem | 44.6 $^{**}$ | 61.7 $^{**}$ | 56.6 $^{**}$ | 54.3 $^{**}$ | **46.8** |
| | | SimpleMem | 47.7 $^{**}$ | 65.3 | 68.4 | 60.5 $^{**}$ | 45.5 |
| | | A-Mem | 43.8 $^{**}$ | 64.8 | 66.3 | 58.3 $^{**}$ | 42.7 $^{**}$ |
| | Skill-Evo. | AutoSkill | 34.1 $^{**}$ | 47.7 $^{**}$ | 60.0 $^{**}$ | 47.3 $^{**}$ | 35.7 $^{**}$ |
| | | MemSkill | 40.4 $^{**}$ | 59.7 $^{**}$ | 64.0 $^{**}$ | 54.7 $^{**}$ | 45.8 |
| | | SkillOpt | 42.4 $^{**}$ | 65.4 | 62.1 $^{**}$ | 56.6 $^{**}$ | 40.3 $^{**}$ |
| | Co-Evo. | SkillWeave | **52.7** | **66.1** | **73.0** | **63.9** | 46.3 |
| Gemini-Flash | | Base | 52.9 $^{**}$ | 66.1 $^{**}$ | 66.1 $^{**}$ | 61.7 $^{**}$ | 48.4 |
| | | RAG-BM25 | 50.8 $^{**}$ | 44.6 $^{**}$ | 56.0 $^{**}$ | 50.5 $^{**}$ | 48.2 |
| | Memory | LightMem | 58.2 $^{**}$ | 65.3 $^{**}$ | 63.2 $^{**}$ | 62.2 $^{**}$ | **49.7** |
| | | SimpleMem | 56.9 $^{**}$ | 66.8 $^{**}$ | 71.5 | 65.1 $^{**}$ | 48.1 |
| | | A-Mem | 59.1 $^{**}$ | 66.2 $^{**}$ | 64.6 $^{**}$ | 63.3 $^{**}$ | 49.5 |
| | Co-Evo. | SkillWeave | **61.9** | **68.3** | **77.0** | **69.0** | **49.7** |

Table 2: Ablation on Legal (DS-V4-Flash). Stage: whether the variant changes training (accumulation) or only inference.

| Stage | Variant | Score | Δ | #Skills |
|---|---|---|---|---|
| – | BM25 (logs + Wiki) | 44.9 | −5.2 | – |
| – | SkillWeave | 50.1 | – | 8 |
| Train | w/o KB | 47.1 | −2.9 | 10 |
| | w/o clustering | 47.8 | −2.2 | 36 |
| | Single topic | 45.0 | −5.1 | 8 |
| | Qwen3-32B feedback | 47.7 | −2.3 | 5 |
| | w/o KB, Wiki at inf. | 48.6 | −1.4 | 10 |
| Inf. | w/o length stats | 46.9 | −3.2 | 8 |
| | w/o neg. triggers | 47.8 | −2.3 | 8 |
| | Hard neg. filter | 43.1 | −6.9 | 8 |

Table 3: Wall-clock time on Legal (960 train, 241 test).

| Method | Build | Predict |
|---|---|---|
| SkillWeave | 2.3 h | 0.4 h |
| Base | 0.0 h | 0.1 h |
| RAG-BM25 | 0.0 h | 1.2 h |
| LightMem | 0.2 h | 0.9 h |
| SimpleMem | 0.6 h | 3.1 h |
| A-Mem | 0.1 h | 1.6 h |
| AutoSkill | 32.8 h | 2.9 h |
| MemSkill | 10.6 h | 2.4 h |
| SkillOpt | 7.9 h | 1.4 h |

before evaluation on the full test set. Under the same protocol, SkillWeave improves steadily in all three domains, while SimpleMem and AutoSkill stay flat, suggesting that their memories benefit less from incremental data. After the last chunk, online SkillWeave also surpasses its offline result in Table 1, which uses the same training queries (Legal 51.8 vs. 50.1, Academic & Knowledge 71.6 vs. 68.8, Open-Domain 77.7 vs. 74.6). Offline logs are all collected from the base model, so their feedback only describes the base model's errors. Online, each chunk is answered with the skills learned so far, and the feedback therefore targets the mistakes that remain after those skills are applied, giving the next update more useful information.

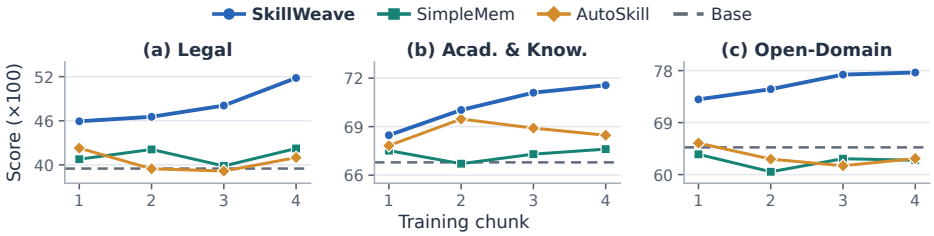



Figure 2: Online accumulation on DeepSeek-V4-Flash. Training logs arrive in four sequential chunks. SkillWeave improves steadily, while the others remain flat. Dashed: Base baseline.

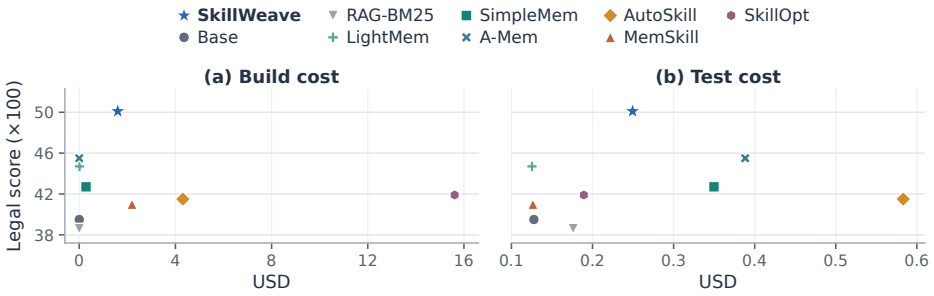



Figure 3: Cost-performance tradeoff on Legal (DeepSeek-V4-Flash, 960 training logs, 241 test cases). Left: build cost. Right: test inference cost. Colors and markers identify methods.

The appendices report further analyses. The agent system prompt alone leaves the average essentially at Base (57.2 vs. 57.0, Appendix D), skills transfer across backbones with average drops of 0.4 and 1.0 points (Appendix E), and over five full retrains SkillWeave's MemoryBench average ($63.6 \pm 0.9$) stays above every baseline with 95% confidence intervals excluding zero (Appendix F).

### 6.3 Cost Analysis

Figure 3 and Table 3 report cost and wall-clock time on Legal. Costs use \$0.14/M input and \$0.28/M output; wall-clock times use Qwen3.5-27B served by vLLM (Kwon et al., 2023) on $8\times$ NVIDIA H20 GPUs. 1) **Build:** Skill-evolution baselines update their skills log by log, which makes building slow and costly. For example, SkillOpt, with over 19k revision and validation calls, takes \$15.61 and 7.9 hours. SkillWeave processes 50 logs per batch, reflecting on them in parallel and writing one skill per topic, so it builds in \$1.60 and 2.3 hours, the cheapest and fastest among skill-based methods. Memory methods build more cheaply (at most \$0.28 and 0.6 hours) but score at least 4.6 points lower. 2) **Test:** SkillWeave's test cost (\$0.25) is below SimpleMem, A-Mem, and AutoSkill (\$0.35 to \$0.58), and its test time (0.4 hours) is the shortest after Base. In total, it reaches the best score at \$1.85, versus \$4.89 for AutoSkill and \$15.80 for SkillOpt.

## 7 Conclusion

We presented SkillWeave, a framework that learns from conversational feedback logs alone by jointly evolving a skill library, an evidence-grounded knowledge base, and the applicability conditions of each skill. By aggregating sparse feedback into task topics and storing when a skill does not apply alongside its procedure, SkillWeave absorbs the procedural, factual, and applicability signals mixed within feedback, and outperforms memory and skill-evolution methods across three LLMs.

Several directions remain open. The knowledge base grows through append-only accumulation; incorporating update and deduplication mechanisms could improve its quality over longer deployment horizons. Exploring how skills accumulated on one backbone model transfer to other models is an additional opportunity, for which our preliminary cross-backbone experiments (Appendix E) provide initial evidence.

## AI USE STATEMENT

In this work, we used generative AI tools to assist in implementing methods (refining code). We have not used generative AI tools for the other required-disclosure tasks: developing theoretical models or conceptual frameworks, proposing or refining hypotheses, designing the research methodology or experiments, translation, cleaning or reformatting datasets, or interpreting results; generating synthetic datasets, formulating mathematical claims, writing proofs, and qualitative and thematic data analysis are not applicable to this work. The model-generated responses, simulated user feedback, and LLM-judge scores in our experiments (Section 5, Appendix C) are the subject of the research, as part of the experimental method, rather than AI assistance in conducting it. Additionally, we used generative AI tools to suggest a structure for the paper and to edit the paper to improve readability. We have reviewed all AI-assisted work: AI-assisted code was reviewed and tested by the authors, the suggested structure was revised and finalized by the authors, and edited 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, we released our implementation of SkillWeave, including log generation, skill and knowledge accumulation, inference, and evaluation scripts, at https://anonymous.4open.science/r/SkillWeave . The evaluation benchmarks (MemoryBench and ExpertLongBench) and the English and Chinese Wikipedia corpora are all publicly available, and the benchmarks come with official evaluation code. The experimental setup is given in Section 5, the accumulation algorithm and hyperparameters in Appendices A and B, the benchmark and evaluation details in Appendix C, and all prompt templates in Appendix G. We hope these measures facilitate the replication of our results and support future work on learning from user feedback logs.

## REFERENCES

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.

Anthropic. Equipping agents for the real world with agent skills, 2025. URL https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills.

Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Jade Goldstein, Alon Lavie, Chin-Yew Lin, and Clare Voss (eds.), *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pp. 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL https://aclanthology.org/W05-0909/.

Valeriia Bolotova, Vladislav Blinov, Falk Scholer, W. Bruce Croft, and Mark Sanderson. A non-factoid question-answering taxonomy. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, pp. 1196–1207, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450387323. doi: 10.1145/3477495.3531926. URL https://doi.org/10.1145/3477495.3531926.

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chengyu Hou, Chenhao Xu, Chenze Shao, Chong Ruan, Conner Sun, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Donghao Li, et al. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In Iryna Gurevych and Yusuke Miyao (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 889–898, Melbourne, Australia,

July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1082. URL https://aclanthology.org/P18-1082/.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Huajun Chen, and Ningyu Zhang. Lightmem: Lightweight and efficient memory-augmented generation, 2026. URL https://arxiv.org/abs/2510.18866.

Google. Introducing Gemini 3.7 Flash, 2026. URL https://blog.google/innovation-and-ai/models-and-research/gemini-models/introducing-gemini-3-7-flash/.

Sikun Guo, Amir Hassan Shariatmadari, Guangzhi Xiong, Albert Huang, Myles Kim, Corey M. Williams, Stefan Bekiranov, and Aidong Zhang. Ideabench: Benchmarking large language models for research idea generation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, pp. 5888–5899, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400714542. doi: 10.1145/3711896.3737419. URL https://doi.org/10.1145/3711896.3737419.

Gongyao Jiang, Xinran Shi, and Qiong Luo. JRE-L: Journalist, reader, and editor LLMs in the loop for science journalism for the general audience. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6579–6594, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.335. URL https://aclanthology.org/2025.naacl-long.335/.

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/.

Jiho Kim, Woosog Chay, Hyeonji Hwang, Daeun Kyung, Hyunseung Chung, Eunbyeol Cho, Yohan Jo, and Edward Choi. Dialsim: A real-time simulator for evaluating long-term multi-party dialogue understanding of conversation systems, 2025. URL https://arxiv.org/abs/2406.13144.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 611–626, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702297. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Haitao Li, You Chen, Qingyao Ai, Yueyue Wu, Ruizhe Zhang, and Yiqun Liu. Lexeval: A comprehensive chinese legal benchmark for evaluating large language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 25061–25094. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/2cb40fc022ca7bdc1a9a78b793661284-Paper-Datasets_and_Benchmarks_Track.pdf.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Simplemem: Efficient lifelong memory for llm agents, 2026. URL https://arxiv.org/abs/2601.02553.

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.

Haoran Que, Feiyu Duan, Liqun He, Yutao Mou, Wangchunshu Zhou, Jiaheng Liu, Wenge Rong, Zekun Moore Wang, Jian Yang, Ge Zhang, Junran Peng, Zhaoxiang Zhang, Songyang Zhang, and Kai Chen. Hellobench: Evaluating long text generation capabilities of large language models, 2024. URL https://arxiv.org/abs/2409.16191.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

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.

Jie Ruan, Inderjeet Nair, Shuyang Cao, Amy Liu, Sheza Munir, Micah Pollens-Dempsey, Tiffany Chiang, Lucy Kates, Nicholas David, Sihan Chen, Ruxin Yang, Yuqian Yang, Jasmine Gump, Tessa Bialek, Vivek Sankaran, Margo Schlanger, and Lu Wang. Expertlongbench: Benchmarking language models on expert-level long-form generation tasks with structured checklists, 2025. URL https://arxiv.org/abs/2506.01241.

Weihang Su, Baoqing Yue, Qingyao Ai, Yiran Hu, Jiaqi Li, Changyue Wang, Kaiyuan Zhang, Yueyue Wu, and Yiqun Liu. Judge: Benchmarking judgment document generation for chinese legal system. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, pp. 3573–3583, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400715921. doi: 10.1145/3726302.3730295. URL https://doi.org/10.1145/3726302.3730295.

Changyue Wang, Weihang Su, Qingyao Ai, Yichen Tang, Runzhong Qiao, Xuancheng Li, Min Zhang, and Yiqun Liu. Adaptive multi-resolution procedural knowledge compression for large language models, 2026. URL https://arxiv.org/abs/2606.12203.

Yuning Wu, Jiahao Mei, Ming Yan, Chenliang Li, Shaopeng Lai, Yuran Ren, Zijia Wang, Ji Zhang, Mengyue Wu, Qin Jin, and Fei Huang. Writingbench: A comprehensive benchmark for generative writing, 2025. URL https://arxiv.org/abs/2503.05244.

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*, 2025a.

Zhijian Xu, Yilun Zhao, Manasi Patwardhan, Lovekesh Vig, and Arman Cohan. Can LLMs identify critical limitations within scientific research? a systematic evaluation on AI research papers. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 20652–20706, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1009. URL https://aclanthology.org/2025.acl-long.1009/.

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.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. Skillopt: Executive strategy for self-evolving agent skills, 2026a. URL https://arxiv.org/abs/2605.23904.

Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, Bo Zhang, and Liang He. Autoskill: Experience-driven lifelong learning via skill self-evolution, 2026b. URL https://arxiv.org/abs/2603.01145.

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

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.

# A Algorithm

Algorithm 1 summarizes the batch-wise accumulation pipeline.

**Algorithm 1** SkillWeave Batch-wise Accumulation

**Require:** Log stream $\mathcal{L}$, batch size $b$, reference corpus $\mathcal{C}$
**Ensure:** Skill library $\mathcal{S}$, knowledge base $\mathcal{K}$

Initialize $\mathcal{S} \leftarrow \emptyset$, $\mathcal{K} \leftarrow \emptyset$, clusters $\leftarrow \emptyset$, topics $\leftarrow \emptyset$
**for** each batch $B_t$ of size $b$ from $\mathcal{L}$ **do**
  *// Step 1: Knowledge Accumulation*
  **for** each log $l \in B_t$ **in parallel do**
    passages $\leftarrow$ BM25($l$, $\mathcal{C}$, $k_{\text{kb}}$)
    **if** passages $\neq \emptyset$ **then**
      (flag, title, text) $\leftarrow$ LLM_Reflect($l$, passages)
      **if** flag = true **then**
        Append (title, text) to $\mathcal{K}$
      **end if**
    **end if**
  **end for**
  *// Step 2: Incremental Clustering*
  **for** each log $l \in B_t$ **do**
    $\mathbf{v}_l \leftarrow$ Embed($l$)
    Assign $l$ to nearest cluster or open new cluster (threshold $\tau$)
  **end for**
  *// Step 3: Topic Extraction*
  **for** each cluster $c$ that received new logs **do**
    $k \leftarrow$ TopicCount($c$, $\sigma_{\text{high}}$, $k_{\max}$)
    Assign cases in $c$ to $k$ topics via LLM
  **end for**
  *// Step 4: Skill Generation*
  **for** each topic $p$ with $|\text{cases}(p)| \geq n_{\min}$ **do**
    $s_p \leftarrow$ LLM_GenerateSkill(all cases of $p$)
    Update $\mathcal{S}$ with $s_p$ (create or rewrite)
  **end for**
**end for**
**return** $\mathcal{S}, \mathcal{K}$

# B Hyperparameters

Table 4 lists all hyperparameters. A single configuration is shared across all domains; the only domain-specific difference is that domains with session corpora (Open-Domain) use larger knowledge-base capacity to accommodate the injected transcripts.

Table 4: Hyperparameters used across all experiments.

| Group | Parameter | Value |
|---|---|---|
| **Clustering & Topics** | Similarity threshold $\theta$ | 0.15 |
| | Intra-cluster sim. threshold $\sigma_{\text{high}}$ | 0.7 |
| | Max topics per cluster $k_{\max}$ | 3 |
| | Min samples per topic $n_{\min}$ | 5 |
| | Batch size $b$ | 50 |
| | Embedding model | all-MiniLM-L6-v2[4] |
| **Skill Library** | Max skills | 64 |
| **Retrieval** | Skill top-$k$ | 5 |
| | KB top-$k$ ($k_{\text{kb}}$) | 5 |
| | Retrieval backend | BM25 |
| **Knowledge Base** | Max entries | 512 (5000 for Open-Domain) |
| | Max entry chars | 2500 (12000 for Open-Domain) |
| **Inference** | Temperature | 0.1 |
| | Max tokens | 4096 |
| | Max tool rounds | 4 |

# C Benchmark Details and Metrics

## C.1 MemoryBench

**Benchmark.** MemoryBench (Ai et al., 2025) evaluates continual learning from user feedback logs. The benchmark provides the training logs as the only learning signal and evaluates the evolved system on held-out test queries. We use its three domain-level sets and follow the official training/test splits: **Legal** (JuDGE (Su et al., 2025); LexEval-Judge, LexEval-QA, and LexEval-Summarization (Li et al., 2024); WritingBench-Politics&Law (Wu et al., 2025); 241 test items), **Academic & Knowledge** (HelloBench-Academic&Knowledge-QA and -Writing (Que et al., 2024); IdeaBench (Guo et al., 2025); JRE-L (Jiang et al., 2025); LimitGen-Syn (Xu et al., 2025b); WritingBench-Academic&Engineering (Wu et al., 2025); 244 test items), and **Open-Domain** (DialSim-BigBang, -Friends, and -TheOffice (Kim et al., 2025); HelloBench-Creative&Design (Que et al., 2024); Locomo (Maharana et al., 2024); NFCats (Bolotova et al., 2022); WritingBench-Creative&Design (Wu et al., 2025); WritingPrompts (Fan et al., 2018); 334 test items).

**User simulator.** Training logs are produced by MemoryBench's user simulator, and we use it exactly as released, with its prompts, model, and settings unchanged. For every training query, the backbone's own response is shown to the simulator, which also sees the benchmark's reference material (e.g., rubric checklists or reference answers) and returns natural-language feedback; the resulting dialogs form the training logs. The reference material only helps the simulator give targeted feedback and is never shown to any method. The simulator's quality was verified by the MemoryBench authors: in A/B tests, human annotators could not distinguish its logs from real user logs (Ai et al., 2025). Every other part of our evaluation, including data splits, metrics, normalization bounds, and judge prompts, likewise follows MemoryBench without modification.

**Per-dataset metrics.** Each sub-dataset is scored with its original metric. Deterministic metrics include F1 (Locomo), accuracy (DialSim), Rouge-L (Lin, 2004) (LexEval), and METEOR (Banerjee & Lavie, 2005) (WritingPrompts). Open-ended datasets (e.g., HelloBench, WritingBench, NFCats, LimitGen-Syn) are scored by an LLM judge, and datasets with several criteria (e.g., IdeaBench, JuDGE) have the judge combine them into a single score. All metrics and judge prompts follow the official MemoryBench protocol; we use DeepSeek-V4-Pro as the judge for every method.

**Domain score.** Because a domain mixes datasets whose metrics have different scales, MemoryBench first rescales each item score $x$ from dataset $d$ as $\tilde{x} = (x - \min_d)/(\max_d - \min_d)$, where $\min_d$ and $\max_d$ are the normalization bounds released with the official repository. The domain score is the mean of $\tilde{x}$ over all test items in the domain, so each dataset contributes in proportion to its

[4] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

number of test items. We report this score $\times 100$. Scores are meaningful only for comparing methods within the same backbone and domain. Significance tests are paired over the same normalized item scores.

### C.2 EXPERTLONGBENCH

ExpertLongBench (Ruan et al., 2025) consists of expert-level long-form generation tasks, each built around a long input document. We use 480 training queries and 120 test items. Training logs are generated on-policy with the same user simulator as for MemoryBench, and every method uses the same backbone, test set, and inference budget. Evaluation follows the official CLEAR protocol. For each item, rubric checklist items are extracted from both the model output and the reference, and an LLM judge (DeepSeek-V4-Pro) decides in both directions whether each item is semantically contained in the other text. Rubric-level precision and recall are combined into F1, which we report as raw scores ($\times 100$) without min-max normalization. Significance tests are paired over the same per-item F1 scores.

## D SYSTEM PROMPT EFFECT

To test whether SkillWeave's gains come from its agent-style system prompt, we run an additional condition (Base + system prompt) that uses the full SkillWeave system prompt with an empty skill catalog and no knowledge base on DeepSeek-V4-Flash.

Table 5: System prompt control (DeepSeek-V4-Flash, $\times 100$).

| Method | Legal | Acad. & Know. | Open-Domain | Avg. |
|---|---|---|---|---|
| SkillWeave | 50.1 | 68.8 | 74.6 | 64.5 |
| Base + system prompt | 43.6 | 65.8 | 62.2 | 57.2 |
| Base | 39.5 | 66.8 | 64.7 | 57.0 |

The system prompt alone leaves the domain average essentially unchanged relative to plain Base (57.2 vs. 57.0). On Legal, the system prompt provides a moderate boost (+4.1), but on Open-Domain it is negative ($-2.5$) and on Academic & Knowledge it is near zero. SkillWeave significantly outperforms Base + system prompt on all three domains, confirming that the improvement originates from the evolved skills and knowledge.

## E CROSS-BACKBONE TRANSFER

To examine whether skills generalize across backbone models, we run transfer experiments where the skill library is trained on one backbone and used for inference on another.

Table 6: Cross-backbone skill transfer ($\times 100$). "Own" is the score when both training and inference use the same backbone.

| Skills from | Inference on | Legal | Acad. & Know. | Open-Domain | $\Delta$ (avg.) |
|---|---|---|---|---|---|
| Qwen3.5-27B | DS-V4-Flash | 51.1 | 66.2 | 74.9 | $-0.4$ |
| (Own DS-V4-Flash) | | 50.1 | 68.8 | 74.6 | – |
| DS-V4-Flash | Qwen3.5-27B | 52.6 | 64.5 | 71.8 | $-1.0$ |
| (Own Qwen3.5-27B) | | 52.7 | 66.1 | 73.0 | – |

Transferred skills retain most of the performance of native skills. The average domain-level drop is small (0.4 points for Qwen→DeepSeek and 1.0 points for DeepSeek→Qwen), indicating that the procedural guidance captured by SkillWeave is largely model-agnostic.

# F RETRAIN VARIABILITY

To check that SkillWeave's advantage is not due to a favorable run, we repeat the full pipeline five times on DeepSeek-V4-Flash with the same configuration and training logs, and do the same for the main baselines (for Base, which has no training stage, only inference and judging are repeated). Table 7 reports the MemoryBench average (mean of the three domains) over the five runs. SkillWeave reaches $63.6 \pm 0.9$, and its margin over every baseline stays positive, with 95% confidence intervals that exclude zero. The run-to-run spread of SkillWeave (0.9) is comparable to that of the baselines (0.6 to 1.5) and much smaller than its margins, so the gains in Table 1 are not an artifact of retraining noise.

Table 7: Retrain variability on DeepSeek-V4-Flash (MemoryBench average, $\times 100$, five runs). $\Delta$: SkillWeave minus the method, with its 95% confidence interval.

| **Method** | **Mean $\pm$ Std** | $\Delta$ | **$\Delta$ 95% CI** |
|---|---|---|---|
| SkillWeave | $63.6 \pm 0.9$ | – | – |
| Base | $57.0 \pm 0.6$ | $+6.6$ | $[+4.6, +8.6]$ |
| LightMem | $57.3 \pm 1.2$ | $+6.3$ | $[+4.3, +8.3]$ |
| SimpleMem | $59.7 \pm 1.5$ | $+3.9$ | $[+2.1, +5.8]$ |
| MemSkill | $55.9 \pm 0.9$ | $+7.7$ | $[+5.9, +9.6]$ |
| SkillOpt | $56.4 \pm 0.8$ | $+7.2$ | $[+5.3, +9.1]$ |

# G PROMPT TEMPLATES

This section lists the prompt templates used in SkillWeave's accumulation and inference pipelines. All templates exist in both Chinese and English versions; the system selects the version matching the sample language. We show the English versions below.

## G.1 KNOWLEDGE REFLECTION

**System Prompt: Knowledge Reflection**

You build the private knowledge base. Reflect only from the plain-text user log about factual knowledge the model may lack, and rewrite retrieved wiki snippets into reusable private knowledge entries. Do not add an entry without reliable retrieved evidence. Use English.

**User Prompt: Reflection and Rewrite**

Language: {lang}

Plain-text interaction log:
{dialog_text}

Retrieved factual snippets:
{retrieved_docs}

Decide whether to add private knowledge. Use only facts supported by the retrieved snippets; do not invent facts. Return JSON: {"should_add": true/false, "title": "...", "text": "rewritten knowledge entry"}

The model call uses temperature 0.1 and max_tokens 2000. Retrieved snippets are truncated to 10,000 characters before insertion.

## G.2 TOPIC EXTRACTION

Each cluster is processed once per batch.

User Prompt: Topic Extraction

Please analyze the following {N} similar cases and extract main topics.

Important notes:

- Decide how many topics to extract based on the actual characteristics of the cases (1 to {k}).
- If the cases are similar, extract only 1 topic.
- If the cases have clearly different subtypes, extract multiple topics (up to {k}).
- Each topic must be a generalization of a task type or application scenario.
- Each case belongs to exactly one topic.

Cases:
{log_summaries}

Output JSON: {"topics": [{"topic": "name", "description": "...", "case_indices": [0, 2, 5]}, ...]}

### G.3 Skill Generation

Each topic with at least $n_{\min}$ cases is processed once. The prompt includes 11 core principles; we show the key ones below.

User Prompt: Skill Generation (abridged)

Generate a generalizable skill from the following similar training cases.

**Task Type**: {topic}
**Similar Cases**: {similar_cases}

Core Principles:

1. Prefer generalizable skills that capture task patterns, quality preferences, and common failure modes.
2. Every skill must be Markdown with YAML frontmatter: name, description, negative_triggers.
3. The description must clearly state when to use; negative_triggers must state when not to use, especially neighboring genres.
4. Do not copy verbatim from specific samples; infer generalizable patterns.
5. Learn output format from the answer style of similar cases.
6. Extract conventional phrasings that genuinely recur in the cases.
7. If the task may need private factual knowledge, the methodology can instruct the agent to call search_private_kb.
8. Make length and structure conditional on task properties; never state an unconditional length requirement.

Output the SKILL.md Markdown content directly (starting with —), with complete YAML frontmatter and methodology body.

### G.4 Inference System Prompt

System Prompt: Inference Agent

You are a careful LLM-Agent. First read the available skill name/description catalog. When a skill's description matches the user's task type, goal, or domain, strongly consider loading it via the platform's standard tool mechanism.

Load a skill when its description indicates relevance to the current task: matching genre (QA, generation, analysis), domain (academic, legal, news), or method (extraction, summarization). Even partial

matches can be valuable.

However, if a catalog line contains "Avoid when" conditions and the current task falls under them, you MUST skip that skill.

Use the private knowledge base only when a loaded skill explicitly requires it and the task truly lacks private factual knowledge. If the user already provided enough source material, use that material directly.

Available skill catalog (internal reference only):
{skill_catalog}

## H SKILL EVOLUTION CASE STUDY

To illustrate how skills evolve as more interactions accumulate, we trace the `legal-essay-analysis` skill from the online experiment (Legal domain, DeepSeek-V4-Flash) across chunk 1 (240 training logs) and chunk 4 (960 cumulative logs). Table 8 summarizes the changes.

Table 8: Evolution of the `legal-essay-analysis` skill from chunk 1 to chunk 4.

| Component | Chunk 1 (18 cases) | Chunk 4 (58 cases) |
|---|---|---|
| **Description** | Analyze legal essay questions: case characterization, legal application, procedural issues. | Answer legal essay questions: case analysis, opinion articulation, legal document drafting. Covers criminal, civil, administrative, and procedural law. |
| **Negative triggers** | (1) User only asks for a short answer<br>(2) User asks for legal documents (indictments, judgments)<br>(3) User asks for legal translation<br>(4) User asks for legal consultation, not academic analysis | (1) Rote recitation of legal provisions or definitions<br>(2) Template teaching for legal document formats<br>(3) Non-legal essay questions<br>(4) Factual Q&A ("what does Article X say") |
| **Length stats** | mean 2692, p75 3054, p90 3384 | mean 2115, p75 2489, p90 3002 |
| **Methodology** | 11 sections: problem type classification (4 types), analysis framework, citation format, handling disputes, distinguishing similar crimes (4 specific pairs), domain phrasings, length control, handling user follow-ups, evidence evaluation, common mistakes, KB retrieval. | 7 sections: conclusion-first principle, layered argumentation structure, citation of legal provisions and principles, case-specific analysis, handling disputes and multiple opinions, criminal law specifics (concurrence, attempt, sentencing), language style and output format. |

The skill broadened its scope (from "case characterization" to multiple task types across all law branches), replaced all four negative triggers with more precise exclusion conditions, and restructured the methodology from a prescriptive 11-section checklist into a more concise 7-section argumentation guide. The length statistics shifted downward as more diverse cases (including shorter tasks) entered the topic, lowering p75 from 3054 to 2489 characters. These changes reflect the cumulative effect of 40 additional training cases: the system observed a wider range of legal essay tasks and generalized the skill accordingly.