SPECULATE WITH MEMORY: LOSSLESS ACCELERATION FOR LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative execution accelerates LLM agents by using a smaller, cheaper model to predict and pre-launch the next step while the environment is idle. However, existing speculators are stateless and discard all information between tasks, preventing prediction quality from improving with experience. We equip the speculator with three online memory systems that learn from past agent trajectories: a contrastive transition table tracking action-sequence statistics, an episodic memory retrieving contextually similar segments, and a confusion tracker suppressing recurring errors. We evaluate this approach on six benchmarks spanning three speculation types: action prediction, observation prediction, and chained prediction. Memory-augmented speculation yields a 19–39% relative accuracy improvement on action prediction and up to a $2.5\times$ increase on observation prediction tasks with repetitive action spaces. These gains grow continuously as memory accumulates and generalize across speculator models of varying cost. All speculation is lossless because it runs during idle time at zero added wall-clock cost, and the actor's trajectory is identical to non-speculative execution.

1 INTRODUCTION

Large language model (LLM) agents interact with external environments through tool calls, browser actions, and physical commands, where each step incurs latency from both the LLM inference and the action execution. In interactive settings such as customer service, web navigation, and embodied control, this sequential latency accumulates and directly affects user experience. Speculative execution addresses this bottleneck by using a smaller, faster model to predict the agent's next action and pre-launch it before the agent commits (Ye et al., 2025). The idea draws on speculative execution in processor architecture and speculative decoding in LLM inference (Leviathan et al., 2023). If the prediction is correct, the result is already available and latency is hidden. If incorrect, the pre-launched work is discarded at no functional cost. However, most speculative execution methods for LLM agents are **stateless**, treating every task independently. Recent work mines tool-calling patterns offline (Sui et al., 2026), but no existing method retrieves contextually similar past trajectories or adapts to its own prediction failures during deployment. A customer-service speculator that has processed 500 prior interactions makes the same predictions as one that has seen none. Yet agent tasks exhibit strong regularity. The same action sequences recur across episodes: looking up a user followed by retrieving their order in customer service, or navigating to a shelf followed by picking up a mug in household tasks. A system that remembers these patterns should predict more accurately over time.

We augment the speculator with structured memory that accumulates online from past agent trajectories. Drawing on recent advances in agent memory systems (Wang et al., 2025; Zhao et al., 2024; Ouyang et al., 2026), we equip the speculator with three complementary components that capture aggregate workflow patterns, retrieve contextually similar past episodes as in-context examples, and correct recurring prediction errors. While the actor is busy, the speculator predicts the outcome and pre-launches the next step along each predicted branch. When the actor's real output arrives, the system checks whether any speculative branch matches. On a hit, the pre-launched work is already complete, so the agent advances immediately instead of issuing a fresh blocking call. On a miss, the speculative work is discarded and execution proceeds normally, so the actor's trajectory is identical to non-speculative execution. Pre-launched work is restricted to side-effect-free operations (LLM calls, read-only API queries, page fetches), ensuring that incorrect predictions never produce

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

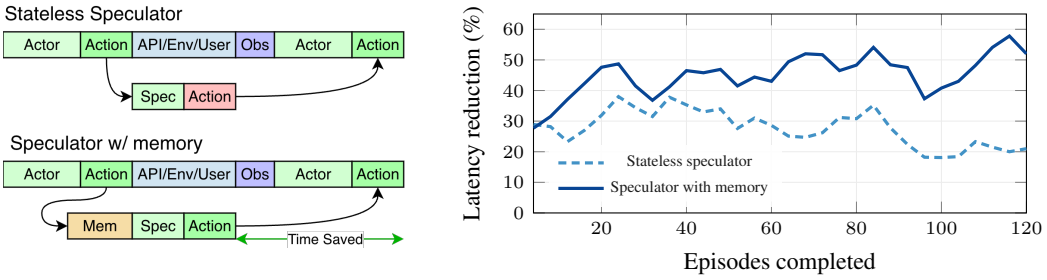

Figure 1: **Left:** The speculator predicts the next output during environment idle time and pre-launches it. A hit skips the actor LLM call; a miss discards the speculative work. Memory of past trajectories increases the hit rate. **Right:** Estimated latency reduction on ALFWorld (10-episode sliding window), counting each correct speculation as hiding the latency of one actor LLM call. The stateless speculator stays near $\sim$28%; with memory the reduction grows to over 50%.

irreversible changes and preserving the lossless guarantee (Section 2.1). Only pre-launchable predictions, those whose execution can overlap with and hide latency, translate into wall-clock savings. We evaluate seven progressive memory settings on six benchmarks spanning web navigation (Zhou et al.; Koh et al., 2024), embodied control (Shridhar et al., 2021), planning (Vallati et al., 2015), customer service (Yao et al., 2024; Barres et al., 2025), and multi-hop QA (Yang et al., 2018).

Memory-augmented speculation consistently outperforms the stateless baseline across all six benchmarks. The magnitude varies by speculation type: 19–39% relative improvement on action-prediction tasks and up to 2.5$\times$ on observation-prediction tasks with structured action spaces. Figure 1 (left) illustrates the mechanism. During environment idle time (API calls, user responses, page loads), the speculator predicts the next output and pre-launches it. Memory of past trajectories increases the hit rate. Figure 1 (right) shows the estimated latency reduction on ALFWorld over 120 episodes. Each correct speculation hides the latency of one actor LLM call, and the memory-augmented speculator improves from $\sim$28% to over 50% as it accumulates experience, while the stateless baseline remains flat. The improvements are model-agnostic and generalize across speculator models of varying capability and cost, and the optimal memory configuration varies by domain. Our contributions are as follows:

- We augment speculative execution for LLM agents with three complementary memory systems that accumulate experience from past trajectories: a contrastive transition table, an episodic memory with miss-episode recording, and a confusion tracker.
- We identify three speculation types based on where latency falls in the agent-environment loop, unifying prior work under a single framework.
- We evaluate seven progressive memory settings across six benchmarks and three speculation types, showing consistent improvement over stateless speculation, with the largest gains on observation-prediction benchmarks with structured action spaces (up to 2.5$\times$).

2 Speculate with Memory

2.1 Speculation Framework

Consider an LLM agent interacting with an environment in a turn-based loop. At each turn t, the agent observes state s_t, comprising the conversation history and environment feedback, and selects an action $a_t = (\mathit{type}, \mathit{args})$ via an LLM call with latency ℓ_{LLM}. The environment returns observation o_t with latency ℓ_{env}, which encompasses waiting for API responses, user input, or environment state computation. A *speculator* is a smaller, faster model that predicts future outputs in this loop. Let ℓ_{spec} denote the speculator's inference latency. Depending on the prediction target, the speculator runs either during environment processing or in parallel with the actor. In both cases, its cost is fully hidden because it overlaps with ongoing computation. With k parallel speculators, a best-of-k strategy succeeds if any single prediction matches.

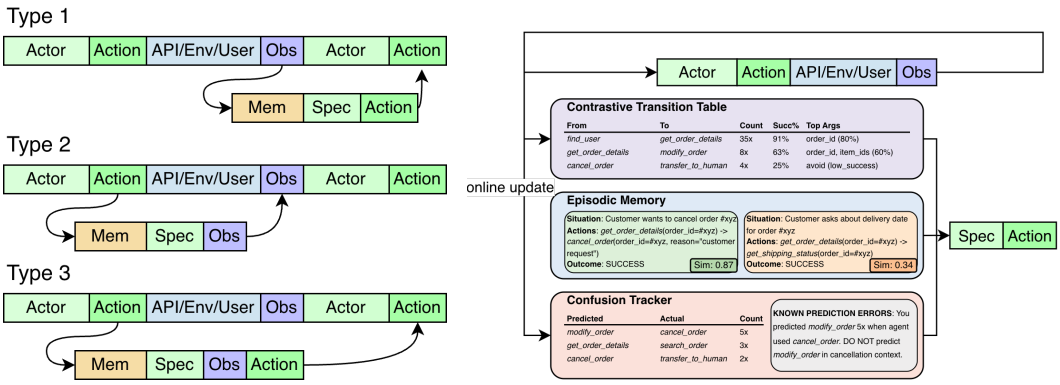

Figure 2: **Left:** Timing of the three speculation types. Each row shows the agent-environment loop (top) and the speculator's parallel work (bottom); shaded regions indicate saved latency on a hit. **Right:** Memory-augmented speculation architecture. The agent loop feeds online updates to three memory components (contrastive transition table, episodic memory, and confusion tracker) that provide structured context to the speculator. Examples are from τ^2-bench.

What to predict, and what latency is saved on a hit, depends on the structure of the agent-environment loop. We identify three speculation types. In *action prediction* (Type 1), the speculator predicts the agent's next action $\hat{a}_{t+1}$ in parallel with the actor. On a hit, the predicted action was already pre-launched, saving up to ℓ_{env}. The realized saving is $\min(\ell_{\text{env}},\ \ell_{\text{LLM}} - \ell_{\text{spec}})$ since the speculator must finish within the actor's ℓ_{LLM} window to pre-launch. In *observation prediction* (Type 2), the speculator predicts the environment's next observation $\hat{o}_t$ during environment processing. On a hit, the actor's next LLM call was already pre-started with the predicted observation, saving up to ℓ_{LLM}. The realized saving is $\min(\ell_{\text{LLM}},\ \ell_{\text{env}} - \ell_{\text{spec}})$. Therefore, Type 2 is most effective when environment feedback is slow relative to the speculator, such as in robot execution, user input delays, or heavy API calls. In *chained prediction* (Type 3), the speculator predicts both the observation and the next action during environment processing, chaining Type 2 and Type 1 in a single round. On a full hit, both the actor call and the action are pre-computed, saving up to $\ell_{\text{LLM}} + \ell_{\text{env}}$. In all cases, the saving approaches its upper bound when $\ell_{\text{spec}} \ll \ell_{\text{window}}$, which motivates using a smaller, faster speculator model. Figure 2 (left) illustrates the three types. The choice of type is determined by which steps incur significant latency and whether the environment's response is tractable to predict.

A correct prediction is useful only if the predicted work can be safely initiated before verification completes. For Type 2, the pre-launched work is an actor LLM call with predicted input. This is always safe since no environment state is modified. For Types 1 and 3, the pre-launched work is action execution, which is safe only for read-only actions such as API queries, page navigations, and database reads. Write actions with irreversible side effects are excluded. In practice, each benchmark defines a static whitelist of read-only action types derived from its API schema or action space. The speculator's predicted action type is checked against this whitelist before pre-launching. In all cases, we require a full exact match. Both the action type and all parameters must agree, because wrong parameters cannot be safely pre-launched.

2.2 Memory Systems

We equip the speculator with three complementary memory systems that learn from past agent trajectories (Figure 2, right): a contrastive transition table, an episodic memory, and a confusion tracker.

Contrastive Transition Table. Agent trajectories contain reusable action patterns (Wang et al., 2025; Fang et al., 2025; Wang et al., 2023). We capture these in a transition table $\mathcal{T}$ indexed by action-type pairs (a_i, a_j). For each observed transition in a completed trajectory $\tau = [(a^{(1)}, \text{args}^{(1)}), \ldots, (a^{(n)}, \text{args}^{(n)})]$ with outcome label $y \in \{+, -\}$, we record:

$$\forall\, 1 \le t < n: \quad n^{y}_{a^{(t)},a^{(t+1)}} \mathrel{+}= 1, \text{ArgSig}(a^{(t)}, a^{(t+1)}) \leftarrow \text{ArgSig}(a^{(t)}, a^{(t+1)}) \cup \{\text{sorted_keys}(\text{args}^{(t+1)})\} \tag{1}$$

where n^{+}_{ij} and n^{-}_{ij} are success and failure counts, and ArgSig accumulates the sorted parameter names observed for action a_j when preceded by a_i (e.g., `order_id,reason`). The table provides transition confidence and per-transition success rate:

$$\text{Confidence}(a_i \to a_j) = \frac{n^{+}_{ij} + n^{-}_{ij}}{\sum_k (n^{+}_{ik} + n^{-}_{ik})} \qquad \text{SuccessRate}(a_i \to a_j) = \frac{n^{+}_{ij}}{n^{+}_{ij} + n^{-}_{ij}} \tag{2}$$

Storing success and failure counts separately provides a contrastive signal. This is motivated by work showing that contrasting successful versus failed trajectories yields richer learning signals (Zhao et al., 2024; Ouyang et al., 2026), and enables the speculator to prioritize high-confidence, high-success-rate continuations. The table is fundamentally unconditional: it captures $P(\text{next_action} \mid \text{current_action})$ but not the dependence on conversational context or specific argument values.

Episodic Memory. To capture context-dependent patterns that the transition table misses, we store full trajectory segments from past episodes in an episodic memory $\mathcal{E}$ (Lewis et al., 2020; Zheng et al., 2023; Kagaya et al., 2024; Wang et al., 2025). Each episode e is a trajectory segment from a completed task:

$$e = (\textit{context}, \textit{actions}, \textit{observation}, \textit{arg_sources}, \textit{outcome}, \textit{lesson}) \tag{3}$$

where *context* is a natural-language summary of the agent's current situation, *actions* is the list of actions with full arguments, *observation* is the environment response, *arg_sources* maps each argument value to its provenance by substring matching against prior tool observations, *outcome* indicates task success, and *lesson* is a takeaway auto-generated from episode metadata via template rules. All episode fields are constructed by rule-based heuristics without auxiliary LLM calls, meaning memory updates add zero inference latency. At speculation time, the current context is embedded using OpenAI's `text-embedding-3-small` model (1536 dimensions) and the top-k most similar episodes are retrieved via cosine similarity, providing the speculator with structured examples of similar past situations. The only retrieval-time API call is this embedding query (<50 ms), which runs during environment idle time alongside the speculator. We also record *speculation-miss episodes* (Ding, 2026; Ouyang et al., 2026) that store the predicted versus actual action, giving the speculator explicit examples of its own past errors. Finally, retrieval-time pruning inspired by Evo-Memory (Wei et al., 2025) did not improve accuracy and is omitted from our reported results (see Appendix A.8).

Confusion Tracker. The confusion tracker identifies recurring prediction errors (Shinn et al., 2023; Zhao et al., 2024). After each speculation verification, we record the (predicted_action_type, actual_action_type) pair. When a specific confusion pattern is observed ≥ 3 times, it is surfaced as a hard constraint in the speculator's prompt, explicitly instructing the speculator to avoid the confused action type. Unlike episodic memory, which requires embedding retrieval, the confusion tracker is a lightweight dictionary of counts that provides immediate feedback as patterns cross the threshold.

All three memory systems are updated online after each completed task, so subsequent speculations benefit from accumulated experience. Speculation itself never affects the actor. The actor sees identical context regardless of prediction outcomes. Correct predictions use the cached result. Incorrect ones are discarded and execution proceeds normally, preserving the trajectory exactly as it would be without speculation.

Table 1: Benchmark characteristics grouped by speculation type.

Type	Benchmark	Metric	Tasks	Win	Action space	Pattern type
1	WebArena	RO action match	812	52.3%	Browser actions	Page navigation
	VWA	RO action match	910	54.9%	Browser actions	Visual navigation
2	ALFWorld	Obs. match	134	100%	13 verbs	Task-type templates
	PDDL	Obs. match	60	83.3%	Grid commands	State transitions
3	τ^2-bench	RO action match	164	74.4%	$\sim$30 APIs	API call sequences
	HotpotQA	Action match	300	67.0%	3 ReAct actions	Multi-hop QA

3 EXPERIMENTS

3.1 SETUP

We evaluate on six benchmarks spanning four agent modalities: customer service, web navigation, embodied tasks, and multi-hop question answering (Table 1). The benchmarks are grouped by speculation type: two benchmarks per type.

Benchmarks. *Type 1 (action prediction):* WebArena (Zhou et al.) tests web navigation across shopping, forum, and content management sites using 812 tasks. VisualWebArena (VWA) (Koh et al., 2024) extends WebArena with visual grounding on 910 tasks. *Type 2 (observation prediction):* ALFWorld (Shridhar et al., 2021) is a household task benchmark with a discrete 13-verb action space over 134 `valid_unseen` episodes. PDDL planning tasks (Vallati et al., 2015; Ma et al., 2024) require agents to navigate grid worlds and solve structured puzzles across 60 tasks. *Type 3 (chained prediction):* τ^2-bench (Barres et al., 2025) is a dual-control customer-service benchmark; we use both the retail and airline domains, totaling 164 tasks. HotpotQA (Yang et al., 2018) is a multi-hop question answering benchmark where a ReAct agent (Yao et al., 2023) iteratively searches Wikipedia over 300 tasks.

Evaluation. Each speculation type determines the evaluation metric. For Type 1, we evaluate read-only action full match; for Type 2, observation exact match. For Type 3, the speculator predicts the environment response and then the agent's next action; we use action full match as the primary metric since it determines pre-launchability. We report *aggregate accuracy*: total correct steps divided by total steps, pooled across all tasks. For τ^2-bench, we exclude turn 0 from the denominator since the user's identity is initially unknown.

Models. Actor trajectories are collected once and replayed, eliminating actor variance across settings and isolating API costs to the speculator. The actor model is GPT-5.4 for ALFWorld, PDDL, τ^2-bench, and HotpotQA, and GPT-5-mini for WebArena and VWA. The primary speculator is GPT-4.1-mini. We also leverage LLMs for visualization of experimental results.

Ablation settings. We evaluate seven progressive memory settings, all running in parallel during environment idle time with $n = 3$ speculator instances per setting: (1) **Stateless**: conversation history only, no memory; (2) **Confusion**: adds confusion pattern constraints. To build cumulatively, all subsequent settings include this confusion tracking as a base layer: (3) **Table**: adds transition table statistics; (4) **Episodic**: replaces the table with retrieved past episodes to isolate retrieval effects; (5) **Table+Episodic**: combines table and episodic memory; (6) **Episodic+Miss**: identical to Episodic but adds speculation-miss episodes; (7) **Full**: all components combined. A best-of-k evaluation checks whether any instance predicted correctly. The prediction targets match the speculation types: Type 1 predicts actions, Type 2 predicts observations, and Type 3 chains both.

External memory baselines. We compare against AWM-Spec and RB-Spec, which adapt Agent Workflow Memory (Wang et al., 2025) and ReasoningBank (Ouyang et al., 2026) to the speculator. AWM-Spec induces reusable workflows from successful prior trajectories. RB-Spec extracts reasoning lessons from successful and failed trajectories and retrieves the most similar prior task

Table 2: Main results: aggregate accuracy (%) and improvement over Stateless (Δ, pp) across all six benchmarks. The table also reports the AWM-Spec and RB-Spec baselines. Columns are grouped by speculation type. Best value per benchmark in **bold**. Cell shading indicates magnitude of Δ. All Stateless→Best improvements are significant ($p < 0.001$, McNemar's test over all evaluation steps; see Table 7).

	Type 1: Action pred.				**Type 2: Obs. pred.**				**Type 3: Chained**			
	WebArena		VWA		ALFWorld		PDDL		τ^2-bench		HotpotQA	
	Acc	Δ	Acc	Δ	Acc	Δ	Acc	Δ	Acc	Δ	Acc	Δ
Stateless	19.8	—	12.5	—	16.3	—	17.6	—	12.7	—	20.5	—
AWM-Spec	20.2	+0.4	13.8	+1.3	17.5	+1.2	18.0	+0.4	18.2	+5.5	15.7	−4.8
RB-Spec	20.3	+0.5	12.0	−0.5	16.0	−0.3	16.8	−0.8	19.7	+7.0	16.7	−3.8
Confusion	21.4	+1.6	15.3	+2.8	16.0	−0.3	19.0	+1.4	13.1	+0.4	21.4	+0.9
Table	21.9	+2.1	15.8	+3.3	19.5	+3.2	21.1	+3.5	15.2	+2.5	21.1	+0.6
Episodic	23.4	+3.6	**17.4**	**+4.9**	23.8	+7.5	19.0	+1.4	17.0	+4.3	25.0	+4.5
Table+Episodic	23.5	+3.7	16.6	+4.1	28.1	+11.8	20.0	+2.4	17.6	+4.9	**27.0**	**+6.5**
Episodic+Miss	**23.7**	**+3.9**	16.7	+4.2	38.6	+22.3	33.0	+15.4	16.8	+4.1	26.0	+5.5
Full	23.5	+3.7	16.2	+3.7	**40.0**	**+23.7**	**33.8**	**+16.2**	**19.9**	**+7.2**	**27.5**	**+7.0**

bundle. We adapt both methods to our speculative-prediction setting and evaluate them on the same benchmarks using GPT-4.1-mini as the speculator.

3.2 Main Results

Table 2 summarizes speculation accuracy across all six benchmarks for seven ablation settings and two adapted memory baselines. Memory-augmented speculation improves over Stateless on every benchmark, with the best setting varying by domain and speculation type. Full exceeds both AWM-Spec and RB-Spec on all six benchmarks. RB-Spec comes closest on τ^2-bench (19.7% vs. 19.9%), whereas both baselines trail Stateless on HotpotQA.

Type 1: Action prediction (WebArena, VWA). On WebArena, Episodic+Miss achieves 23.7%, +3.9 pp over Stateless at 19.8%. The confusion tracker, transition table, and episodic memory each contribute; the largest increments are Stateless→Confusion (+1.6 pp) and the Table→Episodic swap (+1.5 pp). Miss episodes add only +0.3 pp, indicating that web-navigation errors are less systematic. On VWA, Episodic achieves 17.4%, +4.9 pp over Stateless. Confusion alone provides +2.8 pp, while adding the transition table to episodic memory reduces accuracy by 0.8 pp, suggesting that episodic memory is preferable for diverse action spaces.

Type 2: Observation prediction (ALFWorld, PDDL). On ALFWorld, Full achieves 40.0%, 2.5× Stateless at 16.3%, the largest relative improvement across all benchmarks. Miss episodes provide the largest gain (+14.8 pp over Episodic), followed by episodic memory (+7.8 pp over Confusion) and the transition table (+3.5 pp over Confusion). On PDDL, Full achieves 33.8%, 1.9× Stateless. Adding miss episodes to Episodic yields +14.0 pp, again showing that failure learning dominates in structured environments with systematic errors.

Type 3: Chained prediction (τ^2-bench, HotpotQA). On τ^2-bench, Full achieves 19.9%, +7.2 pp over Stateless. Confusion adds +0.4 pp, the transition table adds +2.1 pp over Confusion, and episodic memory adds a further +2.4 pp over Table. On HotpotQA, Full achieves 27.5%, +7.0 pp over Stateless. The gain comes from recurring decisions about when to search, look up, or finish, which episodic memory captures.

Across benchmarks, each component has a distinct profile. Confusion tracking helps on five of six benchmarks but slightly reduces accuracy on ALFWorld. The transition table benefits domains with regular action sequences, while episodic memory provides consistent gains by supplying argument values and observation templates from similar situations. Although the best settings achieve only 17–40% accuracy, speculation is lossless: incorrect predictions are discarded, so every hit hides

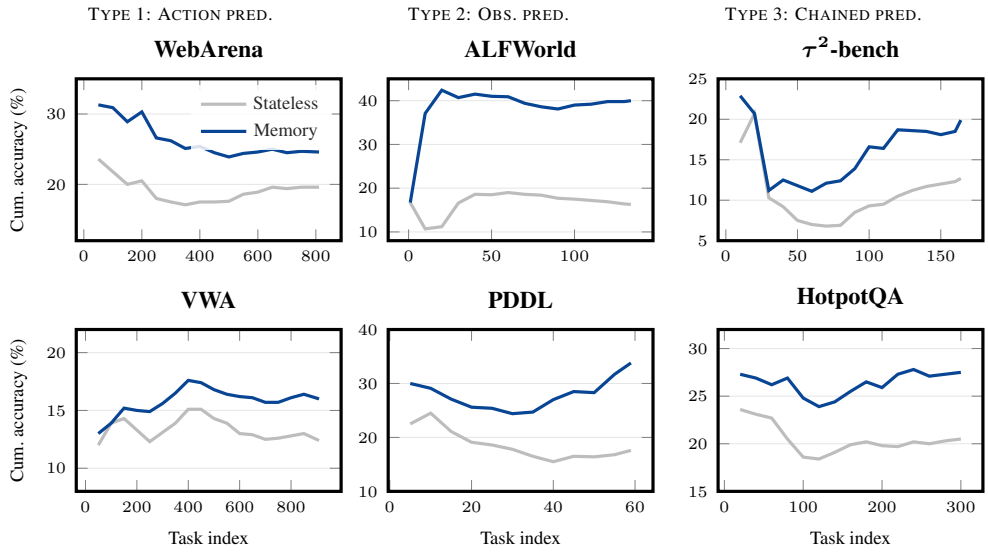

Figure 3: Cumulative accuracy ($k = 1$) across all six benchmarks, grouped by speculation type. Each point shows aggregate accuracy over all tasks up to that index. "Memory" denotes the best-performing memory setting per benchmark. The learning effect is strongest on Type 2 benchmarks, where memory climbs steadily while the stateless baseline remains flat. All seven settings are shown in Appendix A.5.

latency. At 20% accuracy with $\ell_{\text{hit}} = 7$ s, memory hides roughly 140 s per 100 steps. Section 4.3 provides the full analysis; because the speculator is much cheaper than the actor, speculation yields a net benefit above approximately 5% accuracy.

4 ANALYSIS

4.1 EXPERIENCE ACCUMULATION

Figure 3 plots cumulative accuracy as tasks accumulate across all six benchmarks, comparing the stateless speculator against the best-performing memory setting per benchmark. The accumulation effect varies by speculation type. On Type 2 benchmarks, where repetitive action sequences provide reusable patterns, accuracy grows steadily with experience. On ALFWorld, memory quickly climbs to ∼40% and stabilizes, while Stateless stays flat around 16%. On PDDL, memory rises from ∼24% to 34% in the second half, while Stateless remains below 18%. On Type 1 benchmarks, memory maintains a consistent 3–5 pp advantage throughout without a strong upward trend, indicating that gains come primarily from the memory content itself rather than continued accumulation. On Type 3 benchmarks, memory maintains a consistent advantage on τ^2-bench, reaching 19.9% vs. 12.7% for Stateless. HotpotQA shows a growing gap, with Full reaching 27.5% vs. 20.5% for Stateless, driven by action-pattern recognition rather than accumulating observation examples. Results are generally robust to task presentation order, with the Stateless baseline varying by at most 1.6 pp and the top-ranked setting shifting only on benchmarks with smaller evaluation sets (see Appendix A.3).

4.2 SPECULATOR MODEL SENSITIVITY

Table 3 compares three speculator models of varying capability and cost. We omit GPT-5.4-mini for WebArena and VWA because its capability is comparable to their actor model, GPT-5-mini. The central finding is that memory improvement is model-agnostic. The Stateless-to-best delta is consistent across all tested models, ranging from +2–7 pp on τ^2-bench and HotpotQA to +16–27 pp on ALFWorld and PDDL. Model capability primarily affects observation prediction, where GPT-5.4-mini improves over GPT-4.1-mini by +11 pp on ALFWorld and +18 pp on PDDL. The cheapest model, GPT-5.4-nano at $0.6\times$ the cost, achieves competitive memory deltas across all benchmarks, making it a highly cost-effective speculator.

Table 3: Speculator model comparison: aggregate accuracy (%) of the best memory setting and memory improvement (Δ: Stateless→best, pp) across three speculator models. Cost is relative to GPT-4.1-mini. WebArena and VWA use GPT-5-mini as actor; GPT-5.4-mini is omitted for these benchmarks as it is comparable in capability to the actor.

	WebArena		**VWA**		**ALFWorld**		**PDDL**		**τ^2-bench**		**HotpotQA**	
	Acc	Δ	Acc	Δ	Acc	Δ	Acc	Δ	Acc	Δ	Acc	Δ
GPT-4.1-mini	23.7	+3.9	17.4	+4.9	40.0	+23.7	33.8	+16.2	19.9	+7.2	27.5	+7.0
GPT-5.4-mini	—	—	—	—	51.3	+18.9	51.5	+27.2	24.2	+4.2	32.8	+2.1
GPT-5.4-nano	16.3	+9.0	9.1	+4.9	46.7	+20.8	41.0	+21.0	30.3	+6.4	23.4	+4.3

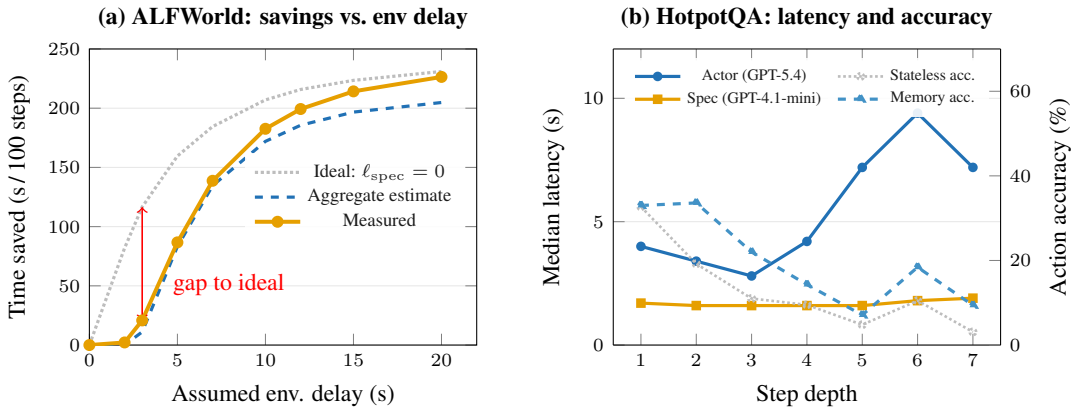

Figure 4: **Latency analysis. (a)** ALFWorld savings under assumed environment delay ℓ_{env}. The dotted curve preserves the observed hits and actor latencies but sets $\ell_{\text{spec}} = 0$; it is an idealized latency reference, not an accuracy bound. The red annotation shows the gap to this reference at a 3 s delay. Measured trace-level results (solid) track the aggregate estimate (dashed). **(b)** HotpotQA latency and accuracy by step. Speculator is 2–5× faster than actor, providing a consistent head start. Memory accuracy (dashed) consistently outperforms Stateless (dotted) across all depths.

4.3 Latency Analysis

Cost and theoretical savings. The speculator LLM call is one to two orders of magnitude cheaper than the actor across all benchmarks.[1] For Types 2 and 3, each speculation additionally pre-launches an actor call, but on a hit the pre-launched result substitutes for the next step's actor call, so the net expected additional cost is $k \cdot C_S + (1 - \text{acc}) \cdot C_{\text{act}}$. Table 6 in Appendix A.4 reports the full cost-latency breakdown per 100 steps for all six benchmarks.

Live validation. To validate that replay-measured accuracy translates to real-world savings, we run the actor and speculator live on the 300 tasks of HotpotQA and 134 episodes of ALFWorld, recording per-step latencies alongside prediction accuracy. On HotpotQA, the GPT-5.4 actor has an overall median latency of 4.9 s, whereas the GPT-4.1-mini speculator finishes in 1.7 s, yielding a ∼3 s head start per hit. The live action hit rate is 40.0% across 610 speculated steps, yielding a 1.05× analytical speedup and saving 892 s over a 19,024 s baseline. On ALFWorld, the actor has an overall median latency of 4.5 s and the speculator 2.7 s, with a 44.9% observation hit rate across 1,638 speculated steps. Because ALFWorld evaluates observation prediction, time savings depend on the environment feedback latency ℓ_{env}. As Figure 4(a) shows, savings computed from the measured per-step latencies closely track the aggregate estimate across assumed environment delays from 2 s to 20 s. The dotted curve is an idealized zero-latency reference: it keeps the observed prediction hits and actor latencies fixed while setting $\ell_{\text{spec}} = 0$. The red annotation shows the gap to this reference at a 3 s delay; this gap reflects the complete trace-level timing constraints rather

[1]Pricing: GPT-4.1-mini $0.40/$1.60 per MTok (input/output); GPT-5.4 $2.50/$15.00 per MTok. Source: `openai.com/api/pricing`, May 2026.

than speculator latency alone. At short environment delays, fewer predictions finish in time; as ℓ_{env} grows, the measured savings approach the idealized reference.

4.4 Step-Depth Accuracy

Memory-augmented speculation maintains its accuracy advantage across step depths on all six benchmarks, as detailed in Figure 8 of Appendix A.7. The advantage is largest at early steps where workflow patterns are most predictable, but persists at later steps where episodic retrieval supplies relevant templates. Figure 4(b) connects accuracy to latency: the speculator remains near 1.7 s regardless of depth, whereas the actor varies from roughly 3 to 9.5 s. Consequently, each correct speculation at deeper steps saves more wall-clock time. As plotted on the right axis, the full memory setting maintains a consistent accuracy advantage over the stateless baseline at every depth. The combined effect of higher accuracy and a larger latency gap makes deeper steps particularly valuable targets for speculation.

5 Related Work

Speculative and parallel execution for agents. Tool-using LLM agents (Yao et al., 2023; Schick et al., 2023; Wang et al., 2024) execute actions sequentially, making latency proportional to the number of tool calls. Ye et al. (2025) propose Speculative Actions, predicting the agent's next tool call during idle time with a stateless speculator. Other approaches target different bottlenecks: LLMCompiler (Kim et al., 2024) parallelizes independent calls within a single turn via a task-dependency DAG, which is complementary to our cross-turn speculation. SpecReason (Pan et al., 2025) and ConfSpec (Liu & He, 2026) apply speculative decoding at the internal reasoning-step level, whereas our work operates at the environment-action level. Most closely related is PASTE (Sui et al., 2026), which learns DAG-structured tool-calling patterns to guide speculation. Unlike PASTE, our approach retrieves contextually similar past trajectories with concrete argument values, learns from prediction failures online, and updates all memory dynamically during deployment.

Memory systems for agents. A growing body of work equips agents with non-parametric memory updated from past trajectories. ExpeL (Zhao et al., 2024) extracts insights by contrasting successful and failed trajectory pairs, while AWM (Wang et al., 2025) distills reusable action workflows from past episodes for retrieval at test time. Other systems store procedural abstractions (Fang et al., 2025; Wang et al., 2023), full trajectories as exemplars (Zheng et al., 2023; Kagaya et al., 2024; Sarukkai et al., 2025), or reasoning strategies derived from success and failure (Ouyang et al., 2026). Retrieval-augmented episodic memory (Lewis et al., 2020; Park et al., 2023; Majumder et al., 2023; Zhang et al., 2026) and failure-aware learning (Shinn et al., 2023; Ding, 2026; Marc-Antoine et al.) have similarly become foundational to agent design. Recent work also addresses memory curation: EvolveR (Wu et al., 2025) applies explicit distill, deduplicate, update, and filter operations, whereas Memory-R1 (Yan et al., 2025) and AgeMem (Yu et al., 2026) train RL controllers over memory management actions. Crucially, all these systems target the *actor*'s task performance; we apply this same family of techniques to improve the *speculator*'s prediction accuracy.

6 Conclusion

We presented memory-augmented speculative execution, which equips LLM agent speculators with three complementary memory systems to learn from past trajectories. Evaluation across six diverse benchmarks shows that structured memory consistently improves accuracy for all speculation types. The largest gains occur in observation prediction, where repetitive environment responses provide highly reusable patterns. These model-agnostic improvements vary by domain: Type 2 tasks benefit from continuous experience accumulation, while Type 1 tasks rely primarily on retrieving specific past content. Because speculation runs during environment idle time, these accuracy gains translate directly to measured latency reductions without adding wall-clock cost to the actor. However, combining all memory components is not always optimal, and unpredictable elements like page-specific UI indices still bottleneck absolute accuracy. Future work includes confidence-based memory routing, structure-aware web prediction, and broader production deployments.

AI USE STATEMENT

We used generative AI tools to polish the manuscript and to support research ideation and execution, including feedback on experimental design, implementation, visualization, and result analysis. The authors reviewed and verified all AI-assisted text, code, and analyses and take responsibility for the final content of this work.

ETHICS STATEMENT

Our method does not alter the actor's decisions, so incorrect speculations do not change agent behavior. Deployments may nevertheless inherit risks from the underlying agent, and trajectory memory may contain user information. Production systems should therefore apply appropriate domain safeguards, access controls, anonymization, and data-retention policies; Appendix B.5 discusses these considerations further.

REPRODUCIBILITY STATEMENT

We specify the memory algorithms and evaluation protocol in Sections 2 and 3.1, and all settings use the same frozen actor trajectories. The appendix reports task-order sensitivity, variance and significance tests, latency assumptions, and prompt formats (Appendix A.3, B.3, A.9, and C).

REFERENCES

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Liang Ding. Agenther: Hindsight experience replay for llm agent trajectory relabeling. *arXiv preprint arXiv:2603.21357*, 2026.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. *CoRR*, abs/2508.06433, 2025. doi: 10.48550/ARXIV.2508.06433. URL https://doi.org/10.48550/arXiv.2508.06433.

Tomoyuki Kagaya, Thong Jing Yuan, Yuxuan Lou, Jayashree Karlekar, Sugiri Pranata, Akira Kinose, Koki Oguri, Felix Wick, and Yang You. Rap: Retrieval-augmented planning with contextual memory for multimodal llm agents, 2024.

Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W Mahoney, Kurt Keutzer, and Amir Gholami. An llm compiler for parallel function calling. In *Forty-first International Conference on Machine Learning*, 2024.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 881–905, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.50. URL https://aclanthology.org/2024.acl-long.50/.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Siran Liu and Cyril Y He. Confspec: Efficient step-level speculative reasoning via confidence-gated verification. *arXiv preprint arXiv:2602.18447*, 2026.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn llm agents. *Advances in neural information processing systems*, 37:74325–74362, 2024.

Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Peter Jansen, Oyvind Tafjord, Niket Tandon, Li Zhang, Chris Callison-Burch, and Peter Clark. Clin: A continually learning language agent for rapid task adaptation and generalization. *arXiv preprint arXiv:2310.10134*, 2023.

Allard Marc-Antoine, Arnaud Teinturier, Victor Xing, and Gautier Viaud. Experiential reflective learning for self-improving llm agents. In *ICLR 2026 Workshop on Memory for LLM-Based Agentic Systems*.

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. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jL7fwchScm.

Rui Pan, Yinwei Dai, Zhihao Zhang, Gabriele Oliaro, Zhihao Jia, and Ravi Netravali. Specreason: Fast and accurate inference-time compute via speculative reasoning. *arXiv preprint arXiv:2504.07891*, 2025.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Vishnu Sarukkai, Zhiqiang Xie, and Kayvon Fatahalian. Self-generated in-context examples improve llm agents for sequential decision-making tasks. *arXiv preprint arXiv:2505.00234*, 2025.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2010.03768.

Yifan Sui, Han Zhao, Rui Ma, Zhiyuan He, Hao Wang, Jianxun Li, and Yuqing Yang. Act while thinking: Accelerating llm agents via pattern-aware speculative tool execution. *arXiv preprint arXiv:2603.18897*, 2026.

Mauro Vallati, Lukas Chrpa, Marek Grześ, Thomas Leo McCluskey, Mark Roberts, Scott Sanner, et al. The 2014 international planning competition: Progress and trends. *Ai Magazine*, 36(3): 90–98, 2015.

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

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. Openhands: An open platform for ai software developers as generalist agents. *arXiv preprint arXiv:2407.16741*, 2024.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *International Conference on Machine Learning*, pp. 63897–63911. PMLR, 2025.

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H Chi, et al. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*, 2025.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, and Botian Shi. Evolver: Self-evolving llm agents through an experience-driven lifecycle, 2025. URL https://arxiv.org/abs/2510.16079.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z Pan, et al. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*, 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *11th International Conference on Learning Representations, ICLR 2023*, 2023.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024. URL https://arxiv.org/abs/2406.12045.

Naimeng Ye, Arnav Ahuja, Georgios Liargkovas, Yunan Lu, Kostis Kaffes, and Tianyi Peng. Speculative actions: A lossless framework for faster agentic systems. *arXiv preprint arXiv:2510.04371*, 2025.

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

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. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory, 2026.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *The Twelfth International Conference on Learning Representations*, 2023.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*.

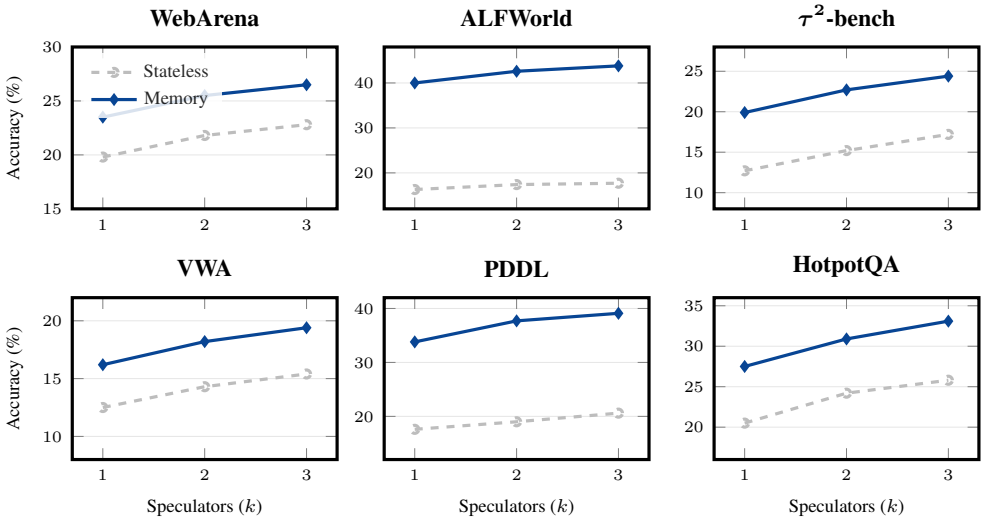

Figure 5: Best-of-k scaling from $k = 1$ to $k = 3$ across all six benchmarks. Running $k = 3$ parallel speculators improves accuracy by 1.4–5.6 pp over $k = 1$ across all domains. Memory-informed settings benefit significantly more from scaling than the Stateless baseline on observation-prediction benchmarks (e.g., ALFWorld Full +3.8 pp vs. Stateless +1.4 pp), suggesting that memory-guided predictions explore a more diverse set of plausible environment continuations.

A Additional Results

A.1 Best-of-k Scaling

Figure 5 illustrates the accuracy scaling from $k = 1$ to $k = 3$ speculators across all six benchmarks. Two primary patterns emerge. First, increasing the number of speculators yields consistent gains across the board, ranging from a +1.4 pp increase for ALFWorld Stateless to +5.6 pp for HotpotQA Memory. Second, memory-informed settings generally benefit more from scaling than the stateless baseline on observation-prediction benchmarks. For instance, on ALFWorld, the Full setting gains +3.8 pp compared to +1.4 pp for Stateless. This divergence suggests that memory-guided predictions explore a more diverse set of plausible environment continuations rather than producing near-identical guesses.

A.2 Won vs. Lost Trajectory Split

Table 4: Won vs. lost trajectory split: aggregate accuracy (%, $k = 1$) for Stateless and best Memory settings. Won trajectories generally achieve higher speculation accuracy, with the gap largest on HotpotQA (+14.0 pp for Stateless, +16.1 pp for Memory). ALFWorld is omitted as all episodes succeeded.

				Won		**Lost**	
Benchmark	#Won	#Lost	Total	Stateless	Memory	Stateless	Memory
WebArena	425	387	812	24.9	28.3	16.9	21.0
VWA	500	410	910	14.8	19.7	11.1	16.1
PDDL	50	10	60	20.4	33.3	10.5	35.0
τ^2-bench	122	42	164	14.9	21.2	7.8	17.0
HotpotQA	201	99	300	26.7	34.6	12.7	18.5

Table 4 analyzes speculation accuracy based on task outcome across five benchmarks; ALFWorld is omitted as all episodes were successful. Successful trajectories generally yield higher specula-

Table 5: Ordering effects on aggregate accuracy (%, $k = 1$) across three task orderings: *sequential* (default), *shuffled* (random permutation), and *grouped* (clustered by type). The rightmost column shows the Stateless baseline's maximum variation. The top-ranked memory setting shifts on some benchmarks, but the absolute differences between top-performing settings remain small.

Benchmark	Ordering	Stateless	Confusion	Table	Episodic	Tbl+Epi	Epi+Miss	Full	Range
WebArena	Sequential	19.8	21.4	21.9	23.4	23.5	**23.7**	23.5	
	Shuffled	19.8	21.5	22.1	**24.2**	23.9	23.6	23.7	
	Grouped	19.6	21.5	22.3	23.5	**23.8**	23.6	23.8	0.2
VWA	Sequential	12.5	15.3	15.8	**17.4**	16.6	16.7	16.2	
	Shuffled	12.7	14.7	16.1	**17.6**	17.1	16.9	16.8	
	Grouped	12.7	14.9	15.4	**17.1**	16.2	16.9	16.4	0.2
ALFWorld	Sequential	16.3	16.0	19.5	23.8	28.1	38.6	**40.0**	
	Shuffled	16.0	16.0	19.9	26.2	29.2	39.6	**42.4**	
	Grouped	15.5	16.3	19.9	22.7	26.5	34.8	**38.0**	0.8
PDDL	Sequential	17.6	19.0	21.1	19.0	20.0	33.0	**33.8**	
	Shuffled	16.1	17.9	20.6	19.4	20.8	33.9	**36.9**	
	Grouped	16.0	19.3	20.4	19.9	20.2	33.5	**34.9**	1.6
τ^2-bench	Sequential	12.7	13.1	15.2	17.0	17.6	16.8	**19.9**	
	Shuffled	11.9	14.3	15.6	16.0	17.2	17.2	**17.6**	
	Grouped	12.3	14.3	14.3	**18.0**	16.0	16.6	17.4	0.8
HotpotQA	Sequential	20.5	21.4	21.1	25.0	27.0	26.0	**27.5**	
	Shuffled	20.0	21.1	22.0	27.8	26.9	27.4	**28.8**	
	Grouped	19.8	21.5	21.4	27.0	**27.5**	25.6	25.8	0.7

tion accuracy, suggesting that they follow more predictable, standard patterns. This gap is most pronounced on HotpotQA, where successful tasks achieve 26.7% Stateless accuracy compared to 12.7% for failures, a 2.1× difference. This suggests that failed tasks involve more exploratory or erratic search behaviors that are inherently harder to speculate.

A similar trend holds for WebArena, VWA, and τ^2-bench. PDDL is a slight outlier, showing marginally higher Memory accuracy on lost tasks (35.0% vs. 33.3% for won). However, the limited number of failures in PDDL ($n = 10$) makes this estimate less reliable than the other benchmarks. Overall, the data confirms that speculation is most effective when the agent is on a successful, predictable path.

A.3 Ordering Effects

Table 5 reports aggregate accuracy across three task orderings on all six benchmarks. Three primary findings emerge from this analysis.

Stability of the Stateless baseline. The Stateless baseline is remarkably stable, varying by at most 1.6 pp across orderings on PDDL and by less than 1 pp on five of the six benchmarks. Since the Stateless setting uses no memory, this minor variation reflects only the inherent difference in predictability among specific task sequences within the benchmarks.

Bounded variation in memory settings. Observation-prediction benchmarks (ALFWorld and PDDL) show the highest sensitivity to task ordering when memory is enabled. For instance, ALFWorld Full ranges from 38.0% in the grouped ordering to 42.4% when shuffled, representing a 4.4 pp spread. The grouped ordering, which clusters similar task types, likely produces less diverse episodic memory during the early stages of a run. Conversely, the shuffled ordering performs best on these benchmarks, suggesting that early exposure to a diverse set of examples improves the quality of subsequent episodic retrieval.

Table 6: Cost-benefit analysis, all values per 100 agent steps. ℓ_{hit} is wall-clock saving per correct speculation; T_{100} is baseline latency without speculation. C_{act}, C_{S}, and C_{M} represent actor and speculator costs. Spec and +Mem columns denote latency reduction in seconds. Cell shading intensity is proportional to the absolute reduction. Speculation occurs at every step except the first of each task.

							$k=1$		$k=2$		$k=3$	
Benchmark	Type	ℓ_{hit}	T_{100}	C_{act}	C_{S}	C_{M}	Spec	+Mem	Spec	+Mem	Spec	+Mem
WebArena	1	12.3 s	3600 s	0.35	0.032	0.044	−244 s	−292 s	−288 s	−333 s	−311 s	−355 s
VWA	1	17.1 s	4197 s	0.44	0.032	0.044	−213 s	−297 s	−244 s	−325 s	−264 s	−343 s
ALFWorld	2	7.1 s	710 s	0.21	0.012	0.024	−116 s	−284 s	−124 s	−303 s	−126 s	−311 s
PDDL	2	6.3 s	630 s	0.23	0.022	0.034	−111 s	−213 s	−120 s	−238 s	−130 s	−246 s
τ^2-bench	3	12.5 s	4000 s	1.65	0.025	0.037	−159 s	−248 s	−190 s	−284 s	−215 s	−305 s
HotpotQA	3	10.0 s	1000 s	0.28	0.023	0.035	−205 s	−275 s	−242 s	−309 s	−258 s	−331 s

Rank shifts among top settings. The optimal memory configuration is consistent for some benchmarks but shifts for others. On VWA, the Episodic setting is consistently best across all orderings, while the Full setting remains top-ranked for ALFWorld and PDDL. On WebArena, the top-ranked setting alternates between Episodic, Tbl+Epi, and Epi+Miss, though the absolute difference between these winners is less than 0.6 pp. On τ^2-bench and HotpotQA, the best setting shifts more substantially, with specific configurations varying by up to 3.0 pp (e.g., HotpotQA Full). Despite these rank shifts, the Stateless-to-best gap remains positive across all orderings for every benchmark, confirming that memory-augmented speculation is robust to task presentation order.

A.4 Cost–Benefit Analysis

Table 6 provides a per-benchmark cost-benefit breakdown. All values are normalized to 100 agent steps for direct comparison. The columns are constructed as follows. The wall-clock time saved per correct speculation, ℓ_{hit}, is determined by the speculation type as described in Section 2.1. For Type 1, $\ell_{\text{hit}} = \ell_{\text{env}}$ represents the environment processing time saved by prelaunching the predicted action. These values are measured from trajectory timestamps, yielding 12.3 s for WebArena and 17.1 s for VWA, which includes visual processing overhead. For Type 2, $\ell_{\text{hit}} = \min(\ell_{\text{LLM}},\ \ell_{\text{env}} - \ell_{\text{spec}})$. The table uses representative deployment latencies where $\ell_{\text{env}} \approx 7$ s for embodied tasks. For Type 3, $\ell_{\text{hit}} = \ell_{\text{LLM}} - \ell_{\text{spec}}$, representing the head start gained by the speculator finishing before the actor. T_{100} is the baseline total latency for 100 steps without speculation, computed from the actor latencies recorded during trajectory collection. C_{act} is the actor LLM cost for 100 steps, computed from token counts in the collected trajectories and the provider's pricing (see footnote in Section 4.3). C_{S} and C_{M} are the speculator costs without and with memory context. Memory adds 200–500 extra input tokens per call, increasing cost by \$0.01–0.02 per 100 steps. The Spec and +Mem columns show the estimated latency reduction calculated as acc $\times\ \ell_{\text{hit}} \times 100$, where acc is the aggregate accuracy from Table 2.

Representative API latencies from our provider include $\ell_{\text{LLM}} \approx$ 5–10 s for GPT-5.4, $\ell_{\text{spec}} \approx$ 1–3 s for GPT-4.1-mini, and ℓ_{env} ranging from less than 0.1 s for text simulators to approximately 17 s for VWA. The speculator call is 10–70$\times$ cheaper than the actor call, so speculation yields a net benefit at any accuracy above roughly 5%. Memory systems improve accuracy at negligible incremental cost because embedding retrieval adds less than 50 ms per turn.

A.5 All-Setting Learning Curves

Figure 3 in the main text compares the stateless baseline against the best-performing memory configuration for each benchmark. Figure 6 plots all seven configurations to show the full performance progression as experience accumulates across the task sequence. On ALFWorld and PDDL, the configurations separate into two clear tiers. The Epi+Miss and Memory settings form an upper tier that diverges sharply from the remaining five settings after roughly 40 tasks. This separation demonstrates that learning from prediction failures is the dominant performance factor for observation-prediction benchmarks. On WebArena and VWA, all memory settings cluster more tightly. In these domains, Episodic retrieval provides the largest initial lift, while additional components offer dimin-

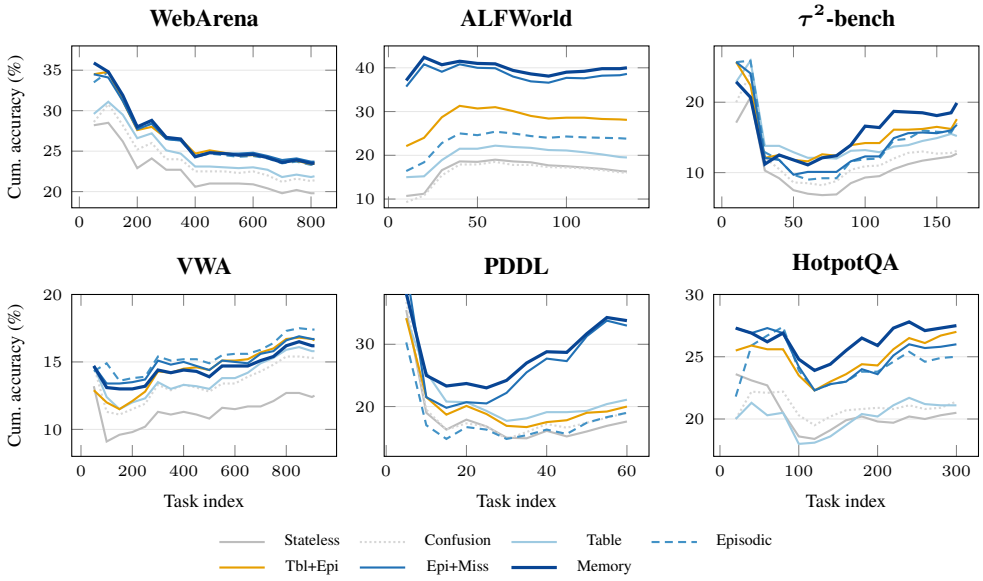

Figure 6: Cumulative accuracy ($k = 1$) across all six benchmarks for all seven memory settings. Each point shows the aggregate accuracy over all tasks processed up to that index. PDDL and ALFWorld show the clearest separation, as the failure-learning settings (Epi+Miss and Memory) pull away from the stateless baseline as experience accumulates.

ishing returns. On τ^2-bench, the Memory setting gradually pulls ahead of all other configurations during the second half of the trajectory. On HotpotQA, Tbl+Epi and Memory track closely throughout the run, and both outperform the other configurations by approximately 7 pp.

A.6 Per-Group Breakdown

Figure 7 details the per-group accuracy for the Stateless and Memory settings across all six benchmarks. On ALFWorld, the Memory setting improves accuracy across all six task types. The largest gains occur on Heat tasks with a +32.6 pp increase and Cool tasks with a +26.1 pp increase. These tasks feature repetitive action sequences that are highly amenable to learning from past trajectories. Similarly, on PDDL, Tyreworld benefits dramatically with a +33.7 pp gain, whereas Blockworld sees a more modest improvement of +6.4 pp because of its more variable state transitions. WebArena shows consistent improvement across all six websites, led by Reddit with a +4.3 pp absolute gain. VWA also shows moderate, uniform gains across its three site categories. On τ^2-bench, the retail domain benefits substantially, achieving 25.2% accuracy with Memory compared to 18.2% for Stateless. The airline domain shows lower absolute accuracy due to its smaller training set and more diverse API patterns. On HotpotQA, comparison questions remain easier to predict than bridge questions for both settings, but Memory provides consistent gains for both types. These results on VWA and HotpotQA demonstrate that episodic retrieval provides stable benefits even across heterogeneous task groups.

A.7 Step-Depth Accuracy

Figure 8 illustrates how speculation accuracy evolves with step depth across all six benchmarks. On WebArena, accuracy is highest at step 0, where Memory reaches 56% compared to 40% for Stateless. Performance drops at step 2 as predictions shift from general navigational commands to more granular, element-specific interactions. ALFWorld maintains a consistent 15–20 pp advantage across all depths, indicating that the benefit of memory is not restricted to initial steps. On PDDL, the Memory setting achieves 37–43% accuracy at steps 1–3 while the Stateless baseline remains below 20%, a gap that persists through step 10. VWA displays a steady 3–5 pp Memory advantage across most depths. On τ^2-bench, Stateless shows low initial accuracy, whereas Memory provides

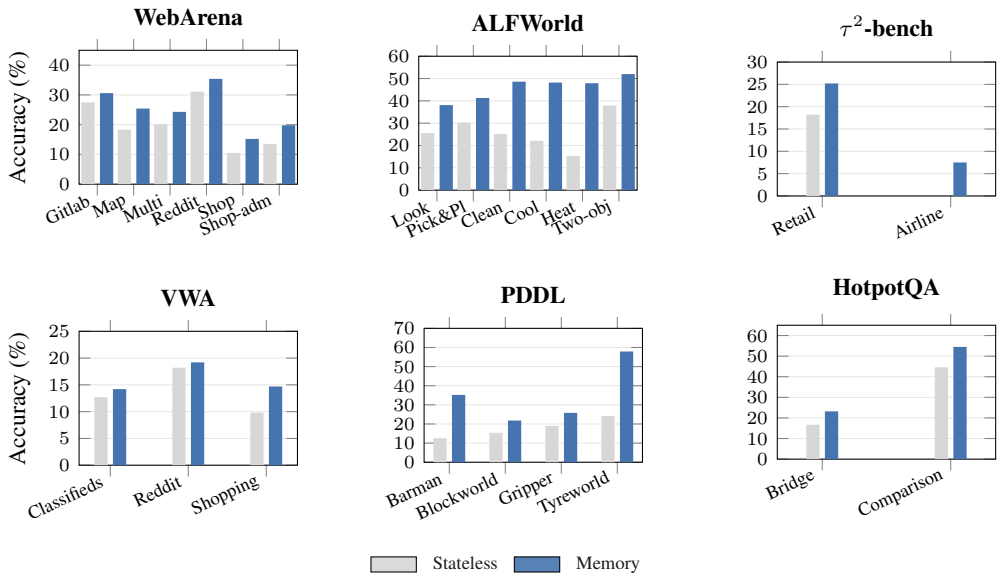

Figure 7: Per-group accuracy breakdown (k=1) across all six benchmarks. Groups correspond to websites for WebArena/VWA, task types for ALFWorld, domains for τ^2-bench, planning domains for PDDL, and question types for HotpotQA. Memory improves over Stateless across nearly all groups, with the largest gains on ALFWorld task types with stereotyped sequences and PDDL Tyreworld.

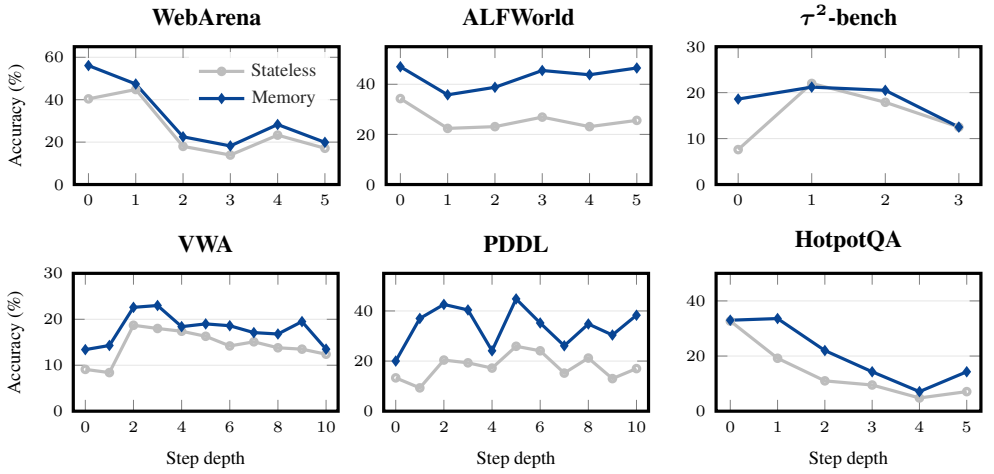

Figure 8: Speculation accuracy by step depth across all six benchmarks. Step depth 0 is the first speculated step (not the initial task step, which is excluded). "Memory" denotes the best-performing memory setting per benchmark. Memory maintains higher accuracy at most depths, with the advantage largest on Type 2 benchmarks (15–20pp on ALFWorld, 10–20pp on PDDL).

an 11 pp advantage at step 0 and peaks at 24.5% by step 2. Finally, HotpotQA exhibits a tighter separation, with Memory maintaining a 6–8 pp advantage during intermediate reasoning steps.

A.8 Episodic Memory Pruning

We explored three retrieval-time pruning passes inspired by Evo-Memory (Wei et al., 2025). The first is a low-utility penalty that halves the similarity score for episodes from failed tasks where speculation was also incorrect. The second is redundancy deduplication, which removes candidates

with an inter-episode cosine similarity above 0.92. The third is a miss-pattern penalty that reduces scores by $0.7\times$ for episodes whose primary action is mispredicted in at least three instances. The pruning process over-fetches by a factor of $3\times$ before filtering the results to k candidates.

Across all benchmarks, pruning did not meaningfully change accuracy. On ALFWorld, the Memory setting increased from 40.0% to 40.4% with pruning enabled, while other benchmarks showed similarly negligible effects. We attribute this to two factors. First, the episodic memory is relatively small, containing hundreds rather than thousands of episodes, so redundancy is not yet a significant bottleneck. Second, the miss-pattern penalty can inadvertently suppress useful episodes if a frequently mispredicted action type also appears in successful retrieved records. Consequently, all results reported in this paper use the configuration with pruning disabled.

A.9 Environment Delay Sensitivity (ALFWorld)

For Type 2 benchmarks, the speculator executes during the environment feedback window. While ALFWorld utilizes a text simulator with near-zero latency, target deployments such as household robotics involve physical execution delays. We measured actual actor and speculator latencies across all 134 episodes and analytically computed the speedup for a range of representative environment delays.

ℓ_{env}	Spec ready	Hits	Save/hit	Total saved	Speedup
2 s	16.9%	146	0.3 s	39 s	1.003×
5 s	90.5%	686	2.2 s	1,541 s	1.085×
7 s	97.3%	723	3.4 s	2,465 s	1.119×
10 s	99.3%	732	4.4 s	3,244 s	1.128×
15 s	99.9%	735	5.2 s	3,806 s	1.113×
20 s	99.9%	735	5.5 s	4,023 s	1.095×

As shown in the table, absolute time saved grows monotonically with the environment delay, increasing from 39 s at a 2 s delay to 4,023 s at a 20 s delay. However, the speedup ratio peaks near $\ell_{\text{env}} = 10$ s. This occurs because larger delays inflate the baseline latency faster than the savings grow. The per-hit savings, defined as $\min(\ell_{\text{env}} - \ell_{\text{spec}}, \ell_{\text{actor}})$, are increasingly capped by the actor latency of the subsequent step rather than the environment delay window. In practical deployments, the environment latency is typically fixed by the hardware, making the speedup at that specific delay the relevant performance metric.

B Limitations and Broader Impact

B.1 Element Index Prediction

The primary source of prediction error in web navigation benchmarks is element index prediction for `click_element` actions. Element indices are arbitrary, page-specific identifiers assigned by the accessibility tree parser. These indices frequently change across different pages, page loads, or even within a single session as dynamic content renders. Because the speculator lacks access to the target page's DOM at the time of prediction, it cannot resolve which specific index corresponds to the intended element. This volatility fundamentally limits full-match accuracy on WebArena and VWA, where `click_element` accounts for the majority of read-only actions. Incorporating lightweight page-structure features, such as a compressed DOM skeleton or element-role embeddings, into the speculation context represents a promising direction to mitigate this bottleneck.

B.2 Memory Combination and Interference

The full memory configuration, which combines all three systems, underperforms simpler settings on benchmarks such as WebArena and VWA. For instance, adding the transition table to episodic memory on VWA reduces accuracy by 0.8 pp. This suggests that aggregate transition statistics can sometimes mislead the speculator when action spaces are highly diverse. More broadly, naively concatenating all memory signals into the speculator's context can introduce redundancy, as episodic examples may already encode the relevant transition patterns, or conflicting guidance, such as when the transition table favors a frequent action that a miss episode explicitly warns against.

These observations motivate selective memory routing: a lightweight classifier or confidence-based mechanism that chooses the optimal memory configuration for each task or turn rather than always utilizing the full stack. Our evaluation of seven distinct settings provides the necessary training signal for such a router, as accuracy data for each configuration is available across various task types.

B.3 Speculator Variance

Table 7: Accuracy (%) for the Stateless baseline and best memory configuration with mean ± standard deviation. For Type 1 and Type 2, columns s_0, s_1, s_2 represent three parallel speculators from the same run. For Type 3, columns represent three independent task orderings. The rightmost column shows p-values from McNemar's test for the Stateless-to-Memory improvement.

Benchmark	Setting	s_0	s_1	s_2	Mean ± Std	p
WebArena	Stateless	19.8	19.8	19.9	19.8 ± 0.05	
	Epi+Miss	23.7	23.9	23.9	23.8 ± 0.09	< 0.001
VWA	Stateless	12.5	12.5	12.8	12.6 ± 0.12	
	Episodic	17.4	17.1	17.3	17.3 ± 0.10	< 0.001
ALFWorld	Stateless	16.3	16.2	16.2	16.2 ± 0.05	
	Memory	40.0	41.5	41.4	41.0 ± 0.65	< 0.001
PDDL	Stateless	17.6	16.0	17.1	16.9 ± 0.66	
	Memory	33.8	34.1	33.4	33.8 ± 0.28	< 0.001
τ^2-bench	Stateless	12.7	11.9	12.3	12.3 ± 0.33	
	Memory	19.9	17.6	17.4	18.3 ± 1.14	< 0.001
HotpotQA	Stateless	20.5	20.0	19.8	20.1 ± 0.29	
	Memory	27.5	28.8	25.8	27.4 ± 1.23	< 0.001

Each experiment utilizes three parallel speculators with identical prompts and temperatures. Table 7 reports the accuracy for the Stateless baseline and the best-performing memory configuration for each benchmark. In Type 1 and Type 2 benchmarks, three parallel speculators from the same execution show standard deviations ranging from 0.04 pp to 0.66 pp. For Type 3 benchmarks, three independent task orderings serve as replications to produce standard deviations between 0.29 pp and 1.23 pp. McNemar's tests conducted over all evaluation steps confirm that the accuracy improvement from the Stateless baseline to the Memory setting is statistically significant across all six benchmarks with $p < 0.001$. This test assesses whether the proportion of steps where memory succeeds but Stateless fails exceeds the reverse, providing a robust per-step significance assessment. Our replay-based evaluation protocol eliminates actor variance across settings because all configurations are evaluated against the identical actor trajectories. The s_0 values in Table 7 correspond exactly to the primary results reported in Table 2.

B.4 Replay vs. Online Evaluation

Our replay-based design evaluates speculation accuracy in isolation. The speculator predicts against pre-recorded actor trajectories, ensuring that correct predictions do not actually accelerate the interaction or change the actor's context during the evaluation. In a live deployment, however, correct speculations would reduce latency and potentially allow the actor to process more turns within a fixed time budget. This live feedback loop could be positive if faster interactions lead to more coherent trajectories, but it could also be negative if a changed temporal context invalidates the replay assumption. A live deployment study incorporating real API latencies is necessary to measure true end-to-end wall-clock savings and validate that the accuracy measured during replay translates to a proportional speedup in practice.

B.5 Broader Impact

Speculative execution for LLM agents is a latency-optimization technique that does not alter the agent's final behavior because the actor's decisions remain identical regardless of whether specula-

tion is employed. The lossless guarantee ensures that incorrect predictions are silently discarded, introducing no new failure modes beyond those already present in the base agent. However, faster agent execution could amplify existing risks if the underlying agent is deployed in sensitive domains such as autonomous customer service or financial transactions without adequate safeguards. Because our memory systems store action-level trajectory data that could contain user information, production deployments should apply appropriate data retention policies and anonymization to the episodic memory store to protect user privacy.

C PROMPT TEMPLATES

Episodic memory format. Retrieved episodes are formatted as structured tuples injected into the speculator's context. The stored *observation* field (Eq. 4) is used for embedding-based retrieval but not surfaced in the prompt, as the speculator already has the current conversation context. The *arg_sources* field is folded into the action formatting:

```
--- Example 1 (similarity=0.87) ---
Situation: User asked to return items from order #W6247578.
Agent actions:
get_order_details(order_id=#W6247578)
order_id: from user message
Outcome: SUCCEEDED
Takeaway: Standard order lookup pattern after user provides
order ID.
```

Speculation-miss episode format. Miss episodes store the predicted versus actual action:

```
--- Speculation miss (similarity: 0.82) ---
Pattern: Speculator predicted click(7) but agent actually
used click(12).
Context: Product listing page with multiple items.
---
```

Contrastive transition table format. The transition table is serialized as a ranked list of next-action candidates with frequency and success rate, conditioned on the current action:

```
Historical patterns from 85 past tasks (71 success, 14
failure):
After get_order_details, the most likely next tools are:
1. get_user_details --- 42% of the time (success rate: 88%)
typical args: order_id,user_id(63%)
2. cancel_pending_order --- 25% of the time (success rate:
72%)
typical args: order_id,reason(91%)

Transitions to AVOID (high failure rate):
- modify_pending_order --- 12% of the time but only 33%
success rate
```

For benchmarks using turn-level speculation (ALFWorld), `get_full_context()` provides the complete table rather than conditioning on a single action.

Confusion tracker format. Confusion patterns exceeding the threshold are injected as hard constraints:

```
KNOWN PREDICTION ERRORS (avoid these):
- You predicted scroll_down 5 times when the agent actually
used
  click_element. Do NOT predict scroll_down in this context.
```