Textualized Learning: A Survey of Non-Parametric Experience Learning in Large Language Models

Anonymous ACL submission

Abstract

Recently, a growing body of work, despite varying in tasks and terminology, shares a new learning structure: deriving explicit, reusable text-based artifacts (instead of updating parameters) from task experience and feedback using LLMs to improve performance. We unify these efforts as *Textualized Learning*. We provide a formal definition of textualized learning and characterize it as a three-stage process, namely, artifact generation, organization, and usage, and survey recent works spanning automatic prompt optimization, agent memory, test-time improvement, skill learning, and beyond. Unlike prior surveys, we analyze these works through the lens of the learning process itself: examining how artifacts are abstracted, evaluated, maintained, and applied, and discuss connections to supervised learning and reinforcement learning. We also summarize its application domains and highlight open challenges and future directions.

1 Introduction

Learning from experience is fundamental to intelligent systems. Expert systems externalize knowledge as symbolic rules curated by human experts (Buchanan and Shortliffe, 1984; Feigenbaum, 1977), whereas deep learning internalizes knowledge in model parameters through backpropagation (Goodfellow et al., 2016). These paradigms involve different trade-offs: human-induced symbolic rules are interpretable but difficult to scale, whereas parametric knowledge scales automatically but is less transparent and controllable.

Building intelligent systems on top of large language models (LLMs) has become increasingly common, yet such systems still face the challenge of rapidly learning domain- and scenario-specific knowledge. Such knowledge is often highly specialized and cannot be reliably acquired from general pretraining alone. For example, software engineering tasks require models to follow compilation constraints, interface specifications, testing requirements, and repository workflows (Yang et al., 2024c; Wang et al., 2024a), while financial tasks depend on business rules, regulatory requirements, and decision boundaries (Yang et al., 2023).

Humans often address such problems by explicitly organizing experience into reusable rules. For example, students write notes and error summaries; annotators refine annotation guidelines after mistakes; engineers document operating procedures after troubleshooting failures. This "experience → rule → reuse" pathway externalizes tacit knowledge into explicit, reusable textual forms, reducing the cognitive burden of future tasks.

As LLMs acquire stronger induction and reasoning abilities, recent work has begun to explore whether they can emulate this human-like learning pathway. We term this process as **Textualized Learning** (TL): given a verifiable dataset or task environment, a system abstracts task experience and feedback into explicit, reusable artifacts, maintains them externally, and applies them during inference to improve performance without updating model parameters. These artifacts include prompts, rules, code, or other structured objects.

Compared with parametric learning, TL offers three advantages. 1) Low-cost: both learning and inference only rely on general-purpose model APIs, without specialized model deployment. 2) Interpretable: generated artifacts are expressed in human-readable forms and can be inspected, corrected, or extended by users (Askari et al., 2024). 3) Easy to maintain: when task rules change or new patterns emerge, artifacts can be edited or supplemented without retraining the underlying model (Cao et al., 2025).

Currently, related work is dispersed across several research directions, including automatic prompt optimization (Pryzant et al., 2023; Spiess et al., 2025), experience memory (Zhao et al., 2023;

Li et al., 2025a), autonomous agents (Wang et al., 2024c), and test-time improvement (Snell et al., 2025). They also use diverse terminologies, such as experience learning (Chen et al., 2025), rule induction (Pang et al., 2023), procedural memory (Mi et al., 2026), and skill evolution (Zheng et al., 2025). Although these methods differ in task form, feedback source, and usage mechanism, they share a common pattern: transforming experience into reusable external artifacts. So, this survey studies them under a unified framework.

The surveys most closely related to this paper focus on memory in LLMs (Wu et al., 2025c; Huang et al., 2026; Hu et al., 2025), since memory is often represented textually. However, these surveys primarily examine memory taxonomies, such as episodic, semantic, and procedural memory, as well as memory usage mechanisms. They provide less analysis of the learning process itself. In contrast, this paper adopts a learning-centered perspective and systematically characterizes the task definition and key stages of Textualized Learning.

This survey makes three contributions. First, Section 2 introduces a formal definition of TL based on the classical definition of machine learning, clarifying which works fall within its scope. Second, we summarize three key stages of TL: artifact generation, organization, and usage. This framework unifies scattered terms such as experience synthesis (Zhao et al., 2023), skill induction (Wang et al., 2025c), memory update (Zhang et al., 2026a), experience refinement (Cai et al., 2025b), retrieval or selection (Fu et al., 2024; Mi et al., 2026), and policy guidance (Liang et al., 2026b). Third, we relate TL to supervised and reinforcement learning, review application domains, and discuss challenges and future directions.

2 Definition and Framework

2.1 Definition

A classical definition of machine learning states that: a computer program learns from experience E with respect to a class of tasks T and a performance measure P, if its performance at task T, as measured by P, improves with experience E (Mitchell, 1997). TL can be viewed as one implementation of this general framework. Specifically, for a task class T and a performance measure P, if a program improves its performance at T by *generating, organizing, and using textual artifacts*, then the program performs Textualized Learning. Here, artifacts may include prompts, code, skills, and other reusable external objects.

2.2 Unified Formalization

To describe the mechanism of TL more rigorously, we formulate it as a unified framework. Let x denote a task input, y^* the target output, $\mathcal{M}$ a pretrained LLM, and $\mathcal{G}$ an artifact library. TL has three key stages—generation, organization, and usage, as shown in Figure 1.

The generation stage abstracts reusable artifacts from task experience and feedback. Given input x, target output y^*, and model prediction $\hat{y}$, a feedback function $\mathcal{F}$ produces signals that indicate the strengths, weaknesses, or errors of the current output. Based on this feedback, a generation function ϕ generates an experience artifact g with potential cross-sample generality:

$$g = \phi(x, \hat{y}, \mathcal{F}(x, \hat{y}, y^*)) \,. \tag{1}$$

The organization stage integrates multiple gs into an artifact library $\mathcal{G}$. Let $\mathcal{G}_t$ denote the artifact library at step t. The system uses a management function $\mathcal{U}$ to incorporate g:

$$\mathcal{G}_{t+1} = \mathcal{U}(\mathcal{G}_t, g), \tag{2}$$

where $\mathcal{U}$ may include filtering, merging, deduplication, pruning, and conflict resolution, to preserve validity and consistency while controlling library size.

The usage stage applies the artifact library during inference. Given a new input x, the system first retrieves a relevant subset from $\mathcal{G}$ through a retrieval function $\mathcal{R}$, and then integrates the retrieved artifacts into the model input through an injection function $\mathcal{I}$. Artifacts may be injected as system instructions, context prefixes, few-shot examples, or structured constraints. The final prediction is:

$$\hat{y} = \mathcal{M}(x, \mathcal{I}(x, \mathcal{R}(\mathcal{G}, x))) \,. \tag{3}$$

The overall objective of TL is to maximize the expected performance of the artifact-enhanced model over the task distribution $\mathcal{D}$, where P evaluates the prediction against the target:

$$\max_{\mathcal{G}} \mathbb{E}_{(x,y^*)\sim\mathcal{D}}[\mathrm{P}(\mathcal{M}(x, \mathcal{I}(x, \mathcal{R}(\mathcal{G}, x))), y^*)]. \tag{4}$$

2.3 Connections with Related Fields

Parametric learning and TL share similar optimization goals but optimize different variables, namely, model parameters θ and textual artifact library $\mathcal{G}$. In classical supervised learning, learning involves defining a loss function L_θ, computing the gradient $\nabla_\theta L$, and updating parameters by $\theta = \theta - \eta \nabla_\theta L$.

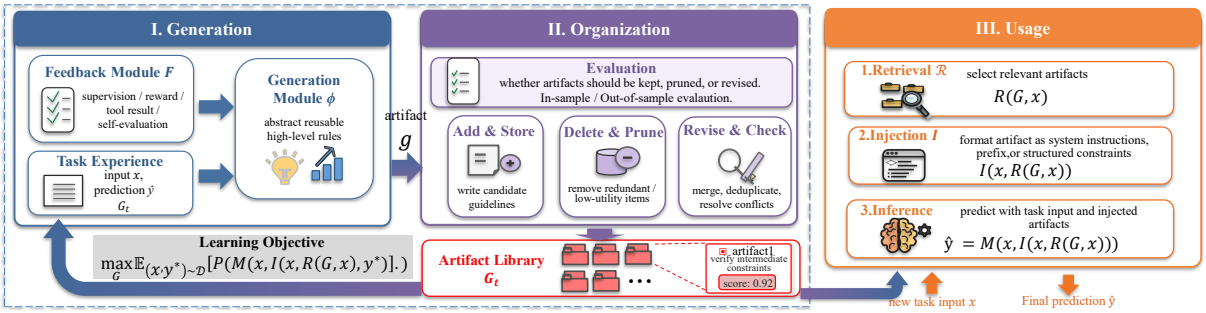

Figure 1: An overview of the unified formalization of Textualized Learning.

The analogous components in TL are the feedback function, artifact generation, and artifact addition during organization. This analogy suggests that techniques from traditional supervised learning may inform future TL systems.

However, unlike gradients, textual artifacts are not rigorously derived from differentiable objective functions and therefore do not inherently guarantee improvement. The utility of a new artifact must be empirically verified on the current sample, validation samples, future tasks, or online interactions. Maintaining reliable and compact artifact libraries is therefore a distinctive challenge of TL.

Appendix C further compares TL with agent memory, in-context learning, retrieval-augmented generation, and self-correction.

3 Forms and Representations of Artifacts

In TL, artifacts are reusable external objects generated by a model from task experience. Depending on their abstraction level, information completeness, and executability, artifacts can take diverse forms, including prompts, experience trajectories, rules, procedural skills, and executable implementations.

Prompt-style artifacts store system prompts, influencing model outputs by modifying the overall prompt. Systems usually retain only a small set of best-performing prompts (Pryzant et al., 2023; Khattab et al., 2024; Agrawal et al., 2025; Spiess et al., 2025).

Raw trajectories store task execution processes and are reused as few-shot examples. Such records may include historical states and decision paths (Zhou et al., 2025; Zheng et al., 2023; Kim et al., 2024a); they may also be organized as graph structures that record tool-call transitions and state summaries (Li et al., 2025a; Tang et al., 2025).

Rule-style artifacts summarize more abstract and generalizable experiences. Typical forms include natural-language reflections (Zhao et al., 2023; Ouyang et al., 2025b; Liang et al., 2025) and condition-specific rules (Fu et al., 2024). Some methods decompose experience into erroneous behavior, error analysis, and corrective clues (Zhang et al., 2026g; Wu et al., 2025a), or use artifacts to characterize environmental regularities and physical constraints (Ren et al., 2026).

Skills further specify triggering conditions, execution steps, and termination criteria on top of rules, forming reusable task procedures. Existing work represents skills as workflow-oriented modular procedures (Wang et al., 2024c; Han et al., 2025; Fan et al., 2025), standardized entries with trigger sets, tags, and versions (Yang et al., 2026; Chen et al., 2026; Zhou et al., 2026b), or procedural memories consisting of goals, preconditions, abstract action sequences, and postconditions (Forouzandeh et al., 2025; Liang et al., 2026b).

Code and tool artifacts are executable code, functions, or tools. This is the most structured and executable form. Rather than injecting complete textual descriptions into the context, the system reuses experience by calling runnable modules (Wang et al., 2023a, 2025c; Qiu et al., 2025b; Liang et al., 2025; Wu et al., 2026).

Table 2 in the Appendix summarizes the major representation forms considered in this paper.

4 Artifact Generation

Artifact generation is a prompt-driven process of experience abstraction, corresponding to the generation function in Eq. 1. The system first uses supervision signals to evaluate model performance on samples or task executions, then selects experience materials at an appropriate granularity as the generation context, and finally uses explicit instructions to prompt the LLM to reflect on, compare, summarize, or rewrite these materials into candidate artifacts. A detailed paper-level taxonomy of

the generation stage is provided in Appendix F.

Specifically, for the i-th sample or task execution, the system observes the input x_i, the model output $\hat{y}_i$, and the supervision signal $\mathcal{F}(x_i, \hat{y}_i)$ returned by the feedback module. The experience set used for artifact generation can therefore be written as:

$$D = \{(x_i, \hat{y}_i, \mathcal{F}(x_i, \hat{y}_i, y_i^*))\}_{i=1}^{n}. \quad (5)$$

The generation function ϕ takes the experience set D and generation context C as input, abstracts model behavior and feedback, and produces an artifact g:

$$g = \phi(D, C), \quad (6)$$

4.1 Feedback Function

The feedback function $\mathcal{F}$ determines the signals used for artifact generation. Existing methods mainly use two types of feedback.

Feedback from outside the model provides explicit evaluation criteria, allowing the system to judge whether model behavior is correct. Such feedback may be based on labels or environmental rewards.

Label-based feedback may involve fine-grained single-sample comparisons, such as individual erroneous examples (Pang et al., 2023), or dataset-level performance metrics, such as validation accuracy (Zhou et al., 2022; Yang et al., 2024b).

External feedback may also come from the environment, especially after agent interaction. Such feedback can be divided into trajectory-level and step-level signals. Trajectory-level feedback evaluates the final outcome of an entire task episode, such as whether the task was completed, the goal was achieved, or the trajectory was successful (Fu et al., 2024; Cai et al., 2025b). Step-level feedback evaluates individual actions, tool calls, or subtasks, such as whether an operation succeeded, whether the current step made progress, or whether the model exhibited high uncertainty at a particular step (Zhang et al., 2026g; Wang et al., 2023a; Xie et al., 2026; Yang et al., 2025b).

Feedback produced by the model itself or by auxiliary verification modules. This feedback is useful when gold answers or external evaluation signals are unavailable, and includes self-critique, self-verification, and self-consistency comparisons. Early work adopted the basic "generate–self-feedback–revise" framework (Madaan et al., 2023; Shinn et al., 2023), but primarily served the current instance rather than forming reusable artifacts. Later work retains internal feedback generated during inference and converts it into reusable artifacts (Zhao et al., 2023). When a single output does not provide a stable criterion, internal feedback may take the form of self-consistency: systems generate multiple candidate answers, reasoning paths, or experience samples in parallel (Cai et al., 2025a; Ouyang et al., 2025b). Some work does not explicitly judge execution correctness, but instead incorporates raw trajectories into the model to provide familiarity with the environment (Liu et al., 2025b).

4.2 Generation Function

The generation function converts model behavior, experience, and feedback into candidate artifacts. In implementation, ϕ is typically instantiated as an LLM prompt. Its input can be written as Eq. 5; the prompt supplies experience materials—samples, trajectories, or candidate outputs to be summarized—and generation instructions specifying how the model should interpret these materials and produce artifacts.

Experience granularity determines the scope of materials from which ϕ performs abstraction, analogous to batch size in gradient descent. **Single-sample** granularity generates an artifact from one experience unit. Because the task context, model behavior, and feedback signal form a relatively complete experience loop, this setting helps identify error causes, effective strategies, and local decision rationales. It is commonly used for local artifact generation (Zhang et al., 2026g) and skillization of single-task experience (Wang et al., 2023a). **Multi-sample** granularity abstracts artifacts from multiple experience units, including rules induced from boundary differences among labeled samples (Pang et al., 2023), self-correction experience summarized from Text-to-SQL errors and corrections (Askari et al., 2024), frequent error patterns extracted from clusters of failed samples (Su et al., 2025), and multiple reasoning paths (Ouyang et al., 2025b) or candidate experience samples (Cai et al., 2025a) generated for the same input. Serial multi-attempt generation further performs repeated interactions and revisions on the same task, obtaining richer experience from continuous feedback chains (Cai et al., 2025b).

Generation instructions specify how the LLM should abstract artifacts. Based on the primary goal of generating instructions, we divide them into four categories: reference-guided error discrimination, positive-experience distillation, process-record or-

ganization, and self-diagnostic error mining.

1. **Reference-guided error discrimination** asks the model to compare correct and incorrect cases. In answer-generation tasks, systems compare erroneous predictions with labels to form transferable checking rules (Pang et al., 2023; Askari et al., 2024), or compare failed patches with successful repairs to form checking rules or self-correction artifacts (Chen et al., 2025). In long-horizon interactive tasks, systems compare successful and failed trajectories to extract reusable experience from overall success/failure contrasts (Zhao et al., 2023; Cai et al., 2025b), differences in state transitions and action selection (Fu et al., 2024; Zhang et al., 2026g), tool-use or procedural execution processes (Qiu et al., 2026), and divergences in reasoning paths (Ouyang et al., 2025b).

2. **Positive-experience distillation** asks the model to extract artifacts from correct trajectories. Outputs often take the form of successful principles (Zhao et al., 2023; Cai et al., 2025b; Kim et al., 2025). For long-horizon tasks, such instructions ask the model to extract workflows, step templates, or subprocesses from successful trajectories (Wang et al., 2024c; Fan et al., 2025; Han et al., 2025; Fang et al., 2025; Yang et al., 2025b). In embodied, tool-use, and code agents, instructions may further ask the model to specify triggering conditions, inputs and outputs, and execution steps (Yang et al., 2026; Mi et al., 2026), extract locally transferable skills from successful trajectories (Ni et al., 2026; Ding et al., 2025), or transform successful trajectories into verifiable code or tool-use procedures (Wang et al., 2023a; Xiao et al., 2025), enabling later invocation or checking.

3. **Process-record organization** asks the model to compress and format raw trajectories. Such methods preserve rich contextual details and are suitable for long-horizon tasks that are difficult to compress into short rules. Some methods directly store complete or near-complete historical trajectories for later decision-making (Zheng et al., 2023; Kim et al., 2024a; Liu et al., 2025b); others retain only key observations, actions, states, and outcomes (Zhou et al., 2025; Chhikara et al., 2025; Zhong et al., 2024; Zhang et al., 2026e). More structured approaches rewrite experience as graph nodes (Xia et al., 2025) or tool-execution processes (Tang et al., 2025), facilitating later retrieval, comparison, and updating.

4. **Self-diagnostic error mining** takes failed experience as input and asks the model to diagnose failure causes without gold answers, then summarize them as reusable artifacts. Typical practices include inducing error patterns from failed samples (Su et al., 2025), improving efficiency from inefficient embodied-task behavior (Xie et al., 2026), forming reflections from execution failures and feedback signals (Liang et al., 2025; Zhang et al., 2024a, 2026d), and transforming code errors into evidence for subsequent skill repair (Zheng et al., 2025).

According to **generation timing**, the generation function can be divided into offline generation and online generation. **Offline** generation relies on a preconstructed dataset, whose samples usually include gold answers. The system generates artifacts on this dataset and keeps the artifact library $\mathcal{G}$ fixed during later testing or deployment. This setting is suitable for batch induction from erroneous samples (Pang et al., 2023; Su et al., 2025) and validation-driven prompt optimization (Yang et al., 2024b; Agrawal et al., 2025; Spiess et al., 2025). **Online** generation can be viewed as a non-parametric form of test-time learning (Sun et al., 2020; Wang et al., 2021): artifacts are generated during testing or task execution, and the system continually accumulates experience and updates artifacts based on environmental feedback, execution results, or repeated attempts. Examples include post-task reflection (Zhao et al., 2023; Zhang et al., 2024a; Liang et al., 2025), execution-feedback-driven skill repair (Wang et al., 2023a; Zheng et al., 2025; Wu et al., 2025a), and experience updates through multi-round attempts (Cai et al., 2025b; Zhang et al., 2026a).

5 Artifact Organization

Artifact organization concerns how artifacts generated are maintained as reusable external knowledge structures; see Eq. 2. It not only writes new artifacts into the library but also maintains the collection over time with three main functions: addition, deletion, and modification. A central technical issue underlying all three functions is artifact evaluation.

5.1 Evaluation

Evaluation determines whether an artifact is worth retaining and therefore supports addition, deletion, and modification.

In-sample evaluation generates an artifact from the current sample, trajectory, or task and verifies its effect on the same object. Common signals

include whether an error is fixed (Askari et al., 2024) and whether code or skills execute successfully (Wang et al., 2023a; Yang et al., 2026). This evaluation is low-cost, but local effectiveness does not necessarily imply cross-task generalization.

Out-of-sample evaluation measures generalization on validation sets, new tasks, future tasks, or cross-environment scenarios. Common metrics include validation accuracy (Pang et al., 2023), post-revision task metrics (Askari et al., 2024), and success rates on new tasks (Xia et al., 2026; Liang et al., 2026b).

Long-term interaction settings further use online cumulative signals, such as recency, importance, and relevance (Park et al., 2023); time decay and user relevance (Zhong et al., 2024); explicit memory-state operations (Chhikara et al., 2025); and retrieval success rate, execution gain, and usage frequency (Tang et al., 2025). When the candidate space is large, systems also perform relative evaluation through validation-score ranking (Yang et al., 2024b; Pryzant et al., 2023; Agrawal et al., 2025), evolutionary selection (Guo et al., 2023; Fernando et al., 2023), or candidate-policy comparison (Zhang et al., 2024a, 2026d).

5.2 Addition

Addition incorporates artifacts into $\mathcal{G}$. The simplest strategy is direct writing, such as storing successful trajectories (Zheng et al., 2023), similar historical cases (Zhou et al., 2025), state-related artifacts after task completion (Fu et al., 2024), or new experiences after iterative refinement (Qiu et al., 2026). Direct writing enables rapid accumulation but may introduce redundancy and retrieval noise.

A more conservative strategy imposes admission rules before writing. Systems may save a code skill only if execution tests pass (Wang et al., 2023a), filter procedural memories based on reward gates or online scores (Mi et al., 2026), retain web skills according to invocation feedback (Zheng et al., 2025), judge cross-task transferability based on shared conditions (Yang et al., 2024a), or control writing through graph-memory weights (Xia et al., 2025). Some methods first induce and compress artifacts based on similarity, error boundaries, or trajectory redundancy, and then write shorter entries, such as structured reasoning experience (Ouyang et al., 2025b), information-extraction artifacts (Pang et al., 2023), self-correction artifacts (Askari et al., 2024), software-repair experience (Chen et al., 2025), and tool-use experience (Xiao et al., 2025).

Regardless of writing strategy, systems usually store source, applicable conditions, tags, scores, timestamps, invocation counts, success/failure statistics, or versions as metadata for later merging, deletion, and updating.

5.3 Deletion

Deletion and pruning control library size and reduce noise. One class of deletion is based on similarity and redundancy. When new and old artifacts are semantically similar, cover the same error boundary, or correspond to the same skill, systems merge duplicate content and remove covered entries (Pang et al., 2023; Ouyang et al., 2025b; Xiao et al., 2025).

Another class targets low utility and capacity constraints, such as eliminating low-utility experience according to retrieval success rate and execution gain (Tang et al., 2025), maintaining retrievable memory based on recency and importance (Park et al., 2023), reducing the weights of old memories using time decay and user relevance (Zhong et al., 2024), or deleting procedural memories based on non-positive gains, semantic duplication, and capacity limits (Mi et al., 2026). In code, skill, and tool-use settings, criteria are more direct: systems remove non-executable skills, failed skills, or persistently low-reward skills (Yang et al., 2026; Xia et al., 2026). Long-term memory methods more often adopt soft deletion, reducing the influence of old experience based on low usage, low relevance (Zhou et al., 2025), or dynamic utility decay (Zhang et al., 2025b). By contrast, eliminating low-scoring prompt candidates in prompt optimization (Yang et al., 2024b; Pryzant et al., 2023; Fernando et al., 2023) is closer to relative selection during search and should not be directly equated with deletion in a long-term artifact library.

5.4 Modification

Modification improves artifact-library quality. The lightest form of updating adjusts scores, invocation counts, success/failure statistics, recency, confidence, or rewards based on usage results. Systems may update procedural memories according to online scores (Mi et al., 2026), update experience utility based on cross-framework transfer effects (Tang et al., 2025), or update tool-capability memory based on tool-use outcomes (Xiao et al., 2025). For code and skill artifacts, modification is often driven by execution failures, test results, or

version performance; outcomes may include code debugging (Wang et al., 2023a), tool-generation repair (Qiu et al., 2025b,a), skill-version updates, or rollbacks (Yang et al., 2026; Zhang et al., 2026c). When new artifacts are related to but not fully redundant with old ones, systems may merge old artifacts with recent feedback based on semantic similarity or coverage relations (Askari et al., 2024; Qiu et al., 2026). Systems may also narrow or expand artifact scope in response to misuse, negative transfer, or mismatched applicability, such as by refining skill applicability conditions (Zheng et al., 2025), merging multi-round modifications into experience (Cai et al., 2025a), or updating memory metadata (Qiao et al., 2026).

6 Artifact Usage

Artifact usage concerns how $\mathcal{G}$ affects new tasks (Eq. 3). Given a new input, the system must decide *which* artifacts to invoke and *how* they should influence model inference.

6.1 Artifact Retrieval

Artifact retrieval identifies usable artifacts g from the library for the current task. Based on whether the retrieval query is actively constructed during task execution, existing methods can be broadly divided into static and dynamic retrieval.

Static retrieval directly uses the current input, state, or task context x as the retrieval condition. It is common when artifact applicability is relatively stable. If artifacts contain explicit fields such as task labels, trigger conditions, tool names, or skill descriptions, systems can use symbolic or metadata matching, e.g., matching skills by description (Yang et al., 2026), selecting procedural memories by applicable conditions (Mi et al., 2026), or invoking code skills by functionality (Wang et al., 2023a). For natural-language experience, reflections, or trajectory fragments, semantic vector retrieval is more common, such as retrieving long-term dialogue memories (Zhong et al., 2024; Chhikara et al., 2025), similar historical cases (Zhou et al., 2025), or web-task trajectories (Kim et al., 2024a). When the library has a hierarchical, graph-based, or modular dependency structure, structural relations can further locate relevant subprocesses or composable skills (Zhang et al., 2025a; Han et al., 2025; Fan et al., 2025; Liang et al., 2026b).

Dynamic retrieval does not treat x as a fixed query. Instead, an LLM or agent constructs a query q according to the current goal, invokes one or more retrievers, and may rewrite the query, expand the search scope, or filter returned content based on intermediate results. Such retrieval can dynamically schedule long- and short-term memories (Yu et al., 2026), select and reorganize context components (Zhang et al., 2026d), or retrieve memories by jointly considering relevance, recency, and importance (Park et al., 2023). In general-purpose agents, dynamic retrieval is often combined with cross-session experience retrieval (Liu et al., 2025a), reflection selection (Zhao et al., 2023), cross-domain knowledge-base retrieval (Tang et al., 2025), and skill-graph retrieval (Huang and Huang, 2025).

6.2 Artifact Injection

Artifact injection concerns when, where, and in what form retrieved artifacts affect the model. **Pre-task injection** places artifacts into the system prompt, task instruction, or retrieved context before generation, so that they serve as global constraints or guidance, as in prompt optimization, reasoning, and self-correction methods (Pryzant et al., 2023; Yang et al., 2024b; Guo et al., 2023; Askari et al., 2024; Zhao et al., 2023; Ouyang et al., 2025b).

In-execution injection selects artifacts according to the current observation, state, or subgoal before each decision, supporting state-conditioned constraints, case-based references, and procedural guidance in interactive tasks (Fu et al., 2024; Liu et al., 2025a; Zheng et al., 2023; Kim et al., 2024a; Zhou et al., 2025; Mi et al., 2026; Wang et al., 2024c; Han et al., 2025; Fan et al., 2025). For executable artifacts, usage may bypass textual insertion: experience is encapsulated as functions, scripts, tools, or programmatic skills that the model can directly invoke (Wang et al., 2023a, 2025c; Qiu et al., 2025b,a; Liang et al., 2025). Thus, artifact usage ranges from textual constraints and case references to procedural guidance and executable calls.

7 Applications of Textualized Learning

Based on existing experimental settings, applications of Textualized Learning can be grouped into seven categories: reasoning, programming, web tasks, embodied decision-making, planning, dialogue, and domain expertise. The main text summarizes their task requirements, while specific datasets and benchmarks are listed in Appendix E.

Reasoning tasks require artifacts that encode solution patterns, error types, evidence-integration strategies, and reasoning constraints, thereby supporting more structured and generalizable inference (Hendrycks et al., 2021b; Yang et al., 2018). Programming and tool-use tasks rely more heavily on verifiable feedback, with artifacts capturing code-repair experience, repository localization, interface constraints, tool-selection rules, and parameter-construction templates (Jimenez et al., 2024; Xiao et al., 2025). Web tasks, including navigation, shopping, browser operation, and GUI control, involve dynamic multi-step interactions, where artifacts typically encode page states, user constraints, interactive elements, action ordering, and failure-recovery strategies (Deng et al., 2023; Zhou et al., 2023; Fu et al., 2024). Embodied and planning tasks emphasize state-conditioned decision-making, including environmental regularities, object and spatial relations, subgoal decomposition, action sequences, constraint checking, resource allocation, collaboration, and replanning (Shridhar et al., 2021; Wang et al., 2022; Xie et al., 2026, 2024a; Liu et al., 2023b; Cai et al., 2025b). Dialogue tasks focus on cross-session memory, user preferences, clarification seeking, and factual or stylistic consistency (Maharana et al., 2024; Wu et al., 2024a; Zhong et al., 2024). Domain-specific expert tasks prioritize correctness, safety, and auditability, requiring artifacts to encode diagnostic procedures, business rules, risk boundaries, formula applicability, and experimental constraints (Arora et al., 2025; Wang et al., 2025a).

8 Challenges and Prospects

Although TL has shown promise across multiple tasks, the field remains at an early stage and raises several open questions.

The first challenge is the lack of unified benchmarks. Most existing work is evaluated in task-specific settings, making direct comparison difficult. This issue is especially salient for agent-related tasks, where environment configurations vary widely and methods evaluated on the same nominal task may still be incomparable. A unified benchmark covering multiple tasks and enabling controlled comparisons among learning methods would help advance the field.

In the generation stage, generated artifacts can be viewed as analogous to gradients, but current methods are primarily based on prompt engineering and lack convergence analyses or guarantees. Theoretical analysis along this direction could deepen our understanding of TL. In addition, the choice of sample set, or "batch," for artifact generation may substantially affect artifact usefulness. Future work may draw on curriculum learning (Bengio et al., 2009), example selection in in-context learning (Su et al., 2023), and automatic prompt search or compiled optimization (Pryzant et al., 2023; Agrawal et al., 2025; Spiess et al., 2025).

In the organization stage, artifact evaluation is crucial. Managing a large artifact library requires estimating the effect of individual artifacts, but evaluating the library as a whole remains challenging. The minimum description length (MDL) principle (Grünwald, 2007) provides an information-theoretic framework for rule quality: using the shortest description to cover the most data. This principle may offer a promising direction.

More broadly, TL and parametric learning may be complementary. Frequently used artifacts may be internalized into model parameters, while occasional or dynamic knowledge may remain in external libraries. How to automatically decide which artifacts are worth distilling into the model remains an open problem. Future work may also draw inspiration from cognitive science and educational psychology, such as the generation effect and spaced repetition.

9 Conclusion

Overall, Textualized Learning has the potential to become an important infrastructure for long-term learning agents. By combining experience memory, automatic verification, artifact-library management, and policy optimization, future agents may continuously accumulate experience through interaction and transform it into reusable and maintainable task capabilities.

Limitations

This paper has several limitations. First, because this line of work is named differently across communities, we can only make our best effort to cover relevant studies. Future versions should provide a more explicit account of the literature search and selection process.

Second, this paper summarizes Textualized Learning as a three-stage process. In practice, some systems may involve more complex work-

flows. Our abstraction is intended to capture the core ideas of most methods rather than exhaustively describe every implementation detail.

References

Eshaan Agarwal, Joykirat Singh, Vivek Dani, Raghav Magazine, Tanuja Ganu, and Akshay Nambi. 2024. Promptwizard: Task-aware prompt optimization framework. *Preprint*, arXiv:2405.18369.

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. 2025. Gepa: Reflective prompt evolution can outperform reinforcement learning. *Preprint*, arXiv:2507.19457.

Wasi Uddin Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. 2023. AVATAR: A parallel corpus for java-python program translation. *Preprint*, arXiv:2308.08227.

Afra Feyza Akyurek, Advait Gosai, Chen Bo Calvin Zhang, Vipul Gupta, Jaehwan Jeong, Anisha Gunjal, Tahseen Rabbani, Maria Mazzone, David Randolph, Mohammad Mahmoudi Meymand, et al. 2025. PRBench: Large-scale expert rubrics for evaluating high-stakes professional reasoning. *Preprint*, arXiv:2511.11562.

David L. Applegate, Robert E. Bixby, Vašek Chvátal, and William J. Cook. 2006. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press.

Rahul K. Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero-Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, Johannes Heidecke, and Karan Singhal. 2025. HealthBench: Evaluating large language models towards improved human health. *Preprint*, arXiv:2505.08775.

Arian Askari, Christian Poelitz, and Xinye Tang. 2024. Magic: Generating self-correction guideline for in-context text-to-sql. *Preprint*, arXiv:2406.12692.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *Preprint*, arXiv:2108.07732.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *International Conference on Machine Learning*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, et al. 2025. AgiBot World Colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *Preprint*, arXiv:2503.06669.

Bruce G. Buchanan and Edward H. Shortliffe. 1984. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, Reading, MA.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ: A large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026.

Yuzheng Cai, Siqi Cai, Yuchen Shi, Zihan Xu, Lichao Chen, Yulei Qin, Xiaoyu Tan, Gang Li, Zongyi Li, Haojia Lin, Yong Mao, Ke Li, and Xing Sun. 2025a. Training-free group relative policy optimization. *Preprint*, arXiv:2510.08191.

Zhicheng Cai, Xinyuan Guo, Yu Pei, JiangTao Feng, Jiangjie Chen, Ya-Qin Zhang, Wei-Ying Ma, Mingxuan Wang, and Hao Zhou. 2025b. Flex: Continuous agent evolution via forward learning from experience. *Preprint*, arXiv:2511.06449.

Zouying Cao, Jiaji Deng, Li Yu, Weikang Zhou, Zhaoyang Liu, Bolin Ding, and Hai Zhao. 2025. Remember me, refine me: A dynamic procedural memory framework for experience-driven agent evolution. *Preprint*, arXiv:2512.10696.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Shiqi Chen, Jingze Gai, Ruochen Zhou, Jinghan Zhang, Tongyao Zhu, Junlong Li, Kangrui Wang, Zihan Wang, Zhengyu Chen, Klara Kaleb, Ning Miao, Siyang Gao, Cong Lu, Manling Li, Junxian He, and Yee Whye Teh. 2026. Skillcraft: Can llm agents learn to use tools skillfully? *Preprint*, arXiv:2603.00718.

Silin Chen, Shaoxin Lin, Xiaodong Gu, Yuling Shi, Heng Lian, Longfei Yun, Dong Chen, Weiguo Sun, Lin Cao, and Qianxiang Wang. 2025. Swe-exp: Experience-driven software issue resolution. *Preprint*, arXiv:2507.23361.

Jiale Cheng, Xiao Liu, Kehan Zheng, Pei Ke, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. 2023. Black-box prompt optimization: Aligning large language models without model training. *Preprint*, arXiv:2311.04155.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. 2019. BabyAI: First steps towards grounded language learning with a human in the loop. In *International Conference on Learning Representations (ICLR)*.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2Web: Towards a generalist agent for the web. *Preprint*, arXiv:2306.06070.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2024. Chain-of-verification reduces hallucination in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3563–3578, Bangkok, Thailand. Association for Computational Linguistics.

Ruomeng Ding, Wei Cheng, Minglai Shao, and Chen Zhao. 2025. Skillgen: Learning domain skills for in-context sequential decision making. *Preprint*, arXiv:2511.14670.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, and Zhifang Sui. 2024. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Alexandre Drouin, Maxime Gasse, Lucas Caccia, Issam H. Laradji, Michele Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. 2024. WorkArena: How capable are web agents at solving common knowledge work tasks? *Preprint*, arXiv:2403.07718.

Pengfei Du. 2026. Memory for autonomous llm agents: Mechanisms, evaluation, and emerging frontiers. *Preprint*, arXiv:2603.07670.

Shengda Fan, Xin Cong, Zhong Zhang, Yuepeng Fu, Yesai Wu, Hao Wang, Xinyu Zhang, Enrui Hu, and Yankai Lin. 2025. Generalizing experience for language agents with hierarchical metaflows. In *Advances in Neural Information Processing Systems*.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. 2025. Memp: Exploring agent procedural memory. *Preprint*, arXiv:2508.06433.

Arsene Fansi Tchango, Rishab Goel, Zhi Wen, Julien Martel, and Joumana Ghosn. 2022. DDXPlus: A new dataset for automatic medical diagnosis. In *Advances in Neural Information Processing Systems*.

Edward A. Feigenbaum. 1977. The art of artificial intelligence- themes and case studies of knowledge engineering. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 1014–1029.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2023. Promptbreeder: Self-referential self-improvement via prompt evolution. *Preprint*, arXiv:2309.16797.

Saman Forouzandeh, Wei Peng, Parham Moradi, Xinghuo Yu, and Mahdi Jalili. 2025. Learning hierarchical procedural memory for llm agents through bayesian selection and contrastive refinement. *Preprint*, arXiv:2512.18950.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. 2024. Autoguide: Automated generation and selection of context-aware guidelines for large language model agents. *Preprint*, arXiv:2403.08978.

Hiroki Furuta, Yutaka Matsuo, Aleksandra Faust, and Izzeddin Gur. 2024. Exposing limitations of language model agents in sequential-task compositions on the web. In *International Conference on Learning Representations (ICLR)*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Peter D. Grünwald. 2007. *The Minimum Description Length Principle*. MIT Press.

Lukas Guertler et al. 2025. TextArena. `https://github.com/LeonGuertler/TextArena`. Benchmark for multi-turn text-based agent environments.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2023. Evoprompt: Connecting llms with evolutionary algorithms yields powerful prompt optimizers. *Preprint*, arXiv:2309.08532.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*.

Dongge Han, Camille Couturier, Daniel Madrigal Diaz, Xuchao Zhang, Victor Rühle, and Saravan Rajmohan. 2025. Legomem: Modular procedural memory for multi-agent llm systems for workflow automation. *Preprint*, arXiv:2510.04851.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024a. WebVoyager: Building an end-to-end web agent with large multimodal models. *Preprint*, arXiv:2401.13919.

Xuehai He, Yichen Zhang, Luntian Mou, Eric Xing, and Pengtao Xie. 2020. PathVQA: 30000+ questions for medical visual question answering. *Preprint*, arXiv:2003.10286.

Zihong He, Weizhe Lin, Hao Zheng, Fan Zhang, Matt W. Jones, Laurence Aitchison, Xuhai Xu, Miao Liu, Per Ola Kristensson, and Junxiao Shen. 2024b. Human-inspired perspectives: A survey on ai long-term memory. *Preprint*, arXiv:2411.00489.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 33–38.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, et al. 2025. Memory in the age of AI agents. *arXiv preprint arXiv:2512.13564*.

Ken Huang and Jerry Huang. 2025. Audited skill-graph self-improvement for agentic llms via verifiable rewards, experience synthesis, and continual memory. *Preprint*, arXiv:2512.23760.

Wei-Chieh Huang, Weizhi Zhang, Yueqing Liang, Yuanchen Bei, Yankai Chen, Tao Feng, Xinyu Pan, Zhen Tan, Yu Wang, Tianxin Wei, Shanglin Wu, Ruiyao Xu, Liangwei Yang, Rui Yang, Wooseong Yang, Chin-Yuan Yeh, Hanrong Zhang, Haozhen Zhang, Siqi Zhu, Henry Peng Zou, Wanjia Zhao, Song Wang, Wujiang Xu, Zixuan Ke, Zheng Hui, Dawei Li, Yaozu Wu, Langzhou He, Chen Wang, Xiongxiao Xu, Baixiang Huang, Juntao Tan, Shelby Heinecke, Huan Wang, Caiming Xiong, Ahmed A. Metwally, Jun Yan, Chen-Yu Lee, Hanqing Zeng, Yinglong Xia, Xiaokai Wei, Ali Payani, Yu Wang, Haitong Ma, Wenya Wang, Chenguang Wang, Yu Zhang, Xin Wang, Yongfeng Zhang, Jiaxuan You, Hanghang Tong, Xiao Luo, Xue Liu, Yizhou Sun, Wei Wang, Julian McAuley, James Zou, Jiawei Han, Philip S. Yu, and Kai Shu. 2026. Rethinking memory mechanisms of foundation agents in the second half: A survey. *arXiv preprint arXiv:2602.06052*.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *International Conference on Learning Representations (ICLR)*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. *Preprint*, arXiv:2403.07974.

Hanqi Jiang, Junhao Chen, Yi Pan, Ling Chen, Weihang You, Yifan Zhou, Ruidong Zhang, Lin Zhao, Yohannes Abate, and Tianming Liu. 2026. Synapse: Empowering llm agents with episodic-semantic memory via spreading activation. *Preprint*, arXiv:2601.02744.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. 2024. SWE-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations (ICLR)*.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Sarunas Kalade and Graham Schelle. 2025. NPUEval: Optimizing NPU kernels with LLMs and open source compilers. *Preprint*, arXiv:2507.14403.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2024. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*.

Minsoo Kim, Victor Bursztyn, Eunyee Koh, Shunan Guo, and Seung-won Hwang. 2024a. Rada: Retrieval-augmented web agent planning with llms. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13511–13525, Bangkok, Thailand. Association for Computational Linguistics.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Ilya Lampe, Pannag Sanketi, et al. 2024b. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning (CoRL)*.

Namyoung Kim, Kai Tzu-iunn Ong, Yeonjun Hwang, Minseok Kang, Iiseo Jihn, Gayoung Kim, Minju Kim, and Jinyoung Yeo. 2025. Principles: Synthetic strategy memory for proactive dialogue agents. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 21329–21368, Suzhou, China. Association for Computational Linguistics.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. 2024. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. *Preprint*, arXiv:2401.13649.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. 2023. Openassistant conversations: Democratizing large language model alignment. In *Advances in Neural Information Processing Systems*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Andrew Kyle Lampinen, Martin Engelcke, Yuxuan Li, Arslan Chaudhry, and James L. McClelland. 2025. Latent learning: Episodic memory complements parametric learning by enabling flexible reuse of experiences. *Preprint*, arXiv:2509.16189.

Chris Latimer, Nicoló Boschi, Andrew Neeser, Chris Bartholomew, Gaurav Srivastava, Xuan Wang, and Naren Ramakrishnan. 2025. Hindsight is 20/20: Building agent memory that retains, recalls, and reflects. *Preprint*, arXiv:2512.12818.

LeetCode. 2026. LeetCode. `https://leetcode.com/`. Accessed 2026-05-20.

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. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. 2023a. Can LLM already serve as a database interface? a BIg bench for large-scale database grounded text-to-SQLs. In *Advances in Neural Information Processing Systems*.

Kai Li, Xuanqing Yu, Ziyi Ni, Yi Zeng, Yao Xu, Zheqing Zhang, Xin Li, Jitao Sang, Xiaogang Duan, Xuelei Wang, Chengbao Liu, and Jie Tan. 2026a. TiMem: Temporal-hierarchical memory consolidation for long-horizon conversational agents. *Preprint*, arXiv:2601.02845.

Minghao Li, Feifan Song, Bowen Yu, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023b. API-Bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3102–3116.

Sijia Li, Yuchen Huang, Zifan Liu, Zijian Li, Jingjing fu, Lei Song, Jiang Bian, Jun Zhang, and Rui Wang. 2025a. Experience-evolving multi-turn tool-use agent with hybrid episodic-procedural memory. *Preprint*, arXiv:2512.07287.

Wenliang Li, Rui Yan, Xu Zhang, Li Chen, Hongji Zhu, Jing Zhao, Junjun Li, Mengru Li, Wei Cao, Zihang Jiang, Wei Wei, Kun Zhang, and Shaohua Kevin Zhou. 2025b. Macd: Multi-agent clinical diagnosis with self-learned knowledge for llm. *Preprint*, arXiv:2509.20067.

Xiaoyuan Li, Moxin Li, Keqin Bao, Yubo Ma, Wenjie Wang, Dayiheng Liu, and Fuli Feng. 2026b. Skillgraph: Skill-augmented reinforcement learning for agents via evolving skill graphs. *arXiv preprint arXiv:2605.12039*.

Sirui Liang, Pengfei Cao, Jian Zhao, Wenhao Teng, Xiangwen Liao, Jun Zhao, and Kang Liu. 2026a. Learning how to remember: A meta-cognitive management method for structured and transferable agent memory. *Preprint*, arXiv:2601.07470.

Xuechen Liang, Meiling Tao, Yinghui Xia, Jianhui Wang, Kun Li, Yijin Wang, Yangfan He, Jingsong Yang, Tianyu Shi, Yuantao Wang, Miao Zhang, and Xueqian Wang. 2025. Sage: Self-evolving agents with reflective and memory-augmented abilities. *Neurocomputing*.

Yuan Liang, Ruobin Zhong, Haoming Xu, Chen Jiang, Yi Zhong, Runnan Fang, Jia-Chen Gu, et al. 2026b. Skillnet: Create, evaluate, and connect ai skills. *Preprint*, arXiv:2603.04448.

Zehao Lin, Chunyu Li, and Kai Chen. 2026. A survey on the security of long-term memory in llm agents: Toward mnemonic sovereignty. *Preprint*, arXiv:2604.16548.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word

problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. 2023a. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. *Preprint*, arXiv:2306.03310.

Genglin Liu, Shijie Geng, Sha Li, Hejie Cui, Sarah Zhang, Xin Liu, and Tianyi Liu. 2025a. Webcoach: Self-evolving web agents with cross-session memory guidance. *Preprint*, arXiv:2511.12997.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2023b. AgentBench: Evaluating LLMs as agents. *Preprint*, arXiv:2308.03688.

Yitao Liu, Chenglei Si, Karthik Narasimhan, and Shunyu Yao. 2025b. Contextual experience replay for self-improvement of language agents. *Preprint*, arXiv:2506.06698.

Zeyuan Liu, Jeonghye Kim, Xufang Luo, Dongsheng Li, and Yuqing Yang. 2026. Exploratory memory-augmented llm agent via hybrid on- and off-policy optimization. *Preprint*, arXiv:2602.23008.

Lefteris Loukas, Manos Fergadiotis, Ilias Chalkidis, Eirini Spyropoulou, Prodromos Malakasiotis, Ion Androutsopoulos, and Georgios Paliouras. 2022. FiNER: Financial numeric entity recognition for XBRL tagging. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4419–4431.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Felix Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, et al. 2024. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. *Preprint*, arXiv:2408.04682.

Jinghao Luo, Yuchen Tian, Chuxue Cao, Ziyang Luo, Hongzhan Lin, Kaixin Li, Chuyi Kong, Ruichao Yang, and Jing Ma. 2026. From storage to experience: A survey on the evolution of llm agent memory mechanisms. *Preprint*, arXiv:2605.06716.

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. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of LLM agents. *Preprint*, arXiv:2402.17753.

Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, Singapore. Association for Computational Linguistics.

Mathematical Association of America. 2026. American invitational mathematics examination (AIME). https://maa.org/math-competitions/american-invitational-mathematics-examination-aime/. Accessed 2026-05-20.

Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. 1998. PDDL: The planning domain definition language. *Yale Center for Computational Vision and Control Technical Report*.

Qirui Mi, Zhijian Ma, Mengyue Yang, Haoxuan Li, Yisen Wang, Haifeng Zhang, and Jun Wang. 2026. Procmem: Learning reusable procedural memory from experience via non-parametric ppo for llm agents. *Preprint*, arXiv:2602.01869.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Tom M. Mitchell. 1997. *Machine Learning*. McGraw-Hill.

Hadi Nekoei, Aman Jaiswal, Patrice Bechard, Oleh Shliazhko, Orlando Marquez Ayala, Mathieu Reymond, Massimo Caccia, Alexandre Drouin, Sarath Chandar, and Alexandre Lacoste. 2025. Jef-hinter: Leveraging offline knowledge for improving web agents adaptation. *Preprint*, arXiv:2510.04373.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Xiaoxi Jiang, and Guanjun Jiang. 2026. Trace2skill: Distill trajectory-local lessons into transferable agent skills. *Preprint*, arXiv:2603.25158.

Pascal Notin, Alexander Kollasch, Daniel Ritter, Lood van Niekerk, Steffanie Paul, Hannes Spinner, Nathan Rollins, Asa Shaw, Rachel Weitzman, Jonathan Frazer, et al. 2023. ProteinGym: Large-scale benchmarks for protein fitness prediction and design. In *Advances in Neural Information Processing Systems*.

OpenAI. 2024. SWE-bench Verified. https://openai.com/index/introducing-swe-bench-verified/. Accessed 2026-05-20.

Anne Ouyang, Simon Guo, Rahul Arora, Alex Zhang, William Hu, Christopher Re Zhang, Jonathan Ragan-Kelley, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. 2025a. KernelBench: Can LLMs write efficient GPU kernels? *Preprint*, arXiv:2502.10517.

Siru Ouyang, Jun Yan, Yanfei Chen, Rujun Han, Zifeng Wang, Bhavana Dalvi Mishra, Rui Meng, Chun-Liang Li, Yizhu Jiao, Kaiwen Zha, Maohao Shen, Vishy Tirumalashetty, George Lee, Jiawei Han, Tomas Pfister, and Chen-Yu Lee. 2026. SkillOS: Learning skill curation for self-evolving agents. *Preprint*, arXiv:2605.06614.

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

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. 2024. Automatically correcting large language models: Surveying the landscape of diverse automated correction strategies. *Transactions of the Association for Computational Linguistics*, 12:484–506.

Wenbo Pan, Shujie Liu, Xiangyang Zhou, Shiwei Zhang, Wanlu Shi, Mirror Xu, and Xiaohua Jia. 2026. M*: Every task deserves its own memory harness. *Preprint*, arXiv:2604.11811.

Chaoxu Pang, Yixuan Cao, Qiang Ding, and Ping Luo. 2023. Guideline learning for in-context information extraction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15372–15389, Singapore. Association for Computational Linguistics.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23, pages 1–22, New York, NY, USA. Association for Computing Machinery.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 48371–48392. PMLR.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. 2022. Grips: Gradient-free, edit-based instruction search for prompting large language models. *Preprint*, arXiv:2203.07281.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968, Singapore. Association for Computational Linguistics.

Ruchir Puri, David S. Kung, Geert Janssen, Wei Zhang, Giacomo Domeniconi, Vladimir Zolotov, Julian Dolby, Jie Chen, Mihir Choudhury, Lindsey Decker, et al. 2021. CodeNet: A large-scale AI for code dataset for learning a diversity of coding tasks. *Preprint*, arXiv:2105.12655.

Jingyang Qiao, Weicheng Meng, Yu Cheng, Zhihang Lin, Zhizhong Zhang, Xin Tan, Jingyu Gong, Kun Shao, and Yuan Xie. 2026. Memory intelligence agent. *Preprint*, arXiv:2604.04503.

Yujia Qin, Shengding Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations (ICLR)*.

Jiahao Qiu, Xuan Qi, Hongru Wang, Xinzhe Juan, Yimin Wang, Zelin Zhao, Jiayi Geng, Jiacheng Guo, Peihang Li, Jingzhe Shi, Shilong Liu, and Mengdi Wang. 2025a. Alita-g: Self-evolving generative agent for agent generation. *Preprint*, arXiv:2510.23601.

Jiahao Qiu, Xuan Qi, Tongcheng Zhang, Xinzhe Juan, Jiacheng Guo, Yifu Lu, Yimin Wang, Zixin Yao, Qihan Ren, Xun Jiang, Xing Zhou, Dongrui Liu, Ling Yang, Yue Wu, Kaixuan Huang, Shilong Liu, Hongru Wang, and Mengdi Wang. 2025b. Alita: Generalist agent enabling scalable agentic reasoning with minimal predefinition and maximal self-evolution. *Preprint*, arXiv:2505.20286.

Libin Qiu, Zhirong Gao, Junfu Chen, Yuhang Ye, Weizhi Huang, Xiaobo Xue, Wenkai Qiu, and Shuo Tang. 2026. Autorefine: From trajectories to reusable expertise for continual llm agent refinement. *Preprint*, arXiv:2601.22758.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2024. From exploration to mastery: Enabling llms to master tools via self-driven interactions. *Preprint*, arXiv:2410.08197.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. GPQA: A graduate-level google-proof q&a benchmark. *Preprint*, arXiv:2311.12022.

Baochang Ren, Yunzhi Yao, Rui Sun, Shuofei Qiao, Ningyu Zhang, and Huajun Chen. 2026. Aligning agentic world models via knowledgeable experience learning. *Preprint*, arXiv:2601.13247.

Alireza Rezazadeh, Zichao Li, Wei Wei, and Yujia Bao. 2025. From isolated conversations to hierarchical schemas: Dynamic tree memory representation for LLMs. In *International Conference on Learning Representations (ICLR)*.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana

Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. 2019. Habitat: A platform for embodied AI research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9339–9347.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. 2017. World of bits: An open-domain platform for web-based agents. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3135–3144.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10740–10749.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations (ICLR)*.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*.

Claudio Spiess, Mandana Vaziri, Louis Mandel, and Martin Hirzel. 2025. AutoPDL: Automatic prompt optimization for LLM agents. In *Proceedings of the Fourth International Conference on Automated Machine Learning*, volume 293 of *Proceedings of Machine Learning Research*, pages 13/1–13/20.

Larry R Squire. 2004. Memory systems of the brain: a brief history and current perspective. *Neurobiology of learning and memory*, 82(3):171–177.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Selective annotation makes language models better few-shot learners. In *International Conference on Learning Representations*.

Xuanbo Su, Yingfang Zhang, Hao Luo, Xiaoteng Liu, and Leo Huang. 2025. Mistake notebook learning: Batch-clustered failures for training-free agent adaptation. *Preprint*, arXiv:2512.11485.

Haoran Sun and Shaoning Zeng. 2025. Hierarchical memory for high-efficiency long-term reasoning in llm agents. *arXiv preprint arXiv:2507.22925*.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. 2020. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9229–9248. PMLR.

Zeyi Sun, Ziyu Liu, Yuhang Zang, Yuhang Cao, Xiaoyi Dong, Tong Wu, Dahua Lin, and Jiaqi Wang. 2025. Seagent: Self-evolving computer use agent with autonomous learning from experience. *Preprint*, arXiv:2508.04700.

Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*, 2 edition. MIT Press.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-Bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. 2025. Dynamic cheatsheet: Test-time learning with adaptive memory. *Preprint*, arXiv:2504.07952.

Yunteng Tan, Zhi Gao, and Xinxiao Wu. 2026. Enhancing web agents with a hierarchical memory tree. *Preprint*, arXiv:2603.07024.

Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, Peng Xia, Fang Wu, He Zhu, Ge Zhang, Jiaheng Liu, Xingyao Wang, Sirui Hong, Chenglin Wu, Hao Cheng, Chi Wang, and Wangchunshu Zhou. 2025. Agent KB: Leveraging cross-domain experience for agentic problem solving. *Preprint*, arXiv:2507.06229.

Zhenheng Tang, Xin He, and Tiancheng Zhao. 2026. Llm agent memory: A survey from a unified representation–management perspective. OpenReview preprint.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: A large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 809–819.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. 2024. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16022–16076.

Endel Tulving. 1972. Episodic and semantic memory. *Organization of memory*.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. ACE 2005 multilingual training corpus.

Chenxi Wang, Zhuoyun Yu, Xin Xie, Wuguannan Yao, Runnan Fang, Shuofei Qiao, Kexin Cao, Guozhou Zheng, Xiang Qi, Peng Zhang, and Shumin Deng. 2026. SkillX: Automatically constructing skill knowledge bases for agents. *Preprint*, arXiv:2604.04804.

Dannong Wang, Jaisal Patel, Daochen Zha, Steve Y. Yang, and Xiao-Yang Liu. 2025a. FinLoRA: Benchmarking LoRA methods for fine-tuning LLMs on financial datasets. *Preprint*, arXiv:2505.19819.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. 2021. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. Voyager: An open-ended embodied agent with large language models. In *Advances in Neural Information Processing Systems*.

Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. 2025b. Reinforcement learning for self-improving agent with skill library. *Preprint*, arXiv:2512.17102.

Ruoyao Wang, Peter A. Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11279–11298.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024a. Executable code actions elicit better LLM agents. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 50208–50232. PMLR.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P. Xing, and Zhiting Hu. 2023b. Promptagent: Strategic planning with language models enables expert-level prompt optimization. *Preprint*, arXiv:2310.16427.

Xuewei Wang, Weiyan Shi, Richard Kim, Yoojung Oh, Sijia Yang, Jingwen Zhang, and Zhou Yu. 2019. Persuasion for good: Towards a personalized persuasive dialogue system for social good. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5635–5649.

Zilong Wang, Yuedong Cui, Li Zhong, Zimin Zhang, Da Yin, Bill Yuchen Lin, and Jingbo Shang. 2024b. OfficeBench: Benchmarking language agents across multiple applications for office automation. *Preprint*, arXiv:2407.19056.

Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. 2025c. Inducing programmatic skills for agentic tasks. *Preprint*, arXiv:2504.06821.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2024c. Agent workflow memory. *Preprint*, arXiv:2409.07429.

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. 2025. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *Preprint*, arXiv:2511.20857.

David Weininger. 1988. SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2024a. LongMemEval: Benchmarking chat assistants on long-term interactive memory. *Preprint*, arXiv:2410.10813.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, and Botian Shi. 2025a. Evolver: Self-evolving llm agents through an experience-driven lifecycle. *Preprint*, arXiv:2510.16079.

Shaokai Wu, Yanbiao Ji, Qiuchang Li, Zhiyi Zhang, Qichen He, Wenyuan Xie, Guodong Zhang, Bayram Bayramli, Yue Ding, and Hongtao Lu. 2025b. Dejavu: Towards experience feedback learning for embodied intelligence. *Preprint*, arXiv:2510.10181.

Yaxiong Wu, Sheng Liang, Chen Zhang, Yichao Wang, Yongyue Zhang, Huifeng Guo, Ruiming Tang, and Yong Liu. 2025c. From human memory to ai memory: A survey on memory mechanisms in the era of llms. *Preprint*, arXiv:2504.15965.

Zhenyu Wu, Qingkai Zeng, Zhihan Zhang, Zhaoxuan Tan, Chao Shen, and Meng Jiang. 2024b. Large language models can self-correct with key condition verification. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12846–12867, Miami, Florida, USA. Association for Computational Linguistics.

Zijun Wu, Yongchang Hao, and Lili Mou. 2026. Tokmem: One-token procedural memory for large language models. In *International Conference on Learning Representations (ICLR)*.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. 2026. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *Preprint*, arXiv:2602.08234.

Siyu Xia, Zekun Xu, Jiajun Chai, Wentian Fan, Yan Song, Xiaohan Wang, Guojun Yin, Wei Lin, Haifeng Zhang, and Jun Wang. 2025. From experience to strategy: Empowering llm agents with trainable graph memory. *Preprint*, arXiv:2511.07800.

Yunzhong Xiao, Yangmin Li, Hewei Wang, Yunlong Tang, and Zora Zhiruo Wang. 2025. Toolmem: Enhancing multimodal agents with learnable tool capability memory. *Preprint*, arXiv:2510.06664.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. 2024a. TravelPlanner: A benchmark for real-world planning with language agents. *Preprint*, arXiv:2402.01622.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J. Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. 2024b. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*.

Zhengwei Xie, Zhisheng Chen, Ziyan Weng, Tingyu Wu, Chenglong Li, Vireo Zhang, and Kun Wang. 2026. Steve-evolving: Open-world embodied self-evolution via fine-grained diagnosis and dual-track knowledge distillation. *Preprint*, arXiv:2603.13131.

Yiming Xiong, Shengran Hu, and Jeff Clune. 2026. Learning to continually learn via meta-learning agentic memory designs. *Preprint*, arXiv:2602.07755.

Frank F. Xu et al. 2024. The AgentCompany: Benchmarking LLM agents on consequential real world tasks. *Preprint*, arXiv:2412.14161.

Renjun Xu and Yang Yan. 2026. Agent skills for large language models: Architecture, acquisition, security, and the path forward. *Preprint*, arXiv:2602.12430.

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

Chen Yang, Chenyang Zhao, Quanquan Gu, and Dongruo Zhou. 2024a. Cops: Empowering llm agents with provable cross-task experience sharing. *Preprint*, arXiv:2410.16670.

Cheng Yang, Xuemeng Yang, Licheng Wen, Daocheng Fu, Jianbiao Mei, Rong Wu, Pinlong Cai, Yufan Shen, Nianchen Deng, Botian Shi, Yu Qiao, and Haifeng Li. 2025a. Learning on the job: An experience-driven self-evolving agent for long-horizon tasks. *Preprint*, arXiv:2510.08002.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2024b. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*.

Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. 2023. Fingpt: Open-source financial large language models. *arXiv preprint arXiv:2306.06031*. Accepted by the FinLLM Symposium at IJCAI 2023.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R. Narasimhan, and Ofir Press. 2024c. Swe-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*.

Wei Yang, Jinwei Xiao, Hongming Zhang, Qingyang Zhang, Yanna Wang, and Bo Xu. 2025b. Coarse-to-fine grounded memory for llm agent planning. *Preprint*, arXiv:2508.15305.

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. 2026. Autoskill: Experience-driven lifelong learning via skill self-evolution. *Preprint*, arXiv:2603.01145.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*.

Shicheng Ye, Chao Yu, Kaiqiang Ke, Chengdong Xu, and Yinqi Wei. 2025a. H^2r: Hierarchical hindsight reflection for multi-task llm agents. *Preprint*, arXiv:2509.12810.

Sixiang Ye, Zeyu Sun, Guoqing Wang, Liwei Guo, Qingyuan Liang, Zheng Li, and Yong Liu. 2025b. Prompt alchemy: Automatic prompt refinement for enhancing code generation. *Preprint*, arXiv:2503.11085.

Simon Yu, Gang Li, Weiyan Shi, and Peng Qi. 2025. PolySkill: Learning generalizable skills through polymorphic abstraction. *Preprint*, arXiv:2510.15863.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task.

In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. 2026. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. *Preprint*, arXiv:2601.01885.

Zhenhang Yuan, Shenghai Yuan, and Lihua Xie. 2026. RPMS: Enhancing llm-based embodied planning through rule-augmented memory synergy. *Preprint*, arXiv:2603.17831.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. 2025. Optimizing generative AI by backpropagating language model feedback. *Nature*, 639(8055):609–616.

Guibin Zhang, Muxin Fu, Kun Wang, Frank Wan, Miao Yu, and Shuicheng Yan. 2025a. G-memory: Tracing hierarchical memory for multi-agent systems. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. 2026a. Memgen: Weaving generative latent memory for self-evolving agents. In *International Conference on Learning Representations (ICLR)*.

Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. 2025b. Memevolve: Meta-evolution of agent memory systems. *Preprint*, arXiv:2512.18746.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, Xue Liu, Xiaoxiao Li, and Philip S. Yu. 2026b. Coevoskills: Self-evolving agent skills via co-evolutionary verification. *Preprint*, arXiv:2604.01687.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. 2026c. Memskill: Learning and evolving memory skills for self-evolving agents. *Preprint*, arXiv:2602.02474.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. 2026d. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations (ICLR)*.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yutao Qi, Bo Tang, and Muning Wen. 2026e. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *Preprint*, arXiv:2601.03192.

Shuo Zhang, Zhenqiang Shen, Xiang Ao, Qinghua Hu, Qing He, and Xiaoqun Chen. 2026f. HiMem: Hierarchical long-term memory for LLM long-horizon agents. *Preprint*, arXiv:2601.06377.

Wenqi Zhang, Ke Tang, Hai Wu, Mengna Wang, Yongliang Shen, Guiyang Hou, Zeqi Tan, Peng Li, Yueting Zhuang, and Weiming Lu. 2024a. Agentpro: Learning to evolve via policy-level reflection and optimization. *Preprint*, arXiv:2402.17574.

Wenyuan Zhang, Xinghua Zhang, Haiyang Yu, Shuaiyi Nie, Bingli Wu, Juwei Yue, Tingwen Liu, and Yongbin Li. 2026g. Expseek: Self-triggered experience seeking for web agents. *Preprint*, arXiv:2601.08605.

Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2024b. A survey on the memory mechanism of large language model based agents. *Preprint*, arXiv:2404.13501.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2023. Expel: Llm agents are experiential learners. *Preprint*, arXiv:2308.10144.

Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. 2025. Skillweaver: Web agents can self-improve by discovering and honing skills. *Preprint*, arXiv:2504.07079.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. 2023. Synapse: Trajectory-as-exemplar prompting with memory for computer control. *Preprint*, arXiv:2306.07863.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19724–19731.

Chenyu Zhou, Huacan Chai, Wenteng Chen, Zihan Guo, Rong Shan, Yuanyi Song, Tianyi Xu, Yingxuan Yang, Aofan Yu, Weiming Zhang, et al. 2026a. Externalization in LLM agents: A unified review of memory, skills, protocols and harness engineering. *arXiv preprint arXiv:2604.08224*.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, and Jun Wang. 2025. Memento: Fine-tuning llm agents without fine-tuning llms. *Preprint*, arXiv:2508.16153.

Huichi Zhou, Siyuan Guo, Anjie Liu, Zhongwei Yu, Ziqin Gong, Bowen Zhao, Zhixun Chen, Menglong Zhang, Yihang Chen, Jinsong Li, Runyu Yang, Qiangbin Liu, Xinlei Yu, Jianmin Zhou, Na Wang, Chunyang Sun, and Jun Wang. 2026b. Memento-skills: Let agents design agents. *Preprint*, arXiv:2603.18743.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. 2023. WebArena: A realistic web environment for building autonomous agents. *Preprint*, arXiv:2307.13854.

Yingli Zhou, Wang Shu, Yaodong Su, Wenchuan Du, Yixiang Fang, and Xuemin Lin. 2026c. A comprehensive survey on agent skills: Taxonomy, techniques, and applications. *Preprint*, arXiv:2605.07358.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. *Preprint*, arXiv:2211.01910.

Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, Yu Qiao, Zhaoxiang Zhang, and Jifeng Dai. 2023. Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. *Preprint*, arXiv:2305.17144.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, et al. 2024. BigCodeBench: Benchmarking code generation with diverse function calls and complex instructions. *Preprint*, arXiv:2406.15877.

A Taxonomy of the Textualized Learning Process

Figure 2 presents a process taxonomy of Textualized Learning following the learning loop. It first organizes the subproblems under the three stages of generation, organization, and usage; representative methods can be further added to the leaf nodes in future versions.

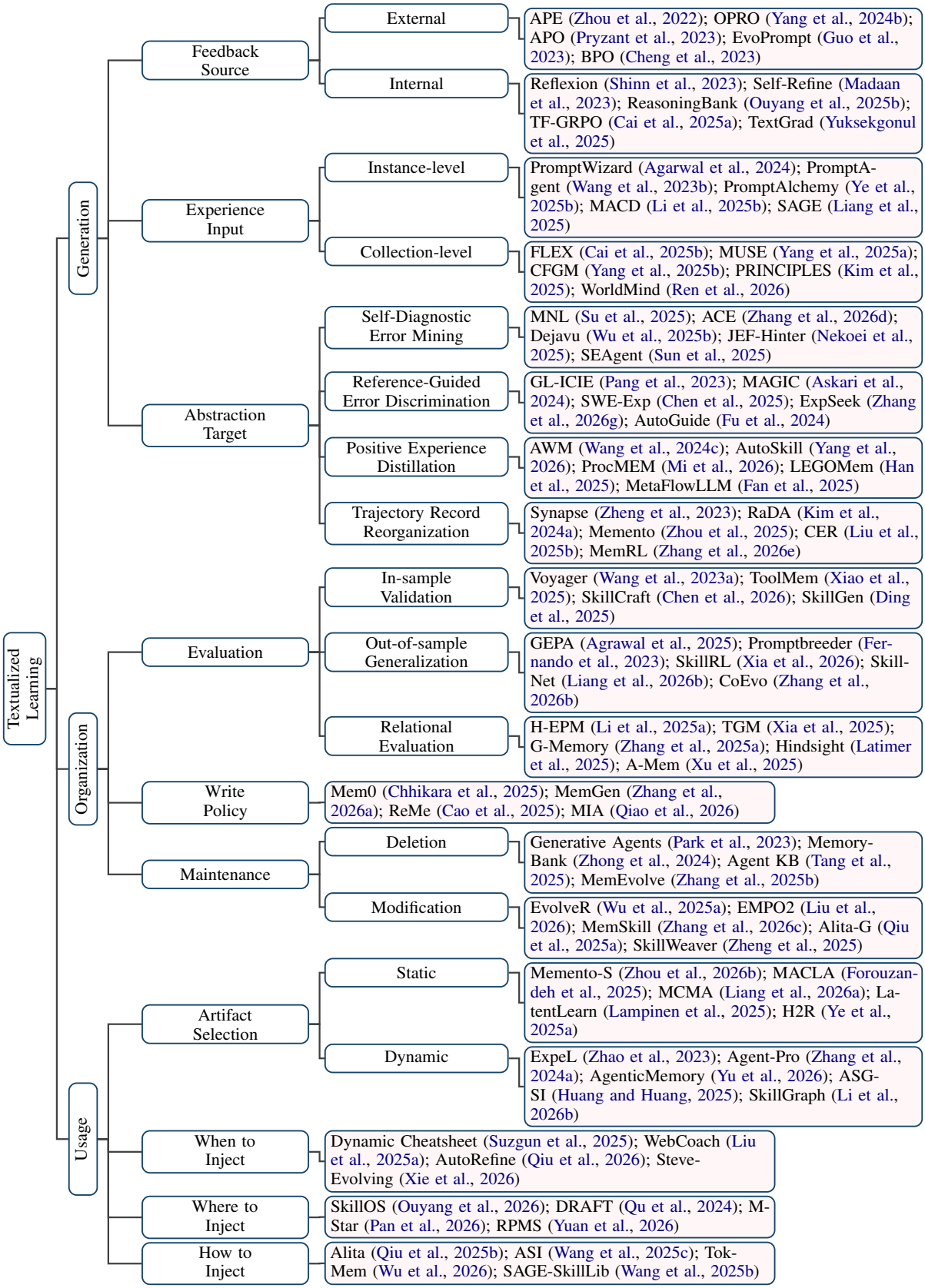

Figure 2: Taxonomy of the learning process in textualized learning. Representative methods are placed under their most characteristic process component to reduce duplication.

B Related Surveys

Existing related surveys mostly organize the literature from the perspective of external knowledge carriers, especially agent memory. These works usually focus on how agents store, retrieve, update, and forget information beyond the limited context window. For example, some surveys start from human memory types to analyze the formation and role of AI memory (Wu et al., 2025c), summarize memory mechanisms, storage structures, and invocation methods from the perspective of foundation agents (Huang et al., 2026), or discuss the forms, functions, and dynamics of memory in agent systems (Hu et al., 2025). These surveys provide systematic frameworks for understanding agent memory modules, but their core question remains: what information do agents store, how is it stored, and how is it retrieved and updated?

Our focus is different. We do not only study memory as one type of external module, but instead focus on a more general learning process: how systems abstract reusable textualized knowledge from task feedback, interaction trajectories, erroneous samples, and successful experience. In other words, memory is one possible carrier of external knowledge, whereas Textualized Learning concerns how experience is transformed into reusable rules, reflections, prompts, workflows, skills, tool-use policies, or code functions. Thus, Textualized Learning may be implemented as memory, but it may also take non-memory forms. Recent surveys on agent externalization have begun to view memory, skills, protocols, and harness engineering as different forms of externalization (Zhou et al., 2026a); however, their main focus remains on externalized structures rather than on systematically characterizing how such external knowledge is learned from experience.

Therefore, the core difference between this paper and existing memory-related surveys is that existing surveys mainly answer how agents store and retrieve external information, whereas we aim to answer how agents learn what to do in the future from experience. Specifically, we organize existing methods along the learning chain of supervision signals, experience granularity, generation instructions, evaluation methods, organization and maintenance, and later usage. We discuss how systems identify valuable experience, abstract it into artifacts, evaluate artifact utility in-sample and out-of-sample, and perform addition, deletion, modification, and storage. This perspective unifies scattered mechanisms across memory, reflection, prompt optimization, procedural memory, skill learning, and self-evolving agents, explaining how agents can achieve cross-task continual improvement through external textualized knowledge without updating model parameters.

C Conceptual Distinctions

Although Textualized Learning often relies on retrieval mechanisms, prompt engineering, and storage modules, and therefore overlaps technically with agent memory, in-context learning, retrieval-augmented generation, and self-correction, these concepts differ in memory storage, prompting paradigms, knowledge sources, and optimization objectives.

Consider a concrete web-search scenario: a user asks the system to search an e-commerce platform for "wireless noise-canceling headphones" and filter products priced between 500 and 800 yuan with ratings no lower than 4.6. The task requires both information acquisition and multi-step operation. Agent memory focuses on storing and reusing traceable process information, such as user preferences or key snippets from visited pages, so that later tasks can preserve consistent preferences. In-context learning typically provides a few examples or output templates, such as an example product table and field definitions, to align the output format within the current task. Retrieval-augmented generation introduces external factual information directly related to the current query, such as product-page evidence, platform rules, or parameter explanations. Self-correction performs automatic checking and regeneration within the same task, such as rewriting the output when fields are missing, prices are inconsistent, or links are malformed.

In contrast, still using the shopping-search task as an example, suppose the system receives feedback after a failed execution: the output table includes products outside the price range, some products report only star ratings without checking review counts, and some product pages contain both list prices and coupon prices, which the system fails to distinguish. The artifact generation module abstracts these concrete errors into textual rules, such as: "In product filtering tasks, first list all hard constraints and build a field checklist; distinguish list price, coupon price, and final price, and keep the price definition consistent with the user-specified

Survey	Main focus	Non-mem.	Learn.	Eval.	Maint.	Key difference from ours
Memory Mechanism of LLM Agents (Zhang et al., 2024b)	Memory module design, necessity, operations, and evaluation for LLM agents.	×	△	✓	✓	Focuses on memory modules; does not center the feedback-to-artifact learning process.
From Human Memory to AI Memory (Wu et al., 2025c)	Human-inspired AI memory taxonomy over memory objects, forms, and time scales.	×	△	△	△	Organizes memory types; ours organizes how reusable artifacts are learned.
Human-inspired AI Long-term Memory (He et al., 2024b)	Human long-term memory perspectives and adaptive long-term memory architectures.	×	△	△	△	Emphasizes long-term memory architecture rather than artifact generation and use.
Memory in the Age of AI Agents (Hu et al., 2025)	Agent memory forms, functions, and dynamics, with conceptual distinctions.	△	△	△	✓	Treats memory as the core object; ours treats memory as one possible backend.
Rethinking Memory Mechanisms of Foundation Agents (Huang et al., 2026)	Foundation-agent memory mechanisms, substrates, subjects, and operation policies.	△	△	✓	✓	Covers memory operations; ours further details generation, evaluation, and organization of artifacts.
Memory for Autonomous LLM Agents (Du, 2026)	Memory mechanisms, evaluation, and frontiers under a write–manage–read loop.	△	△	✓	✓	Uses a memory lifecycle view; ours uses a feedback-to-artifact learning view.
LLM Agent Memory (Tang et al., 2026)	Unified representation–management perspective on LLM agent memory systems.	△	△	△	✓	Focuses on representation and management; ours focuses on how experience becomes reusable rules.
From Storage to Experience (Luo et al., 2026)	Evolution of memory mechanisms from storage to reflection and experience.	△	✓	△	✓	Closest to ours, but still framed around memory evolution rather than general textualized artifacts.
Security of Long-Term Memory (Lin et al., 2026)	Security risks and governance across long-term memory lifecycles.	×	×	△	✓	Studies memory safety; ours studies Textualized Learning mechanisms.
Externalization in LLM Agents (Zhou et al., 2026a)	Unified view of memory, skills, protocols, and harness engineering as externalization.	✓	△	△	△	Covers external forms; ours explains how such reusable knowledge is learned from experience.
Agent Skills for LLMs (Xu and Yan, 2026)	Skill architecture, acquisition, deployment, security, and future directions.	✓	✓	△	✓	Centers on skills; ours covers rules, reflections, prompts, workflows, skills, and code artifacts.
Comprehensive Survey on Agent Skills (Zhou et al., 2026c)	Taxonomy of agent skills, including representation, acquisition, retrieval, and evolution.	✓	✓	✓	✓	Covers skill lifecycle; ours generalizes beyond skills to Textualized Learning.
Ours	Feedback-driven learning of reusable textualized artifacts from experience.	✓	✓	✓	✓	Studies the full loop: signal, data granularity, generation, evaluation, organization, and use.

Table 1: Comparison with related surveys.

constraint; rating filtering should check both rating and review count to avoid high ratings with too few reviews; if a product lacks key fields, mark it as insufficient information rather than filling it in; before producing the final table, verify each row against all constraints." This illustrates that Textualized Learning focuses on extracting rules from errors rather than simply remembering the failure itself.

After generation, the artifact is stored in an external artifact library, together with metadata such as applicable scenarios, source tasks, and success or failure feedback. When a similar web search or product-filtering task appears later, the system can retrieve relevant artifacts based on the task description and inject them into the reasoning context as rule blocks. During inference, these artifacts affect multiple stages of the model: in planning, the model first extracts hard constraints such as price, rating, and review count; in execution, it reads web information according to the field checklist; in generation, it organizes the table following the artifact; and in self-checking, it verifies each candidate product against all constraints. If new errors arise in later tasks, such as confusing coupon prices with membership prices, the system can further revise or extend the existing artifact based on new feedback.

Thus, the key to Textualized Learning is a cross-task learning and reuse loop: the system identifies stable error patterns or successful strategies from concrete execution experience, abstracts them into retrievable, injectable, and updatable textual artifacts, and finally uses these artifacts to constrain planning, action, and verification in later inference. This distinguishes Textualized Learning from agent memory, in-context learning, retrieval-augmented generation, and self-correction: it is not merely recording what happened in the past, imitating examples in the current prompt, or supplementing factual evidence for the current problem, but learning what to do when similar tasks arise in the future.

To clarify the distinct role of Textualized Learning in agent cognitive architectures, we further compare it with the following four paradigms.

C.1 Textualized Learning vs. Agent Memory

Textualized Learning is closely related to agent memory, but the two focus on different dimensions. Agent memory is usually an external information carrier (Zhou et al., 2026a) concerned with how information is written, stored, retrieved, and updated (Wu et al., 2025c; Huang et al., 2026). Textualized Learning, by contrast, focuses on how reusable knowledge is abstracted from task feedback, interaction trajectories, erroneous samples, and successful experience. On the one hand, artifacts can be stored as memory, such as reflection records (Shinn et al., 2023; Zhao et al., 2023; Ouyang et al., 2025b), error patterns (Askari et al., 2024; Su et al., 2025; Zhang et al., 2026g), experience entries (Park et al., 2023; Zhong et al., 2024; Tang et al., 2025), or retrievable rules (Fu et al., 2024; Zhang et al., 2026d; Cai et al., 2025b). On the other hand, artifacts may also be implemented outside memory, as skills (Wang et al., 2023a; Yang et al., 2026; Ding et al., 2025; Ni et al., 2026), system prompts or rule blocks (Pryzant et al., 2023; Yang et al., 2024b; Fernando et al., 2023; Guo et al., 2023), and code functions or programmatic skills (Wang et al., 2023a, 2025c; Zheng et al., 2025; Mi et al., 2026). In other words, memory is one possible storage backend for Textualized Learning, rather than its defining boundary.

In LLM agents, memory usually appears as an external read-write storage and retrieval module that preserves information beyond the limited context window and retrieves it when needed (Wu et al., 2025c; Huang et al., 2026). Inspired by cognitive neuroscience, human long-term memory is often divided into episodic, semantic, and procedural memory (Tulving, 1972; Squire, 2004). Episodic memory records concrete experiences and event trajectories, such as user preferences during shopping; semantic memory stores factual knowledge and concepts to support reasoning (Park et al., 2023). However, episodic and semantic memory mainly carry individualized experience and factual knowledge. They are useful as background context or evidence, but they do not necessarily provide behavioral strategies for a concrete task. Thus, in scenarios requiring multi-step planning, strict rule following, or cross-task strategy reuse, relying only on these memory types is often insufficient.

By contrast, Textualized Learning does not simply store experience or knowledge. Its goal is to extract reusable rules, strategies, or procedures from scattered interaction data (Mi et al., 2026; Han et al., 2025). From this perspective, Textualized Learning can be viewed as an externalization mechanism for procedural and strategic knowledge: it abstracts and compresses raw interaction trajectories, error feedback, and successful experience into compact, retrievable textual rules that can directly affect later inference, reducing the burden of re-inducing strategies from raw experience in future tasks.

C.2 Textualized Learning vs. In-Context Learning

In-context learning refers to the ability of a model to adapt to a target task at inference time, without parameter updates, simply by conditioning on a small number of examples, instructions, or output formats in the input context (Brown et al., 2020; Dong et al., 2024). Existing studies typically distinguish two prompt structures for in-context learning: example-based prompts, which concatenate input-output pairs to induce analogical generation, and instruction-based prompts, which describe task goals and output constraints in natural language (Min et al., 2022). Textualized Learning and in-context learning both achieve non-parametric adaptation through textual context and do not change the model's underlying weights. Formally, generated artifacts are often injected into the context as system prompts, rule blocks, or structured templates.

However, they differ substantially in their inference mechanisms. In-context learning primarily relies on analogies induced by few-shot examples. Models often imitate surface patterns without explicitly understanding the logical structure behind the examples, which may limit stability on complex problems. Textualized Learning, by contrast, does not simply concatenate raw examples; instead, it provides abstracted reasoning paths, constraints, and rule instructions. In tasks requiring strict logical control, such strategic artifacts can compensate for the limitations of in-context learning in logical induction.

C.3 Textualized Learning vs. Retrieval-Augmented Generation

Retrieval-augmented generation (RAG) combines external textual knowledge with generative models. The system retrieves relevant passages from a document collection and feeds them together with the original input into the model, thereby improv-

ing factual accuracy and traceability in knowledge-intensive tasks (Lewis et al., 2020; Gao et al., 2024). Representative methods include RAG frameworks that explicitly use retrieved evidence during generation and REALM-style approaches that introduce retrieval during pretraining. In implementation, dense retrieval, such as DPR, and multi-evidence fusion, such as FiD, are common technical routes (Lewis et al., 2020; Guu et al., 2020; Karpukhin et al., 2020; Izacard and Grave, 2021).

Textualized Learning and RAG both involve retrieving and injecting external information, but they differ in knowledge objects and optimization goals. First, RAG retrieves factual textual evidence from external corpora to support answers and reduce hallucinations (Lewis et al., 2020; Gao et al., 2024); Textualized Learning focuses on strategic textual artifacts for task execution, explicitly describing reasoning steps, constraints, and boundary cases. Second, RAG knowledge bases are typically pre-collected external documents, and retrieval is based on semantic relevance to the current input. Artifacts, in contrast, are usually induced from task supervision, interaction trajectories, or feedback signals, emphasizing correspondence with task structure and error patterns. Third, RAG improves factual correctness by supplementing parametric knowledge, whereas Textualized Learning constrains reasoning paths and behavioral choices through reusable strategic artifacts, thereby improving success rate, consistency, and controllability in complex tasks. Thus, the two are often complementary in practice: RAG provides factual evidence, while Textualized Learning provides strategic guidance (Lewis et al., 2020; Gao et al., 2024).

C.4 Textualized Learning vs. Self-Correction

Self-correction studies how a system can automatically verify, critique, and rewrite an initial output for the same input instance, without updating model parameters, to obtain a higher-quality final answer. Existing work is often categorized by intervention timing and feedback form, including correction signals during training, iterative revision during generation, and post-generation evaluation and editing (Pan et al., 2024). Concrete implementations include using model self-feedback for multi-round rewriting (Madaan et al., 2023), generating reflections after failed trajectories to guide retry (Shinn et al., 2023), introducing explicit verification steps to reduce hallucination (Dhuliawala et al., 2024), triggering verify-and-revise loops through key-condition verification (Wu et al., 2024b), and using black-box checks to decide whether revision is needed (Manakul et al., 2023).

Textualized Learning and self-correction both rely on reflection and evaluation mechanisms, using the model's introspective ability to identify problems in outputs. However, they differ in scope. Self-correction mainly targets the current instance, aiming to improve the current output, such as fixing a piece of code or correcting an answer; after the task ends, these experiences are usually not retained. Textualized Learning, by contrast, is a cross-task, long-term optimization process. It summarizes general experience from correction processes and solidifies it as reusable artifacts for future similar tasks.

D Forms of Artifact Generation

The output forms of artifact generation can be classified by abstraction level, structural explicitness, and executability into five major categories: Prompt, Trajectory, Guideline, Skill, and Code/-tool, as summarized in Table 2. Prompt artifacts mainly optimize task descriptions and output constraints (Pryzant et al., 2023; Khattab et al., 2024). They are the cheapest to acquire and maintain, but they are weakly structured and non-executable, and usually affect model behavior only by changing the overall prompt. Trajectory artifacts preserve complete or partial task execution processes (Zheng et al., 2023; Kim et al., 2024a), including states, actions, observations, feedback, and outcomes. They retain richer contextual information than prompts, but have lower abstraction; reuse often requires the model to re-induce useful strategies from concrete traces, leading to higher context cost and reasoning burden. Guideline-style artifacts further abstract trajectories and feedback into rules, reflections, checklists, or strategies (Zhao et al., 2023; Fu et al., 2024). Compared with trajectories, they compress redundant details and provide more direct textual guidance, but they often lack explicit triggers and execution interfaces. Skill artifacts add applicability conditions, triggers, inputs and outputs, execution steps, and failure-recovery procedures (Yang et al., 2026; Mi et al., 2026), thereby organizing abstract guidance as procedural knowledge for long-horizon tasks, tool use, and compositional workflows. Code/tool artifacts further encapsulate stable procedures as executable programs, tool interfaces, or action macros (Wang

et al., 2023a; Qiu et al., 2025b). They have the highest degree of structure and executability, can directly participate in environment interaction, and reduce the need for long textual injection, but also require higher generation, verification, and maintenance cost.

E Application Tasks and Benchmarks

Table 3 summarizes the main task types and representative datasets discussed in the application section. The table is intended to illustrate the evaluation scope covered by existing work, rather than to imply that all methods are evaluated under directly comparable settings. The link column prioritizes dataset websites, project pages, or commonly used code repositories; entries without stable links are marked as "–". Rows are grouped by the seven application categories in Section 7. To avoid repeated one-column/two-column switches in the appendix, the full-width appendix tables are grouped at the end.

F Extended Generation-stage Taxonomy Table

Table 4 keeps the paper-level structure of the original extended generation-stage table, but normalizes each paper to the seven task types used in Section 7: reasoning, programming, web, embodied decision-making, planning, dialogue, and domain expertise. When a method spans multiple settings, the task-type column lists up to two categories. The granularity column distinguishes single-instance generation from batch or contrastive generation. The instruction column uses compact labels: Ref.-guided error discr. denotes reference-guided error discrimination, Positive exp. distill. denotes positive-experience distillation, Process-record org. denotes process-record organization, and Self-diagnostic mining denotes self-diagnostic error mining. The form column is normalized to Prompt, Raw trajectories, Rule, Skill, and Code. The full table is placed with the other full-width appendix table at the end.

G Storage Structures for Artifact Collections

Section 5 mainly discusses the evaluation, addition, deletion, and modification mechanisms of artifact collections. This appendix supplements the discussion with the underlying storage structures. Existing organization structures can be roughly divided into three categories: flat entry-based storage, hierarchical or nested storage, and graph-based or hybrid storage. Their differences lie not in the textual form of artifacts, but in whether the collection explicitly maintains dependencies, compositions, similarities, conflicts, or transition relations among entries.

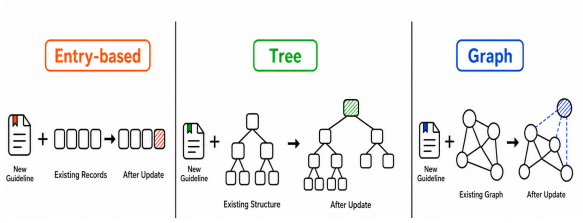

Figure 3: Major storage structures for artifact collections, including entry-based, tree-based, and graph-based or hybrid storage.

Figure 3 illustrates three common organization structures inside artifact libraries. Entry-based storage supports fast writing and retrieval; tree-based storage represents hierarchical decomposition; graph-based storage is better suited for dependencies, conflicts, and compositional relations.

G.1 Entry-based Storage

Entry-based storage organizes the artifact library as a set of independent records. Each entry stores reusable content together with retrieval and maintenance metadata, such as applicable scenarios, source samples, invocation records, historical performance, or validation scores. Prompt-optimization methods typically store candidate prompts and validation scores (Yang et al., 2024b; Pryzant et al., 2023), or store candidate prompts and fitness values during evolutionary search (Fernando et al., 2023; Guo et al., 2023), enabling later comparison, replacement, and further optimization. Experience-learning methods often organize erroneous samples and cause analyses as self-correction records (Askari et al., 2024), decompose erroneous behavior, error analysis, and corrective clues into retrievable entries (Zhang et al., 2026g), or store trajectory fragments (Zheng et al., 2023), reflective experience (Zhao et al., 2023), and reasoning experience (Ouyang et al., 2025b) as independent records.

Entries can also attach metadata to support later ranking and maintenance, such as using time, importance, and relevance to determine memory recall (Park et al., 2023), using time decay and user

Form	Learned object	Struct.	Exec.	Cost	Key characteristic	Representative methods
Prompt	task instruction, system prompt, prompt template	Low	Low	Low	Controls model behavior through reusable input text; mainly answers how to phrase the task.	APO (Pryzant et al., 2023); OPRO (Yang et al., 2024b); EvoPrompt (Guo et al., 2023); PromptAgent (Wang et al., 2023b); GEPA (Agrawal et al., 2025)
Trajectory	execution trace, episode, case, success/failure example	Low	Low	Mid	Stores concrete past task processes for later analogical reuse, without necessarily abstracting rules.	Synapse (Zheng et al., 2023); Memento (Zhou et al., 2025); RaDA (Kim et al., 2024a); SYNAPSE (Jiang et al., 2026)
Guideline	rule, reflection, lesson, checklist, strategy	Mid	Low	Mid	Abstracts transferable natural-language guidance from traces, feedback, or self-reflection.	AutoGuide (Fu et al., 2024); ExpeL (Zhao et al., 2023); MAGIC (Askari et al., 2024); ReasoningBank (Ouyang et al., 2025b); TGM (Xia et al., 2025)
Skill	workflow, reusable procedure, tool-use routine	High	Mid	High	Adds triggering conditions, inputs/outputs, steps, and recovery logic to reusable guidance.	AWM (Wang et al., 2024c); AutoSkill (Yang et al., 2026); ProcMEM (Mi et al., 2026); LEGOMem (Han et al., 2025); MetaFlowLLM (Fan et al., 2025)
Code/tool	executable code, API wrapper, action macro	High	High	High	Encapsulates stable procedures as callable programs or tools that directly participate in execution.	Voyager (Wang et al., 2023a); Alita (Qiu et al., 2025b); Alita-G (Qiu et al., 2025a); Programmatic Skills (Wang et al., 2025c); SAGE (Liang et al., 2025)

Table 2: Comparison of major artifact representation forms. They differ in structure, executability, acquisition cost, and how directly they can be reused by agents.

relevance to maintain long-term dialogue memory (Zhong et al., 2024), or managing entry states through explicit memory operations and metadata updates (Chhikara et al., 2025). Overall, entry-based storage is easy to write and maintain, and is suitable for rapidly accumulating prompts, reflections, error patterns, experience fragments, and memory entries. Its limitation also comes from this independence: the system cannot directly determine whether entries are duplicate, special cases of one another, mutually supportive, or conflicting. Additional similarity comparison, quality evaluation, or pruning mechanisms are therefore often required to control redundancy and retrieval noise.

G.2 Tree-based Storage

Tree-based storage represents parent-child support relations among artifacts, where high-level artifacts are supported by lower-level experience, events, skills, steps, or implementations. When writing a new artifact, the system must not only store its content but also decide which existing parent node it should attach to, or whether it should become a new root node. Tree structures are therefore suitable for representing hierarchical relations from fine-grained experience to high-level summaries, from atomic actions to composite skills, or from general interfaces to concrete implementations.

In memory-management settings, tree structures are used to consolidate low-level interaction fragments into high-level summaries (Rezazadeh et al., 2025), organize event experience as hierarchical memory (Sun and Zeng, 2025; Zhang et al., 2026f), or aggregate fine-grained dialogue memories into long-term profiles by temporal granularity (Li et al., 2026a; Tan et al., 2026). In skill-learning settings, tree or nested structures represent compositions from basic actions to composite skills (Wang et al., 2026), abstraction–implementation relations between general skill interfaces and scenario-specific implementations (Yu et al., 2025), or reusable modules encapsulating programmatic skills (Wang et al., 2025c), code skills (Wang et al., 2023a), and sub-agent procedures (Qiu et al., 2026). Tree-based storage explicitly preserves decomposition and reuse relations, but requires maintaining parent selection, subtree updates, and the influence of lower-level skill or environment changes on high-level artifacts.

G.3 Graph-based Storage

Graph-based storage represents explicit relations among artifacts. When writing a new artifact, the system adds it as a node to the graph and determines its edges to existing nodes. Compared with parent-child expansion in trees, graphs better represent many-to-many connections, cross-dependencies, state transitions, skill composition, similarity links, and conflicts.

In experience and long-term memory, graphs can record state transitions and decision paths (Xia et al., 2025), connect factual knowledge (Chhikara et al., 2025), or link memories by similarity (Xu et al., 2025). They can also connect memories and experience at different granularities through multi-layer graphs (Zhang et al., 2025a) or multi-network structures (Latimer et al., 2025). In tool, skill, and procedural knowledge, graphs often represent tool-call order (Li et al., 2025a), skill dependencies and preconditions (Huang and Huang, 2025;

Liang et al., 2026b), node relations in evolving skill graphs (Li et al., 2026b), and executable skill generation or composition relations (Ding et al., 2025), enabling retrieval, composition, and updating along graph relations. Graph-based storage can express complex relations and is suitable for cross-task reuse, conflict detection, and multi-hop retrieval, but it also requires maintaining edge types, directions, and validity. It is thus more appropriate for complex and heterogeneous artifact libraries, whereas entry-based or tree-based storage is usually simpler when relations are weak.

H Full Appendix Tables

Table 3: Representative tasks and benchmarks for Textualized Learning applications.

Category	Task	Dataset / Benchmark	Link
Reasoning	Fact verification	FEVER (Thorne et al., 2018)	Website
Reasoning	Financial information extraction	FiNER (Loukas et al., 2022)	–
Reasoning	General reasoning	BBH (Suzgun et al., 2023)	GitHub
Reasoning	General reasoning	BIG-Bench (Srivastava et al., 2023)	GitHub
Reasoning	Information extraction	ACE05 (Walker et al., 2006)	–
Reasoning	Information extraction	SemEval (Hendrickx et al., 2010)	–
Reasoning	Knowledge reasoning	MMLU (Hendrycks et al., 2021a)	GitHub
Reasoning	Math competition reasoning	AIME (Mathematical Association of America, 2026)	–
Reasoning	Math reasoning	GSM8K (Cobbe et al., 2021)	GitHub
Reasoning	Math reasoning	MATH (Hendrycks et al., 2021b)	GitHub
Reasoning	Math word problems	AQuA (Ling et al., 2017)	–
Reasoning	Multi-hop QA	2WikiMultihopQA (Ho et al., 2020)	–
Reasoning	Multi-hop QA	HotpotQA (Yang et al., 2018)	Website
Reasoning	Open-domain QA	Natural Questions (Kwiatkowski et al., 2019)	Website
Reasoning	Open-domain QA	TriviaQA (Joshi et al., 2017)	–
Reasoning	Science QA	GPQA (Rein et al., 2023)	GitHub
Reasoning	Semantic parsing	Spider (Yu et al., 2018)	Website
Reasoning	Text-to-SQL	BIRD (Li et al., 2023a)	Website
Programming	Algorithmic programming	LeetCode (LeetCode, 2026)	–
Programming	Code generation	BigCodeBench (Zhuo et al., 2024)	–
Programming	Code generation	HumanEval (Chen et al., 2021)	GitHub
Programming	Code generation	LiveCodeBench (Jain et al., 2024)	Website
Programming	Code generation	MBPP (Austin et al., 2021)	GitHub
Programming	Code understanding	AVATAR (Ahmad et al., 2023)	–
Programming	Code understanding	CodeNet (Puri et al., 2021)	–
Programming	Hardware-aware code optimization	NPUEval (Kalade and Schelle, 2025)	–
Programming	Kernel optimization	KernelBench (Ouyang et al., 2025a)	–
Programming	Software repair	SWE-bench (Jimenez et al., 2024)	Website
Programming	Verified software repair	SWE-bench Verified (OpenAI, 2024)	Website
Web	API use	API Bank (Li et al., 2023b)	GitHub
Web	App and tool interaction	AppWorld (Trivedi et al., 2024)	–
Web	Function calling	BFCL-V3 (Patil et al., 2025)	–
Web	GUI control	OSWorld (Xie et al., 2024b)	Website
Web	GUI control	ScienceBoard (Sun et al., 2025)	–
Web	Office web tasks	WorkArena (Drouin et al., 2024)	–
Web	Tool use	ToolBench (Qin et al., 2024)	GitHub
Web	Tool-use safety	ToolSandbox (Lu et al., 2024)	–
Web	Visual web control	VisualWebArena (Koh et al., 2024)	Website
Web	Web control	WebArena (Zhou et al., 2023)	Website
Web	Web interaction	CompWoB (Furuta et al., 2024)	–
Web	Web interaction	MiniWoB (Shi et al., 2017)	–
Web	Web navigation	Mind2Web (Deng et al., 2023)	Website
Web	Web navigation	WebVoyager (He et al., 2024a)	–
Web	Web shopping	WebShop (Yao et al., 2022)	GitHub
Embodied Decision	Embodied navigation	EB-Habitat (Savva et al., 2019; Wei et al., 2025)	–
Embodied Decision	Game and simulation interaction	TextArena (Guertler et al., 2025)	–
Embodied Decision	Instruction-following navigation	BabyAI (Chevalier-Boisvert et al., 2019)	–
Embodied Decision	Open-world exploration	Minecraft tasks (Wang et al., 2023a; Zhu et al., 2023)	Website
Embodied Decision	Robot manipulation	AgiBot (Bu et al., 2025)	–
Embodied Decision	Robot manipulation	LIBERO (Liu et al., 2023a)	Website
Embodied Decision	Science environment	ScienceWorld (Wang et al., 2022)	Website
Embodied Decision	Text-world navigation	ALFWorld (Shridhar et al., 2021)	Website
Embodied Decision	Visual navigation	EB-ALFRED (Shridhar et al., 2020; Wei et al., 2025)	–
Embodied Decision	VLA tasks	OpenVLA (Kim et al., 2024b)	Website
Planning	Abstract planning	gridworld (Sutton and Barto, 2018)	–
Planning	Agent decision-making	AgentBench (Liu et al., 2023b)	GitHub
Planning	Combinatorial planning	Traveling Salesman Problem (Applegate et al., 2006)	–
Planning	Enterprise task planning	TheAgentCompany (Xu et al., 2024)	–
Planning	Lifelong learning	Lifelong Agent Bench (Yang et al., 2026)	–
Planning	Long-horizon planning	TravelPlanner (Xie et al., 2024a)	GitHub
Planning	Multi-agent interaction	Smallville (Park et al., 2023)	Website
Planning	Office tasks	OfficeBench (Wang et al., 2024b)	–
Planning	Symbolic planning	PDDL (McDermott et al., 1998)	–
Dialogue	Cross-session memory	LoCoMo (Maharana et al., 2024)	–
Dialogue	Long-memory evaluation	LongMemEval (Wu et al., 2024a)	–
Dialogue	Open-ended dialogue	OASST1 (Köpf et al., 2023)	–
Dialogue	Personalized interaction	SiliconFriend (Zhong et al., 2024)	–
Dialogue	Persuasive dialogue	P4G (Wang et al., 2019)	–
Dialogue	Task-oriented dialogue	MultiWOZ (Budzianowski et al., 2018)	GitHub
Domain Expertise	Business reasoning	PRBench (Akyurek et al., 2025)	–
Domain Expertise	Clinical decision	HealthBench (Arora et al., 2025)	Website
Domain Expertise	Clinical decision	MIMIC-MACD (Li et al., 2025b)	–
Domain Expertise	Financial formula reasoning	Formula (Wang et al., 2025a)	–
Domain Expertise	Medical diagnosis	DDXPlus (Fansi Tchango et al., 2022)	–
Domain Expertise	Medical VQA	PathVQA (He et al., 2020)	GitHub
Domain Expertise	Molecular representation	SMILES (Weininger, 1988)	–
Domain Expertise	Protein modeling	ProteinGym (Notin et al., 2023)	Website

Table 4: Extended generation-stage taxonomy of Textualized Learning papers. Task types are normalized to seven application categories, generation instructions are grouped into four categories, and artifact forms are grouped into five categories.

Paper	Task Type	Gran.	Instruction	Form	Link
Agent KB (Tang et al., 2025)	Reasoning; Programming	Single	Positive exp. distill.	Skill	-
AWM (Wang et al., 2024c)	Web	Batch	Positive exp. distill.	Skill	GitHub
Memento (Zhou et al., 2025)	Reasoning	Single	Process-record org.	Raw trajectories	GitHub
ACE (Zhang et al., 2026d)	Reasoning; Web	Single	Self-diagnostic mining	Rule	GitHub
AgeMem (Yu et al., 2026)	Reasoning; Web	Single	Positive exp. distill.	Rule	-
Agent-Pro (Zhang et al., 2024a)	Web	Single	Self-diagnostic mining	Rule	GitHub
WorldMind (Ren et al., 2026)	Embodied Decision	Batch	Self-diagnostic mining	Rule	-
Alita (Qiu et al., 2025b)	Reasoning; Programming	Single	Positive exp. distill.	Code	GitHub
Alita-G (Qiu et al., 2025a)	Reasoning; Web	Single	Positive exp. distill.	Code	GitHub
ASG-SI (Huang and Huang, 2025)	Planning; Web	Single	Positive exp. distill.	Skill	GitHub
AutoGuide (Fu et al., 2024)	Reasoning; Web	Batch	Ref.-guided error discr.	Rule	-
APO (Pryzant et al., 2023)	Reasoning; Web	Batch	Ref.-guided error discr.	Prompt	-
AutoPDL (Spiess et al., 2025)	Reasoning	Single	Ref.-guided error discr.	Prompt	Website
AutoRefine (Qiu et al., 2026)	Reasoning; Web	Batch	Ref.-guided error discr.	Rule	-
AutoSkill (Yang et al., 2026)	Reasoning; Programming	Single	Positive exp. distill.	Skill	GitHub
BPO (Cheng et al., 2023)	Reasoning; Web	Single	Ref.-guided error discr.	Prompt	GitHub
CFGM (Yang et al., 2025b)	Reasoning; Web	Batch	Ref.-guided error discr.	Rule	-
CoEvo (Zhang et al., 2026b)	Reasoning; Web	Batch	Positive exp. distill.	Skill	Website
CER (Liu et al., 2025b)	Web	Single	Process-record org.	Raw trajectories	-
CoPS (Yang et al., 2024a)	Reasoning; Web	Single	Ref.-guided error discr.	Rule	GitHub
Dejavu (Wu et al., 2025b)	Reasoning; Embodied Decision	Single	Self-diagnostic mining	Rule	Website
Dyn. Cheatsheet (Suzgun et al., 2025)	Reasoning; Embodied Decision	Single	Self-diagnostic mining	Rule	GitHub
EvolveR (Wu et al., 2025a)	Reasoning; Web	Single	Self-diagnostic mining	Rule	GitHub
Evo-Memory (Wei et al., 2025)	Reasoning	Single	Process-record org.	Raw trajectories	-
EvoPrompt (Guo et al., 2023)	Reasoning; Web	Single	Ref.-guided error discr.	Prompt	GitHub
ExpeL (Zhao et al., 2023)	Reasoning; Web	Batch	Ref.-guided error discr.	Rule	GitHub
H-EPM (Li et al., 2025a)	Reasoning; Web	Batch	Process-record org.	Raw trajectories	-
EMPO2 (Liu et al., 2026)	Web; Domain Expertise	Single	Self-diagnostic mining	Rule	-
ExpSeek (Zhang et al., 2026g)	Reasoning; Web	Batch	Ref.-guided error discr.	Rule	-
FLEX (Cai et al., 2025b)	Reasoning; Domain Expertise	Batch	Ref.-guided error discr.	Rule	Website
TGM (Xia et al., 2025)	Reasoning	Single	Ref.-guided error discr.	Raw trajectories	-
DRAFT (Qu et al., 2024)	Planning; Web	Batch	Positive exp. distill.	Skill	GitHub
GenAgents (Park et al., 2023)	Reasoning; Web	Single	Self-diagnostic mining	Rule	-
GEPA (Agrawal et al., 2025)	Reasoning; Programming	Batch	Ref.-guided error discr.	Prompt	GitHub
GITM (Zhu et al., 2023)	Reasoning; Embodied Decision	Batch	Positive exp. distill.	Rule	-
G-Memory (Zhang et al., 2025a)	Reasoning; Programming	Single	Positive exp. distill.	Rule	GitHub
GRIPS (Prasad et al., 2022)	Reasoning	Single	Ref.-guided error discr.	Prompt	GitHub
GL-ICIE (Pang et al., 2023)	Reasoning; Web	Single	Ref.-guided error discr.	Rule	-
H2R (Ye et al., 2025a)	Planning	Single	Positive exp. distill.	Rule	-
Hindsight (Latimer et al., 2025)	Reasoning; Web	Single	Positive exp. distill.	Rule	-
ASI (Wang et al., 2025c)	Programming; Web	Single	Positive exp. distill.	Code	GitHub
JEF-Hinter (Nekoei et al., 2025)	Reasoning; Web	Batch	Self-diagnostic mining	Rule	-
APE (Zhou et al., 2022)	Reasoning; Web	Single	Ref.-guided error discr.	Prompt	GitHub
OPRO (Yang et al., 2024b)	Reasoning	Single	Ref.-guided error discr.	Prompt	GitHub
LatentLearn (Lampinen et al., 2025)	Programming; Web	Single	Process-record org.	Rule	-
MACLA (Forouzandeh et al., 2025)	Reasoning; Programming	Batch	Positive exp. distill.	Skill	-
MCMA (Liang et al., 2026a)	Reasoning; Embodied Decision	Batch	Positive exp. distill.	Rule	GitHub
MUSE (Yang et al., 2025a)	Reasoning; Programming	Batch	Ref.-guided error discr.	Skill	GitHub
ALMA (Xiong et al., 2026)	Planning	Single	Self-diagnostic mining	Rule	-
LEGOMem (Han et al., 2025)	Reasoning	Batch	Positive exp. distill.	Skill	-
M-Star (Pan et al., 2026)	Reasoning; Embodied Decision	Batch	Positive exp. distill.	Skill	-
MACD (Li et al., 2025b)	Reasoning; Domain Expertise	Single	Self-diagnostic mining	Rule	-
MAGIC (Askari et al., 2024)	Reasoning; Web	Batch	Self-diagnostic mining	Rule	GitHub
Mem0 (Chhikara et al., 2025)	Reasoning; Web	Single	Process-record org.	Rule	-
Memento (Zhou et al., 2025)	Reasoning; Web	Single	Process-record org.	Raw trajectories	GitHub
Memento-S (Zhou et al., 2026b)	Planning; Web	Single	Positive exp. distill.	Skill	-
MemEvolve (Zhang et al., 2025b)	Reasoning; Web	Single	Positive exp. distill.	Code	GitHub
MemGen (Zhang et al., 2026a)	Reasoning; Programming	Single	Process-record org.	Rule	GitHub
MIA (Qiao et al., 2026)	Reasoning	Single	Process-record org.	Raw trajectories	GitHub
MemBank (Zhong et al., 2024)	Dialogue; Planning	Single	Process-record org.	Rule	GitHub
Memp (Fang et al., 2025)	Reasoning; Embodied Decision	Single	Positive exp. distill.	Skill	GitHub
MemRL (Zhang et al., 2026e)	Reasoning; Programming	Single	Process-record org.	Skill	-
MemSkill (Zhang et al., 2026c)	Reasoning; Embodied Decision	Batch	Positive exp. distill.	Skill	GitHub
MetaFlow (Fan et al., 2025)	Reasoning; Web	Batch	Positive exp. distill.	Skill	GitHub

Continued on next page

Paper	Task Type	Gran.	Instruction	Form	Link
MNL (Su et al., 2025)	Reasoning; Web	Single	Self-diagnostic mining	Rule	-
TextGrad (Yuksekgonul et al., 2025)	Reasoning; Programming	Single	Ref.-guided error discr.	Prompt	GitHub
PRINCIPLES (Kim et al., 2025)	Dialogue; Reasoning	Batch	Ref.-guided error discr.	Rule	-
ProcMEM (Mi et al., 2026)	Reasoning; Embodied Decision	Batch	Positive exp. distill.	Skill	GitHub
PromptAlchemy (Ye et al., 2025b)	Reasoning; Programming	Single	Ref.-guided error discr.	Prompt	GitHub
PAgent (Wang et al., 2023b)	Reasoning	Single	Ref.-guided error discr.	Prompt	GitHub
PB (Fernando et al., 2023)	Reasoning	Single	Ref.-guided error discr.	Prompt	-
PW (Agarwal et al., 2024)	Reasoning; Web	Single	Ref.-guided error discr.	Prompt	-
RaDA (Kim et al., 2024a)	Web	Single	Process-record org.	Raw trajectories	-
RBank (Ouyang et al., 2025b)	Reasoning; Programming	Batch	Ref.-guided error discr.	Rule	-
SAGE-Lib (Wang et al., 2025b)	Reasoning; Web	Single	Positive exp. distill.	Code	GitHub
ReMe (Cao et al., 2025)	Reasoning; Web	Batch	Positive exp. distill.	Skill	GitHub
RPMS (Yuan et al., 2026)	Reasoning; Embodied Decision	Batch	Positive exp. distill.	Rule	-
SAGE (Liang et al., 2025)	Reasoning; Web	Single	Self-diagnostic mining	Rule	-
SEAgent (Sun et al., 2025)	Reasoning; Programming	Single	Self-diagnostic mining	Rule	GitHub
SkillCraft (Chen et al., 2026)	Reasoning; Web	Batch	Positive exp. distill.	Skill	GitHub
SkillGen (Ding et al., 2025)	Reasoning; Embodied Decision	Single	Positive exp. distill.	Skill	GitHub
SkillNet (Liang et al., 2026b)	Reasoning; Web	Single	Positive exp. distill.	Skill	GitHub
Skill-Pro (Mi et al., 2026)	Web; Embodied Decision	Batch	Positive exp. distill.	Skill	GitHub
SkillRL (Xia et al., 2026)	Reasoning; Web	Batch	Positive exp. distill.	Skill	GitHub
SkillWeaver (Zheng et al., 2025)	Web	Batch	Positive exp. distill.	Code	GitHub
Steve-Evo (Xie et al., 2026)	Reasoning; Embodied Decision	Single	Self-diagnostic mining	Rule	GitHub
SWE-Exp (Chen et al., 2025)	Reasoning; Programming	Batch	Positive exp. distill.	Rule	-
Synapse (Zheng et al., 2023)	Reasoning; Web	Batch	Process-record org.	Raw trajectories	Website
TokMem (Wu et al., 2026)	Reasoning; Web	Single	Positive exp. distill.	Rule	-
ToolMem (Xiao et al., 2025)	Reasoning; Web	Batch	Positive exp. distill.	Skill	-
T2S (Ni et al., 2026)	Reasoning; Web	Single	Positive exp. distill.	Skill	GitHub
TF-GRPO (Cai et al., 2025a)	Reasoning; Programming	Batch	Ref.-guided error discr.	Rule	GitHub
Voyager (Wang et al., 2023a)	Embodied Decision	Batch	Positive exp. distill.	Code	-
WebCoach (Liu et al., 2025a)	Web	Single	Ref.-guided error discr.	Rule	-