# Skill Form Matters: A Train-Free Study of Skill Representation Transfer for LLM Agents

**Anonymous ACL submission**

## Abstract

LLM agents increasingly externalize successful experience as retrievable skills, but the same experience can be stored as traces, scripts, hierarchical summaries, or artifact-style directories. We study whether representation form itself changes transfer behavior when retrieval and prompting are otherwise held fixed. We present a train-free protocol that converts the same source trajectories into multiple skill forms and compares them under matched retrieval, prompting, and context-budget regimes. On ALFWorld, the results show form-dependent trade-offs rather than a universal winner: trace-like representations are strong for near-neighbor reuse under tight budgets, scripts are more robust on the unseen split, and more structured forms add maintenance affordances at non-trivial context cost. The broader contribution is methodological: skill form should be treated as an explicit experimental variable when building portable, context-efficient skill banks.

## 1 Introduction

Long-horizon LLM agents increasingly rely on *externalized experience*—reusable *skills* that can be retrieved and injected at test time—rather than parameter updates alone (Zhao et al., 2023; Fang et al., 2025; Liu et al., 2026; Yang et al., 2026). Yet the same underlying experience is stored in incompatible *forms*: raw trajectories, summarized scripts, hierarchical reflections, executable code, or filesystem artifacts such as local `SKILL.md` directories (Fang et al., 2025; Ye et al., 2025; Yang et al., 2025; Wang et al., 2023; Yang et al., 2026; Anthropic, 2025). Because form controls what retrieval can match and what survives truncation, this heterogeneity obscures a basic empirical question: holding experience and the retrieval–execution backbone fixed, what does a particular *form* buy, and at what context cost?

This paper asks a focused question:

> *Given the same source experience, which skill **form** transfers best under context constraints and distribution shift, and which forms are most context-efficient and maintainable?*

We treat form as an explicit variable. This framing is motivated by two observations. First, recent procedural-memory results already suggest a non-trivial form trade-off: script-like abstractions can generalize better, while trajectory-like traces excel on near-neighbor reuse (Fang et al., 2025). Second, industrial practice increasingly packages skills as editable artifacts with explicit metadata and optional resources, moving beyond free-form "strings in context" toward portable, schema-guided representations (Anthropic, 2025; Yang et al., 2026).

**Why form matters empirically.** Skill form controls what is made *explicit* versus left implicit, what retrieval can reliably match, what information survives truncation under context limits, and how easily skills can be curated (edited, versioned, migrated). Raw traces preserve local affordances and recovery patterns but are long and instance-specific; scripts are compact and often more invariant but may drop execution-critical details. Hierarchical forms expose compositional structure, while directory-backed artifacts separate metadata from optional references, enabling progressive loading and targeted edits. Despite these differences, prior comparisons rarely hold the source experience and the retrieval–execution backbone fixed across multiple forms, entangling representation with other design choices (retriever, prompting, update dynamics) and making attribution difficult (Liang et al., 2025).

**Our contribution.** We present a *train-free* protocol that converts the *same* source trajectories into four canonical forms and evaluates them under

controlled retrieval and injection budgets:

1. TRAJ: compressed execution traces;
2. SCRIPT: flat SOP-style scripts;
3. HIER: hierarchical skills separating planning and execution;
4. ARTIFACT: artifactized skill directories with `SKILL.md` and optional references.

Across all forms, we keep the retrieval key, retriever, base prompt, decoding settings, and evaluation protocol fixed. On ALFWorld (in-domain and unseen splits), this controlled comparison reveals mean-level trade-offs and yields actionable representation guidelines.

**Contributions in brief.** (1) A controlled, train-free protocol for converting one shared trajectory set into multiple skill forms. (2) A measured progressive-loading ablation for directory-backed artifact skills, comparing metadata-only, body-level, and full views under matched budgets. (3) Empirical evidence and analysis linking form choice to near-neighbor transfer, unseen-split behavior, and context efficiency under shared retrieval and prompting.

## 2 Controlled Protocol

We study skill-form transfer in an agent loop (Figure 1). Each source trajectory is first mapped to a *canonical experience* record with a stable retrieval key and shared metadata. That record is then serialized into different forms, so the injected content is the only varying component. At test time, a task query retrieves top-ranked skills and injects their form-specific content into a fixed executor prompt.

**Canonical experience record.** For every trajectory $\tau$, we construct a canonical record $(q, f, r)$ where $q$ is the natural-language task query, $f$ is a list of extracted constraints/facts, and $r$ is a bounded raw trace. Crucially, $(q, f, r)$ is shared across all forms; only the form-specific serialization differs (Section 3). This design ensures that differences in performance can be attributed to representational *form* rather than to changes in retrieval keys or metadata.

**Retrieval.** To isolate form effects, we use the same retrieval key $q$ across forms and keep the retriever fixed. Given a query $q$, retrieval returns a ranked list $\{(m_i, s_i)\}_{i=1}^{N}$ of memory items $m_i$ and similarity scores $s_i$. All forms share the same retrieval scores and ranking; only the injected representation differs. Appendix A gives the exact retriever specification.

**Injection budgets.** We compare two regimes that capture realistic context constraints:

- **Fixed-$k$**: retrieve and inject a fixed number of top-ranked items, treating each form's natural length as part of the trade-off;
- **Fixed-tokens**: enforce a hard budget over the injected skill block, so forms are compared under the same context allowance.

The fixed-$k$ regime exposes each form's *natural context footprint*, while fixed-tokens isolates how each representation allocates information under the same budget. Appendix A specifies the exact budget values and truncation rule.

**Progressive loading for artifacts.** For artifact skills, the injected representation can be assembled at different granularities: `meta` (frontmatter + file manifest), `body` (full `SKILL.md`), and `full` (`SKILL.md` plus referenced files such as `references/steps.md`). This enables a controlled ablation of where information should live in a directory-backed representation.

**Executor.** We use deterministic decoding and a fixed base prompt across all settings. The executor sees (i) the task goal, (ii) the injected skill block, and (iii) environment feedback, and must output admissible actions. This isolates representational differences from sampling variance and makes failure cases attributable to retrieval, representation, or environment complexity rather than decoding randomness.

**Controlled variables.** Across all comparisons we fix: (i) the set of source trajectories, (ii) the retrieval key and retriever, (iii) the budget regime, (iv) the executor prompt and decoding settings, and (v) the evaluator. The only primary change is the skill form (i.e., how the same experience is externalized into the injected representation).

## 3 Skill Forms

We define four forms that cover common representations in the literature and in practice (Figure 2). All forms are derived from the same canonical experience record, ensuring that retrieval keys and task metadata are matched.

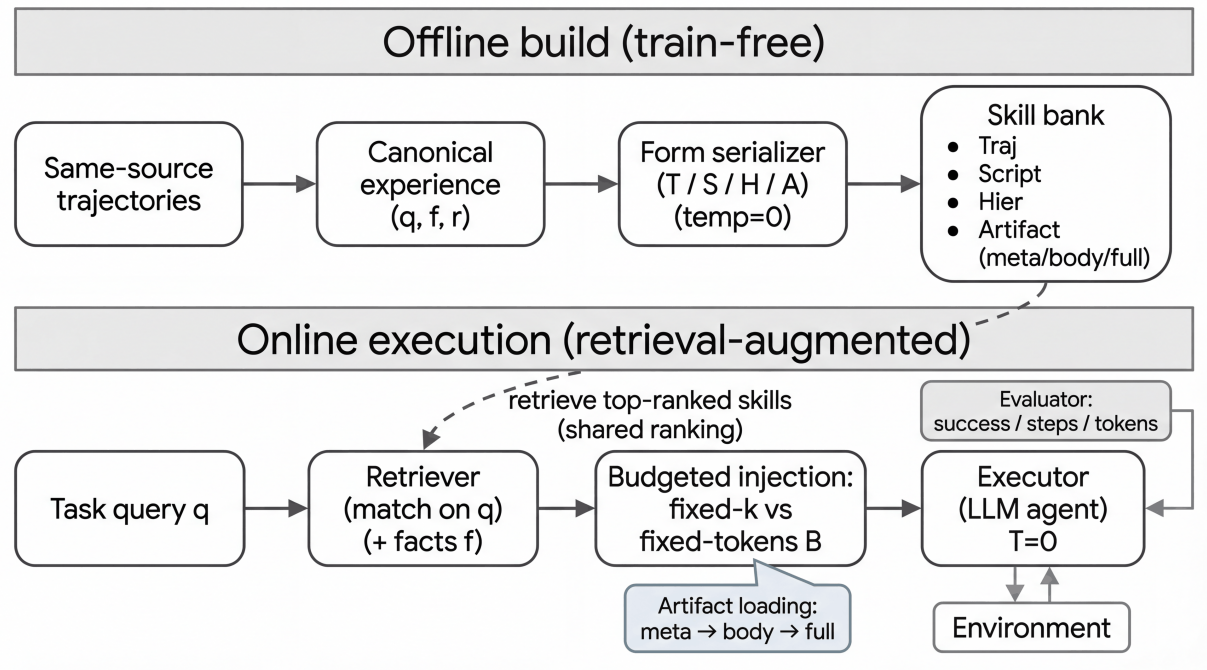


Figure 1: **Controlled form comparison.** We convert *the same* source trajectories into four skill forms, build a shared bank, retrieve by the same query key, and inject under two budget regimes (fixed top-$k$ vs. fixed injected-token budget). Artifact skills support progressive loading (`meta` → `body` → `full`).

| Form | Fidelity | Abstraction | Structure | Editability |
|---|---|---|---|---|
| TRAJ | **H** | L | L | L |
| SCRIPT | M | **H** | L | M |
| HIER | M | M | **H** | M |
| ARTIFACT | M | M | **H** | **H** |

**Hypothesized form-level inductive biases.** Coarse summary of what each form is designed to preserve and expose. Cells use an H/M/L scale (high/medium/low; shading encodes the same scale). ARTIFACTis hypothesized to score high on editability because information is organized into explicit fields and files.

**TRAJ: Trajectory trace.** The trajectory form stores a compressed execution trace that preserves local observations and actions, following trajectory-style agent memories and procedural traces (Fang et al., 2025; Liu et al., 2026). It favors high-fidelity reuse when the retrieved case is a close neighbor, but can be long and brittle under context limits.

**SCRIPT: Flat script (SOP).** The script form summarizes the same experience as an 8–12 step numbered procedure with explicit checkpoints and common failure modes, in the spirit of proceduralized scripts and textual skill abstractions (Fang et al., 2025; Zhao et al., 2023). This form emphasizes abstraction and compressibility, but may omit low-level details that matter for precise execution.

**HIER: Hierarchical skill.** Inspired by hierarchical memory and reflection (Ye et al., 2025; Yang et al., 2025), the hierarchical form separates *planning* from *execution*. It structures experience as a small set of subgoals (each with intent, how-to, stop condition, and pitfalls) followed by concise execution guidance. This form aims to preserve compositional structure while keeping low-level action advice accessible.

**ARTIFACT: Artifact directory.** The artifact form packages a skill as a local directory whose entrypoint is a `SKILL.md` file (YAML frontmatter + Markdown body), optionally accompanied by reference files (e.g., `references/steps.md`). This mirrors emerging "skill bank" practice (Anthropic, 2025; Yang et al., 2026): skills are portable, versionable, and editable, and can support *progressive loading* by injecting only metadata vs. full body vs. full body plus references. This directory structure supports *controlled loading*: under tight budgets, the executor can be conditioned on a compact metadata view, while detailed references are injected only when additional context is warranted; Appendix D reports the resulting ablation.

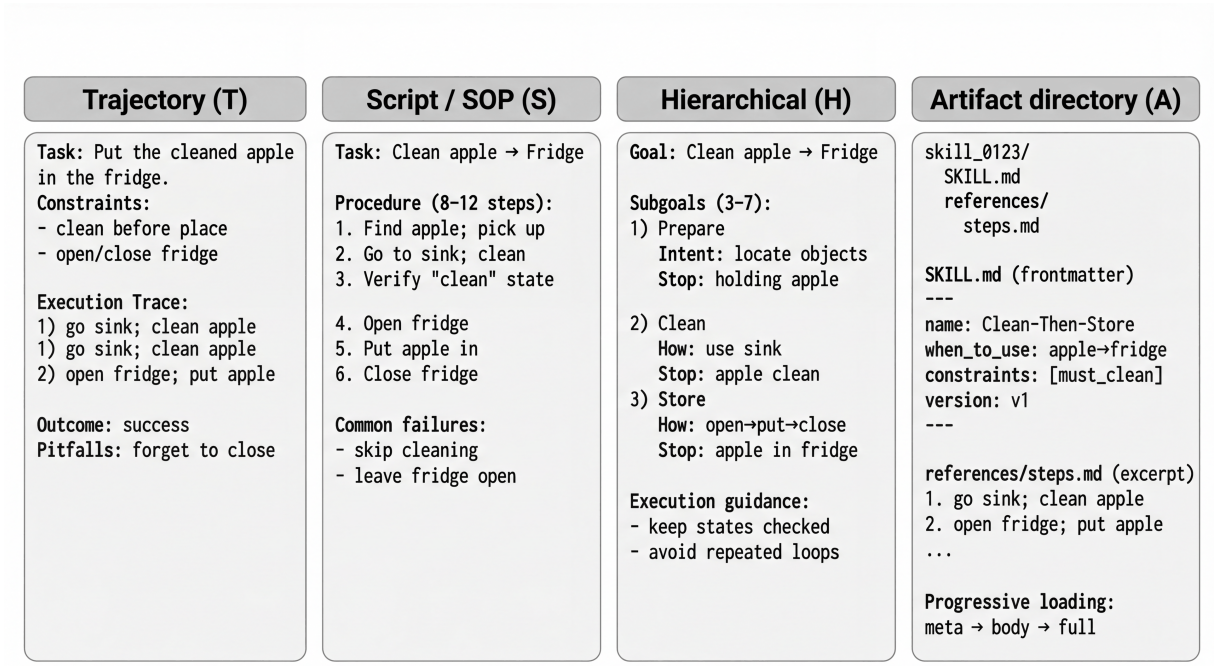


Figure 2: **Same experience, different forms.** One source episode becomes four forms. The artifact view is a directory centered on `SKILL.md` with optional reference files.

**Conversion is train-free.** All forms are derived from the same canonical experience record without gradient updates. Appendix B gives the exact serialization constraints and the `SKILL.md` schema.

## 4 Experimental Setup

**Benchmark.** We evaluate on ALFWorld (Shridhar et al., 2020), a text-based embodied environment where agents complete household tasks via admissible commands. We report results on the standard in-domain split (`eval_id`) and the OOD-style split (`eval_ood`); exact split sizes are given in Appendix A.

**Compared forms.** We compare four train-free skill forms: trajectory traces (TRAJ; Fang et al. 2025; Liu et al. 2026), flat procedural scripts (SCRIPT; Fang et al. 2025; Zhao et al. 2023), hierarchical skills (HIER; Ye et al. 2025; Yang et al. 2025), and directory-backed artifacts (ARTIFACT; Anthropic 2025; Yang et al. 2026).

**Controlled evaluation.** All forms are built from the same pool of successful training trajectories and share the same retrieval and execution backbone. For a fixed test query, the identity and rank of retrieved items are therefore comparable across forms; differences arise only from how those items are expressed. The exact bank size, retriever, budgets, and truncation rule are given in Appendix A.

**Metrics.** We report task success rate, average steps, and injected-token cost. Confidence intervals and the full reporting protocol are described in Appendix A.

## 5 Results

**Main finding: form interacts with distribution shift and budgets.** Table 1 and Figure 3 show mean-level trade-offs across regimes. Under a fixed injected-token budget, TRAJattains the highest mean success on the in-domain split (`eval_id`: 82.1%). On the unseen split, SCRIPTachieves the highest mean success under both budgets. In raw counts, however, some gaps are very small: for example, on `eval_ood` under fixed-tokens, SCRIPTsucceeds on 99/134 episodes versus 98/134 for TRAJ. Appendix D reports paired exact McNemar tests; none of the block winners is significantly better than all challengers at $p < 0.05$. We therefore treat the mean rankings as directional evidence and ask in Section 6 where those differences arise.

**Context efficiency: how many items fit under the same budget.** Even under fixed-tokens, forms differ in how many distinct retrieved items can be included. Table 2 shows that SCRIPTfits

| Split | Budget | Form | Succ. | Success (%) | Steps | Inj. toks. |
|---|---|---|---|---|---|---|
| eval_id | fixed-$k$ | TRAJ | 110/140 | 78.6 [72.1, 85.0] | 39.50 | 936.8 |
| | | SCRIPT | 109/140 | 77.9 [71.4, 84.3] | 38.04 | 730.2 |
| | | HIER | 112/140 | 80.0 [73.6, 86.4] | 37.44 | 1550.8 |
| | | ARTIFACT | 107/140 | 76.4 [69.3, 82.9] | 44.48 | 1100.7 |
| | fixed-tokens | TRAJ | 115/140 | 82.1 [76.4, 88.6] | 35.17 | 690.8 |
| | | SCRIPT | 107/140 | 76.4 [69.3, 83.6] | 37.88 | 693.3 |
| | | HIER | 109/140 | 77.9 [71.4, 84.3] | 36.11 | 690.0 |
| | | ARTIFACT | 106/140 | 75.7 [68.6, 82.9] | 45.73 | 682.9 |
| eval_ood | fixed-$k$ | TRAJ | 95/134 | 70.9 [62.7, 78.4] | 43.53 | 913.6 |
| | | SCRIPT | 100/134 | 74.6 [67.2, 81.3] | 42.09 | 745.6 |
| | | HIER | 95/134 | 70.9 [62.7, 78.4] | 42.92 | 1533.9 |
| | | ARTIFACT | 97/134 | 72.4 [64.2, 79.8] | 46.90 | 1094.6 |
| | fixed-tokens | TRAJ | 98/134 | 73.1 [65.7, 80.6] | 42.36 | 691.3 |
| | | SCRIPT | 99/134 | 73.9 [66.4, 81.3] | 41.02 | 692.9 |
| | | HIER | 95/134 | 70.9 [62.7, 78.4] | 43.89 | 689.7 |
| | | ARTIFACT | 95/134 | 70.9 [62.7, 77.6] | 47.04 | 682.9 |

Table 1: **ALFWorld main results.** Success rates (95% bootstrap CI), steps, and average injected tokens (estimated) for each skill form under fixed-$k$ retrieval ($k$=3) and fixed injected-token budget ($B$=700). We also report raw success counts. Shading marks the highest mean success rate within each split×budget block; paired exact McNemar tests are reported in Appendix Table 4. Query-based retrieval and deterministic execution are shared across settings.

| Form | Avg. items (ID) | Avg. items (OOD) | Trunc.% (ID) | Trunc.% (OOD) |
|---|---|---|---|---|
| TRAJ | 2.76 | 2.82 | 98.6 | 97.8 |
| SCRIPT | **3.40** | **3.30** | 98.6 | 97.8 |
| HIER | 2.00 | 2.00 | 100.0 | 100.0 |
| ARTIFACT | 2.00 | 2.00 | 100.0 | 100.0 |

Table 2: **Context efficiency under fixed-tokens.** With a fixed injected-token budget ($B$=700), different forms allow different numbers of retrieved items to fit. Truncation rate is the fraction of episodes where the injected block required truncating the last included item. Boldface marks the largest average number of included items per split.

the largest number of items on average (∼3.3–3.4), while HIERand ARTIFACTtypically fit only two. One plausible explanation for the unseen-split pattern is this "breadth" effect: within the same budget, the executor sees a more diverse set of precedents.

**Hierarchy and artifacts trade structure for context.** In the fixed-$k$ regime, HIERattains the best eval_id success (80.0%) but at a substantially larger injected context footprint (∼1,551 tokens on average), compared to SCRIPT(∼730) and TRAJ(∼937). Structure can improve decision quality but competes directly with context when retrieval injects multiple items. The artifact form occupies an intermediate region: it is competitive in success while providing a representation designed for portability and editing. Appendix D shows that this internal organization matters: metadata-only loading matches full loading on eval_id fixed-$k$ (both 108/140) and attains the highest mean on eval_ood fixed-tokens (101/134), indicating that progressive loading is useful as a representational choice rather than only as a systems convenience.

## 6 Analysis

We use two diagnostics to understand *why* forms differ: (i) retrieval similarity buckets and (ii) task-family breakdowns. Full breakdown tables are provided in Appendix D.

**Retrieval similarity reveals near-neighbor vs. abstraction regimes.** We bucket test episodes by the top-1 retrieval similarity score (low/mid/high). On the in-domain split, the pattern is clear: when retrieval is a strong match, trajectory-like guidance works well under a strict token budget. On the unseen split, the contrast is more local than universal: script-like abstraction is strongest in the mid-similarity bucket, but not in every low-similarity case.

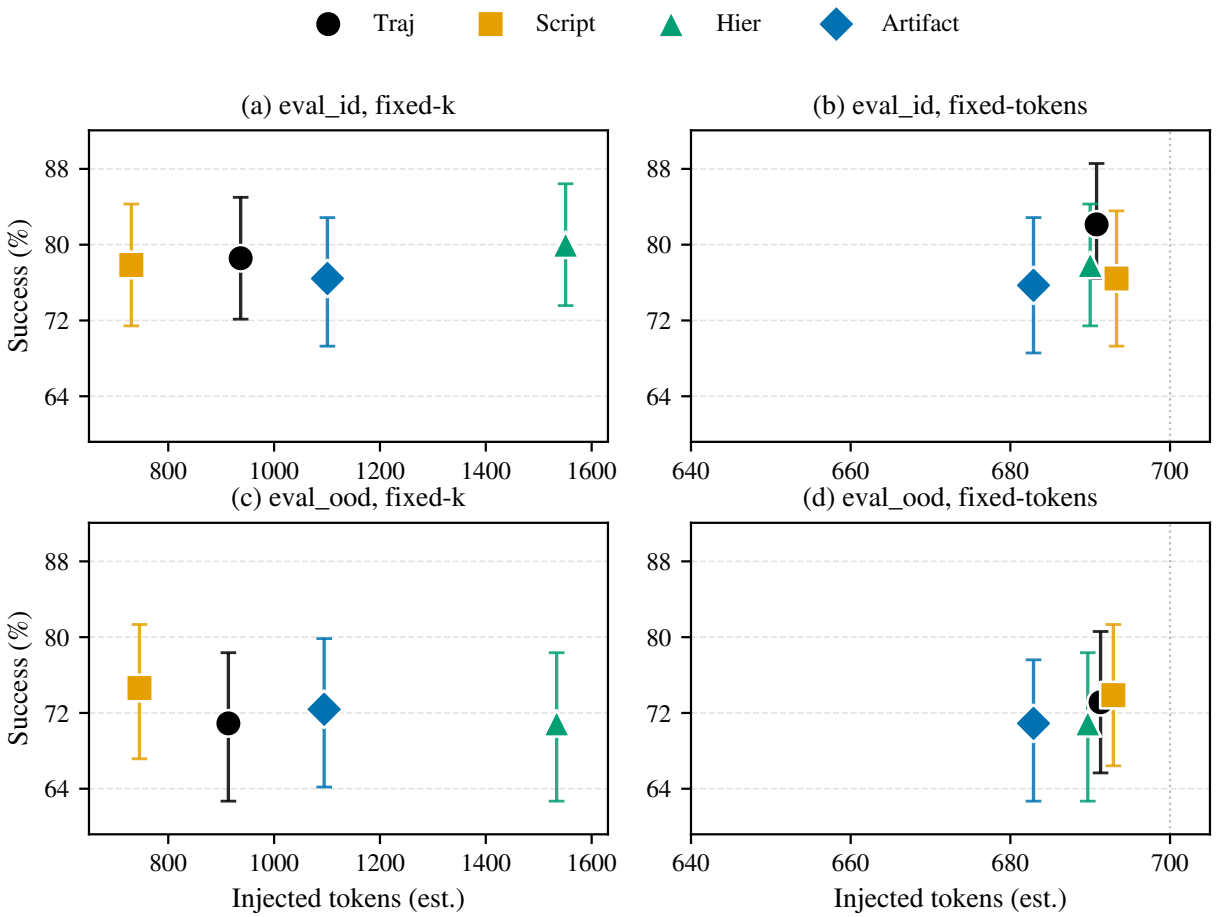


Figure 3: **Cost–performance trade-offs.** Each point is a skill form; $x$ is average injected tokens (estimated) and $y$ is success (%). Error bars show 95% bootstrap CIs. Panels: (a) `eval_id`, fixed-$k$; (b) `eval_id`, fixed-tokens ($B$=700); (c) `eval_ood`, fixed-$k$; (d) `eval_ood`, fixed-tokens. Fixed-$k$ reveals each form's natural context footprint; fixed-tokens enforces a strict injected-token budget (dotted line).

This supports the view that TRAJprimarily transfers *instance-level* procedural details, while SCRIPTtransfers *task-level* invariants. For example, under fixed-tokens on `eval_ood` (Table 7, panel (a)), SCRIPTimproves markedly from the low bucket (63.6%) to mid/high buckets (72.7–84.8%), while TRAJexhibits a sharper degradation in the mid bucket (61.4%).

**Task families expose execution vs. planning.** ALFWorld task types differ in whether success depends more on robust action sequences or on higher-level task structure. Descriptively, HIERis comparatively competitive on task families that require explicit transformations or prerequisite tracking. Its downside is context cost: under fixed-$k$, the larger injected block can crowd the prompt. These findings suggest that hierarchy is most useful when it compresses *decision structure*, not when it redundantly restates low-level commands. Table 7 (panel (b)) shows the same pattern. TRAJis especially strong on multi-object delivery, while SCRIPTand ARTIFACTare comparatively strong on inspection-style tasks and simple pick-and-place. Because some task-family buckets are small, we treat these breakdowns as descriptive evidence rather than as standalone significance claims.

**Why artifacts matter even when success is similar.** The artifact form exposes a complementary axis not captured by success alone: portability and editability. By distributing information across metadata, body, and optional references, artifact skills allow progressive loading and targeted edits without rewriting an unstructured blob. In this paper, we therefore treat it as a representational affordance rather than as a validated empirical advantage on ALFWorld alone (Appendix B).

**Truncation is the default under fixed-tokens.** Table 2 shows that under fixed-tokens, truncation

is nearly ubiquitous (98–100% of episodes), meaning the executor almost always sees a *prefix* of the retrieved skill content. This makes information ordering part of the representation: forms that surface constraints and invariants early are less sensitive to truncation, while forms that delay key details to later trace steps or reference sections may degrade. This observation also motivates progressive loading for artifacts: if detailed files are rarely reached under strict budgets, they should be loaded on demand rather than injected by default. Appendix Table 6 further shows that the main pattern is not only an item-count effect: even when we inject only the single top-ranked item, TRAJremains the weakest form on both splits.

**Failures are dominated by step-limit timeouts.** Across all 2,192 ALFWorld episodes, failures are dominated by hitting the 100-step environment cap rather than by invalid-action termination. This makes *step efficiency* a consequential secondary metric: representations that reduce dithering or enforce prerequisite structure can convert timeouts into successes even when the underlying action knowledge is similar. In Table 1, ARTIFACTuses more steps on average, suggesting that artifact organization benefits from staged loading or sharper execution cues to translate structure into efficient behavior.

## 7 Discussion

Within our ALFWorld setting under a shared lexical retriever, no single form dominates. Each form encodes a different inductive bias and behaves differently as retrieval quality and context constraints change. The comparison is therefore most useful as a guide to *when* each form is preferable. Because the pairwise gaps are modest, we treat these patterns as design tendencies rather than as universal prescriptions.

**Retrieval quality determines whether fidelity or abstraction helps.** When retrieval finds close neighbors, trajectory traces transfer precise local procedures efficiently under a strict budget. When similarity is weaker or the target distribution shifts, script abstractions tend to be more stable because they preserve task-level invariants rather than instance-specific details.

**Structure helps only when it compresses decisions.** Hierarchical representations are useful when failures stem from missing prerequisites or poorly organized subgoals, but their benefits weaken when structure is paid for with substantial context overhead. This is why hierarchy is most effective when it compresses decision structure rather than restating low-level commands that the trace or script already conveys.

**Artifacts matter when portability and maintenance matter.** Directory-backed artifacts are not the strongest representation on raw task success in our current ALFWorld runs, but they expose a different design axis: stable metadata can be separated from longer execution details, making skills easier to edit, version, and selectively load. This observation suggests a practical design strategy: keep a canonical experience record and materialize different views on demand, so truncation-sensitive settings can prioritize concise forms while curation-oriented settings retain structured artifact views.

**Form should be treated as a first-class variable.** Many agent comparisons change several ingredients at once, including the retriever, the memory source, and the update loop. Our results show that even when all of those components are held fixed, changing only the externalized form of the same experience can move the success–cost frontier. This makes skill form an experimental variable rather than a presentational detail. In practical systems, the strongest default may therefore be a bank that preserves one canonical record but can expose trace-like, script-like, or artifact-like views as retrieval quality and context budgets change.

## 8 Related Work

**Externalizing experience as memory or skills.** Many agent-memory systems distill experience into reusable textual memories or skills without parameter updates (Zhao et al., 2023; Liu et al., 2026). Procedural memory work highlights that the *shape* of stored experience matters: trajectories preserve fidelity, while scripts/proceduralizations emphasize abstraction (Fang et al., 2025). Recent surveys separate memory along axes such as form, function, and dynamics, suggesting that representational choices should be evaluated explicitly (Liang et al., 2025).

**Hierarchy and structured procedural knowledge.** Hierarchical memory and reflection separate planning-level knowledge from execution-level details (Ye et al., 2025; Yang et al., 2025).

Our hierarchical form borrows this design cue, but our goal is different: we do not introduce a new memory-update algorithm; instead, we compare hierarchy to other forms under matched source experience and a fixed retrieval backbone.

**Skills as code and as artifacts.** Several systems store skills as executable code or API-like routines (Wang et al., 2023; Zheng et al., 2025). In parallel, practical skill banks increasingly use filesystem-backed artifacts with explicit metadata and optional resources, enabling portability and human editing (Anthropic, 2025; Yang et al., 2026). Our artifact form operationalizes this industrial representation within a controlled study. It can be viewed as an explicit separation of *what* to do from *how* to do it, echoing arguments for polymorphic abstraction in skill learning (Yu et al., 2025).

**What we do not study.** We focus on train-free comparisons and do not study training-heavy skill selection or self-evolution, which confound representation with learning dynamics (Zhang et al., 2026; Gen-Verse, 2026).

**Long-term memory and improvement loops.** Our work is complementary to systems that emphasize long-term interaction memory or self-improvement loops, including persistent simulated societies (Park et al., 2023), explicit memory controllers (Packer et al., 2023), and reflective improvement signals (Shinn et al., 2023). Our contribution is to isolate one axis of that design space: the representation form of reusable experience under a controlled, train-free comparison.

## 9 Limitations

This study isolates skill form under a fixed retrieval and execution protocol, but several limitations remain. First, our main results are on a single embodied benchmark (ALFWorld); additional domains (e.g., tool-heavy planning) may yield different boundaries. Second, conversion quality is a potential confound: although converters are deterministic (temperature 0) and conditioned on matched canonical records, alternative prompts or stronger summarizers could shift absolute performance. Third, token costs are estimated; absolute counts may differ under other tokenizers, although comparisons are consistent within our reporting protocol. Fourth, we use a lexical retriever and a fixed injection policy, and we do not yet include cross-model transfer or external anchors such as no-skill and random-retrieval baselines. Finally, artifact directories primarily target portability and editability; quantifying these axes beyond task success (e.g., minimal-edit repair under new constraints) and systematically evaluating progressive loading remain important future work.

## 10 Conclusion

Skill representation is often treated as a secondary design choice, yet modern systems externalize experience in markedly different forms. We presented a train-free, controlled protocol that converts the same source trajectories into four skill forms and compares them under matched retrieval, prompting, and budget regimes. Our ALFWorld results show mean-level trade-offs rather than a universal winner: trajectory traces are strongest for near-neighbor reuse under tight budgets, scripts achieve the highest mean success on the unseen split, and hierarchical or artifactized forms add structure and maintenance affordances at non-trivial context cost. At the same time, paired gaps between forms are often modest, so the main contribution is controlled measurement of form effects, not a claim that one representation dominates. This suggests a simple design rule: keep one canonical experience record, then materialize different forms as retrieval quality, distribution shift, and context budget change. Under high-similarity retrieval and tight budgets, faithful traces are often the right interface; under weaker retrieval or stronger maintenance requirements, more abstract or more structured views become preferable. The broader point is methodological: form should not be treated as a hidden formatting choice inside agent memory systems. Future work should report the source experience, the externalized form, and the budgeted injection protocol as separate variables. For practical skill banks, a system can preserve one canonical record offline, then surface trace-like, script-like, or artifact-like views as retrieval quality and maintenance needs change.

## References

AgentMemory. 2025. Huaman-agent-memory: A curated repository of agent memory resources. GitHub repository.

Anthropic. 2025. Agent skills: Filesystem-backed skill artifacts (documentation and specification).

Online documentation/specification for `SKILL.md`-based skill directories.

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.

Shangbin Feng, Weijia Shi, Yike Wang, Wenxuan Ding, Vidhisha Balachandran, Yiming Yang, Yejin Choi, Noah A. Smith, and Luke Zettlemoyer. 2024. Don't hallucinate, abstain: Identifying LLM knowledge gaps via multi-LLM collaboration. *Preprint*, arXiv:2402.00367.

Tao Feng, Lizhen Qu, Niket Tandon, and Gholamreza Haffari. 2025. IRIS: An iterative and integrated framework for verifiable causal discovery in the absence of tabular data. *Preprint*, arXiv:2510.09217.

Gen-Verse. 2026. Openclaw-rl: Train any agent simply by talking. GitHub repository.

Jiafeng Liang, Hao Li, Chang Li, Jiaqi Zhou, Shixin Jiang, Zekun Wang, Changkai Ji, Zhihao Zhu, Runxuan Liu, Tao Ren, Jinlan Fu, See-Kiong Ng, Xia Liang, Ming Liu, and Bing Qin. 2025. Ai meets brain: Memory systems from cognitive neuroscience to autonomous agents. *Preprint*, arXiv:2512.23343.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. 2026. Simplemem: Efficient lifelong memory for LLM agents. *Preprint*, arXiv:2601.02553.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2023. Memgpt: Towards LLMs as operating systems. *Preprint*, arXiv:2310.08560.

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. *Preprint*, arXiv:2304.03442.

Bo Peng, Sirui Chen, Lei Xu, and Chaochao Lu. 2026. Causcientist: Teaching LLMs to respect data for causal discovery. *Preprint*, arXiv:2601.13614.

Xin Quan, Marco Valentino, Louise A. Dennis, and André Freitas. 2024. Verification and refinement of natural language explanations through LLM-symbolic theorem proving. *Preprint*, arXiv:2405.01379.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Preprint*, arXiv:2303.11366.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. ALFWorld: Aligning text and embodied environments for interactive learning. *Preprint*, arXiv:2010.03768.

Martin Tutek, Fateme Hashemi Chaleshtori, Ana Marasović, and Yonatan Belinkov. 2025. Measuring chain of thought faithfulness by unlearning reasoning steps. *Preprint*, arXiv:2502.14829.

Guangya Wan, Yunsheng Lu, Yuqi Wu, Mengxuan Hu, and Sheng Li. 2024. Large language models for causal discovery: Current landscape and future directions. *Preprint*, arXiv:2402.11068.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *Preprint*, arXiv:2305.16291.

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. 2025. Learning on the job: An experience-driven self-evolving agent for long-horizon tasks. *Preprint*, arXiv:2510.08002.

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.

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

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

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

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.

## A Reproducibility and Protocol

This appendix records the evaluation protocol, reporting conventions, and reproduction recipe used for the ALFWorld experiments. Table 3 summarizes the default settings referenced throughout the paper. Appendix B lists converter templates and the artifact schema; Appendix D provides additional breakdown tables.

| Component | Default setting |
|---|---|
| Data | ALFWorld TextWorld; bank built from train; evaluation on `eval_id` (140) and `eval_ood` (134). |
| Bank size | 100 source trajectories per form for the reported ALFWorld bank. |
| Canonical record | $(q, f, r)$ with query $q$, extracted facts $f$, bounded trace $r$ (first 50 steps; observations $\leq$200 chars), plus metadata. |
| Retriever | TF–IDF cosine over $q$ with English stopwords and at most 50k features. |
| Budgets | Fixed-$k$ ($k$=3) and fixed-tokens ($B$=700); max item length 900 tokens; last-item-only truncation with `[TRUNCATED]`. |
| Models | Executor: `gpt-5.2-low`; converter/bank builder: `gpt-5.4-high`; both called through an OpenAI-compatible endpoint. |
| Executor | Temperature 0, fixed base prompt, and a 100-step environment cap. |
| Metrics | Success, steps, injected tokens (estimated), and 95% bootstrap CI with 1,000 resamples. |

Table 3: **Default protocol settings.**

**Data and splits.** We use the TextWorld variant of ALFWorld (Shridhar et al., 2020). The skill bank is built from the training split. Evaluation uses `eval_id` ("valid_seen": 140 episodes) and `eval_ood` ("valid_unseen": 134 episodes).

**Source record.** Source trajectories are obtained by executing the provided walkthrough for each training instance, recording observation–action pairs. From each trajectory, we build a canonical record shared across forms: query $q$, extracted facts/constraints $f$, and a bounded raw trace $r$ (first 50 steps; observations truncated to 200 characters), plus metadata (e.g., task type and step count).

**Retriever.** We use a lexical retriever: TF–IDF with English stopwords (max 50k features), ranking items by cosine similarity. To isolate form effects, all forms share the same retrieval key ($q$) and ranking; only the injected representation changes.

**Budget regimes and truncation.** We compare (i) fixed-$k$ injection ($k$=3) and (ii) fixed injected-token budget ($B$=700) over the injected skill block. Each retrieved item is additionally capped (max item length 900 tokens) to prevent outliers. Under fixed-tokens, we concatenate items by rank until exceeding $B$; if the next item would exceed $B$, we truncate *only that last included item*, append `[TRUNCATED]`, and stop.

**Token estimation.** Token costs are computed by a deterministic estimator: when an OpenAI model name is provided, we use `tiktoken` encoding for that model; otherwise we fall back to a stable heuristic ($\approx$1.3$\times$ word count). All reported token numbers in the paper refer to this estimated count.

**Executor and determinism.** Execution uses temperature 0 and a fixed base prompt across all runs. We cap environment interaction to 100 steps per episode (timeouts terminate failures). We fix the random seed (42) for run ordering and bootstrapping.

**Bootstrap confidence intervals.** Success-rate CIs are 95% bootstrap intervals computed over episodes with 1,000 resamples.

**Paired tests.** For each split–budget block, we also report exact McNemar tests on paired binary outcomes from the same task instances. These tests are descriptive checks on whether the highest-mean form wins significantly more discordant episodes than each challenger.

**Reproduction recipe.** The pipeline uses four lightweight entry points:

- *run* for a single experimental setting;
- *analyze* for aggregate CSVs and plots;
- *make_tables.py* for LaTeX tables;
- *make_results_figures.py* for paper figures.

Typical usage is equally simple:

- run one setting by specifying the benchmark, split, skill form, and budget regime;
- aggregate logs from a results directory into summary CSVs and plots;
- regenerate paper tables and figures from the analysis directory.

## B Converter Constraints and Artifact Schema

**Canonical experience record.** All forms are generated from the same canonical record $(q, f, r)$: the task query $q$, extracted facts/constraints $f$, and a bounded raw trace $r$. This record is stored in each bank item and is *shared* across forms; converters only change how the injected skill is serialized.

**Form templates.** For reproducibility, we fix form-level templates that front-load constraints and checks:

- **TRAJ(trace):** task, constraints, bounded execution trace, outcome, and key pitfalls.
- **SCRIPT(script):** 8–12 numbered steps, each one sentence; explicit checkpoints; $\leq$5 failure modes.
- **HIER(hierarchy):** goal, prereqs, 3–7 subgoals, and concise execution guidance; each subgoal includes intent, how, stop condition, and pitfalls.
- **ARTIFACT(artifact):** a filesystem-backed directory with `SKILL.md` and optional reference files such as `steps.md`.

**Converter constraints.** Both LLM-based converters consume the same $(q, f, r)$ record and are run deterministically. The script converter is constrained to output: (i) a one-line task description, (ii) an 8–12 step SOP with one sentence per step, and (iii) at most five failure modes. The hierarchical converter is constrained to output: (i) a goal and prereqs/tools block, (ii) 3–7 subgoals, each with `Intent`, `How`, `Stop condition`, and `Pitfalls`, and (iii) concise execution guidance. These constraints are what make the form comparison reproducible; the exact wording of prompts is not used as an independent variable.

**Artifact directory layout.** Each artifact skill is a directory with a stable `SKILL.md` entrypoint and optional references:

```
<skill_id>/
  SKILL.md
  references/
    steps.md   (optional)
```

**`SKILL.md` schema.** Artifact skills follow a directory format with an entry file `SKILL.md` that contains YAML frontmatter and a Markdown body. This schema is inspired by public skill-artifact practice, but it is an experimental extension used for controlled comparison rather than a claim of exact compatibility with any external standard. The frontmatter stores four kinds of information: identity and usage, I/O, constraints and tooling, and provenance. Concretely, the frontmatter follows the template below (types are illustrative):

```
---
name: <string>
description: <string>
when_to_use: <string>
inputs: [<string>, ...]
outputs: [<string>, ...]
constraints: [<string>, ...]
tools: [<string>, ...]
tags: [<string>, ...]
version: <string>
source_task_ids: [<id>, ...]
---
```

The body then presents four short sections: Overview, high-level steps, troubleshooting notes, and a references manifest for optional attached files.

**Progressive loading variants.** Given an artifact directory, we define three injection granularities: `meta` (frontmatter + file list), `body` (full `SKILL.md`), and `full` (`SKILL.md` plus `references/steps.md` if present).

## C Additional Related-Work Pointers

To keep the main related-work narrative focused while preserving breadth, we list a few additional pointers grouped by theme:

- **Reliability signals and knowledge gaps:** abstention and "unknown unknowns" (Feng et al., 2024; Tutek et al., 2025).
- **Faithfulness and refinement of explanations:** verification/refinement workflows (Quan et al., 2024).
- **Verifiable reasoning and causal discovery (orthogonal to our agent setup):** Wan et al. (2024); Feng et al. (2025); Peng et al. (2026).
- **Community resource list:** curated pointers on agent memory (AgentMemory, 2025).

## D Additional Controls and Breakdowns

We provide paired significance tests, progressive-loading ablations, top-1 controls, and compact breakdowns for ALFWorld.

| Split | Budget | Winner | Challenger | Winner-only | Challenger-only | $p$ |
|---|---|---|---|---|---|---|
| eval_id | fixed-$k$ | HIER | ARTIFACT | 17 | 12 | 0.458 |
| | | HIER | SCRIPT | 15 | 12 | 0.701 |
| | | HIER | TRAJ | 12 | 10 | 0.832 |
| eval_id | fixed-tokens | TRAJ | ARTIFACT | 17 | 8 | 0.108 |
| | | TRAJ | HIER | 13 | 7 | 0.263 |
| | | TRAJ | SCRIPT | 16 | 8 | 0.152 |
| eval_ood | fixed-$k$ | SCRIPT | ARTIFACT | 9 | 6 | 0.607 |
| | | SCRIPT | HIER | 12 | 7 | 0.359 |
| | | SCRIPT | TRAJ | 11 | 6 | 0.332 |
| eval_ood | fixed-tokens | SCRIPT | ARTIFACT | 11 | 7 | 0.481 |
| | | SCRIPT | HIER | 11 | 7 | 0.481 |
| | | SCRIPT | TRAJ | 10 | 9 | 1.000 |

Table 4: **Paired exact McNemar tests against the highest-mean form in each ALFWorld block.** Winner-only and challenger-only count discordant episodes on the same task instances. Bold $p$-values indicate $p < 0.05$; in the current ALFWorld E1 results, no block winner reaches that threshold against all challengers.

| Split | Budget | Metric | Meta | Body | Full |
|---|---|---|---|---|---|
| eval_id | fixed-$k$ | Success | **77.1 [108/140]** | 76.4 [107/140] | **77.1 [108/140]** |
| | | Inj. toks. | 596.0 | 1100.7 | 1358.3 |
| | fixed-tokens | Success | 74.3 [104/140] | **75.7 [106/140]** | **75.7 [106/140]** |
| | | Inj. toks. | 694.3 | 682.9 | 687.0 |
| eval_ood | fixed-$k$ | Success | **73.1 [98/134]** | 72.4 [97/134] | **73.1 [98/134]** |
| | | Inj. toks. | 590.0 | 1094.6 | 1347.3 |
| | fixed-tokens | Success | **75.4 [101/134]** | 70.9 [95/134] | 71.6 [96/134] |
| | | Inj. toks. | 693.9 | 682.9 | 686.8 |

Table 5: **Artifact progressive-loading ablation.** Within the directory-backed artifact family, we compare `meta` (frontmatter + manifest), `body` (the main `SKILL.md` view used in E1), and `full` (`SKILL.md` plus referenced files). Cells report success (%) with raw counts in brackets; boldface marks the highest mean success within each split×budget block.

| Split | Metric | TRAJ | SCRIPT | HIER | ARTIFACT |
|---|---|---|---|---|---|
| eval_id | Success | 65.0 [91/140] | **74.3 [104/140]** | 73.6 [103/140] | 72.1 [101/140] |
| | Inj. toks. | 316.9 | 246.9 | 510.0 | 367.2 |
| eval_ood | Success | 63.4 [85/134] | 68.7 [92/134] | **70.2 [94/134]** | **70.2 [94/134]** |
| | Inj. toks. | 306.2 | 253.7 | 503.4 | 365.7 |

Table 6: **Top-1 control ($k$=1).** Each setting injects only the single highest-ranked retrieved item, reducing the confound that shorter forms may fit more retrieved items under larger budgets. Cells report success (%) with raw counts in brackets; boldface marks the highest mean success within each split.

**(a) Similarity buckets.**

| Split | Bucket | $n$ | TRAJ | SCRIPT | HIER | ARTIFACT |
|---|---|---|---|---|---|---|
| eval_id | low | 46 | **84.8** | 71.7 | 80.4 | 69.6 |
| | mid | 45 | **75.6** | 73.3 | **75.6** | 73.3 |
| | high | 49 | **85.7** | 83.7 | 77.5 | 83.7 |
| eval_ood | low | 44 | **72.7** | 63.6 | 65.9 | 61.4 |
| | mid | 44 | 61.4 | **72.7** | 59.1 | 70.5 |
| | high | 46 | 84.8 | 84.8 | **87.0** | 80.4 |

**(b) Task-type breakdown.**

| Split | Task type | $n$ | TRAJ | SCRIPT | HIER | ARTIFACT |
|---|---|---|---|---|---|---|
| eval_id | look_at_obj_in_light | 13 | 76.9 | 76.9 | 76.9 | **84.6** |
| | pick_and_place_simple | 35 | **88.6** | 85.7 | **88.6** | 82.9 |
| | pick_clean_then_place_in_recep | 27 | 74.1 | **77.8** | 70.4 | 66.7 |
| | pick_cool_then_place_in_recep | 25 | **80.0** | 64.0 | 76.0 | 68.0 |
| | pick_heat_then_place_in_recep | 16 | 75.0 | **81.2** | 75.0 | **81.2** |
| | pick_two_obj_and_place | 24 | **91.7** | 70.8 | 75.0 | 75.0 |
| eval_ood | look_at_obj_in_light | 18 | 77.8 | **83.3** | 72.2 | **83.3** |
| | pick_and_place_simple | 24 | 70.8 | **83.3** | 75.0 | 75.0 |
| | pick_clean_then_place_in_recep | 31 | **74.2** | **74.2** | 71.0 | 71.0 |
| | pick_cool_then_place_in_recep | 21 | **76.2** | **76.2** | 71.4 | 71.4 |
| | pick_heat_then_place_in_recep | 23 | 60.9 | 60.9 | **65.2** | 60.9 |
| | pick_two_obj_and_place | 17 | **82.3** | 64.7 | 70.6 | 64.7 |

Table 7: **Additional ALFWorld breakdowns (fixed-tokens).** (a) Success rates (%) by top-1 retrieval similarity bucket (low/mid/high; thresholds computed per split: `eval_id` $q_1 = 0.623$, $q_2 = 0.771$; `eval_ood` $q_1 = 0.584$, $q_2 = 0.677$). (b) Success rates (%) by ALFWorld task family. Boldface marks the row-wise best form.