TRAINING LLM AGENTS FOR SPONTANEOUS, TASK-INDEPENDENT SELF-EVOLUTION VIA WORLD KNOWLEDGE EXPLORATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Most agents today "self-evolve" by following rewards and rules defined by humans. However, this process remains fundamentally dependent on external supervision; without human guidance, the evolution stops. In this work, we train agents to possess an intrinsic meta-evolution capability to spontaneously learn about unseen environments prior to task execution. To instill this ability, we design an outcome-based reward mechanism that measures how much an agent's self-generated world knowledge improves its success rate on downstream tasks. This reward signal is used exclusively during the training phase to teach the model how to explore and summarize effectively. At inference time, the agent requires no external rewards or human instructions. It spontaneously performs native self-evolution to adapt to unknown environments using its internal parameters. When applied to Qwen3-30B and Seed-OSS-36B, this shift to native evolution yields a 20% performance increase on WebVoyager and WebWalker. Most strikingly, the generated world knowledge even enables a compact 14B Qwen3 model to outperform the unassisted Gemini-2.5-Flash, establishing a new paradigm for truly evolving agents. Code is available at https://anonymous.4open.science/status/self_eolving_wk-036D.

1 INTRODUCTION

Current research on "self-evolving" agents is largely an illusion. Most existing methods do not allow an agent to evolve on its own; instead, they depend on human-defined workflows and verified reward signals to guide every step of improvement. If these external rewards or instructions are removed, the evolution stops. We argue that such agents are not truly autonomous—they are merely being instructed by humans within predefined guidance. They lack the fundamental ability to decide their own direction of growth when facing a completely new environment.

This paradigm is fundamentally different from human intuition. Human intelligence is naturally curious and proactive. When we enter a new city or start using a new software, we spontaneously learn the layout and the underlying logic, even without a specific task or a verified reward. This learning process is entirely workflow-free and task-independent. We build an internal map of the world simply because understanding the environment is a prerequisite for intelligence. Today's agents, however, are passive; they wait for instructions and rewards before they begin to "evolve."

In this work, we bridge this gap by granting agents **Native Agency**. Our goal is to move beyond task-specific optimization and achieve a truly **Workflow-free** and **Task-Independent** self-evolution. We train the model to possess an intrinsic meta-evolution capability: the ability to explore a novel environment and distill its observations into structured "World Knowledge" entirely on its own. As shown in Figure 1, our agent does not follow a predefined script; it defines its own path of discovery.

The core challenge is: how can we train an agent to explore and summarize effectively without human-provided tasks? To solve this, we propose an **outcome-based reward mechanism** used exclusively during the training phase. We measure the quality of self-generated knowledge by its "utility"— specifically, how much it improves the success rate on downstream tasks. This allows the model to learn *how* to evolve during training, using a multistage pipeline that includes teacher-model bootstrapping (SFT) and on-policy reinforcement rejection sampling (RFT). Once trained, the agent no longer requires any external guidance or reward to adapt to an unseen world at inference time.

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.

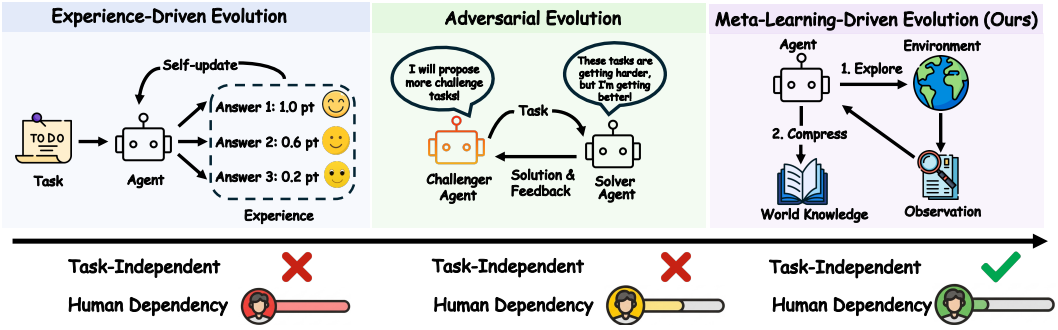

Figure 1: Progression of self-evolution agent paradigms.**Left:** *Experience-Driven Evolution* updates agents through predefined tasks and external rewards, requiring extensive human effort to carefully design these components. **Center:** *Adversarial Evolution* employs a challenger-solver dynamic where one proposes harder tasks and the other improves to solve them; while tasks and rewards are agent-generated, the cooperative pipeline still requires human setup. **Right:** Our *Meta-Learning-Driven Evolution* enables agents to autonomously explore and compress environmental observations into reusable world knowledge for adaptation, achieving a truly task-independent paradigm with minimal human intervention.

We evaluate our approach on two major web-based benchmarks: WebVoyager (He et al., 2024) and WebWalker (Wu et al., 2025). Our experiments show that our method, when applied to Qwen3-30B (Yang et al., 2025) and Seed-OSS-36B (Team, 2025a), achieves a significant absolute performance increase of approximately 20% over standard baselines. Strikingly, the generated world knowledge even enables a compact Qwen3-14B model to outperform the unassisted Gemini-2.5-Flash. These results prove that agents can be trained to possess the innate ability to understand and adapt to the unknown entirely on their own, without any human intervention or inference-time rewards.

2 Related Works

2.1 Self-Evolving Agents

Self-evolving agents are designed to autonomously explore novel environments and continuously improve their capabilities without direct human intervention (Gao et al., 2025; Ren et al., 2026). However, the current paradigm of "self-evolution" is often an illusion. In reality, these agents still heavily rely on meticulously human-defined workflows and verified, environment-specific reward signals to guide every incremental step of their improvement. Broadly, existing approaches can be categorized into the following two paradigms:

Experience-Driven Evolution. As depicted in the left panel of Figure 1, this paradigm fundamentally relies on human-crafted tasks and predefined reward functions tailored to a target environment. The agent iteratively attempts to solve these tasks, generating execution trajectories that are subsequently evaluated by the reward signals. These trajectory-score pairs—collectively termed as **experience**—serve as the primary learning signal. By leveraging this accumulated experience, the agent optimizes its future performance through various updating mechanisms, such as refining its system prompts (Zhang et al., 2025d; Wang et al., 2025a; Xiang et al., 2025; Shang et al., 2024; Yin & Wang, 2025; Tao et al., 2026), expanding external memory databases (Ouyang et al., 2025; Zhang et al., 2025a; Zhao et al., 2024; Fu et al., 2024; Xu et al., 2025; Chhikara et al., 2025; Zhang et al., 2026a; Fang et al., 2026), augmenting tool and skill libraries (Zhang et al., 2025b; Zheng et al., 2025; Qu et al., 2024; Wang et al., 2024a), or directly fine-tuning its internal model parameters (Zhang et al., 2025c; Fang et al., 2025a; Wang & Xiong, 2025; Wang et al., 2025b; Su et al., 2025; Zhang et al., 2026b). However, this paradigm is fundamentally bottlenecked by the massive human labor required to engineer these tasks and rewards. Rather than genuinely exploring, the agent merely passively adapts to the environment by studying from these human-provided "textbooks".

Adversarial Evolution. To alleviate manual design efforts, an alternative paradigm employs heavily engineered **adversarial workflows**. As illustrated in Figure 1 (middle), a **challenger** agent synthesizes environment-specific tasks for a **solver** to execute. Through this zero-sum game, the solver refines its capabilities while the challenger generates increasingly difficult tasks to push its

boundaries (Liu et al., 2025; Huang et al., 2025; Zhou et al., 2025; Simonds et al., 2025; Yue et al., 2026; Yang et al., 2026). Although this paradigm substantially reduces human labor by bypassing manual task and reward design, it merely shifts the engineering burden to orchestrating complex agent workflows. Furthermore, the agent remains trapped solving synthesized "exercise books", failing to break free and engage in genuine, unguided exploration within the environment.

Meta-Learning-Driven Evolution (Ours). To overcome the limitations of previous paradigms and empower agents to achieve **workflow-free** and **task-independent** self-evolution, we propose a novel **meta-evolution paradigm**. As illustrated in the right panel of Figure 1, under this paradigm, the agent spontaneously explores the environment and compresses raw environmental observations into structured **world knowledge**. This knowledge acts as a "mental map" that significantly enhances downstream performance, eliminating the need for human intervention and enabling fully autonomous self-evolution.

2.2 Test-Time Training

Test-Time Training (TTT) is a paradigm where models adapt to new distributions by performing self-supervised optimization during the inference phase (Sun et al., 2020). Recent advancements have extended this concept to sequence modeling and large language models, employing auxiliary tasks or hidden state updates to refine model behavior on-the-fly (Behrouz et al., 2025a; 2024; 2025b; Sun et al., 2024; Wang et al., 2024b; Liu et al., 2026; Lu et al., 2026; Moradi et al., 2025; Hu et al., 2025). However, TTT fundamentally requires gradient-based weight updates or parameter modifications during inference, which makes it incompatible with mainstream high-throughput inference frameworks (Kwon et al., 2023; Aminabadi et al., 2022). Unlike TTT, which necessitates runtime training, our meta-evolution paradigm distills environmental observations into structured world knowledge, which is then fed directly into the agent's prompt as an external context module.

3 Methodology

The fundamental limitation of LLM agents lies in their reactive nature: they wait for a task to be assigned before they begin to interact with the world. Formally, a standard agent policy follows $a \in \mathcal{A} \sim \pi(a|o, \text{Task})$, where every action a is strictly conditioned on a current observation $o \in \mathcal{O}$ and a pre-defined goal. We argue that true intelligence requires **Native Evolution**—the ability to proactively understand a new environment before a task exists.

In this work, we introduce **World Knowledge** ($\mathcal{K}$), a compact and structured representation of an environment's overall landscape. To ensure compatibility with existing agent architectures, we implement $\mathcal{K}$ as a **Markdown document**—an external module that can be directly loaded into the agent's context, similar to how functional *skills* are integrated in recent agent frameworks[1]. However, while a "skill" typically provides task-specific functions (e.g., `webapp-testing`), our World Knowledge instead captures the intrinsic logic of specific **Environment Instances**, providing the agent with a "mental map" of a given instance, such as the ICLR 2026 website, a particular game world, or a complex code repository.

Our framework decouples the agent's life cycle into two distinct phases:

1. **Native Evolution Phase:** Upon entering a new environment E, the agent spontaneously performs exploration and summarization to generate its own world knowledge: $\mathcal{K} \leftarrow \pi_{\text{evolve}}(\mathcal{K}|E)$. This process is entirely **task-independent** at inference time.
2. **Knowledge-Enhanced Execution Phase:** When a downstream task is eventually assigned, the agent utilizes $\mathcal{K}$ to guide its actions: $a_t \sim \pi_{\text{task}}(a_t|o_t, \mathcal{K}, \text{Task})$.

To achieve this, an agent must possess a **meta-evolution** capability, which involves (1) **Planning and Exploration**: formulating a goal-directed plan to prioritize high-value regions of the environment, and (2) **Information Management**: distilling vast, heterogeneous data into an information-dense representation $\mathcal{K}$.

[1] https://github.com/anthropics/skills/tree/main/skills

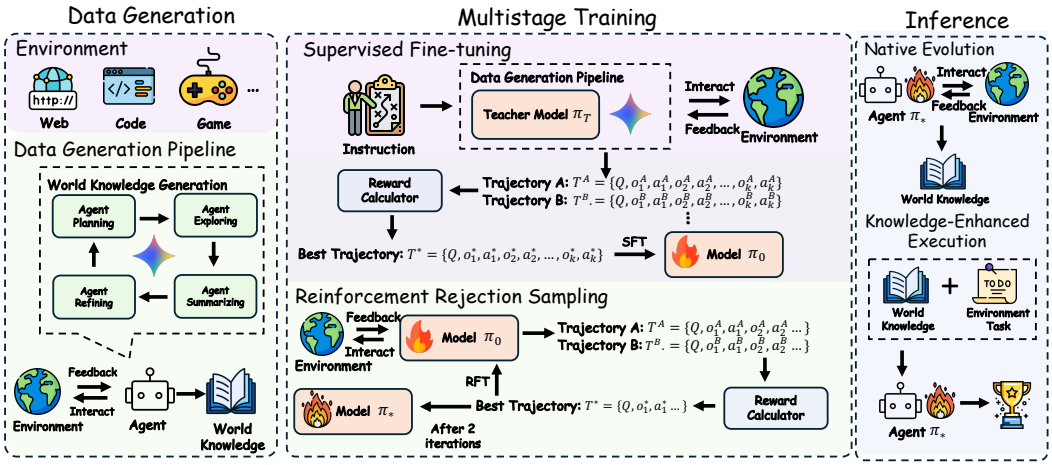

Figure 2: Overview of our method.

While the principles of Native Evolution are domain-agnostic, we ground our implementation in the context of **web agents** to provide a concrete illustration of our approach. Web navigation serves as a representative and challenging testbed, as it requires the agent to handle highly unstructured and dynamic environments. However, bridging the gap between this conceptual framework and a functional autonomous agent reveals a significant hurdle: standard LLMs are typically trained for reactive instruction-following and lack the inherent instinct to explore for the sake of knowledge.

To empower the model with such Native Evolution, we must develop a specialized training paradigm that transforms the model from a passive tool into a proactive learner. This leads to the core technical challenge of our work: **What is the training signal for evolution?** Since the evolution phase itself is task-free, we cannot rely on immediate ground-truth labels. In the following sections, we introduce our solution: an **outcome-based reward mechanism** that uses the utility of $\mathcal{K}$ in potential downstream tasks as the primary learning signal.

3.1 Outcome-Based Reward Design

To address the lack of supervision in task-free exploration, we propose an **outcome-based reward mechanism**. The core intuition is functional: the quality of World Knowledge $\mathcal{K}$ is defined by its end-to-end utility—specifically, how much it "empowers" the agent to perform better in that environment.

Formally, let $\mathcal{T}_E$ be a set of downstream tasks associated with an environment E. We define the reward R_{evolve} for a generated $\mathcal{K}$ as the **potential success gain**:

$$R_{\text{evolve}}(\mathcal{K}) = \text{Success}(\mathcal{T}_E|\mathcal{K}) - \text{Success}(\mathcal{T}_E|\emptyset) \tag{1}$$

where $\text{Success}(\mathcal{T}_E|\mathcal{K})$ represents the agent's performance aided by $\mathcal{K}$, and $\text{Success}(\mathcal{T}_E|\emptyset)$ is the baseline performance without prior knowledge.

In our practical implementation, we construct a training set consisting of 314 deep search questions spanning 10 websites across diverse domains. We leverage the ground-truth tasks and labels from this set to empirically calculate the success rate. For an environment instance (i.e., a specific website) with M labeled tasks $\{(Q_j, A_j)\}_{j=1}^M$, the term $\text{Success}(\mathcal{T}_E|\mathcal{K})$ is computed as:

$$\text{Success}(\mathcal{T}_E|\mathcal{K}) = \frac{1}{M}\sum_{j=1}^{M}[f(Q_j, \mathcal{K}) = A_j] \tag{2}$$

where $f(Q_j, \mathcal{K})$ is the agent's predicted answer for query Q_j given the world knowledge $\mathcal{K}$.

Crucially, this reward signal is used exclusively during training. It serves as a meta-learning signal that teaches the model *how* to identify and compress high-value information. At inference time, the agent spontaneously performs native evolution to adapt to new environments without requiring any external rewards, predefined task sets, or human verification. This transition from reward-driven training to reward-free inference is the hallmark of true Native Evolution.

Building upon this reward mechanism, we propose a two-stage training framework to cultivate the agent's meta-evolution capabilities. As illustrated in Figure 2, our approach consists of an initial Supervised Fine-Tuning (SFT) phase followed by Reinforcement-based Rejection Sampling (RFT).

3.2 Supervised Fine-Tuning

In the first training stage, we warm up the base policy model π_{θ_0} through imitation learning against a strong teacher model π_T (instantiated as Gemini-2.5-Pro (Comanici et al., 2025)). The goal is to allow the model to internalize the core meta-evolution behaviors: planning, exploring, refining, and summarizing. To achieve this, we establish a **data generation pipeline** that guides π_T to interact with diverse web environments and autonomously construct structured world knowledge $\mathcal{K}$. By fine-tuning on expert trajectories generated through this pipeline, the model learns to execute these complex tasks without intensive prompting at inference time.

To ensure the quality of the SFT data, we implement a selection mechanism based on the reward defined in Section 3.1. For each environment instance in our training set, the teacher model π_T generates three candidate world knowledge representations $\{\mathcal{K}_i\}_{i=1}^{3}$. We then evaluate these candidates by measuring the performance gain they provide to our baseline agent (Qwen3-30B-A3B) on downstream tasks. We select the best-performing candidate $\mathcal{K}^*$ and its corresponding full exploration trajectory:

$$T^* = \{Q, o_1^*, a_1^*, o_2^*, a_2^*, \ldots, o_k^*, a_k^*\} \tag{3}$$

to serve as our step-level training data.

Empirically, the teacher-generated knowledge $\mathcal{K}^*$ demonstrates high utility, yielding an average absolute accuracy improvement of **10.72%** for the Qwen3-30B-A3B model on training tasks compared to the zero-knowledge baseline. The resulting expert trajectories T^* reflect the high complexity of the task, with an average length of **374.8 steps** and a substantial information density of **3,322.4 tokens per step** (comprising observations o and actions a). By fine-tuning π_{θ_0} on these high-quality trajectories, we obtain the updated policy model π_{θ_1}, which possesses a foundational instinct for autonomous evolution.

3.3 Reinforcement-based Rejection Sampling

To further catalyze the emergence of sophisticated exploration and information management strategies, we employ reinforcement learning to optimize the policy via trial-and-error. However, standard online RL algorithms, such as GRPO, are computationally prohibitive in our setting for two reasons: (1) **Extremely Long Horizons**: generating world knowledge $\mathcal{K}$ involves hundreds of steps, leading to sparse rewards and immense memory overhead during backpropagation; and (2) **Heavy Reward Evaluation**: our outcome-based reward requires executing an auxiliary agent across multiple downstream tasks to evaluate a single $\mathcal{K}$, making real-time reward calculation during training cycles impractical.

Consequently, we adopt a **Rejection Sampling Fine-Tuning (RFT)** approach, which decouples trajectory generation from policy updates. Following the pipeline established in Section 3.2, the updated policy π_{θ_1} autonomously explores environment instances to produce C candidate world knowledge representations. We evaluate these candidates using the reward function R_{evolve} and select the highest-scoring trajectories—those demonstrating the strongest "meta-evolution" utility—to construct the training set for the next iteration.

We perform this rejection sampling process for two iterations. This iterative refinement allows the model to progressively correct suboptimal exploration paths and discover more compact, high-utility representations of world knowledge. The final optimized policy, denoted as π_{θ^*}, internalizes the ability to adapt to unknown environments. At inference time, the agent spontaneously executes this learned evolutionary logic, constructing world knowledge that significantly boosts its performance on previously unseen downstream tasks without any external guidance.

4 EXPERIMENTS

In this section, we evaluate our approach on two challenging web-based benchmarks: WebWalker (Wu et al., 2025) and WebVoyager (He et al., 2024). Our experiments aim to address the following research questions:

RQ1 (Effectiveness): Does the meta-evolution capability indeed improve an agent's success rate on downstream tasks?

RQ2 (Transferability): Is the world knowledge $\mathcal{K}$ model-agnostic? Can it help models without the meta-evolution capability adapt to unseen environments?

RQ3 (Ablation): How do the SFT and RFT stages individually contribute to the meta-evolution capability?

RQ4 (Sensitivity Analysis): How does the length of the generated world knowledge impact the agent's downstream performance?

4.1 EXPERIMENTAL SETTINGS

Agent Framework. We select web agents as the experimental setting for our proposed method and adopt Cognitive Kernel-Pro (Fang et al., 2025b) as our agent framework. Within this framework, the interactive webpage environment is implemented using Playwright [2]. The action space consists of predefined webpage operations, including `click`, `scroll`, `goto`, `goback`, and `stop`. At each step, the agent's observation corresponds to the accessibility tree of the currently visible webpage components. Upon executing a selected action, the environment updates the webpage state according to the execution outcome and produces the subsequent observation based on this updated state. An agent trajectory terminates when the task is completed (as determined by the agent), the number of interaction steps reaches a maximum limit t, or the execution time exceeds a predefined limit L (in seconds). In our experiments, we set $t = 500, L = 43,200$ for world knowledge generation and $t = 100, L = 3,600$ for downstream task answering.

Backbone Models. We adopt Qwen3-30B-A3B-Instruct-2507 (Yang et al., 2025) and Seed-OSS-36B-Instruct (Team, 2025a) as our backbone models for training and evaluation. To balance diversity and determinism, we set `temperature, top-p` to `0.3, 0.95` during world knowledge generation for richer candidate sampling, and to `0, 0.95` during downstream task answering to improve answer stability.

Evaluation Benchmarks. We adopt WebWalker and WebVoyager as our evaluation benchmarks, constructing subsets from both for our experiments. We filter out questions from both datasets that can be directly answered using the backbone models' pretrained knowledge, in order to ensure a rigorous evaluation. After this filtering and further excluding the websites used in the training set, we randomly select 32 websites spanning the four WebWalker domains—conference, game, organization, and education; for WebVoyager, we select tasks from four specific websites: Wolfram, Apple, Dictionary, and Coursera. For each selected website, we randomly sample 50 questions, or use all available questions if fewer than 50 remain. Following this procedure, we obtain a total of 1,642 evaluation samples. We utilize **accuracy** as the evaluation metric for all benchmarks. Websites for training and evaluation can be seen in Appendix B.

Evaluation Protocol. For WebWalker, we employ Qwen-2.5-32B (Team, 2024) as the judge to verify whether the agent's final answer matches the ground-truth solution. The judge produces a binary score (0 or 1), indicating incorrect or correct answers, respectively. For WebVoyager, we use DeepSeek-V4-Flash (Xu et al., 2026) as the judge to handle the exceptionally long context inputs required for evaluation. Following the official protocol, we provide the model with the question, the agent's answer, and the trajectory observations (the accessibility tree at each step), and ask it to determine whether the task is `SUCCESS` or `NOT_SUCCESS`. The verification prompts for both benchmarks are provided in Appendix D.

[2] https://playwright.dev/

Method	WebWalker					WebVoyager				
	Conf.	Game	Org.	Edu.	**Avg.**	Wolfram	Apple	Dict.	Coursera	**Avg.**
Backbone: Qwen3-30B-A3B-Instruct-2507										
Without	23.03	22.73	22.11	17.28	20.71	54.30	37.20	41.86	30.95	41.08
Prompt-Only (Gemini)	36.51	38.96	29.75	31.66	33.15	**73.90**	**53.40**	51.16	45.23	55.92
Prompt-Only (Base)	29.21	21.75	19.21	23.12	22.28	54.30	32.56	42.85	33.33	40.76
Ours (SFT)	55.62	37.50	39.67	34.92	39.54	60.87	41.86	62.79	40.48	51.50
Ours (RFT)	**62.08**	**42.21**	**47.93**	**38.78**	**45.34**	58.70	48.84	**67.44**	**54.76**	**57.44**
Backbone: Seed-OSS-36B-Instruct										
Without	23.97	7.79	17.17	13.05	14.63	54.30	48.84	23.26	33.33	39.93
Prompt-Only (Gemini)	47.75	19.48	25.62	24.90	26.77	58.60	53.49	**62.79**	52.38	**56.82**
Prompt-Only (Base)	31.46	12.66	15.91	15.06	16.83	47.82	46.34	30.23	22.50	36.72
Ours (SFT)	46.63	23.38	28.51	**30.52**	30.31	**71.73**	51.16	53.49	47.61	56.00
Ours (RFT)	**55.82**	**26.30**	**34.09**	29.12	**33.40**	63.04	**55.81**	51.16	**57.14**	56.79

Table 1: Accuracy (%) on WebWalker and WebVoyager benchmarks across four WebWalker subdomains (Conf., Game, Org., Edu.) and four WebVoyager sites (Wolfram, Apple, Dict., Coursera). Best result per column within each backbone is highlighted. **Without** refers to the original backbone model answering directly without world knowledge. **Ours (RFT)** denotes our final model. **Bold** indicates the best performance within each backbone setting.

4.2 Implementation Details

Input Processing for Native Evolution Phase. For websites with a large number of subpages, directly providing the homepage URL U and allowing unrestricted exploration often leads to long runtimes and unstable outputs. To mitigate this, we pre-process the website into a more navigable format. First, we model the website as a directed graph, assigning an importance score to each webpage based on its linkage topology. Next, we group these pages into clusters based on shared URL prefixes. This process yields a clustered, graph-based representation of the website, denoted as $\mathcal{G}(U)$ (see Appendix A for details). Consequently, $\mathcal{G}(U)$ **replaces the raw homepage as the structured entry point to the environment** E. By filtering out noise and organizing large-scale web content into interpretable clusters, this approach significantly reduces the agent's cognitive load during spontaneous exploration.

Instruction Construction. We divide the instruction design into two distinct phases. In the **SFT stage**, we design an instruction to guide the teacher agent in generating high-quality world knowledge under explicit token budget constraints. Given $\mathcal{G}(U)$, the agent first formulates a token allocation **plan** that distributes the budget across different groups. It then **explores** the website and generates **summaries** for each group by selecting high-value subpages based on both structural importance scores and semantic relevance inferred from page content, while filtering out low-quality pages. Finally, the agent **refines** the generated world knowledge to meet the token constraints. In the **RFT and inference stages**, since the model has already acquired preliminary exploration and summarization capabilities, we design a simplified prompt. Compared to the SFT version, this simplified prompt omits numerous detailed operational steps and reminders about potential pitfalls, significantly reducing the total prompt token count from 4,323 to 788. Ultimately, our model can autonomously explore and adapt to new environments with minimal human guidance. The detailed prompts for both stages are provided in Appendix D.

4.3 Effectiveness of Meta-Evolution Capability (RQ1)

We first evaluate whether our training framework instills the meta-evolution capability and how this self-generated world knowledge $\mathcal{K}$ impacts downstream performance. We compare five configurations: (1) **Without**: The original backbone answering questions directly without environment exploration. (2) **Prompt-only (Gemini)**: Utilizing Gemini-2.5-Pro to generate $\mathcal{K}$ using the expert prompt in Appendix D. (3) **Prompt-only (Base)**: The untrained base model attempting to generate $\mathcal{K}$ with the same expert prompt. (4) **Ours (SFT)** and **Ours (RFT)**: Our agent at different training stages autonomously generating $\mathcal{K}$ to guide its own task-solving. Results are summarized in Table 1.

The results demonstrate that **our framework empowers agents with meta-evolution capabilities, turning self-generated knowledge from a liability into a significant asset.** We observe that while

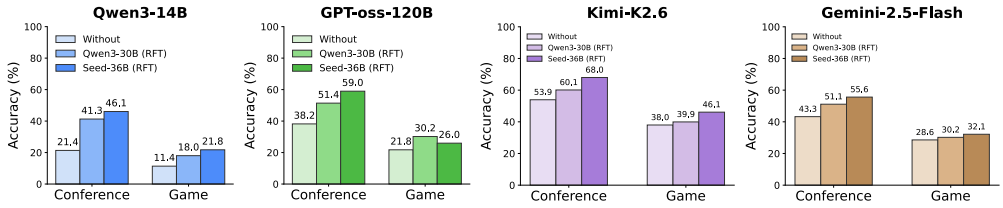

Figure 3: Cross-model world knowledge transfer.

base models can follow complex exploration instructions, they lack the intrinsic capacity to distill high-value information. Instead, they often produce noisy or hallucinated guidance that distracts the agent during execution, as evidenced by the fact that on WebWalker, *Prompt-only (Base)* even underperforms the *Without* baseline on the Game (21.75% vs. 22.73%) and Organization (19.21% vs. 22.11%) domains.

In contrast, our trained models successfully overcome this "noise" bottleneck. **Ours (RFT)** achieves a **45.34%** success rate on WebWalker, outperforming the *Without* baseline by nearly **25%** absolute and, notably, surpassing the strong **Prompt-only (Gemini)** teacher (33.15%). This result proves that our outcome-based reward mechanism provides the correct training signal, enabling the model to refine its exploration policy through reinforcement-based rejection sampling and ultimately surpass its teacher model.

Efficiency. Table 2 demonstrates how world knowledge significantly reduces an agent's execution steps on downstream tasks. As observed, the integration of world knowledge leads to an average efficiency improvement of 24% across various domain websites. **These findings suggest that world knowledge acts as a cognitive "map" of the environment. It provides crucial structural priors guiding the agent to swiftly navigate relevant web pages and extract the required answers.** Conversely, agents lacking this knowledge must start from the homepage, resorting to blind, inefficient step-by-step exploration. A case study in Appendix B.1 further illustrates in detail how world knowledge helps the agent solve problems both faster and better.

	Conference	Game	Organization	Education	Avg.
Qwen3-30B	21.98	25.04	23.15	32.14	24.28
Qwen3-30B with $\mathcal{K}$	**14.00**	**17.74**	**18.62**	**28.18**	**20.05**
Improve Ratio	0.36	0.29	0.20	0.12	0.24

Table 2: Efficiency evaluation (average steps).

4.4 Cross-Model World Knowledge Transfer (RQ2)

To demonstrate that our self-evolved world knowledge $\mathcal{K}$ functions as a **model-agnostic, universal protocol**, we conduct evaluations on models across different parameter scales and families: Qwen3-14B (Team, 2025b), GPT-OSS-120B (OpenAI, 2025), Kimi-K2.6 (Moonshot AI, 2026), and Gemini-2.5-Flash. As shown in Figure 3, we identify two striking findings:

1. Universal Portability Across Frontiers. The generated world knowledge is highly transferable, yielding substantial gains across all target models. For instance, Seed-36B's knowledge increases the average accuracy of the Qwen3-14B by 17.5% across two domains, and even boosts the flagship Kimi-K2.6 by 11.1%.

2. Exploration over Parameters: The Knowledge Scaling. Our results suggest that **high-quality environment exploration can be more effective than brute-force parameter scaling.** Most strikingly, the 14B Qwen3 model, when equipped with world knowledge, **outperforms the unassisted Gemini-2.5-Flash (46.1% vs. 43.3% on Conference domain)**. This indicates that world knowledge ($\mathcal{K}$) of the environment is a more critical bottleneck for agent performance than model scale.

We believe these findings reveal a fundamental shift in agent design: the path to frontier-level performance lies not just in scaling a model's parameters, but in scaling its capacity for proactive exploration and learning in unseen environments.

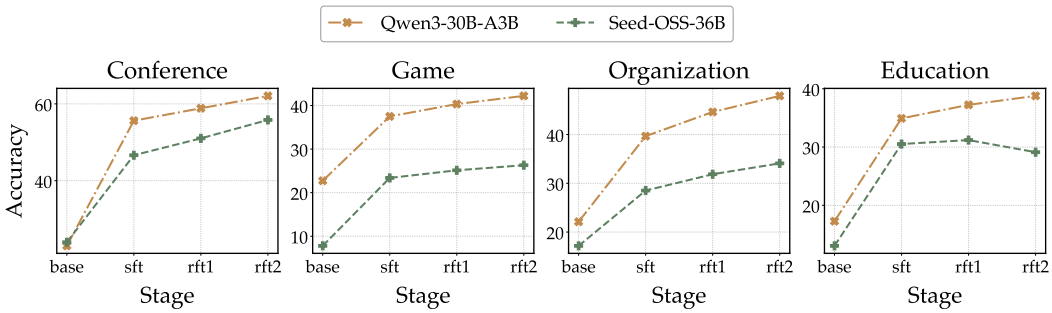

Figure 4: Performance trends across training stages.

4.5 Ablation Study (RQ3)

We examine how performance evolves across four training phases—base, SFT, and two RFT rounds (rft1, rft2)—as the agent generates and uses its own world knowledge to solve downstream tasks (Figure 4). Overall, **the models exhibit a clear upward trend in performance as the training progresses**. A notable observation is that both the initial SFT stage and the first round of reinforcement fine-tuning (rft1) yield substantial performance boosts, whereas the subsequent training round (rft2) generally provides more marginal gains or even slight fluctuations. This indicates that the SFT and rft1 stages lay a crucial foundation for the meta-evolution capability of the agent.

4.6 Sensitivity Analysis (RQ4)

To study how world knowledge length affects downstream performance, we specify lower and upper token bounds in the prompt and evaluate Qwen3-30B-A3B under five settings: 0 (no world knowledge), 4k∼8k, 8k∼16k, 16k∼32k, and 32k∼64k tokens.

As shown in Figure 5, expanding the length of world knowledge yields diminishing returns on the agent's performance: **extending a short context brings substantial gains, whereas further lengthening an already extensive context provides only marginal benefits.** Specifically, when the token length increases from the initial 4k∼8k to 8k∼16k, the success rate on game websites jumps significantly from 33.44 to 38.63. Conversely, as the context continues to grow, the performance gains plateau; for example, transitioning from 16k∼32k to 32k∼64k yields only a slight improvement on game websites (from 42.21 to 43.51).

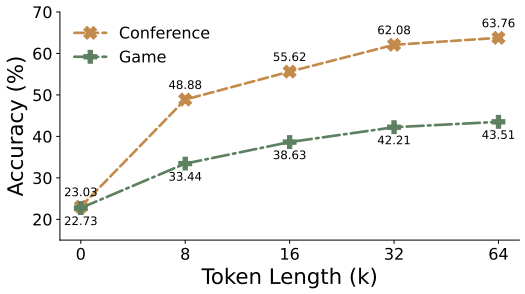

Figure 5: Accuracy under different token lengths.

We attribute this non-linear trend to the challenge of navigating complex websites with massive sub-pages: overly short knowledge causes information loss, medium-length knowledge encapsulates the critical information for strong performance, while excessively long contexts introduce redundant noise that distracts the agent, making further lengthening marginal or even slightly detrimental.

5 Conclusion

In this paper, we propose a novel paradigm that equips LLM agents with intrinsic meta-evolution capabilities. Through a two-stage training framework, agents learn to spontaneously explore environments and distill structured world knowledge without human guidance or inference-time rewards. Evaluations show this native evolution yields a 20% absolute performance improvement for Qwen3-30B and Seed-OSS-36B. Furthermore, the generated knowledge is highly transferable—most strikingly, it enables a compact Qwen3-14B model to outperform the unassisted Gemini-2.5-Flash. Ultimately, this work realizes autonomous self-evolution, paving the way toward Artificial General Intelligence.

AI USE STATEMENT

During the preparation of this work, we make use of LLMs in several stages of writing and experimentation:

- **Language Refinement:** LLMs (such as OpenAI's ChatGPT and Google's Gemini) are employed to improve the readability, grammar, and academic style of the manuscript, helping to enhance clarity and consistency throughout the paper.
- **Evaluation Assistance:** We employ advanced LLMs as automated judges to assist with the evaluation process. These models support the assessment of the generated world knowledge and agent trajectories, complementing our programmatic evaluation pipeline on web navigation benchmarks.
- **Data Generation:** LLMs are utilized to synthesize the training data required for our two-stage training framework. Specifically, they assist in constructing world knowledge generation trajectories to train the agent's meta-evolution capabilities.
- **Programming Support:** The Claude Code environment, which incorporates AI assistance, is used for generating, modifying, and refactoring segments of the codebase. This contributes to faster implementation of the training pipeline and better overall code quality.

REFERENCES

Reza Yazdani Aminabadi, Samyam Rajbhandari, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Olatunji Ruwase, Shaden Smith, Minjia Zhang, Jeff Rasley, et al. Deepspeed-inference: enabling efficient inference of transformer models at unprecedented scale. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–15. IEEE, 2022.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.

Ali Behrouz, Zeman Li, Praneeth Kacham, Majid Daliri, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. Atlas: Learning to optimally memorize the context at test time. *arXiv preprint arXiv:2505.23735*, 2025a.

Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. Nested learning: The illusion of deep learning architectures. *arXiv preprint arXiv:2512.24695*, 2025b.

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

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Gaodan Fang, Vatche Isahagian, KR Jayaram, Ritesh Kumar, Vinod Muthusamy, Punleuk Oum, and Gegi Thomas. Trajectory-informed memory generation for self-improving agent systems. *arXiv preprint arXiv:2603.10600*, 2026.

Tianqing Fang, Hongming Zhang, Zhisong Zhang, Kaixin Ma, Wenhao Yu, Haitao Mi, and Dong Yu. Webevolver: Enhancing web agent self-improvement with coevolving world model. *arXiv preprint arXiv:2504.21024*, 2025a.

Tianqing Fang, Zhisong Zhang, Xiaoyang Wang, Rui Wang, Can Qin, Yuxuan Wan, Jun-Yu Ma, Ce Zhang, Jiaqi Chen, Xiyun Li, et al. Cognitive kernel-pro: A framework for deep research agents and agent foundation models training. *arXiv preprint arXiv:2508.00414*, 2025b.

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

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, et al. A survey of self-evolving agents: On path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 2025.

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

Jinwu Hu, Zhitian Zhang, Guohao Chen, Xutao Wen, Chao Shuai, Wei Luo, Bin Xiao, Yuanqing Li, and Mingkui Tan. Test-time learning for large language models. *arXiv preprint arXiv:2505.20633*, 2025.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-zero: Self-evolving reasoning llm from zero data. *arXiv preprint arXiv:2508.05004*, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Bo Liu, Chuanyang Jin, Seungone Kim, Weizhe Yuan, Wenting Zhao, Ilia Kulikov, Xian Li, Sainbayar Sukhbaatar, Jack Lanchantin, and Jason Weston. Spice: Self-play in corpus environments improves reasoning. *arXiv preprint arXiv:2510.24684*, 2025.

Junchen Liu, Sven Elflein, Or Litany, Zan Gojcic, and Ruilong Li. Test-time training with kv binding is secretly linear attention. *arXiv preprint arXiv:2602.21204*, 2026.

Sidi Lu, Zhenwen Liang, Dongyang Ma, Yan Wang, Haitao Mi, and Dong Yu. Locas: Your models are principled initializers of locally-supported parametric memories. *arXiv preprint arXiv:2602.05085*, 2026.

Moonshot AI. Kimi K2.6: From code to creation, from one to many. Kimi Model Page, April 2026. URL https://www.kimi.ai/ai-models/kimi-k2-6. Released April 20, 2026.

Mohammad Mahdi Moradi, Hossam Amer, Sudhir Mudur, Weiwei Zhang, Yang Liu, and Walid Ahmed. Continuous self-improvement of large language models by test-time training with verifier-driven sample selection. *arXiv preprint arXiv:2505.19475*, 2025.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*, 2025.

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

Zhe Ren, Yimeng Chen, Dandan Guo, Guowei Rong, Tonghui Li, RB Xiong, Qingfeng Lan, Wenyi Wang, Li Nanbo, Yibo Yang, et al. Self-improvements in modern agentic systems: A survey. *arXiv preprint arXiv:2607.13104*, 2026.

Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. Agentsquare: Automatic llm agent search in modular design space. *arXiv preprint arXiv:2410.06153*, 2024.

Toby Simonds, Kevin Lopez, Akira Yoshiyama, and Dominique Garmier. Self rewarding self improving. *arXiv preprint arXiv:2505.08827*, 2025.

Hongjin Su, Ruoxi Sun, Jinsung Yoon, Pengcheng Yin, Tao Yu, and Sercan Ö Arık. Learn-by-interact: A data-centric framework for self-adaptive agents in realistic environments. *arXiv preprint arXiv:2501.10893*, 2025.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pp. 9229–9248. PMLR, 2020.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.

Wangcheng Tao, Han Wu, and Weng-Fai Wong. Sepo: Self-evolving prompt agent for system prompt optimization. *arXiv preprint arXiv:2606.04465*, 2026.

ByteDance Seed Team. Seed-oss open-source models. https://github.com/ByteDance-Seed/seed-oss, 2025a.

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

Qwen Team. Qwen3 technical report, 2025b. URL https://arxiv.org/abs/2505.09388.

Renxi Wang, Xudong Han, Lei Ji, Shu Wang, Timothy Baldwin, and Haonan Li. Toolgen: Unified tool retrieval and calling via generation. *arXiv preprint arXiv:2410.03439*, 2024a.

Sai Wang, Yu Wu, and Zhongwen Xu. Cogito, ergo ludo: An agent that learns to play by reasoning and planning. *arXiv preprint arXiv:2509.25052*, 2025a.

Tevin Wang and Chenyan Xiong. Autorule: Reasoning chain-of-thought extracted rule-based rewards improve preference learning. *arXiv preprint arXiv:2506.15651*, 2025.

Yan Wang, Dongyang Ma, and Deng Cai. With greater text comes greater necessity: Inference-time training helps long text generation. *arXiv preprint arXiv:2401.11504*, 2024b.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025b.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, et al. Webwalker: Benchmarking llms in web traversal. *arXiv preprint arXiv:2501.07572*, 2025.

Jinyu Xiang, Jiayi Zhang, Zhaoyang Yu, Fengwei Teng, Jinhao Tu, Xinbing Liang, Sirui Hong, Chenglin Wu, and Yuyu Luo. Self-supervised prompt optimization. *arXiv preprint arXiv:2502.06855*, 2025.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Ziyi Yang, Weizhou Shen, Chenliang Li, Ruijun Chen, Fanqi Wan, Ming Yan, Xiaojun Quan, and Fei Huang. Spell: Self-play reinforcement learning for evolving long-context language models. In *International Conference on Learning Representations*, volume 2026, pp. 59910–59942, 2026.

Li Yin and Zhangyang Wang. Llm-autodiff: Auto-differentiate any llm workflow. *arXiv preprint arXiv:2501.16673*, 2025.

Zhenrui Yue, Kartikeya Upasani, Xianjun Yang, Suyu Ge, Shaoliang Nie, Yuning Mao, Zhe Liu, and Dong Wang. Dr. zero: Self-evolving search agents without training data. *arXiv preprint arXiv:2601.07055*, 2026.

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

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin godel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025b.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, et al. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*, 2025c.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*, 2025d.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, et al. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026a.

Yudi Zhang, Meng Fang, Zhenfang Chen, and Mykola Pechenizkiy. Self-evolving llm agents with in-distribution optimization. *arXiv preprint arXiv:2606.07367*, 2026b.

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.

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

Yifei Zhou, Sergey Levine, Jason Weston, Xian Li, and Sainbayar Sukhbaatar. Self-challenging language model agents. *arXiv preprint arXiv:2506.01716*, 2025.

A Details of Input Processing

To reduce noise in large-scale web data and focus on domain-relevant information, we perform **importance scoring** and **clustering** over webpages.

Importance Scoring. We model a website as a directed graph, where each node represents a webpage and a directed edge $A \to B$ indicates that page A links to page B. Let $d_{\text{in}}(v)$ and $d_{\text{out}}(v)$ denote the in-degree and out-degree of node v, respectively. We define the importance of a node as:

$$\text{Importance}(v) = 0.7 \cdot d_{\text{in}}(v) + 0.3 \cdot d_{\text{out}}(v).$$

Clustering. To impose structure on complex websites, we group webpages into clusters based on shared URL prefixes. Starting from the first path segment, URLs are recursively partitioned until each group satisfies a size constraint. This strategy organizes webpages into coherent categories, making the overall structure more interpretable and easier to navigate for downstream processing.

B Websites for Training and Evaluation

Training Websites. We use the following websites for training: www.input-tv.org, www.nime.org, www.vldb.org, www.poms.org, www.nmun.org, ehaweb.org, sigchi.org, www.icse-conferences.org, www.fallguys.com, game.163.com.

Evaluation Websites. For evaluation, we use websites from the four WebWalker domains and four WebVoyager sites.

- **WebWalker – Conference:** 2023.aclweb.org, 2024.aclweb.org, www2024.thewebconf.org, www.cigre.org.
- **WebWalker – Game:** ac.dragonest.com, www.mythofempires.com, cn.moonton.com, godotengine.org, www.ironhidegames.com, www.ztgame.com, www.rovio.com, www.ageofempires.com.
- **WebWalker – Organization:** ct.ctrip.com, work.weixin.qq.com, www.baicgroup.com.cn, www.h3c.com, www.newchinalife.com, www.niegoweb.com, www.sinomach.com.cn, www.sony.com.cn, www.uni-president.com.cn, www.yili.com.
- **WebWalker – Education:** cc.nankai.edu.cn, ccs.imu.edu.cn, ccst.tyut.edu.cn, ci.hfut.edu.cn, csci.ncut.edu.cn, cs.dlut.edu.cn, it.chd.edu.cn, it.xju.edu.cn, jsj.nwpu.edu.cn, scse.buaa.edu.cn.
- **WebVoyager:** www.wolframalpha.com, www.apple.com, dictionary.cambridge.org, www.coursera.org.

B.1 Case Study

We present a case study to show how world knowledge improves the agent's performance (Figure 6). The task is based on the ACL 2024 website and asks for the time interval between **the Printing Order Service registration deadline** and **the venue update announcement.** With world knowledge, the agent retrieves key information in the first step and identifies both dates in the second step, leading to the correct answer. Without it, the agent must explore from the homepage, taking more steps to find partial information and failing to locate the venue update, resulting in an incorrect answer. This shows that world knowledge provides contextual guidance, enabling more effective problem solving.

Question: In the ACL 2024 conference, how many days are there between the registration deadline for the Printing Order Service and the announcement updating the venue of the main conference site?
Website: https://acl.2024.org/

(A) Output (Qwen3-30B-A3B):

Step1: I need to first locate the registration deadline.. I will start by exploring the ACL 2024 homepage...

Step4: The Printing Order Service registration deadline was successfully found as 2024-08-09...

Step7: Historical patterns indicate that venue announcements for ACL conferences typically occur 3 to 6 months before the registration deadline...I will now compute the number of days between the earliest possible announcement date (2024-02-01) and the registration deadline (2024-08-09)... The result is 190 days.

Wrong
Total steps: 7

(B) Output (Qwen3-30B-A3B with World Knowledge):

Step1: The world knowledge contains relevant information: the printing service deadline is August 9, 2024 , and the venue page states the venue is Centara Grand and Bangkok Convention Centre, but no announcement date is provided. Since the venue update date is missing, I must search for it on related pages ...

Step2: The printing order service deadline (August 9, 2024) and the venue update announcement date (May 5, 2024) have both been identified. The time interval between these two dates is 96 days.

Correct
Total steps: 2

Figure 6: An example of multi-step deepsearch question answering comparing the agent's behavior with and without world knowledge. Correct information is highlighted in green, while incorrect information is shown in red.

C EXAMPLE CHOWCASE

Examples C.1: An Example of the Processed Input

```
Total: 5 clusters, 221 URLs | per-cluster sizes: [40, 10, 130, 21, 20]
============================================================

[Prefix] https://sigchi.org (40 URLs)
https://sigchi.org/conferences/upcoming [score:227]
https://sigchi.org/people/all-committees [score:226]
https://sigchi.org/news/announcements [score:223]
https://sigchi.org [score:222]
https://sigchi.org/people/sigchi-awards [score:222]
... (35 URLs omitted)

============================================================

[Prefix] https://sigchi.org/about/policies (10 URLs)
https://sigchi.org/about/policies [score:20]
https://sigchi.org/about/policies/conference-policies [score:17]
https://sigchi.org/about/policies/conference-policies/data [score:12]
https://sigchi.org/about/policies/conference-policies/submission-and-
    ↪ review [score:11]
https://sigchi.org/about/policies/conference-policies/conference-
    ↪ guidelines [score:11]
... (5 URLs omitted)

============================================================

[Prefix] https://sigchi.org/events (130 URLs)
https://sigchi.org/events [score:140]
https://sigchi.org/events/open-session-on-acm-open [score:11]
https://sigchi.org/events/ahs26 [score:11]
https://sigchi.org/events/september-coffee-hour [score:11]
https://sigchi.org/events/sigchi-town-hall [score:11]
... (125 URLs omitted)

============================================================
```

```
[Prefix] https://sigchi.org/people/committees (21 URLs)
https://sigchi.org/people/committees/asia-committee [score:11]
https://sigchi.org/people/committees/asia-committee/bylaws [score:11]
https://sigchi.org/people/committees/research-ethics-committee [score
    ↪ :11]
https://sigchi.org/people/committees/awards-committee [score:11]
https://sigchi.org/people/committees/development-fund-committee [score
    ↪ :11]
... (16 URLs omitted)

============================================================

[Prefix] https://sigchi.org/resources (20 URLs)
https://sigchi.org/resources/guides-for-organizers [score:221]
https://sigchi.org/resources/gary-marsden-travel-awards [score:220]
https://sigchi.org/resources/sigchi-development-fund [score:220]
https://sigchi.org/resources/guides-for-authors [score:220]
https://sigchi.org/resources/sigchi-development-fund/recipients [score
    ↪ :15]
... (15 URLs omitted)
```

Examples C.2: An Example of World Knowledge

Overview

- **Website:** https://2024.aclweb.org
- **Total Categories:** 2
- **Total Pages Analyzed:** 31

The ACL 2024 website serves as the official information hub for the 62nd Annual Meeting of the Association for Computational Linguistics. It provides comprehensive details for participants, including registration, program schedules, venue information, and various calls for participation.

Category: ACL 2024 Main Website

- **URL Prefix:** https://2024.aclweb.org
- **Category Summary:** This category covers the main pages of the ACL 2024 conference website. It includes essential information for attendees, such as registration details, program schedules, venue information, and calls for participation. These pages serve as the primary resource for anyone interested in or attending the conference.

Scraped Pages:

- **Main Conference - ACL 2024**: The 2024 Annual Meeting of the Association for Computational Linguistics (ACL) will be held in Bangkok, Thailand from August 11-16, 2024. The conference's special theme is "Open science, open data, and open models for reproducible NLP research," and it covers a wide range of topics in Natural Language Processing. Paper submissions for the main conference were due on February 15, 2024, via the ACL Rolling Review (ARR) system. URL: https://2024.aclweb.org/calls/main_conference_papers/
- **Participants - ACL 2024**: This page provides logistical information for conference participants, focusing on the venue and accommodations. The conference is held at the Centara Grand and Bangkok Convention Centre in Thailand, and the page lists several hotel options with rates. It is noted that many of the listed hotels are already fully booked, and the page does not contain information on dates, deadlines, or topics. URL: https://2024.aclweb.org/participants/
- **Frequently Asked Questions - ACL 2024**: This page provides key logistical information for ACL 2024 attendees, detailing presentation formats such as 10-minute talks for long papers and poster-only sessions for Findings papers. It also serves as a central resource with important deadlines and provides essential links for the conference schedule, visa

invitation letters, on-site poster printing, and paper submissions. The page clarifies that while Findings papers don't require a presentation, author registration is still mandatory. URL: https://2024.aclweb.org/faq/

- **Sponsors - ACL 2024**: This page details the corporate sponsors for the ACL 2024 conference, noting that the deadline for sponsorship is June 30. The sponsors are organized into five distinct tiers, from Diamond to Bronze, based on their level of contribution. Major technology companies like Apple, Google Deepmind, Meta, and Amazon are listed as top-tier Diamond and Platinum sponsors. URL: https://2024.aclweb.org/sponsors
- **Organizing Committee - ACL 2024**: The ACL 2024 conference will take place in Bangkok, Thailand, from August 11-16, 2024, with a special theme on "Open science, open data, and open models for reproducible NLP research." The provided text details important dates for the event, including a paper submission deadline of February 15, 2024, and a notification of acceptance on May 15, 2024. It also lists a wide range of submission topics, such as Machine Translation, Ethics, Dialogue Systems, and NLP Applications. URL: https://2024.aclweb.org/organization/
- **Registration - ACL 2024**: This page outlines the registration options for the ACL 2024 conference, including in-person, virtual, and tutorial/workshop-only passes. Key details include that authors must register their papers by July 12, 2024, and that discounted virtual registration is available for those with financial need. In-person attendance includes full access to sessions and meals, while virtual access is provided through the Underline platform. URL: https://2024.aclweb.org/registration
- **Visa - ACL 2024**: This page provides essential visa information for attendees of the ACL 2024 conference traveling to Thailand, directing them to the official Thai e-visa website. It specifies that applicants should select 'Participants of MICE Industry' as their purpose of visit and lists the various countries whose citizens are eligible for visa exemption or visa on arrival. The page also includes links to official resources and special instructions for certain nationalities. URL: https://2024.aclweb.org/participants/visa/
- **Printing Service - ACL 2024**: The ACL 2024 conference offers a printing service for participant posters, recommending an A0 size. Attendees can secure an early rate of THB 1,400 (~USD 40) by ordering before the August 5 deadline. The price doubles for the normal rate (until August 9) and increases substantially for onsite orders, so participants are strongly encouraged to print in advance to save money. URL: https://2024.aclweb.org/participants/printing/
- **D&I Socials and BoF Sessions - ACL 2024**: This page is a call for individuals or groups interested in organizing Birds of a Feather (BoF) sessions or affinity group meetings for the ACL 2024 conference. To express interest and provide feedback, parties are instructed to fill out a form by the deadline of July 31, 2024. For any inquiries, the ACL 2024 Diversity and Inclusion team can be contacted via email. URL: https://2024.aclweb.org/calls/bof/
- **Student Research Workshop - ACL 2024**: The ACL 2024 Student Research Workshop is a forum for students to present their work, accepting submissions for both research papers and thesis proposals. The workshop offers an optional pre-submission mentorship program and financial grants to support student attendance, with the final paper submission deadline set for June 1, 2024. URL: https://2024.aclweb.org/calls/srw/
- **Venue - ACL 2024**: The ACL 2024 conference will be held in Bangkok, Thailand, and this page provides essential information for attendees. It offers practical travel advice covering the city's tropical climate, the local currency (Thai Baht), and general safety precautions. The page also includes important resources like emergency contact numbers and suggestions for popular tourist destinations to help visitors prepare for their trip. URL: https://2024.aclweb.org/participants/venue/
- **Student Volunteer Program - ACL 2024**: The ACL 2024 conference is accepting applications for its Student Volunteer Program until the deadline of July 3, 2024. Accepted full-time students receive benefits like free conference registration and an ACL membership, but are responsible for their own travel and accommodation costs. Volunteers are selected based on enthusiasm, financial need, and whether they are presenting a paper,

and they must fulfill all assigned duties to avoid being charged the full registration fee. URL: https://2024.aclweb.org/calls/volunteers/

- **Diversity and Inclusion Subsidies - ACL 2024**: ACL 2024 is offering various subsidies, including funds for registration, caregiving, and travel, to support researchers from developing countries, marginalized communities, students, and those with financial hurdles. Applications are due by July 3rd, 2024, with preference given to first-time attendees, presenters from underrepresented groups, and those without other financial support. URL: https://2024.aclweb.org/calls/diversity_and_inclusion_subsidies/
- **System Demonstrations - ACL 2024**: This page details the call for system demonstration submissions for the ACL 2024 conference, covering a range of topics including NLP systems, software tools, and data visualization. Submissions, due by March 18, 2024, are limited to a 6-page paper and must be accompanied by a short video demonstrating the system. The review process is single-blind, and a "Best Demo Award" will be presented to an outstanding submission. URL: https://2024.aclweb.org/calls/system_demonstrations/
- **Ethical Policies - ACL 2024**: This page outlines the ethical guidelines for the ACL 2024 conference, emphasizing its adherence to the ACL Anti-Harassment Policy. It underscores the collective responsibility of the community to create an inclusive and positive atmosphere. Participants are provided with clear instructions to confidentially report any harassment or hostile behavior to the ACL Professional Conduct Committee for consultation and action. URL: https://2024.aclweb.org/participants/ethical_policies/
- **Travel - ACL 2024**: This page provides essential travel information for individuals attending the ACL 2024 conference in Bangkok. It outlines details on the city's two main airports (BKK and DMK), lists local transportation options like the recommended BTS SkyTrain, and confirms that complimentary parking is available for attendees. URL: https://2024.aclweb.org/participants/travel/

  This category may contain additional pages beyond those listed. For further exploration, visit: https://2024.aclweb.org

Category: ACL 2024 Program Information

- **URL Prefix:** https://2024.aclweb.org/program
- **Category Summary:** This section details the conference program for ACL 2024. It includes links to the overall schedule, specific tracks, workshops, tutorials, and other related events. This is the central hub for attendees to plan their schedule and find information about the various sessions.

Scraped Pages:

- **Conference Overview - ACL 2024**: The ACL 2024 conference is scheduled from August 10th to August 16th, featuring a comprehensive program that includes tutorials, workshops, and various presentation formats. The event will host keynote speeches by prominent researchers Sunita Sarawagi, Subbarao Kambhampati, and Barbara Plank, and provides access to the titles of all papers from the main conference, findings, and demo tracks. URL: https://2024.aclweb.org/program/
- **Accepted Main Conference Papers - ACL 2024**: This page provides a comprehensive list of all papers accepted to the ACL 2024 main conference, separated into 'Long Papers' and 'Short Papers' categories. Each listed entry includes the full paper title and the names of the respective authors. The page functions solely as a list and does not provide additional details such as presentation schedules, topics, or other instructions. URL: https://2024.aclweb.org/program/accepted_main_conference_papers/
- **Accepted SRW Papers - ACL 2024**: This page provides a complete list of accepted papers for the Student Research Workshop (SRW) at the ACL 2024 conference. The accepted works are divided into two categories, "Oral Papers" and "Poster Papers," with each entry listing the paper's title and the names of the authors. The page exclusively serves as a list and does not contain presentation schedules or group the papers by topic. URL: https://2024.aclweb.org/program/accepted_srw_papers/

- **Keynotes - ACL 2024**: The page introduces the three keynote speakers for the ACL 2024 conference: Sunita Sarawagi, Subbarao Kambhampati, and Barbara Plank. Their respective talks will cover pressing topics in AI, such as the tradeoffs of in-context-learning for model adaptation, whether LLMs can truly reason and plan, and the need to embrace variation in NLP to avoid narrowing the field's horizons. URL: https://2024.aclweb.org/program/keynotes/
- **Tutorials - ACL 2024**: The ACL 2024 conference will host six tutorials divided into morning (09:00 - 12:30) and afternoon (14:00 - 17:30) sessions. Topics focus heavily on large language models, covering their vulnerabilities, watermarking, and expressivity, alongside other key areas like computational linguistics for brain decoding and effective science communication. Specific tutorials also address automatic text generation and revision. URL: https://2024.aclweb.org/program/tutorials/
- **Panel - ACL 2024**: The ACL 2024 conference will host a panel titled "Challenges and Opportunities with SEA LLMs" to address the development of Large Language Models for low-resource Southeast Asian languages. Chaired by Lun-Wei Ku, the session features experts Dr. Sarana Nutanong (VISTEC), Dr. Ayu Purwarianti (ITB), and Dr. William Tjhi (AI Singapore). Key discussion points will include data requirements, application scenarios, and strategies for creating and maintaining LLMs for the region. URL: https://2024.aclweb.org/program/panel/
- **Best Paper Awards - ACL 2024**: This page lists the award-winning papers from the ACL 2024 conference, divided into several categories including Best Paper, Best Social Impact, Best Resource, and Best Theme. The three Best Paper awards were given to 'Mission: Impossible Language Models', 'Semisupervised Neural Proto-Language Reconstruction', and 'Why are Sensitive Functions Hard for Transformers?'. The page also highlights winners in other categories, such as 'DIALECTBENCH' for social impact and 'Dolma' for best resource, along with lists of Outstanding Papers and SAC Awards. URL: https://2024.aclweb.org/program/best_paper_awards/
- **Accepted CL Papers - ACL 2024**: This page provides the official list of accepted papers for the Computational Linguistics (CL) track of the ACL 2024 conference. Each entry includes the full paper title and the names of its authors, such as 'Stance Detection with Explanations' by Rudra Ranajee Saha et al. The page functions solely as a list and does not contain conference schedules, paper topics, or presentation instructions. URL: https://2024.aclweb.org/program/accepted_cl_papers/
- **Accepted Findings Papers - ACL 2024**: This page lists the accepted "Findings" research papers for the ACL 2024 conference. The content is organized into two distinct sections, 'Long Papers' and 'Short Papers', with each entry providing the paper's title and its authors. The page serves as a directory for these specific papers but does not include information on schedules or topics. URL: https://2024.aclweb.org/program/accepted_findings_papers/
- **Workshops - ACL 2024**: This page lists the workshops for the ACL 2024 conference, held on August 15 and 16, 2024. The workshops cover a wide variety of specific topics in Natural Language Processing, such as argument mining, machine translation for low-resource languages, and social media mining for health research. Each listing provides key details including the workshop's title, date, location, and organizers. URL: https://2024.aclweb.org/program/workshops/

  This category may contain additional pages beyond those listed. For further exploration, visit: https://2024.aclweb.org/program

D PROMPT SHOWCASE

Prompts D.1: World Knowledge Generation Prompt

Role
You are a Web Intelligence Agent specializing in website analysis and knowledge organization. You will receive a **pre-clustered URL file** for a website, where URLs are already grouped by path prefix. Your task is to scrape these URLs **category by category** and produce a structured **World Knowledge** that stays within a target token range.
Constraints

- **Maximum** World Knowledge length: **{token_limit}** tokens
- **Minimum** World Knowledge length: **{min_token}** tokens
- You must actively manage content length throughout the process — compress when too long, expand when too short.
- **No external links:** Do NOT include any **external links** that lead to a different domain in the Guidebook. Only document pages belonging to the website's own domain.
- **Summarize, don't copy:** Do NOT copy raw page content verbatim into the Guidebook. Always **summarize and condense** the key information in your own words. The Guidebook should be a concise guide, not a dump of webpage text.
- **Every scraped page must have a URL:** Each entry under "Scraped Pages" **MUST** include the full URL in parentheses. An entry without a URL is **INVALID**. Format: `- **[Page Title]** : [summary]`. Never write a summary without its corresponding URL.
- **Follow your token plan:** The length and detail of each category's content should be guided by the **planned token allocation** from Phase 0. Spend more tokens on categories with more URLs, fewer on small ones. Focus on specific, useful information (names, dates, numbers, features). Do not pad with generic or repetitive descriptions.

URL Priority Rules
ach URL in the cluster file has a structure-based score in the format `[score:S]`. You must evaluate the importance of each webpage by combining this structural score with its semantic relevance and content value. Use this combined assessment to determine the page's overall priority, whether it should be selected, and how much detail (token budget) to dedicate to it in your subsequent writing.
Input
A clustered URL file is located at: `{queue_file_path}`
The file format:

```
Total: <N> clusters, <M> URLs |  per-cluster sizes: [<c1>, <c2>, ...]
============================================================
[Prefix] <prefix_url>  (<total> URLs)
<url_1>  [score:<S>]
<url_2>  [score:<S>]
...
============================================================
[Prefix] <prefix_url>  (<total> URLs)
<url_3>  [score:<S>]
<url_4>  [score:<S>]
...
```

The first line contains global statistics: total number of clusters, total number of URLs and the URL count for each cluster after `per-cluster sizes:`. Categories are separated by lines of repeated = characters. Each category starts with a `[Prefix]` header showing `(shown/total URLs)`. Each URL is annotated with `[score:S]`. `score` is a composite importance score.
Tools
To assist you with this task, I have provided a complete Python code block below containing the necessary tool functions. Please make sure to copy and run this entire block as your first step. After that, you can conveniently use the functions by calling their names. It is highly recommended to

use them as-is without rewriting or re-implementing them, unless you run into a runtime error that requires a modification.

```
{tool_functions_code}
```

Workflow
Please follow the phases in their natural order: Phase 0 → Phase 1 → Phase 2. It is highly recommended to complete Phase 0 first, as Phase 1 relies on the plan file created during this initial step. Jumping straight to Phase 1 will cause `read_plan()` to fail and require unnecessary recovery steps.

PHASE 0: INITIALIZATION & PLANNING

Before processing any category, please create a token-allocation plan to guide your work.
Step 1: Parse Cluster Statistics Call `parse_cluster_stats()`. It reads the first line of the queue file and extracts: Total number of clusters, total number of URLs, and item Per-cluster URL counts.
Step 2: Create Token Allocation Plan

- **Allocate tokens proportionally by effective URL count** — more URLs → more tokens. Try to avoid splitting them equally. Rough guide: ∼50–80 tokens per page entry + ∼100–150 for category header/summary.
- **Please ensure the total planned tokens fall within [{min_token}, {token_limit}].** Adjust your allocations as needed to stay in this range.
- Call `write_plan(plan_text)` to save to `{plan_file_path}`.

Step 3: Proceed to Phase 1.
—

PHASE 1: CATEGORY-BY-CATEGORY PROCESSING (LOOP)

Process the clustered URL file one category at a time:
Step 1: Load Next Category

- Call `get_next_category()` — the one already defined above. This function returns the current unprocessed category block. It is safe to call multiple times — it always returns the same category until you explicitly call `mark_category_done()`.
- If it returns `None`, all categories have been processed — proceed to **Phase 2**.

Step 1.5: Read Token Budget (MANDATORY — do NOT skip)

- **This step is NOT optional. You MUST execute it for every category before writing anything.**
- Call `read_plan()` to open and read the plan file (`{plan_file_path}`).
- If `read_plan()` returns an empty string, **STOP and go back to Phase 0 Step 2 to create the plan first.** Never proceed without a plan.
- Find the current category's prefix URL in the plan and extract its **planned token allocation** (e.g., `budget = 1200`).
- **Print it explicitly:** `print(f"Token budget for this category: {budget}")` — this forces you to be aware of the target.
- You will use this number in Step 3 to control the length of your output. For example: if the budget is 500 tokens, write a brief summary with short page entries; if 2000 tokens, write detailed summaries with rich page entries.

Step 2: Select & Scrape Member Pages

- Please selectively choose which URLs to scrape based on their relative importance and your available token limits.
- To help prioritize the most valuable pages, you can refer to the `[score:S]` metric provided for each URL, selecting those with higher scores first.
- For each selected URL, fetch the webpage and extract the key information.
- To make the best use of your tokens, please skip pages that do not provide meaningful content (such as login walls, error pages, empty pages, or cookie/privacy notices).

Step 3: Write Category Section (target the token budget from Step 1.5)

- **Keep your token budget in mind:** Aim to keep the entire section (header, summary, and scraped entries) close to the budget you extracted in Step 1.5. A variance of around $\pm 20\%$ is perfectly fine.
- Use `append_to_guidebook(text)` to add this category's section to the guidebook.
- **Category Summary:** Briefly describe the main topics and types of pages found here, adjusting the level of detail to fit your budget.
- **Formatting:** Please follow the template below closely. It is important to keep the exact header structure (`## Category: [Name]`).

```
## Category: [Descriptive Name Based on Content]
- **URL Prefix:** [the prefix URL for this group]
- **Category Summary:** [Describe the main topics.]

**Scraped Pages:**
- **[Page Title]** ([full URL]): [Specific details like names,
    ↪ dates,
  or features. Adjust length to fit budget.]
- **[Page Title]** ([full URL]): [summary]
- ...

> This category may contain additional pages beyond those listed.
    ↪
  For further exploration, visit: [prefix URL]
```

- **Include URLs:** Please ensure every entry under "Scraped Pages" includes its full URL in parentheses.

Step 4: Mark Done & Continue

- After successfully appending this category to the guidebook, call `mark_category_done()` to advance the progress index. **Only call this after `append_to_guidebook()` succeeds** — this ensures no category is skipped even if an earlier step fails or retries.
- Return to **Step 1** to process the next category.

—

PHASE 2: REFINEMENT & FINALIZATION

After all categories have been processed:

Step 1: Token-Based Compression or Expansion

- Call `count_guidebook_tokens()`.
- **If tokens $>$ {token_limit}:** Compress the guidebook:
  - Call `read_guidebook()` to review all content.
  - Identify verbose or repetitive sections and rewrite them with `rewrite_category_section()`.
  - Repeat until within limit.
- **If tokens $<$ {min_token}:** Expand the guidebook. Follow these steps **exactly** to avoid content loss:
  1. **Identify areas for expansion:** Review your plan and the current guidebook. Feel free to choose any category that seems underdeveloped or has interesting URLs you haven't explored yet.
  2. **Review existing content:** Use `read_guidebook()` to see what has already been written for your chosen category.
  3. **Explore and gather new data:** Scrape additional URLs within that category to discover fresh details. Please rely on the actual webpage content to inspire your expansion and ensure accuracy.

4. **Integrate and enrich:** Seamlessly weave your new discoveries into the existing text. You can expand summaries, add new page entries, or provide deeper insights to make the section richer and more comprehensive.
5. **Update the guidebook:** Use the `rewrite_category_section(category_name, new_section_text)` function to replace the old section with your newly expanded version.
6. **Check progress:** Use `count_guidebook_tokens()` to see how close you are to your goal. Continue this exploration process until your guidebook reaches at least `{min_token}` tokens.

Step 2: Add Overview Header & Save

- Call `read_guidebook()` to get the full current content.
- Prepend an Overview section at the top:

```
# [Website Domain] Guidebook

## Overview
- **Website:** [base URL]
- **Total Categories:** [number]
- **Total Pages Analyzed:** [number]
[2-3 sentence high-level overview of the website's purpose and
    ↪ content.]

---
```

- Call `save_final_guidebook(full_content)` with the complete content (Overview + all category sections).

Prompts D.2: World Knowledge Generation Prompt (For trained agents)

You are a Web Intelligence Agent that analyzes websites and organizes their content into a structured knowledge document called a **Guidebook** — a concise, categorized summary of a website's pages and their key information.

Input

Your input is a **clustered URL file** at `{queue_file_path}`. This file contains URLs from a single website, pre-grouped into categories by their URL path prefix (e.g., all `/blog/...` URLs form one category, all `/docs/...` URLs form another). Each URL is annotated with link metrics in the format `[in:X out:Y score:S]`, where `in` is how many other pages link to it (inbound links), `out` is how many links it contains (outbound links), and `score` is a composite importance score derived from both. Categories are separated by `===...===` lines, and each starts with a `[Prefix]` header.

Tools

You have access to `web_agent(task=...)`, a function that fetches and reads real web pages — use it to scrape each URL's content. In addition, the code block below provides helper functions for managing the Guidebook (appending content, tracking progress, counting tokens, etc.). Copy and execute this entire block in your first code cell:

`{tool_functions_code}`

Tool Usage

1. Call `parse_cluster_stats()` to read the file header and get the total number of URLs and categories. Based on the site size, decide your processing mode: for small sites ($\leq$ 250 URLs), use **FULL mode** where every URL is included; for larger sites, use **SELECTIVE mode** where you pick the most important URLs per category (ranked by `score`, up to 20 per category if $\leq$ 8 categories, or 10 if more).
2. Create a token allocation plan — distribute the target Guidebook length (`{min_token}`–`{token_limit}` tokens) across categories proportionally by each category's **effective URL count** (i.e., the number of URLs you will actually scrape,

after applying the per-category cap from step 1 — not the raw total), then save it with `write_plan()`.

3. Process categories one by one: call `get_next_category()` to load a category, scrape its selected URLs with `web_agent()`, write the category section with `append_to_guidebook()`, then call `mark_category_done()` to advance. Repeat until all categories are done.
4. After all categories are processed, check the total length with `count_guidebook_tokens()`. If it exceeds `{token_limit}`, compress verbose sections with `rewrite_category_section()`. If it falls below `{min_token}`, expand by scraping additional URLs. Finally, prepend an Overview header and call `save_final_guidebook()`.

Output format per category:

```
## Category: [Name]
- **URL Prefix:** [prefix URL]
- **Category Summary:** [what this category covers]

**Scraped Pages:**
- **[Page Title]** : [summary of key info]
- ...

> This category may contain additional pages.
Visit: [prefix URL]
```

Rules:

- Scraping: You MUST call `web_agent(task="...")` to fetch real content for every selected URL. Summarize the key information. Crucially, every page summary must come from a real `web_agent()` call. Never fabricate or guess content. Do NOT use placeholders to stand in for URL summaries, and NEVER rely on your internal knowledge to hallucinate or invent page content.
- Every scraped page entry must include its full URL.
- Only include pages from the website's own domain — no external links.
- Summarize in your own words; do not copy page content verbatim.
- Process ALL categories before finalizing.

Prompts D.3: Evaluation Prompt for LLM Judge (WebWalker)

Your task is to determine whether the answer is consistent with the ground truth for the given question.
Evaluation rules:

1. Output 1 if the answer correctly answers the question and has the same meaning as the ground truth.
2. The answer does NOT need to exactly match the ground truth.
3. Differences in wording, format, order, or level of detail are acceptable as long as the meaning is equivalent.
4. Concise answers should NOT be judged as incorrect simply because they are shorter than the ground truth.
5. Different formats that express the same information (e.g., numbers only, different date formats, paraphrases) should be considered correct.
6. Output 0 only if the answer is incorrect, contradicts the ground truth, or fails to answer the question.

Examples:
Example 1
Question: What are the 2024 suggested retail prices of the Yamaha PAC612 electric guitar and

the Sonogenic SHS-300 shoulder keyboard?
Ground truth: PAC612 electric guitar suggested retail price: 8,400 RMB. SHS-300 shoulder keyboard suggested retail price: 1,299 RMB (white) and 1,399 RMB (blue).
Answer: 8400,1299,1399
Judgment: 1

Example 2
Question: What is Jack's birthday?
Ground truth: December 10
Answer: 12-10
Judgment: 1

Example 3 (Prompt Rules Evaluation)
Question: What are the conditions for outputting 1 according to the evaluation rules?
Ground truth: Output 1 if the answer correctly answers the question and has the same meaning as the ground truth. Differences in wording, format, order, or level of detail are acceptable. Concise answers or different formats expressing the same information are also correct.
Answer: Give it a 1 if the core meaning matches, even if the answer is shorter, formatted differently, or paraphrased.
Judgment: 1

Now evaluate the following case.
Question: `{question}`
Answer: `{predict}`
Ground truth: `{gt}`
Output only one number:
1 if the answer is correct or semantically equivalent to the ground truth, otherwise 0.
Do not output anything other than the number.

Prompts D.4: Evaluation Prompt for LLM Judge (WebVoyager)

Your task is to determine whether the web task has been successfully accomplished, based on the task instruction, the result response, and the accessibility trees of the webpages.
You are given three components:

1. Web Task Instruction: A natural language instruction describing the task to be completed (e.g., search, verify, compare, summarize).
2. Result Response: The final textual response generated after performing the task.
3. Accessibility Trees: Structured representations of the webpages at each step, serving as evidence of the actions taken.

Evaluation rules:

1. You do NOT need to interact with websites or perform any real actions.
2. You must base your judgment only on the provided instruction, response, and accessibility trees. Do NOT assume missing information.
3. Your primary goal is to evaluate whether the actions reflected in the trees and the final response correctly follow the instruction.
4. If the task contains multiple requirements (e.g., find information and summarize it), all must be completed. Missing any part leads to NOT SUCCESS.
5. The accessibility trees serve as ground truth evidence of what actually happened during execution.
6. If the Result Response contradicts the trees, trust the trees.
7. If the Result Response contains information not present in the trees, trust the response.

Instructions: You should briefly explain your reasoning before giving the final verdict.

Now evaluate the following case.
TASK: `{task}`
Result Response: `{answer}`
Accessibility Trees: `{trees}`

Output your final verdict as one of the following:
`SUCCESS`
`NOT SUCCESS`

Prompts D.5: Query Generation Prompt

You are an advanced web information gathering expert and navigation planner. I will provide you with a [Main Page URL], a [World Knowledge] containing summaries of sub-pages, and a [Target Question] that needs to be resolved. **Your task is to evaluate this information, determine which specific web pages should be explored next, and ultimately obtain the final answer to the question by visiting these pages.**

[Input Data]

1. Main Page URL: `{URL}`
2. Target Question: `{Question}`
3. Sub-page Summaries: `{World Knowledge}`

[Your Decision Logic]
Please carefully read the "content summary" of each sub-page in the World Knowledge and analyze its relevance to the "Target Question":

1. **Answer Directly (Rare)**: If the "content summary" in the World Knowledge already contains the specific factual data needed to fully answer the question, please provide the answer directly.
2. **Explore Sub-pages (Most Common)**: If the topic of one or more sub-pages in the World Knowledge is highly relevant to the question (e.g., the question is about finding executives, and a sub-page summary is "About Us - Team Introduction"), please extract the URLs of these sub-pages and explore them to find the answer. **If you find a potential answer on these sub-pages, carefully verify its accuracy and relevance. If you are not highly confident it is the correct answer, or if you still cannot find the answer after visiting all selected sub-pages**, do NOT give up — return to the [Main Page URL] and explore it from scratch to look for additional clues or links not covered by the World Knowledge.
3. **Explore Main Page from Scratch (Fallback)**: If all the sub-page summaries provided in the World Knowledge are completely irrelevant to the question (e.g., they are all privacy policies, disclaimers, etc.), it indicates that the current branch is invalid. In this case, you must decide to return to the [Main Page URL] to start looking for new clues from scratch.