PROCED-MEM: BENCHMARKING PROCEDURAL MEMORY RETRIEVAL IN LANGUAGE AGENTS ACROSS DOMAINS

Ishant K[*]
QpiAI

Aswanth Krishnan[*]
QpiAI

ABSTRACT

We introduce Proced-Mem, a benchmark for procedural memory retrieval in language agents with two sub-domains: text-based household tasks (ALFWorld) and real computer environments (OSWorld). Evaluating retrieval independently of downstream execution is critical because current agent evaluations conflate retrieval with planning and execution, masking whether agents retrieve relevant procedures or succeed despite poor memory access. Proced-Mem evaluates up to seven methods across text, visual, and lexical modalities, using an LLM-as-judge protocol for ALFWorld and a leave-one-out protocol with hierarchical ground truth at two granularity levels for OSWorld. Across both sub-domains, we find a *generalization cliff* (30–42% MAP degradation on novel contexts) and a *granularity-method reversal* where visual features rank first at coarse retrieval but last at fine-grained procedural matching. Proced-Mem provides the first diagnostic framework for identifying such failure modes, enabling the principled design of retrieval systems that generalize across granularity levels and modalities.

1 INTRODUCTION

An agent who learned to "clean an apple and place it in a cabinet" should recognize this experience as relevant when asked to "clean a salt shaker and place it in a drawer," despite an entirely different object vocabulary. Retrieving procedurally similar trajectories across contexts is essential for generalization in memory-augmented agent systems (Shinn et al., 2023; Fang et al., 2025; Han et al., 2025), yet no existing evaluation can distinguish retrieval systems that understand procedural structure from those that memorize surface-level lexical patterns.

The core problem is that current evaluations conflate retrieval with planning, grounding, and execution (Wang et al., 2023; Zhang et al., 2024; Lewis et al., 2020). An agent may retrieve wrong procedures yet succeed through robust planning, or retrieve right ones yet fail due to poor execution. This conflation masks a fundamental question: *can retrieval systems identify relevant procedures based only on structural similarity?*

We address this gap with Proced-Mem, a benchmark that evaluates procedural memory retrieval independently of execution across two complementary sub-domains:

ALFWorld Sub-domain. We construct a corpus from a text-based household environment (Shridhar et al., 2021) comprising 336 LLM-generated trajectories from AgentInstruct (zai-org, 2024), which forms the main benchmark. A separate set of 78 expert trajectories, generated by ALFWorld's HandCodedTWAgent, serves exclusively as an exploratory corpus for seen/unseen distribution-shift analysis. Using an LLM-as-judge evaluation protocol validated against human annotations, we evaluated six retrieval methods under distribution shift with a coverage-balanced query bank stratified by procedural complexity.

OSWorld Sub-domain. We curate 206 computer tasks from the `xlangai/ubuntu_osworld_verified_trajs` dataset (Xie et al., 2024; 2025), drawing from `claude-sonnet-4-5-20250929_15steps` model trajectories. These tasks span 15

[*]{ishant, ashwanth.krishnan}@qpiai.tech. Code and benchmark data: https://github.com/qpiai/Proced_mem_bench

application patterns (e.g., Chrome settings, terminal commands, spreadsheet operations) organized into a two-level hierarchy with 5 fine-grained procedural clusters. We evaluated seven retrieval methods that span text embeddings, visual (screenshot) embeddings, multimodal embeddings, and lexical baselines, using a leave-one-out protocol where every task serves as both query and corpus element.

Figure 1 illustrates the core challenge: tasks that share procedural structure can differ visually, while tasks that share a UI can follow entirely different procedures. Together, these sub-domains expose fundamental limitations of current retrieval architectures that are consistent across domains, modalities (text, visual, multimodal), embedding models (384-D all-MiniLM-L6-v2 and 2048-D Omni-Embed-Nemotron-3B), and evaluation protocols.

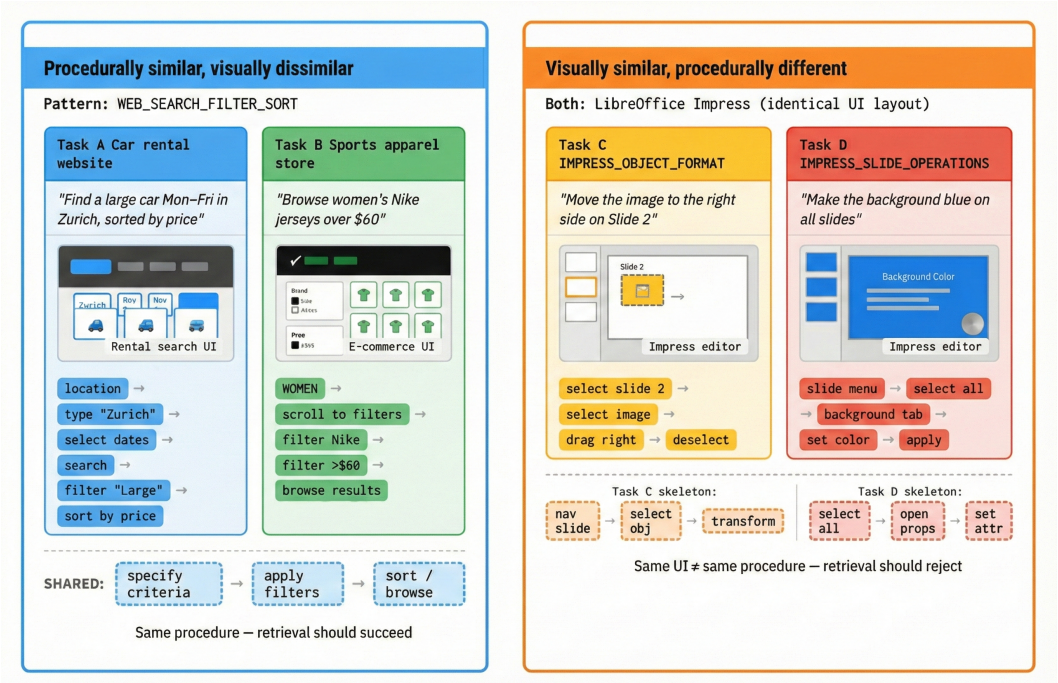

Figure 1: The visual similarity trap. **Left**: two tasks from WEB_SEARCH_FILTER_SORT share the same procedural skeleton (specify criteria, apply filters, sort/browse) despite different UIs. **Right**: two LibreOffice Impress tasks share identical UI layout but follow different procedures (object formatting vs. slide operations). Visual retrieval clusters the right pair together and separates the left pair, inverting the correct ranking.

Contributions.

1. Proced-Mem, a unified benchmark isolating procedural memory retrieval from end-to-end execution across two sub-domains: text-based household tasks (ALFWorld, 6 retrieval methods) and real computer environments (OSWorld, 7 methods spanning text, visual, and lexical modalities).
2. Discovery of a *generalization cliff*: embedding methods suffer 30–42% MAP degradation on out-of-distribution queries, while LLM-generated procedural abstractions degrade by only 11%.
3. Identification of a *granularity-method reversal* across modalities: visual embeddings rank first at coarse pattern matching but last at fine-grained procedural clustering, where action-only text representations dominate, revealing a *visual similarity trap* where screenshots capture application identity rather than procedural structure.

A preliminary version of this work covering only the ALFWorld sub-domain appeared as Kohar & Krishnan (2025). This paper extends it with the OSWorld sub-domain (206 computer tasks, 7

retrieval methods spanning text, visual, and multimodal modalities), the granularity-method reversal finding, and cross-domain analysis.

2 Related Work

2.1 Procedural Memory Systems for Agents

Procedural memory is a core component of agent architectures, yet existing systems vary in how they structure and access stored procedures. One line of work builds trajectory-level procedural stores: Memp (Fang et al., 2025) stores step-level instructions and higher-level abstractions from agent trajectories, while LEGOMem (Han et al., 2025) modularizes memory across multiple agents for workflow automation. A second line argues that procedural memory alone is insufficient: Wheeler and Jeunen (Wheeler & Jeunen, 2025) advocate integrating procedural, semantic, and associative memory, and MIRIX (Wang & Chen, 2025) operationalizes this view through six coordinated memory types managed by a meta-controller. Both lines assess performance through downstream task success rather than evaluating retrieval quality directly, the gap our benchmark addresses.

2.2 Retrieval Architectures and Memory Access

Agent retrieval mechanisms rely predominantly on sentence-transformer encoders (Reimers & Gurevych, 2019) or lexical baselines such as BM25 (Robertson & Zaragoza, 2009). TRACE (Chen et al., 2025) grounds time-series embeddings in aligned textual context. Hong and He (Hong & He, 2024) propose LLM-trained cross-attention networks for memory retrieval. PAL-UI (Sun et al., 2025) uses an active look-back mechanism for computer-use agent planning. These methods improve retrieval within agent pipelines but do not measure retrieval quality independently from downstream planning and execution.

2.3 Evaluation Gaps

Large-scale trajectory datasets such as AgentInstruct (zai-org, 2024), PAL-UI (Sun et al., 2025), and TrajAgent (Du et al., 2024) support agent training but do not isolate retrieval performance. The OSWorld-Verified trajectory dataset (Xie et al., 2025) provides verified model trajectories with screenshots and action sequences across 1000+ evaluation episodes, yet targets agent evaluation rather than retrieval benchmarking. We draw on its verified trajectory data as a source for our OS-World sub-domain. Zhang et al. (Zhang et al., 2024) identify memory as central to self-improving agents, yet existing evaluations assess memory implicitly through final task success. No prior work evaluates whether a retrieval system can recognize that two trajectories instantiate the same underlying procedure despite surface-level differences in objects, UI elements, or application contexts. Proced-Mem fills this gap with dedicated evaluation of procedural retrieval under distribution shift and at multiple granularity levels.

3 Benchmark Design

3.1 Problem Formulation

A *procedural trajectory* $t \in T$ is a sequence of state-action pairs $\langle (s_1, a_1), \ldots, (s_n, a_n) \rangle$ where $s_i \in S$ represents the environmental state and $a_i \in A$ represents the action taken. Given a corpus $\mathcal{C} = \{t_1, \ldots, t_N\}$ and a query q representing a novel task, the *procedural memory retrieval problem* requires a retrieval function $f : Q \times \mathcal{C} \to \mathcal{P}(\mathcal{C})$ that returns a ranked subset of trajectories maximizing procedural relevance:

$$\max_{f} \; \mathbb{E}_{q \sim Q} \left[\text{MAP}(f(q, \mathcal{C}), \text{rel}_{\text{proc}}(q)) \right] \tag{1}$$

where $\text{rel}_{\text{proc}}(q)$ denotes trajectories that are *procedurally* relevant to q, sharing the same underlying action structure regardless of surface-level differences such as object names and UI element identifiers.

3.2 ALFWORLD SUB-DOMAIN

Corpora. We construct two complementary corpora from ALFWorld (Shridhar et al., 2021), a text-based household environment. The *expert corpus* contains 78 trajectories generated by ALFWorld's HandCodedTWAgent, capturing realistic exploration patterns including 15 exploration-interrupted sequences. The *AgentInstruct corpus* contains 336 GPT-4-generated trajectories filtered from 954 initial traces (zai-org, 2024), providing a 4.3× scale increase. Both corpora span six ALFWorld task types.

Retrieval methods. We evaluate six text-only methods: (1) ACTION-ONLY EMBEDDINGS encoding raw action sequences, (2) ENRICHED EMBEDDINGS including task metadata, (3) SUMMARY EMBEDDINGS using LLM-generated procedural abstractions that remove object-specific references, (4) COMBINED EMBEDDINGS concatenating enriched representations with summaries, (5) BM25 lexical retrieval on state-action text, and (6) KEYWORD MATCHING using Jaccard similarity. All embedding methods use the all-MiniLM-L6-v2 sentence transformer (384-dimensional).

Evaluation. We employ an LLM-as-judge protocol (GPT-5) for scalable relevance scoring on a 1–10 scale, validated against manual annotation of 200 query-trajectory pairs achieving 89.5% specificity (OpenAI, 2025). Binary relevance uses threshold $\geq$6. The evaluation operates in two complementary regimes:

(a) **Dual-condition analysis (exploratory).** We evaluate 18 seen and 18 unseen queries against the 78 expert trajectories, measuring pool-based Recall@k at k=5. This regime diagnoses generalization: seen queries share vocabulary with corpus trajectories, while unseen queries introduce novel object combinations.

(b) **Coverage-balanced benchmark (main evaluation).** We evaluate 40 queries, stratified by procedural complexity (Easy/Medium/Hard) and filtered to ensure $\geq$8 relevant trajectories per query, against the full 336 AgentInstruct trajectories, measuring pool-based Recall@k at k=10.

3.3 OSWORLD SUB-DOMAIN

Corpus. We curate 206 computer tasks from OSWorld (Xie et al., 2024) spanning Chrome, LibreOffice Calc, LibreOffice Impress, Terminal, and GNOME applications. Each task is represented by its natural language instruction and an *abstracted action sequence*, a GPT-4o-generated procedural summary that captures the essential steps while removing specific UI element identifiers (e.g., "click settings icon → navigate to privacy section → toggle option").

Hierarchical ground truth. Tasks are organized into a two-level hierarchy:

- **Level 1 (L1):** 15 coarse-grained patterns defined by application and workflow type (e.g., `CHROME_SETTINGS_SIDEBAR`, `WEB_SEARCH_FILTER_SORT`, `CALC_CELL_OPERATIONS`), including 5 cross-application patterns (`MULTI_DOCUMENT`, `MULTI_WEB`, etc.). Mean pattern size: 13.7 tasks.
- **Level 2 (L2):** 5 fine-grained procedural clusters derived from hierarchical clustering within L1 patterns, capturing sub-procedural similarities (e.g., within terminal commands, distinguishing file manipulation from system configuration). Mean cluster size: 7.8 tasks.

Retrieval methods. We evaluate seven methods spanning three modalities (one more than ALFWorld's six text-only methods) because the OSWorld domain provides screenshot observations that enable visual retrieval:

- *Text embeddings*: INSTRUCTION-ONLY, INSTRUCTION+ACTIONS, and ACTIONS-ONLY encode task instructions, concatenated instruction–action text, or abstracted action sequences alone.
- *Visual embeddings* (OSWorld-specific): SCREENSHOT-ONLY mean-pools per-step screenshot embeddings across all post-action screenshots. SCREENSHOT+ACTIONS encodes each step as a multimodal (image, action-text) pair before mean-pooling.

- *Lexical baselines*: BM25 uses TF-IDF retrieval on action text; LCS computes Longest Common Subsequence similarity on normalized actions.

All embedding methods use nvidia/Omni-Embed-Nemotron-3B (2048-dimensional, bfloat16, L2-normalized), which supports both text and image input natively. BM25 and LCS are excluded from L2 evaluation because L2 clusters were derived via hierarchical clustering on the LCS similarity matrix, making their L2 results circular.

Leave-one-out evaluation. Each of the 206 tasks serves as both query and a corpus element. For each L1 round (206 total) or L2 round (39 total), the query task is removed, the remaining tasks form the corpus, and the retrieval is evaluated against the same-pattern (L1) or the same-cluster (L2) ground truth using full-corpus ranking. This protocol maximizes the effective query count and eliminates query selection bias. We precompute the full $N \times N$ similarity matrix once and slice per round for efficiency.

3.4 Evaluation Metrics

Both sub-domains report Mean Average Precision (MAP) as the primary metric, alongside Precision@k, Recall@k, and NDCG@k at $k \in \{5, 10\}$. The two sub-domains differ in retrieval scope: ALFWorld uses pool-based Recall@k (k=5 for the dual-condition analysis, k=10 for the coverage-balanced benchmark), where each query retrieves from a fixed candidate pool; OSWorld uses full-corpus leave-one-out evaluation, where each query retrieves from the entire remaining corpus. MAP captures ranking quality across the full retrieval depth:

$$\text{MAP} = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{|R_q|} \sum_{k=1}^{N} P@k \cdot \text{rel}_q(k) \tag{2}$$

4 Experimental Results

4.1 ALFWorld: The Generalization Cliff

Table 1 presents the dual-condition evaluation on 78 ALFWorld expert trajectories. All methods perform strongly on seen queries but degrade sharply on unseen queries with novel object vocabularies, a *generalization cliff*.

Table 1: ALFWorld dual-condition evaluation (78 expert trajectories, 18+18 queries). Summary Embeddings, which use LLM-generated procedural abstractions, show the smallest degradation (11%), while lexical methods (BM25, Keyword) suffer the largest drops (42%). Rank reversals between conditions indicate that in-distribution performance does not predict out-of-distribution robustness.

Method	**Seen**		**Unseen**		**Drop**
	MAP	P@1	MAP	P@1	(%)
Combined Embeddings	**0.844**	**0.87**	0.592	0.61	29.9
BM25	0.815	0.83	0.469	0.45	42.5
Enriched Embeddings	0.794	0.81	0.565	0.58	28.9
Keyword Matching	0.780	0.77	0.456	0.44	41.5
Action-Only Embeddings	0.756	0.74	0.488	0.49	35.5
Summary Embeddings	0.754	0.72	**0.671**	**0.69**	**11.0**

The ranking reverses: Summary Embeddings rank last on seen queries but first on unseen ones. These embeddings use LLM-generated procedural abstractions that remove object-specific references (e.g., "locate item $\rightarrow$ heat item $\rightarrow$ place in storage"), decoupling procedure identity from surface vocabulary. Their minimal degradation (-11% versus -42% for BM25) demonstrates that abstracting away surface-level object references produces more robust representations than retaining richer contextual detail.

Coverage-balanced validation. Table 2 validates these patterns on the full 336-trajectory AgentInstruct corpus using a coverage-balanced 40-query benchmark. Only embedding methods are evaluated here: the query selection procedure uses keyword matching to identify candidate trajectories before LLM validation, so evaluating BM25 or keyword-based methods would replicate the same signal used to construct the benchmark, introducing circularity.

Table 2: ALFWorld coverage-balanced benchmark (336 AgentInstruct trajectories, 40 queries stratified by complexity). State-aware embeddings outperform action-only overall but show competitive or inferior performance on medium-complexity tasks.

Representation	MAP	Easy	Medium	Hard
State-Aware	**0.795**	**0.842**	0.746	**0.791**
Action-Only	0.723	0.668	**0.802**	0.699

State-aware embeddings outperform action-only representations overall (0.795 vs. 0.723 MAP), particularly on simple tasks. Action-only embeddings achieve superior performance on medium-complexity procedures (0.802 vs. 0.746), suggesting that excessive state descriptions dilute the procedural signal for multi-step operations. A corpus size ablation (336 vs. 78 trajectories) shows that scale yields a 27.7% improvement versus only 9.9% from adding richer state context, establishing that corpus scale dominates representation enrichment.

4.2 OSWorld: Hierarchical Retrieval

Table 3 presents the leave-one-out evaluation on 206 computer tasks across seven retrieval methods at two granularity levels. All seven methods are reported at L1; at L2, BM25 and LCS are excluded due to circularity with ground truth construction (§3.3), leaving five valid methods.

Table 3: OSWorld leave-one-out evaluation across seven retrieval methods. L1 evaluates coarse pattern matching (206 queries, 15 patterns); L2 evaluates fine-grained procedural clustering (39 queries, 5 clusters). The **winning method reverses** between levels: Screenshot-Only at L1 versus Actions-Only at L2. Visual methods rank first at L1 but last at L2, revealing a *visual similarity trap*. A dash indicates circularity with L2 ground truth (see §3.3).

	L1: Pattern-Level			L2: Cluster-Level		
Method	MAP	P@5	NDCG@5	MAP	P@5	NDCG@5
Visual methods						
Screenshot-Only	**0.236**	0.564	0.588	0.134	0.190	0.221
Screenshot+Actions	0.218	0.534	0.555	0.126	0.205	0.220
Text embedding methods						
Instruction+Actions	0.227	**0.573**	**0.596**	0.231	0.349	0.391
Actions-Only	0.202	0.559	0.579	**0.339**	**0.477**	**0.520**
Instruction-Only	0.184	0.477	0.499	0.162	0.267	0.272
Lexical methods						
BM25 (actions)	0.190	0.512	0.523	—	—	—
LCS similarity	0.150	0.443	0.473	—	—	—

L1: Visual features capture application identity. Screenshot-Only achieves the highest L1 MAP (0.236), narrowly surpassing Instruction+Actions (0.227), though their 95% bootstrap confidence intervals overlap substantially ([0.207, 0.263] vs. [0.205, 0.252]; p=0.011).[1] Visually homogeneous patterns drive this advantage: `CHROME_SETTINGS_SIDEBAR` achieves MAP=0.938 under Screenshot-Only because all nine tasks in this pattern navigate Chrome's settings UI, which the visual encoder fingerprints almost perfectly. Excluding this single pattern (4.4% of queries)

[1] All confidence intervals computed via 1000 bootstrap resamples of queries at each level (α=0.05).

drops Screenshot-Only below Instruction+Actions, revealing that visual superiority at L1 depends on application-level UI consistency rather than procedural understanding. Among text methods, Instruction+Actions outperforms Actions-Only (0.202) by 12%, confirming that instruction semantics aid coarse category matching.

L2: Actions dominate, visual methods collapse. The ranking *inverts* at the cluster level. Actions-Only achieves 0.339 MAP [0.263, 0.408], while visual methods collapse to last place: Screenshot-Only drops to 0.134 [0.095, 0.175] and Screenshot+Actions to 0.126 [0.092, 0.165], both worse than Instruction-Only (0.162). The Actions-Only versus Screenshot-Only gap is statistically significant ($p<0.001$) with non-overlapping confidence intervals. This pattern is the *visual similarity trap*: screenshots cluster tasks by *where* they occur (same UI $\approx$ high similarity) rather than *what procedure* they follow (same UI $\neq$ same). A clean three-way stratification emerges: action text (0.339) $>$ semantic instruction text (0.231, 0.162) $>$ visual features (0.134, 0.126). Adding action text to screenshots (Screenshot+Actions) slightly *hurts* performance at both levels, suggesting the model conflates rather than fuses the two signals.

L2 ground truth caveat. The L2 clusters were derived from the similarity of LCS in action sequences, which may favor methods whose representations correlate with string-level action overlap. Actions-Only operates on neural embeddings rather than raw string matching, so the correlation is indirect, but its L2 advantage should be interpreted with this caveat. The reversal pattern, where visual methods drop from rank 1 (L1) to rank 5 (L2), is independent of this confounding: visual embeddings have no relationship to LCS-based clustering. Similarly, Instruction-Only (0.162), which does not encode action text and does not have a relationship to LCS-based clustering, also outperforms both visual methods at L2. The visual collapse is therefore the most robust L2 finding, supported by two independent, circularity-free comparisons.

Figure 2 visualizes this granularity-modality reversal across all five valid L2 methods.

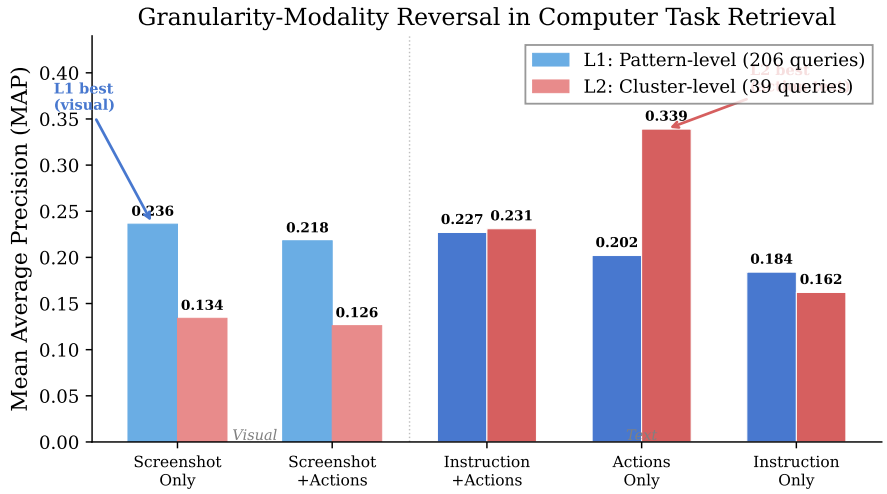

Figure 2: Granularity-modality reversal in computer task retrieval. Visual methods (Screenshot-Only) rank first at L1 but last at L2, while Actions-Only shows the opposite pattern. The more visual the representation, the worse it performs at fine-grained procedural retrieval.

Cross-application patterns are hardest. Among the L1 patterns, the five `MULTI_*` cross-application patterns (e.g., tasks combining browser navigation with document editing) achieve the lowest MAP per-pattern in all methods, including visual ones (MAP < 0.10). These patterns lack both distinctive vocabulary and distinctive visuals, leaving no surface-level signal for current methods to exploit.

4.3 CROSS-DOMAIN SYNTHESIS

Proced-Mem reveals a consistent finding across sub-domains, embedding models, and evaluation protocols: *current retrieval architectures fail to capture procedural structure*, regardless of input modality.

In the ALFWorld sub-domain, this limitation manifests as a generalization cliff: methods that memorize vocabulary co-occurrences (Combined Embeddings, 0.844 MAP on seen) collapse under distribution shift (-30%), while LLM-generated procedural abstractions (Summary Embeddings) maintain stability. This finding parallels practice in deployed coding agents, where tools such as Claude Code (Zhang et al., 2025) and Cursor (Anysphere, Inc., 2025) store procedural knowledge as abstracted natural-language summaries rather than raw interaction logs, independently converging on the Summary Embeddings approach. In the OSWorld sub-domain, the limitation manifests as a granularity-modality reversal: visual features capture application identity at the coarse level but fail at fine-grained procedural clustering, where only action text discriminates sub-procedural structure. The three-way L2 stratification (action text $>$ instruction text $>$ visual features) demonstrates that the higher the surface-level of the signal, the less useful it becomes for procedural retrieval.

Across both sub-domains, the consistency of these failures spans three modalities (text, visual, multimodal), two embedding models of vastly different capacity (384-D MiniLM vs. 2048-D Nemotron), and two evaluation protocols, suggesting an architectural bottleneck. All evaluated methods employ mean pooling over token or patch embeddings, which discards the temporal ordering that may be critical for procedural understanding.

5 DISCUSSION

5.1 ANALYSIS: MEAN POOLING AS A CANDIDATE BOTTLENECK

All evaluated retrieval methods rely on mean pooling over token embeddings, the standard aggregation strategy in sentence-transformer architectures (Reimers & Gurevych, 2019). This operation computes a fixed-size representation by averaging all token (or patch) embeddings:

$$\text{embed}(S) = \frac{1}{n} \sum_{i=1}^{n} e(w_i) \tag{3}$$

For two action sequences $S_1 = [a_1 \to a_2 \to a_3]$ and $S_2 = [a_2 \to a_1 \to a_3]$, which are procedurally distinct yet nearly identical under mean pooling, this aggregation removes the temporal ordering that distinguishes one procedure from another. For visual inputs, the same pooling operates over screenshot patch embeddings, producing a representation that captures *what application is on screen* (spatial layout, color scheme, UI elements) rather than *what sequence of actions was performed*. This architectural property is consistent with the visual similarity trap: screenshots from the same application produce similar mean-pooled representations regardless of procedural differences between tasks.

Mean pooling is consistent with four observations from our benchmarks: (1) the generalization gap arises because vocabulary memorization is fragile under distribution shift, while temporal structure (which pooling discards) would transfer across object vocabularies; (2) adding contextual information yields only modest gains (9.9% improvement from state-aware over action-only format), because mean pooling dilutes additional tokens into the same fixed-dimensional average; (3) action-only text outperforms richer representations at fine granularity, where subtle step order differences carry the discriminative signal; and (4) visual methods achieve their strongest results on visually homogeneous patterns (`CHROME_SETTINGS_SIDEBAR`: MAP=0.938) but collapse when procedural similarity diverges from visual similarity.

Preliminary experiments with a graph neural network (GNN) on a 39-task Chrome subset drawn from the OSWorld corpus provide initial evidence for this hypothesis: encoding procedural dependencies as graph structure improves the retrieval MAP from 0.685 (mean-pooled baseline) to 0.769, a 12.3% gain. Although this result is in a limited subset and requires validation on the full benchmark, it suggests that preserving temporal structure in the representation yields measurable retrieval improvements.

5.2 Implications for Memory System Design

Our findings motivate three architectural directions for procedural memory systems:

Structure-aware encoders. Architectures that preserve temporal dependencies would better represent procedural structure than mean-pooled encoders. The GNN results in §5.1 provide initial evidence: encoding procedural dependencies as the graph structure improves MAP by 12.3%, suggesting that step ordering carries a discriminative signal that flat embeddings discard. The failure of visual embeddings in L2 confirms that richer input modalities alone do not compensate for architectural limitations in temporal modeling.

Two-stage retrieval. Separating structure extraction from similarity computation would avoid forcing static embeddings to encode both content and order. An initial stage extracts action graphs or procedural dependency trees; a second stage computes similarity over these structures using graph-matching methods, retrieving over procedural graphs rather than flat embedding vectors.

Adaptive granularity and modality. No single modality dominates across retrieval levels. A practical system would use visual or semantic retrieval for coarse filtering (L1) followed by action-based reranking for fine-grained matching (L2), exploiting the complementary strengths our benchmarks quantify.

5.3 Future Work

The two-level hierarchy evaluated here naturally extends to finer granularities. Subtask-level retrieval (L2+) would use LLM-based functional similarity judgments, and step-level retrieval (L4) would define ground truth via per-step screenshot similarity. The L2 reversal, where the discriminating signal shifts from semantic (*what to do*) to procedural (*how to do it*), suggests that this trend will strengthen at finer levels.

More broadly, retrieval is one component of a full procedural memory pipeline. Current approaches reuse document-retrieval primitives without adapting them to the temporal and compositional structure of procedures (Yang et al., 2026). A complete system requires procedurally-aware rerankers that score candidates by step-level alignment rather than aggregate similarity, structured representations such as knowledge graphs that capture sub-procedural dependencies, and update rules governing how stored procedures evolve as the agent encounters new task variations. Proced-Mem provides a diagnostic substrate for the development and evaluation of these pipeline components.

6 Limitations

Our benchmark covers two domains (ALFWorld, OSWorld); verification in robotics, software workflows, or multi-agent settings remains future work. The ALFWorld LLM's as a judge shows moderate human agreement (Cohen's $\kappa = 0.178$), reflecting a construct mismatch where humans accept partial procedural utility that the LLM's stricter criteria reject. The OSWorld L2 ground truth derives from LCS-based clustering, constraining valid comparison methods at that level. Our results characterize off-the-shelf encoders; fine-tuning could improve quality, but requires labeled procedural similarity data, a resource our benchmark now provides.

7 Conclusion

We introduced Proced-Mem, a benchmark isolating procedural memory retrieval from end-to-end agent execution across two sub-domains (ALFWorld, OSWorld) with up to seven methods spanning text, visual, and lexical modalities. Across both sub-domains, we identify consistent failures: a generalization cliff (30–42% MAP degradation) and a granularity-method reversal where visual features rank first at coarse retrieval but collapse at fine-grained procedural clustering. These failures are consistent with mean pooling discarding temporal structure. Proced-Mem provides diagnostic tools for developing retrieval architectures that model action ordering and procedural dependencies.

REFERENCES

Anysphere, Inc. Rules — Cursor docs, 2025. URL https://cursor.com/docs/context/rules. Accessed: 2026-03-11.

Jialin Chen, Ziyu Zhao, Gaukhar Nurbek, Aosong Feng, Ali Maatouk, Leandros Tassiulas, Yifeng Gao, and Rex Ying. Trace: Grounding time series in context for multimodal embedding and retrieval. *arXiv preprint arXiv:2506.09114*, 2025. URL https://arxiv.org/abs/2506.09114.

Yuwei Du, Jie Feng, Jie Zhao, and Yong Li. Trajagent: An LLM-agent framework for trajectory modeling via large-and-small model collaboration. *arXiv preprint arXiv:2410.20445*, 2024. URL https://arxiv.org/abs/2410.20445.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. *arXiv preprint arXiv:2508.06433*, 2025. URL https://arxiv.org/abs/2508.06433.

Dongge Han, Camille Couturier, Daniel Madrigal Diaz, Xuchao Zhang, Victor Rühle, and Saravan Rajmohan. Legomem: Modular procedural memory for multi-agent LLM systems for workflow automation. *arXiv preprint arXiv:2510.04851*, 2025. URL https://arxiv.org/abs/2510.04851.

Yucheng Hong and He He. Enhancing memory retrieval in generative agents through LLM-trained cross attention networks. *arXiv preprint arXiv:2402.11729*, 2024. URL https://arxiv.org/abs/2402.11729.

Ishant Kohar and Aswanth Krishnan. A benchmark for procedural memory retrieval in language agents. *arXiv preprint arXiv:2511.21730*, 2025. URL https://arxiv.org/abs/2511.21730.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. *arXiv preprint arXiv:2005.11401*, 2020. URL https://arxiv.org/abs/2005.11401.

OpenAI. OpenAI GPT-5 system card. *arXiv preprint arXiv:2601.03267*, 2025. URL https://arxiv.org/abs/2601.03267.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *EMNLP-IJCNLP*, pp. 3982–3992, 2019. URL https://arxiv.org/abs/1908.10084.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019.

Noah Shinn, Joseph Cassano, Aaron Labash, and Matthew A Finlayson. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023. URL https://arxiv.org/abs/2303.11366.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.03768.

Shuo Sun, Lingfeng Yang, Xingyao Wang, Wenhao Zhang, Xuehai Pan, Hang Su, and Jun Zhu. Pal-ui: Planning with active look-back for vision-based GUI agents. *arXiv preprint arXiv:2510.00413*, 2025. URL https://arxiv.org/abs/2510.00413.

Guanzhi Wang et al. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023. URL https://arxiv.org/abs/2305.16291.

Yu Wang and Xi Chen. Mirix: Multi-agent memory system for LLM-based agents. *arXiv preprint arXiv:2507.07957*, 2025. URL https://arxiv.org/abs/2507.07957.

Andrew Wheeler and Olivier Jeunen. Procedural memory is not all you need. *arXiv preprint arXiv:2505.03434*, 2025. URL https://arxiv.org/abs/2505.03434.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shi, Joel Lu, et al. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972*, 2024. URL https://arxiv.org/abs/2404.07972.

Tianbao Xie, Mengqi Yuan, Danyang Zhang, Xinzhuang Xiong, Zhennan Shen, Zilong Zhou, Xinyuan Wang, Yanxu Chen, Jiaqi Deng, Junda Chen, Bowen Wang, Haoyuan Wu, Jixuan Chen, Junli Wang, Dunjie Lu, Hao Hu, and Tao Yu. Introducing OSWorld-verified trajectories, 2025. URL https://huggingface.co/datasets/xlangai/ubuntu_osworld_verified_trajs.

Chang Yang, Chuang Zhou, Yilin Xiao, Su Dong, Luyao Zhuang, Yujing Zhang, Zhu Wang, Zijin Hong, Zheng Yuan, Zhishang Xiang, Shengyuan Chen, Huachi Zhou, Qinggang Zhang, Ninghao Liu, Jinsong Su, Xinrun Wang, Yi Chang, and Xiao Huang. Graph-based agent memory: Taxonomy, techniques, and applications. *arXiv preprint arXiv:2602.05665*, 2026. URL https://arxiv.org/abs/2602.05665.

zai-org. Agentinstruct dataset, 2024. URL https://huggingface.co/datasets/zai-org/AgentInstruct.

Barry Zhang, Keith Lazuka, and Mahesh Murag. Equipping agents for the real world with agent skills, 2025. URL https://claude.com/blog/equipping-agents-for-the-real-world-with-agent-skills. Anthropic Engineering Blog. Accessed: 2026-03-11.

Yuan Zhang et al. A survey on memory mechanisms in LLM-based agents. *arXiv preprint arXiv:2404.13501*, 2024. URL https://arxiv.org/abs/2404.13501.

A Additional Results

Corpus size ablation (ALFWorld). Comparing the entire 336-trajectory AgentInstruct corpus with 78-trajectory stratified subsamples (averaged over three random seeds), the entire corpus achieves 0.795 MAP versus 0.644 MAP for subsamples, a 23.4% improvement with 81% retention. This establishes that the corpus scale delivers larger gains than representation enrichment (9.9% from the state-aware format vs. action-only format).

Semantic task space analysis (ALFWorld). Embedding 506 ALFWorld task descriptions reveals minimal separation between seen and unseen groups (within-group similarity: 0.323 vs. cross-group: 0.319, Cohen's $d = 0.026$, Silhouette score $s = 0.006$). This confirms that performance gaps arise from retrieval limitations, not intrinsic query difficulty differences.

Coverage-balanced query stratification (ALFWorld). The 40-query coverage-balanced benchmark stratifies queries into three complexity tiers: Easy ($n = 15$) consists of single-object placement tasks (e.g., pick up an object and place it in a receptacle); Medium ($n = 14$) consists of multi-step state-change tasks requiring intermediate actions (e.g., heat an object before placing it); Hard ($n = 11$) consists of multi-object coordination tasks that require the manipulation of multiple objects or sequential use of multiple receptacles. Each level guarantees $\geq$8 relevant trajectories per query to ensure stable precision and recall estimates. State-aware embeddings dominate on Easy (0.842 vs. 0.668 MAP) and Hard (0.791 vs. 0.699 MAP) tasks, while action-only embeddings achieve superior performance on Medium tasks (0.802 vs. 0.746 MAP), suggesting that state descriptions dilute the procedural signal for multi-step operations.

Per-pattern analysis (OSWorld). Among L1 patterns, single-application patterns achieve a MAP ranging from 0.09 to 0.67. `CHROME_SETTINGS_SIDEBAR` is easiest for Screenshot-Only (MAP=0.938) due to its visually homogeneous UI, but excluding this pattern drops Screenshot-Only below Instruction+Actions at L1. `TERMINAL_COMMANDS` shows high P@5 (0.68) but low

MAP (0.14) due to its large pattern size ($n = 39$). The five `MULTI_*` cross-application patterns average MAP < 0.10 across all methods, including visual ones, confirming that cross-application procedural retrieval remains a significant open challenge.

B LLM USAGE DISCLOSURE

In accordance with ICLR 2026 policy, we disclose the following uses of large language models in this work. **Methodology:** GPT-5 served as the LLM-as-judge for scalable relevance scoring in the ALFWorld evaluation protocol (§3.2). GPT-4o generated abstracted action sequences for the OSWorld sub-domain (§3.3), summarizing raw GUI trajectories into procedural step descriptions. **Writing:** LLMs assisted with drafting and editing portions of the paper text. All claims, experimental results, and scientific conclusions were verified by the authors.