WebATLAS: Actor-Critic Task-Completion with Look-ahead Action Simulation

Anonymous ACL submission

Abstract

Current web agents lack the ability to anticipate action outcomes, leading to inefficient plans that fail to account for the structure and dynamics of unfamiliar environments. We introduce **WebATLAS** (**A**ctor-Critic **T**ask-completion with **L**ook-ahead **A**ction **S**imulation for **Web** Applications), a memory-augmented web agent that grounds its planning in environment reality by simulating action consequences in *cognitive space* before execution. WebATLAS first constructs a multi-layered agentic memory, comprising a *cognitive map* that encodes action transition dynamics, and an *episodic memory* that captures procedural experience, through lightweight curiosity-driven exploration. During execution, *Look-Ahead Action Simulation* (LAS) leverages this memory to simulate action outcomes over candidate actions using real, previously observed transitions rather than LLM-hallucinated outcomes. A critic evaluates the resulting trajectories, selects the most promising action sequence, and triggers dynamic replanning when observed states diverge from predictions. WebATLAS achieves state-of-the-art results on three web agent benchmarks (WebArena-Lite, WebChoreArena, WebVoyager) without any website-specific LLM fine-tuning. Ablations confirm that multi-layered memory, look-ahead simulation, and replanning serve complementary roles.

1 Introduction

Autonomous web agents that navigate sites and execute complex tasks on behalf of users—information gathering, transactions, configuration—remain far from human-level reliability on long-horizon benchmarks (Yao et al., 2022; Sodhi et al., 2023; Koh et al., 2024). The difficulty is multifold: partial observability over sprawling page structures, vast action spaces requiring multi-step planning, and the risk of irreversible mistakes (e.g., placing an unintended order or deleting data).

Current LLM-based web agents are largely reactive. They lack structured knowledge of the environment and have limited ability to anticipate action outcomes before execution (Erdogan et al., 2025). Existing methods employ a general-purpose LLM to predict outcomes, but such predictions are ungrounded in the actual environment and prone to hallucination. Alternatively, methods that perform tree search by executing candidate actions in the real environment risk triggering irrecoverable states and incur prohibitive time costs.

To bridge these gaps, we introduce WebATLAS, a web agent framework that *simulates before acting*, with two major novel mechanisms:

1. **Multi-Layered Agentic Memory**, containing a cognitive map of state transitions and an episodic memory of environment-specific constraints (formats, irrecoverable states, etc.), constructed via curiosity-driven exploration;

2. **Look-Ahead Action Simulation (LAS)**, simulating multi-step action outcomes in conceptual space with retrieved memory, and enabling foresight without blind or greedy execution.

Experiments on three web agent benchmarks demonstrate that WebATLAS achieves state-of-the-art (SOTA) performance. On WebArena-Lite and WebVoyager, WebATLAS outperforms Plan-and-Act (), the previous SOTA, by 15.0 and 12.0 absolute points in success rate, respectively. On long-horizon and memory-heavy tasks on WebChoreArena, WebATLAS outperforms all baselines by a significant margin.

We summarize our contributions:

- **Novel memory design**. We design a multi-layer agentic memory, with cognitive map and episodic memory, built from light-weight curiosity-driven exploration, enabling outcome-aware reasoning and hazard avoidance.

- **Look-ahead action simulation**. We propose a framework that evaluates actions via multi-step simulated outcomes retrieved from the cognitive map, replacing unreliable LLM predictions with grounded environment-specific transitions.

- **SOTA performance**. WebATLAS integrates memory, planner, actor-critic and achieves state-of-the-art performances on three agent environments.

2 Method

Problem Formulation We cast web navigation as a Partially Observable Markov Decision Process (POMDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R)$, where $\mathcal{S}$ denotes the state set, $\mathcal{A}$ denotes the action set, $\mathcal{O}$ denotes the observation set, T denotes the state transition function, and R denotes the reward. Given a natural-language goal q, the agent must synthesize a plan and execute a sequence of actions $(a_0, \ldots, a_T)$ that reaches a goal-consistent terminal state. At each time step t, the agent receives partial observations $o_t \in \mathcal{O}$. Based on the observation o_t, the agent chooses an action to take $a_t \in \mathcal{A}$, such as `click` or `type`. The goal of the agent is to maximize the reward, i.e. fulfilling task q.

2.1 Overview

Two components distinguish WebATLAS from prior agent systems:

- **Multi-Layered Memory** (Section 2.2): consists of 1) cognitive map of state transitions and 2) episodic memory of environment-specific knowledge, which supports outcome-aware reasoning. This memory is constructed via curiosity-driven exploration (Section 2.3) before task execution and updated online during inference.

- **Look-Ahead Action Simulation** (Section 2.5): simulates multi-step action outcomes by retrieving outcomes from the multi-layered memory, enabling foresight without executing actions. This avoids greedily selecting the next action or performing ungrounded outcome guessing.

Figure 1(a) illustrates the workflow of WebATLAS. Given a task q, the **Planner** decomposes the task into a structured plan of subtasks. At each step t, the raw observation o_t is summarized to lower cognitive load. The **Actor** proposes a small set of action candidates C_t; the **Critic** evaluates each candidate and selects the most goal-advancing action. The planner dynamically updates the plan when observations diverge from expectations.

2.2 Multi-Layered Memory

Existing agents often fail because they (1) cannot anticipate action outcomes–for instance, placing an order on a shopping website may lead to irrecoverable states–and (2) are unfamiliar with environment-specific conventions such as date formats or search syntax (Yang et al., 2024). This represents a fundamental gap between LLM agents and human intelligence: humans routinely predict action consequences using world knowledge or environment-specific regulations. Our memory system bridges this gap by encoding prior exploration into structured, retrievable representations.

To bridge this gap, we build three complementary memory layers:

Layer 1: Working Memory This is a task-specific buffer where intermediate observations and a short task history are stored. This provides immediate situational awareness for the current status.

Layer 2: Cognitive Map The cognitive map encodes structured knowledge about the environment's dynamics, capturing how actions transform observations. It is represented as a graph of transitions $M = \{(o_t, a_t, o_{t+1})\}$, where o_t and o_{t+1} denote observations (e.g., simplified HTML or URLs) at consecutive steps and a_t is the executed action. Each entry stores both the raw observation and a concise *agentic summary* highlighting state deltas and newly available affordances (e.g., "clicking `Reports` reveals {Sales, Products, ...}"). The construction of these entries is described in Section 2.3.

For retrieval, the cognitive map is queried with a state–action pair (o_t, a_t), returning the next observation o_{t+1} with summarized outcome and new observations. When the query hits an unexplored node, a generic placeholder is returned, signaling a retrieval miss to downstream modules.

Layer 3: Episodic Memory Episodic memory captures environment-specific knowledge such as constraints, formatting conventions, and idiosyncratic behaviors. For instance, it records facts like "the date box only accepts `MM/DD/YYYY` format" or 'exporting CSV on the admin portal leads to an unrecoverable waiting state". By distilling recurring

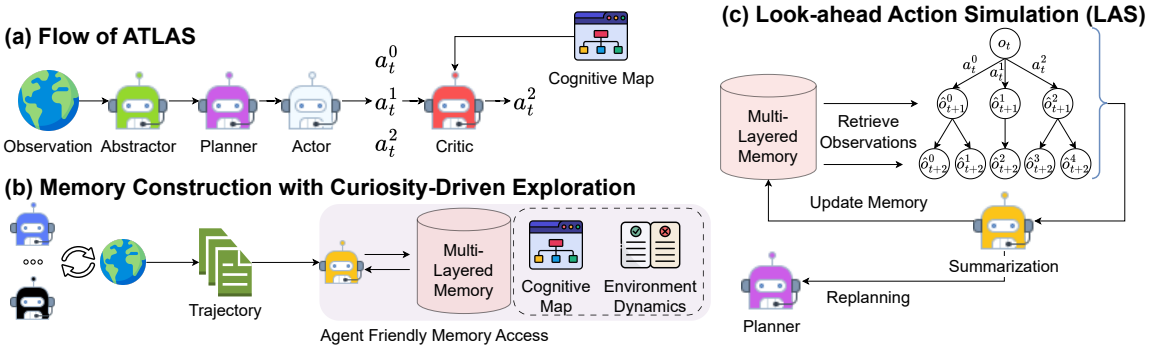

Figure 1: **Architecture of WebATLAS.** (a) **Flow of WebATLAS:** The planner decomposes a task into a series of subtasks. Given the summarized observations, the actor proposes next steps simulated by the critic. The best goal-advancing and least harmful action is selected for execution. (b) **Multi-layered agentic memory & curiosity-driven exploration:** We employ lightweight exploratory agents to interact with the environment and build the multi-layered agentic memory (cognitive map + episodic memory). (c) **Look-ahead Action Simulation (LAS): WebATLAS** simulates all candidate actions by retrieving observations from the memory, providing ability to look-ahead. We employ the memory agent to learn from LAS trajectories to make a better plan and update memory if necessary.

patterns from prior exploration, episodic memory enables agents to avoid known pitfalls and adapt to site-specific requirements.

Both the cognitive map and episodic memory are optionally updated online when execution encounters previously unseen transitions or dynamics.

2.3 Memory Construction via Curiosity-Driven Exploration

Prior work on artificial curiosity (Pathak et al., 2017) has demonstrated that intrinsically curious agents can drive effective environment exploration and task completion. Inspired by this insight, we construct the memory through curiosity-driven exploration before evaluation begins.

Exploration Policies We launch a set of lightweight explorer agents with diverse policies, implemented with varying LLM generation temperatures, sampling methods and hyperparameters, coverage incentive prompts. Crucially, *no task-completion reward is used*, avoiding information leakage from the test set. We balance breadth and depth, limiting explorers to visit the most promising states within a fixed compute budget.

Agentic Memory Access Given the exploration trajectories, a dedicated memory agent converts each raw observation o_t and transition (o_t, a_t, o_{t+1}) into concise agentic summaries emphasizing:

- Salient differences between successive observations (o_t, o_{t+1});
- Newly available actions or affordances revealed in o_{t+1};
- Site-specific rules, constraints, and hazards.

These summaries, together with the raw observations, are stored in the memory. This curation step is essential: naively storing raw observations (e.g., full HTML) would overwhelm the task agent at retrieval, whereas agentic summarization exposes only the most decision-relevant information.

2.4 Planner

Given the natural language task q and initial observation o_0, the planner produces a structured plan P_0. At subsequent steps, it may trigger replanning conditioned on the current observation, execution state, and memory:

$$P_0 = \text{Planner}(q, o_0), \quad (1)$$
$$P_t = \text{Planner}(q, o_t, s_t, M). \quad (2)$$

Plans are concise lists of sub-goals with success predicates (e.g., "Reports → Sales → Set dates → Read table"). The current plan is included in the context for both the actor and critic. Our planner follows the design of Chae et al. (2025) with extensions for dynamic replanning described in Section 2.6.

2.5 Look-Ahead Action Simulation (LAS)

At each step t, the actor proposes N executable candidates $C_t = \text{Actor}(q, P_t, o_t, s_t, M)$, and the

critic evaluates each candidate based on goal alignment, state recoverability, action coherence, plan consistency, and outcome risk (e.g., destructive or dead-end transitions), selecting the highest-scoring action for execution.

Simulated Action While the standard actor-critic loop provides a working baseline, it may suffer from insufficient exploration and myopic action selection. To address this, we extend the framework with *Look-Ahead Action Simulation* (LAS), which performs multi-step simulations in conceptual space using the cognitive map.

Unlike prior approaches that prompts a plain LLM for implicit world model (Chae, 2024), we leverage the cognitive map to retrieve *observed* action outcomes and environment-specific information, granting the critic the ability to look ahead beyond the current step. For each candidate $a_t^i \in C_t$, the critic retrieves the predicted next observation from the memory:

$$\hat{o}_{t+1}^i = M(o_t, a_t^i), \tag{3}$$

where i indexes candidates at step t. From each predicted state, the process recurses: the actor proposes new candidates and the critic evaluates them, repeating for D steps to produce a set of simulated trajectories.

The critic then scores each simulated trajectory holistically, assessing whether the full sequence of predicted states makes consistent progress toward the goal. Trajectories whose transitions were directly observed in past experience are weighted more heavily than those relying on interpolated or uncertain transitions, since observed transitions provide more reliable predictions. The action initiating the highest-scoring trajectory is selected for execution.

Efficiency Because simulations query the cognitive map via hash-based lookup, simulation incurs negligible computational overhead compared to environment interaction. No actual actions are executed during simulation, preserving environment state and avoiding irrecoverable transitions.

Comparison to Prior Work Existing agents perform tree search using LLMs as both reward functions and world models, scoring candidate actions and pruning low-scoring branches. Our approach offers three advantages:

1. *Trustworthiness*: Prior methods rely on LLM-imagined outcomes, which are prone to hallucination since LLMs are not explicitly trained as world models. Our method retrieves real, previously observed transitions, yielding substantially more reliable predictions.

2. *Comprehensiveness*: Prior methods conduct greedy one-step evaluation, pruning branches without further consideration. Actions that appear suboptimal at step t but enable high-value states at step $t{+}1$ are lost. Our multi-step simulations resemble beam search, evaluating the joint outcome of action sequences.

3. *Efficiency*: Simulation occurs entirely in conceptual space—no environment actions are executed—making it orders of magnitude faster than approaches requiring actual environment branching while avoiding irrecoverable state changes.

2.6 LAS-Based Dynamic Replanning and Memory Update

Replanning We trigger replanning when the observed state diverges significantly from the expected state predicted by simulation:

$$\text{Replan} = \mathbb{1}\Big[\, \|o_t^{\text{obs}} - \hat{o}_t^{\text{exp}}\| > \varepsilon \,\Big]. \tag{4}$$

Effective planning requires foresight about likely future states. We leverage the results of LAS to inform the planner: as illustrated in Figure 1(c), the planner integrates a brief *exploration digest*—summarizing what worked, what failed, newly exposed affordances, and uncovered prerequisites—then updates the plan P_t. This mechanism can be viewed as a lightweight form of causal model updating, inspired by Sontakke et al. (2021). By conditioning replanning on divergence rather than triggering it at every step, we also prevent catastrophic forgetting of important context accumulated in prior plans.

Online Memory Update. In addition to replanning, the agent updates its memory during execution. When simulation or action execution encounters previously unseen transitions or environment dynamics, the memory agent selectively incorporates new entries into both the cognitive map and episodic memory. Decisions about what to retain, update, or discard are delegated to the memory agent, which curates information based on task relevance and environmental novelty. This selective updating prevents memory overload while ensuring the agent's world model remains current.

Table 1: Task success rate comparison on WebArena-Lite, WebChoreArena, and WebVoyager environments. Best performance is in **bold** and second best is _underlined. Baseline performances are directly referenced from original papers if available. Additional results are shown in Appendix.

Agent	WebArena-Lite	WebChoreArena	WebVoyager
WebPilot	35.3	8.9	35.7
AWM	33.0	15.1	29.6
WebRL	48.1	9.6	35.6
AgentOccam-Judge	43.1	23.5	40.1
WebAgent-R1	44.8	13.1	32.0
Plan-and-Act	53.9	12.5	58.1
WebATLAS (Ours)	**68.5**	**33.7**	**63.2**

3 Experimental Setup

Agent Environments We evaluate WebATLAS on three web agent benchmarks spanning diverse task complexities and web environments.

- WebArena-Lite (Liu et al., 2024b) is a curated subset of WebArena (Zhou et al., 2024) adapted as a higher quality and more scalable benchmark for evaluating web agents in the most realistic setting possible, incorporating realistic scenarios such as unexpected environment failures. WebArena-Lite consisting of 165 tasks across websites of multiple domains (E-commerce, forums, content management, GitLab, and maps). Tasks require multi-step interactions and are evaluated via functional correctness criteria such as exact URL match, string match, or program-based verification.

- WebChoreArena (Miyai et al., 2025) introduces challenging tasks under the WebArena environment that require navigating through multiple websites, long-horizon planning, and massive memory to finish.

- WebVoyager (He et al., 2024) evaluates agents on real-world websites (e.g., Google Maps, Amazon, GitHub) with open-ended tasks graded by an LLM judge. It tests generalization to unseen, live web pages where the DOM structure and content change over time, requiring robustness to layout variability and dynamic content. We focus on the text-only setting.

Baselines We compare WebATLAS to a wide range of baselines: 1) WebPilot (Zhang et al., 2025), 2) Agentic Workflow Memory (AWM) (Wang et al., 2024a), 3) WebRL (Qi et al., 2024), 4) AgentOccam (Yang et al., 2024), 5) WebAgent-R1 (Wei et al., 2025), and 6) Plan-and-Act (Erdogan et al., 2025).

4 Results

4.1 Comparison with Baselines

Table 1 presents the task success rates across all three benchmarks. WebATLAS achieves state-of-the-art performance on all environments, specifically 68.5% on WebArena-Lite, 33.7% on WebChoreArena, and 63.2% on WebVoyager.

On WebArena-Lite, WebATLAS outperforms the strongest baseline, Plan-and-Act (Erdogan et al., 2025), by 14.6 absolute point, demonstrating the benefit of integrating memory-augmented reasoning with look-ahead simulation beyond static planning alone. On WebChoreArena, WebATLAS surpasses AgentOccam-Judge by 10.2 absolute point, indicating that our memory layers and replanning mechanism are particularly effective for long-horizon procedural tasks where tracking completed sub-steps is critical. On WebVoyager, WebATLAS improves over Plan-and-Act by 5.1 absolute point, showing that our approach generalizes to live, dynamic websites despite being developed primarily on self-hosted environments.

Notably, no single baseline performs consistently well across all three benchmarks—for instance, Plan-and-Act excels on WebVoyager but underperforms on WebChoreArena, while AgentOccam-Judge is competitive on WebChoreArena but lags on WebArena-Lite. In contrast, WebATLAS achieves the best performance on every benchmark, suggesting that the combination of structured memory, hierarchical planning, and look-ahead simulation provides robustness across diverse task types and web environments.

4.2 Ablation Study

We conduct an ablation study to isolate the contribution of each component in WebATLAS, starting from the AgentOccam base agent, shown in Table 2.

Memory Layers Adding cognitive map yields a substantial improvement of +6.6 point, as it enables the agent to reason about site structure and avoid redundant exploration. Episodic memory provides a gain of +2.8 point by supplying relevant procedural knowledge from prior episodes. Combining both memory layers produces a complementary effect (55.2%), confirming that structural and experiential knowledge address different failure modes.

Traditional Memory vs. Agentic Memory A key design decision is *how* the agent accesses its memory stores. Traditional refers to retrieving the raw observations, while agentic memory employs a memory agent to decide what to read / write with content filtering, summarization, and highlighting. We find that replacing this with *agentic memory access* yields a further 3.0 point improvement. This result indicates that allowing the agent to actively reason about its information needs, rather than passively receiving memory contents, is critical for effective utilization of stored knowledge.

Look-ahead Action Simulation Look-ahead action simulation (LAS) allows the agent to mentally simulate the consequences of candidate actions before committing to execution. With a single simulation step, LAS provides a +10.3% improvement over the base agent (58.2%), enabling the agent to reject actions likely to lead to dead ends or irreversible states. Extending to two simulation steps yields further gains (62.7%, +14.8% over base), as the agent can evaluate short action sequences and identify paths that require multi-step commitments. The return is diminishing beyond two steps.

Planning An initial high-level plan, generated at the start of each task and decomposing the goal into sub-goals, provides a 3.0 absolute point improvement (50.9%). While the planner helps structure exploration, static plans degrade when the agent encounters unexpected page layouts or action failures. Adding *replanning* dramatically amplifies the benefit (+10.9 absolute point total, reaching 58.8%). Replanning allows the agent to recover from plan failures and adapt its strategy as it gathers new information about the environment, closing the gap between the idealized plan and the actual web state.

Full System The complete WebATLAS agent integrates all components, achieving 68.5% success rate. Notably, this exceeds the sum of individual component gains, indicating positive interactions.

Table 2: Ablation study of on WebArena-Lite.

Agent	Succ. Rate
Base	47.9
Memory	
+ Cognitive Map	54.5
+ Episodic Memory	50.7
+ Both	55.2
+ Both, Agentic Memory Access	58.2
LAS	
+ LAS 1 step	58.2
+ LAS 2 steps	62.7
Planning	
+ Initial Plan	50.9
+ Initial Plan, Replanning	58.8
WebATLAS (Full)	68.5

The cognitive map informs replanning by providing structural context, look-ahead action simulation is more effective when guided by a high-level plan, and agentic memory access allows the agent to retrieve targeted knowledge conditioned on both its plan and simulated outcomes. The 20.6% absolute point improvement over the base agent demonstrates that structured reasoning and memory, when tightly integrated, substantially extend the capabilities of web agents.

4.3 Discussion

Error Analysis We manually categorize failure cases to identify the primary bottlenecks limiting agent performance shown in Figure 2. Observer error is the dominant failure mode (23.5%), where the agent misinterprets or overlooks critical information from the environment state, suggesting that grounding perception remains a core challenge. Long-horizon tasks account for 20.3% of failures, indicating that agents struggle to maintain coherent plans over extended action sequences with compounding errors. Planning errors and irrelevant actions (15.7%) reflect cases where the agent selects suboptimal or off-task actions despite correct observations. Unfamiliarity with the environment (13.8%) highlights generalization limitations when agents encounter novel interfaces or unseen configurations. Massive memory requirements (11.6%) point to failures when tasks demand retention and reasoning over large amounts of contextual information that exceed the agent's effective working memory. Environment errors (6.7%) represent environment-related failures such as evaluator failed or API timeouts outside the agent's control.

The remaining 8.4% comprise miscellaneous edge cases such as agent termination. These findings suggest that improving observation grounding and long-horizon reasoning should be prioritized in future agent architectures.

Cognitive Map Hit Rate During LAS, we track whether each retrieved transition corresponds to a previously observed state-action pair (hit) or requires interpolation from neighboring entries (miss). Across WebArena-Lite, the cognitive map achieves a hit rate of 73.2% at depth 1 and 58.6% at depth 2, reflecting that most immediate transitions have been encountered in prior exploration.

Time Cost of LAS The cognitive map is constructed during an **offline** exploration phase prior to task execution, meaning its memory footprint does not contribute to online inference cost. During task execution, LAS queries the pre-built cognitive map via hash-based lookup to simulate rollouts for each of the 3 candidate actions. Since no LLM calls or environment interactions are required for state prediction, the overhead is minimal: 1-step LAS increases total task completion time by only 1.6%, and 2-step LAS by 2.9%. This stands in contrast to tree-search methods that invoke the LLM or execute real environment actions at each branch, which scale multiplicatively with branching factor and depth. The negligible cost of LAS enables deeper deliberation without meaningful impact on wall-clock efficiency.

5 Related Work

Web Agents Early web agents predominantly rely on rule-based approaches, which lack the adaptability required for dynamic web environments. Recent advances have shifted toward learning-based methods. Zhou et al. (2024) introduced WebArena, a comprehensive benchmark for evaluating web agents across realistic multi-step tasks, establishing a foundation for systematic evaluation in this domain. Building on this, (Liu et al., 2024a) introduced WebArena-Lite, a curated subset addressing quality and scalability concerns in the original benchmark. Recent work has focused on enhancing LLM agents with structured approaches to planning and memory. Qi et al. (2024) applied reinforcement learning principles to web navigation, achieving notable improvements through policy optimization. Zhang et al. (2025) developed WebPilot, focusing on multimodal understanding

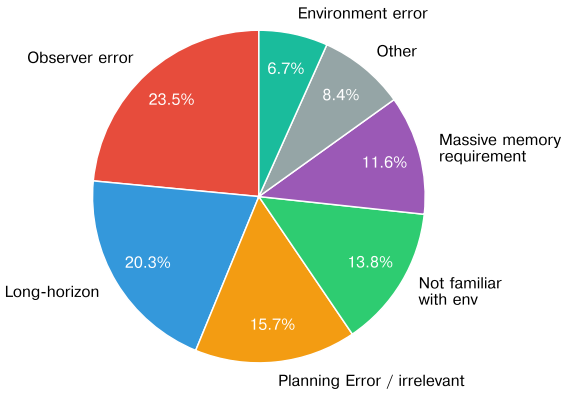

Figure 2: Error analysis for WebATLAS.

of web content, while Erdogan et al. (2025) introduced Plan-and-Act, emphasizing hierarchical task decomposition. (Wang et al., 2024b) introduced Agent Workflow Memory (AWM), demonstrating the importance of persistent memory in multi-step web tasks. (Yang et al., 2024) developed AgentOccam and demonstrated the effectiveness of simplifying the action space to natural language.

Memory-Augmented Agents Memory mechanisms have emerged as crucial components for long-horizon task completion. (Weston et al., 2014) established foundational work on memory networks, which has been adapted for sequential decision-making contexts. More recently, (Zhong et al., 2023) proposed MemoryBank, a comprehensive framework for managing episodic and semantic memory in LLM-based agents. The concept of cognitive maps, originally from cognitive science (Tolman, 1948), has been adapted for artificial agents. (Wayne et al., 2018) demonstrated neural implementations of cognitive mapping in reinforcement learning contexts, while (Park et al., 2023) showed how LLMs can maintain and utilize spatial-temporal memory representations for complex behavioral simulation.

6 Conclusion

We introduce WebATLAS, a web agent that consists of look-ahead action simulation (LAS) and multi-layered agentic memory. With LAS, WebATLAS simulates action outcomes with observations that are grounded to environments, avoiding blindly executing without aware of the potential consequences. Experiments on three agent benchmarks demonstrate that WebATLAS achieves state-of-the-art performance, outperforming previous best-performing baselines significantly.

Limitations

For a web agent, system robustness needs to be measured via stress tests that include UI drift, authentication flows, stochastic failures, and long-horizon, multi-session tasks. Additionally, next-generation planning should be budget-aware and safety-aware by design, trading off success, latency, and risk through calibrated uncertainty and constraint handling. We defer the study of these aspects to future works.

References

Hyungjoo Chae, Sunghwan Kim, Junhee Cho, Seungone Kim, Seungjun Moon, Gyeom Hwangbo, Dongha Lim, Minjin Kim, Yeonjun Hwang, Minju Gwak, and 1 others. 2025. Web-shepherd: Advancing prms for reinforcing web agents. *arXiv preprint arXiv:2505.15277*.

Hyungjoo et al. Chae. 2024. Web agents with world models": Learning and leveraging environment dynamics in web navigation. *arXiv preprint arXiv:2410.13232*.

Lutfi Eren Erdogan, Hiroki Furuta, Sehoon Kim, Nicholas Lee, Suhong Moon, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. 2025. Plan-and-act: Improving planning of agents for long-horizon tasks. In *Forty-second International Conference on Machine Learning*.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024. Webvoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6864–6890.

Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. 2024. Tree search for language model agents. *arXiv preprint arXiv:2407.01476*.

Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Yifan Xu, Xixuan Song, Shudan Zhang, Hanyu Lai, Xinyi Liu, Hanlin Zhao, Jiadai Sun, Xinyue Yang, Yu Yang, Zehan Qi, Shuntian Yao, Xueqiao Sun, Siyi Cheng, Qinkai Zheng, Hao Yu, and 11 others. 2024a. Visualagentbench: Towards large multimodal models as visual foundation agents. *Preprint*, arXiv:2408.06327.

Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Yifan Xu, Xixuan Song, Shudan Zhang, Hanyu Lai, Xinyi Liu, Hanlin Zhao, and 1 others. 2024b. Visualagentbench: Towards large multimodal models as visual foundation agents. *arXiv preprint arXiv:2408.06327*.

Atsuyuki Miyai, Zaiying Zhao, Kazuki Egashira, Atsuki Sato, Tatsumi Sunada, Shota Onohara, Hiromasa Yamanishi, Mashiro Toyooka, Kunato Nishina, Ryoma Maeda, and 1 others. 2025. Webchorearena: Evaluating web browsing agents on realistic tedious web tasks. *arXiv preprint arXiv:2506.01952*.

Joon Sung Park and 1 others. 2023. Generative agents: Interactive simulacra of human behavior. *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. 2017. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, and 1 others. 2024. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *arXiv preprint arXiv:2411.02337*.

Paloma Sodhi, SRK Branavan, Yoav Artzi, and Ryan McDonald. 2023. Step: Stacked llm policies for web actions. *arXiv preprint arXiv:2310.03720*.

Sumedh A Sontakke, Arash Mehrjou, Laurent Itti, and Bernhard Schölkopf. 2021. Causal curiosity: Rl agents discovering self-supervised experiments for causal representation learning. In *International conference on machine learning*, pages 9848–9858. PMLR.

Edward C Tolman. 1948. Cognitive maps in rats and men. *Psychological review*, 55(4):189.

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

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

Greg Wayne and 1 others. 2018. Unsupervised predictive memory in a goal-directed agent. *arXiv preprint arXiv:1803.10760*.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, and 1 others. 2025. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 7920–7939.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. 2024. Agentoccam: A simple yet strong baseline for llm-based web agents. *arXiv preprint arXiv:2410.13825*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, pages 20744–20757. Curran Associates, Inc.

Yao Zhang, Zijian Ma, Yunpu Ma, Zhen Han, Yu Wu, and Volker Tresp. 2025. Webpilot: A versatile and autonomous multi-agent system for web task execution with strategic exploration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 23378–23386.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2023. Enhancing large language models with long-term memory. *Preprint*, arXiv:2305.10250.

Shuyan Zhou, Arvind Neelakantan, and 1 others. 2024. Webarena: A realistic web environment for building autonomous agents. In *Proceedings of ICLR*.

A Additional Results

We present the detailed agent performances for each website on WebArena-Lite in Figure 3.

Table 3: Task success rate comparison on WebArena-Lite and WebVoyager environments. Best performance is in **bold** and second best is _underlined. Baseline performances are directly referenced from original papers if available.

Agent	Avg w/ Multi-site	Avg w/o Multi-site	Gitlab	Reddit	Shopping	Shopping Admin	Maps	Multi-Site
WebPilot	35.3	39.4	65.1	36.9	24.7	33.9	–	–
AWM	33.0	31.8	50.9	30.8	29.1	43.3	–	–
WebRL	48.1	50.0	78.9	44.4	54.3	40.0	–	–
Plan-and-Act	53.9	57.5	53.3	84.2	**55.6**	48.6	46.6	30.0
AgentOccam-Judge	43.1	37.8	61.3	40.6	45.6	46.8	14.6	–
AgentOccam-Judge + Claude-4-Sonnet	46.7	49.7	66.7	68.4	40.0	42.9	30.8	30.0
WebATLAS (Ours)	**68.5**	**71.3**	**73.3**	**84.2**	53.3	**77.1**	**42.3**	**40.0**

B Agent Prompts

B.1 Planner Prompt

Planner Prompt

You are an AI assistant tasked with generating structured checklists that highlight key subgoals necessary to complete a task.

Task Description

- User Instruction (Goal): `{task_objective}`
- Start Website URL: `{initial_url}`
- Initial Observation: `{initial_html}`

Guidelines for Checklist Generation

1. **Identify Essential High-Level Subgoals:**
   - A subgoal should represent a significant step involving user interaction that leads to noticeable page transitions or meaningful changes in system state.
   - Consolidate closely related user actions (such as applying multiple filters or selecting several options) into a single subgoal.
   - Prioritize only the most critical interactions necessary for meaningful progression, avoiding minor or unnecessary steps (e.g., scroll, hover).
2. **Provide a Concise Subgoal Analysis:**
   - Before creating the checklist, offer a brief paragraph summarizing the main subgoals, emphasizing significant transitions or page-level interactions.
3. **Ensure Clear Goal:**
   - If multiple related interactions occur (e.g., setting filters 1, 2, and 3), combine them into one subgoal with clear criteria verifying all required conditions.
   - The checklist should contain only essential steps and should not exceed five critical subgoals.
   - It is not necessary to use all five items if fewer steps adequately represent the essential subgoals.

Output Format: Before generating the checklist, first produce a concise subgoal analysis in a single paragraph summarizing the required interactions. Then generate the checklist.

B.2 Actor Prompt

Actor Prompt

You are an AI assistant performing tasks on a web browser. You will be provided with task objective, current step, web page observations, previous plans, and interaction history. You need to issue an action for this step.
Generate the response in the following format: `{output_specifications}`
You are ONLY allowed to use the following action commands. Only issue one single action.
Planning Actions

- `branch [parent_plan_id] [new_subplan_intent]`: Create a new subplan based on previous plans. Ensure the new subplan is connected to the appropriate parent plan by using its ID.
- `prune [resume_plan_id] [reason]`: Return to a previous plan state when the current plan is deemed impractical.

Web Interaction Actions

- `click [id]`: Click on an element with its numerical ID on the webpage.
- `go_back`: Return to the previously viewed page.
- `go_home`: Return to the homepage.
- `note [content]`: Take note of important info w.r.t. completing the task.
- `stop [answer]`: Stop interaction and return response.
- `type [id] [content] [press_enter_after=0|1]`: Type content into a field with a specific ID. By default, "Enter" is pressed after typing unless `press_enter_after` is set to 0.

B.3 Critic Prompt

Critic Prompt

You are a seasoned web navigator. You now assess the value and risk of several web navigation actions based on the objective, the previous interaction history, and the web's current state. Then, you select the action with the most value and least risk with which you would earn the maximum objective fulfillment reward in the future.
Adhere to the following output format: `{output_specifications}`
Note that `branch` and `prune` are planning actions that will modify the previous plan section and won't interact with the web environment.

B.4 Episodic Memory Prompt

Episodic Memory Prompt

You are an expert in summarizing agent exploration. You are given a list of exploration trajectories and a previous environment dynamics summary. Your task is to update the summary by incorporating new evidence. Focus on rules, constraints, and possibilities revealed by the environment.

Output Must Include:

1. **Allowed Actions** – Actions that successfully changed the state or were permitted.
2. **Prohibited / Invalid Actions** – Actions that failed, were blocked, or had no effect.
3. **Environment-Specific Formats** – Date format, search format, URL format.
4. **Newly Exposed Options** – Actions that became visible during exploration.
5. **Environment Reliability** – Inconsistent behavior, errors, or misleading cues.
6. **Coverage & Unknowns** – Confirmed dynamics vs. uncertain or untested areas.

Notes: Keep concise ($<$5 bullets per section, $<$500 tokens). If new findings contradict older ones, mark explicitly. Do not discard prior information unless contradicted.