SIBYL: AN EFFICIENT SMALL–LARGE MODEL COLLABORATION FRAMEWORK FOR LONG-HORIZON TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Small language models (SLMs) offer a promising foundation for on-device agents through low-latency, resource-efficient inference, yet limited reasoning and planning capabilities constrain their performance on long-horizon tasks requiring multi-step interaction with the environment. Step-level collaboration between SLMs and larger cloud-hosted models can bridge this gap, but identifying states that warrant cloud assistance remains challenging: the contribution of each cloud call is entangled with subsequent actions and can be assessed only from the final task outcome. Compounding this challenge, the SLM must balance two competing objectives: maximizing task success and minimizing cloud calls. To address this, we propose Sibyl, an algorithm that trains SLM agents to selectively consult cloud models at the step level and internalize their guidance for subsequent decisions, achieving strong task performance with minimal cloud reliance. Sibyl follows a three-stage training pipeline that (1) builds a robust base policy through consultation-free self-evolving reinforcement learning (RL); (2) cold-starts consultation behavior via decisive-disagreement state mining; and (3) jointly optimizes consultation decisions and guidance internalization through consultation-aware RL. Experiments on ALFWorld and WebShop demonstrate that Sibyl, using only a 0.6B-parameter model, outperforms state-of-the-art baselines, including agent training and routing methods, by 95.2% and 80.4% in success rate while averaging only 0.8 and 3.9 cloud calls per trajectory, respectively.

1 INTRODUCTION

Language model agents have recently shown strong promise on long-horizon tasks such as embodied interaction (Seo et al., 2026; Kwon et al., 2026), web navigation (Yang et al., 2025b; Liu et al., 2026), and software engineering (Antoniades et al., 2025; Zhang et al., 2025), which require sustained iteration over reasoning, tool use, observation, and revision across many interdependent steps (Shinn et al., 2023; Xu et al., 2026). However, much of this progress has been driven by cloud-hosted large language models (LLMs) (e.g., GPT-5.6 (OpenAI, 2026), Gemini 3.1 Pro (Gemini Team, 2026), Qwen3.7-Max (Qwen Team, 2026), and Kimi K3 (Kimi Team, 2026)), while small language models (SLMs) suitable for edge deployment remain markedly less capable, particularly in the 3B-and-below regime (Liu et al., 2024; Mai et al., 2026). Their limited reasoning and planning degrade decision quality over sustained interaction with the environment, undermining reliability on long-horizon tasks and posing a critical barrier to autonomous-agent deployment in latency- and resource-constrained applications that require efficient on-device execution (Wan et al., 2025; Wen et al., 2024).

A natural solution is edge–cloud collaboration: on-device SLMs handle most steps locally and selectively consult stronger cloud-hosted LLMs (Ong et al., 2025; Liao et al., 2026), combining efficient local execution with advanced reasoning and planning. However, collaboration on long-horizon agentic tasks is fundamentally more challenging than conventional SLM–LLM routing. First, model selection is no longer a one-time decision for each task, but a sequence of decisions throughout an evolving trajectory (Zhang et al., 2026). Consequently, the utility of step-level consultation is entangled with subsequent actions and often discernible only from the final outcome, complicating the precise identification of beneficial consultation states required to maximize task success with minimal cloud calls (Feng et al., 2025). Second, existing SLM–LLM collaboration methods typically rely on an external router (Ong et al., 2025; Chen et al., 2024), which is ill-suited to long-horizon agents: routing decisions are tightly coupled with the agent's acting policy, making it difficult to train

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.

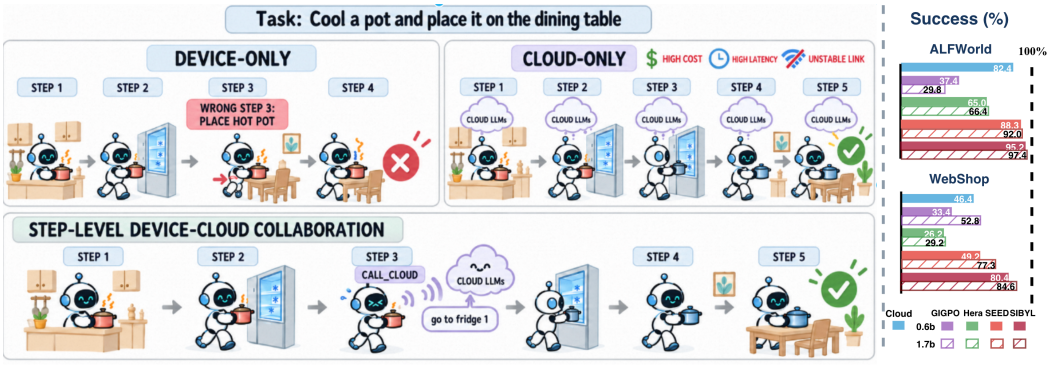


Figure 1: Sibyl enables step-level device–cloud collaboration on long-horizon tasks, combining high success with low latency and cost. Right: success-rate comparisons with representative baselines; see § 5 for further results.

a separate router that generalizes across diverse, evolving states and remains aligned as the policy is updated. Moreover, a separate router adds supervision overhead and system complexity, complicating end-to-end agent training.

We address these challenges with Sibyl, a policy optimization method that trains SLM agents to selectively consult cloud-hosted LLMs and internalize their guidance; Figure 1 illustrates the step-level collaboration paradigmd. Without a separate router, Sibyl treats consultation as an SLM action and optimizes it alongside local actions from trajectory-level outcomes, while progressively reducing cloud reliance through guidance internalization. Sibyl training proceeds in three stages: (1) consultation-free self-evolving reinforcement learning (RL) distills hindsight skills from on-policy trajectories to strengthen SLM agent before introducing cloud assistance; (2) decisive-disagreement state mining identifies states where a one-step cloud intervention turns a failed SLM trajectory into a successful one, providing targeted supervision to cold-start consultation behavior; (3) consultation-aware RL jointly optimizes local actions and cloud calls using trajectory-level outcomes while internalizing LLM guidance through hindsight skill distillation, thereby improving task success and reducing cloud reliance.

Experiments on challenging ALFWorld and WebShop benchmarks demonstrate that Sibyl establishes a new task-success–execution-cost Pareto frontier relative to both state-of-the-art agent training and routing methods. Notably, with only a 0.6B-parameter SLM, Sibyl achieves success rates of 95.2% and 80.4% while averaging only 0.8 and 3.9 cloud calls per trajectory, respectively.

Our contributions are threefold:

- We formulate long-horizon SLM–LLM collaboration as SLM policy optimization over local actions and step-level cloud consultation, eliminating the need for an external router.
- We propose Sibyl, a three-stage training algorithm that enables SLM agents to learn when to ask for cloud assistance and internalize the resulting guidance, thereby minimizing cloud calls while preserving task success.
- Across challenging agent benchmarks, Sibyl advances the success–cost Pareto frontier over state-of-the-art agent training and routing methods.

2 Related Work

Reinforcement learning for language-model agents. Language-model agents show substantial promise for autonomous decision-making by coupling reasoning with sustained environment interaction (Yao et al., 2023; Shinn et al., 2023). RL has become a central paradigm for training agents through direct policy improvement from interaction outcomes. Building on the widely adopted clipped actor–critic objective (Schulman et al., 2017), GRPO (Shao et al., 2024) replaces the learned critic with group-relative advantage estimation. GiGPO (Feng et al., 2025) further introduces hierarchical trajectory- and step-level grouping for long-horizon credit assignment. RAGEN (Wang et al.,

2025) studies stability in self-evolving agentic RL, while SEED (Wu et al., 2026) uses on-policy distillation (OPD) to distill hindsight skills from on-policy trajectories into the acting policy. However, these methods optimize individual agents rather than multi-agent collaboration, making them less effective for capacity-limited SLMs, particularly those with < 1B parameters.

SLM–LLM collaboration. Leveraging assistance from stronger LLMs has emerged as a prominent approach to mitigating SLM capacity limitations while balancing execution efficiency and cost. FrugalGPT (Chen et al., 2024) and Hybrid LLM (Ding et al., 2024) route requests based on predicted quality–cost trade-offs, while Eagle (Zhao et al., 2024) and RouteLLM (Ong et al., 2025) use historical similarity or preference-based predictions. MinionS instead assigns subtask decomposition and verification to the cloud model (Narayan et al., 2025). However, these methods target independent requests rather than evolving long-horizon trajectories. Most related to our work, Hera (Zhang et al., 2026) extends collaboration to the step level but relies on a separately trained router over fixed device and cloud policies, leaving SLM capability unchanged. In contrast, Sibyl integrates consultation and guidance internalization into the SLM policy, eliminating the external router while improving the system's capability ceiling and cost efficiency.

3 Problem Formulation

Given a long-horizon task $x \sim p(\mathcal{X})$, an on-device SLM agent with policy $\pi_{\theta^{\mathrm{S}}}$, parameterized by θ^{S}, interacts with the environment over T steps. At step t, the agent observes a state $s_t \in \mathcal{S}$ and selects an action $a_t \in \mathcal{A}(s_t)$ according to $a_t \sim \pi_{\theta^{\mathrm{S}}}(\cdot \mid s_t, x)$, where $\mathcal{A}(s_t)$ denotes the admissible action set at state s_t. After executing a_t, the environment transitions to the next state s_{t+1}. A completed episode induces a trajectory:

$$\tau = \{(s_t, a_t)\}_{t=1}^{T}, \tag{1}$$

with a trajectory-level return $R(\tau) \in \mathbb{R}$. In many agentic environments, $R(\tau)$ is sparse and observed only upon trajectory termination. The training objective is to optimize policy parameters θ^{S} to maximize expected return:

$$\max_{\theta^{\mathrm{S}}} \ \mathbb{E}_{x\sim p(\mathcal{X}),\, \tau\sim\pi_{\theta^{\mathrm{S}}}(\cdot|x)}[R(\tau)]. \tag{2}$$

To compensate for the SLM's limited capabilities, we augment its action space to enable one-step consultation with a stronger cloud-hosted LLM:

$$\widetilde{\mathcal{A}}(s_t) = \mathcal{A}(s_t) \cup \{\texttt{call_cloud}\}. \tag{3}$$

If the agent selects an action in $\mathcal{A}(s_t)$, it acts locally as usual; if it selects `call_cloud`, a stronger cloud-based policy $\pi_{\theta^{\mathrm{L}}}$ receives the current state s_t and the task instruction x, and returns one executable action $\tilde{a}_t \in \mathcal{A}(s_t)$ according to $\tilde{a}_t \sim \pi_{\theta^{\mathrm{L}}}(\cdot \mid s_t, x)$ for step t. The environment executes $\tilde{a}_t$, after which the local policy $\pi_{\theta^{\mathrm{S}}}$ resumes action selection.

Under this formulation, $\pi_{\theta^{\mathrm{S}}}$ jointly learns which local actions to take and when to consult the cloud $\pi_{\theta^{\mathrm{L}}}$, balancing expected task return against cloud-invocation cost. Let $C(\tau)$ denote the cumulative cost of $\pi_{\theta^{\mathrm{L}}}$ along trajectory τ, such as API charges. We optimize the following cost-regularized policy objective:

$$\max_{\theta^{\mathrm{S}}} \ \mathbb{E}_{x\sim p(\mathcal{X}),\, \tau\sim\pi^{\mathrm{coll}}_{\theta^{\mathrm{S}}}(\cdot|x)} \left[R(\tau) - \lambda C(\tau)\right]. \tag{4}$$

where $\lambda \geq 0$ controls the trade-off between task return and cloud cost, and $\pi^{\mathrm{coll}}_{\theta^{\mathrm{S}}}$ denotes the SLM policy augmented with `call_cloud` action, and $\pi_{\theta^{\mathrm{L}}}$ remains fixed.

4 Method

4.1 Overview

Building on § 3, we present Sibyl, a method that trains SLM agents to selectively consult stronger cloud models and internalize their guidance to maximize task success while minimizing cloud calls. Sibyl follows a three-stage training pipeline, as illustrated in Figure 2:

- **Consultation-Free Self-Evolving RL (§ 4.2).** The SLM is first trained through RL with skill-conditioned on-policy distillation, providing a reliable basis for consultation.

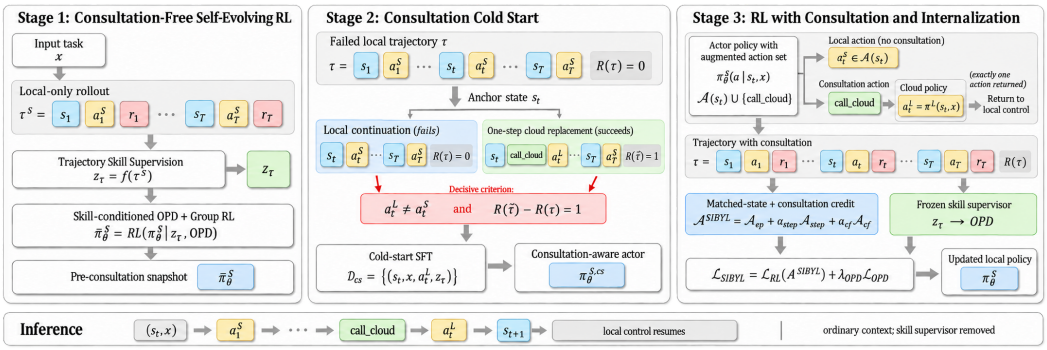


Figure 2: Three-stage training pipeline of Sibyl, which trains an SLM to act locally, consult the cloud at critical steps, and internalize its guidance to achieve high task success with few cloud calls.

- **Consultation Cold Start (§ 4.3).** Matched local and one-step cloud-assisted branches identify decisive-disagreement states to cold-start selective consultation.
- **RL with Consultation and Internalization (§ 4.4).** On-policy RL uses step-level credit to jointly optimize local actions and consultation decisions while internalizing cloud guidance to strengthen local capability and reduce repeated calls.

4.2 Consultation-Free Self-Evolving RL

Learning effective consultation requires a sufficiently capable base policy; otherwise, the SLM may improve task success by overusing cloud assistance rather than identifying states where consultation is truly necessary. We therefore first train $\pi_{\theta^{\mathrm{S}}}$ as a robust standalone agent without consultation. Inspired by self-evolving agent training (Wu et al., 2026), we distill reusable hindsight guidance from the agent's own trajectories, providing dense supervision beyond sparse outcome rewards for long-horizon behaviors such as subgoal planning and failure avoidance.

Trajectory-level skill supervision. We first execute training task x using the SLM policy $\pi_{\theta^{\mathrm{S}}}$ alone, yielding a trajectory τ^{S}. Given x, τ^{S}, and its return $R(\tau^{\mathrm{S}})$, the cloud LLM $\pi_{\theta^{\mathrm{L}}}$ serves as a trajectory teacher $\mathcal{T}_{\theta^{\mathrm{L}}}$ to extract a concise causal summary and reusable hindsight skill:

$$z_{\tau^{\mathrm{S}}} = \mathcal{T}_{\theta^{\mathrm{L}}}(x, \tau^{\mathrm{S}}, R(\tau^{\mathrm{S}})). \tag{5}$$

Beyond the sparse return, $z_{\tau^{\mathrm{S}}}$ captures trajectory-level behavioral structure, such as effective subgoal ordering, decisive observations, and failure-avoidance rules. We then fine-tune $\pi_{\theta^{\mathrm{S}}}$ on the resulting trajectory–skill pairs, using the trained model both as the trajectory analyzer $\mathcal{T}_{\theta^{\mathrm{S}}}$ for subsequent on-policy distillation and as the initialization of $\pi_{\theta^{\mathrm{S}}}$ itself. This bootstrapping eliminates repeated cloud queries for trajectory annotation during online training.

Skill-conditioned on-policy distillation. Since the trajectory–skill fine-tuning above trains $\pi_{\theta^{\mathrm{S}}}$ for trajectory analysis rather than action generation, we first restore its acting capability through a supervised cold start. We collect action trajectories from the stronger $\pi_{\theta^{\mathrm{L}}}$ and fine-tune $\pi_{\theta^{\mathrm{S}}}$ on standard state–action pairs $(s_t, x) \to a_t$, providing both protocol-consistent action generation and an initial capability transfer before on-policy optimization.

Next, we perform self-evolving RL with skill-conditioned on-policy distillation. Specifically, for each task x_i, cold-started $\pi_{\theta^{\mathrm{S}}}$ generates a trajectory $\tau_i = \{(s_{i,t}, a_{i,t})\}_{t=1}^{T_i}$, which is then analyzed by the trajectory analyzer $\mathcal{T}_{\theta^{\mathrm{S}}}$ to produce a trajectory-level hindsight skill $z_i = \mathcal{T}_{\theta^{\mathrm{S}}}(x_i, \tau_i, R(\tau_i))$. Each action $a_{i,t}$ is a token sequence $a_{i,t} = (y_{i,t,1}, \ldots, y_{i,t,L_{i,t}})$ of length $L_{i,t}$. The sampled action tokens are kept fixed and re-scored by $\pi_{\theta^{\mathrm{S}}}$ under the ordinary context $(s_{i,t}, x_i)$ and its skill-augmented counterpart $H(s_{i,t}, x_i, z_i)$, yielding the respective log-probabilities:

$$\ell^{\varnothing}_{i,t,k} = \log \pi_{\theta^{\mathrm{S}}}(y_{i,t,k} \mid s_{i,t}, x_i, y_{i,t,<k}), \tag{6}$$

$$\ell^{z}_{i,t,k} = \log \pi_{\theta^{\mathrm{S}}}(y_{i,t,k} \mid H(s_{i,t}, x_i, z_i), y_{i,t,<k}), \tag{7}$$

The skill-induced probability lift forms a detached confidence weight:

$$w_{i,t,k} = \sigma\Big(\beta_{\text{opd}}\,\text{sg}[\ell^{z}_{i,t,k} - \ell^{\varnothing}_{i,t,k}]\Big), \tag{8}$$

and the OPD objective is:

$$\mathcal{L}_{\text{opd}}(\theta^{\text{S}}) = \mathbb{E}_{i,t,k}\left[w_{i,t,k}\left(\text{sg}[\ell^{z}_{i,t,k}] - \ell^{\varnothing}_{i,t,k}\right)\right], \tag{9}$$

where β_{opd} controls the confidence weighting sharpness, and sg denotes stop-gradient. Gradients flow only through the ordinary branch, enabling $\pi_{\theta^{\text{S}}}$ to learn from online skill-supported behaviors.

The final consultation-free policy is jointly optimized with the OPD loss above and a group-based RL objective. For N trajectories sampled from the same task x_i, we define the group-relative advantage as $A_{i,n} = R(\tau_{i,n}) - \frac{1}{N}\sum_{n'=1}^{N} R(\tau_{i,n'})$. Let $\ell^{\text{old}}_{i,n,t,k}$ denote the token log-probability under the rollout policy and $\rho_{i,n,t,k}(\theta^{\text{S}}) = \exp(\ell^{\varnothing}_{i,n,t,k} - \ell^{\text{old}}_{i,n,t,k})$ the corresponding importance ratio. Following GRPO (Shao et al., 2024), the RL loss is:

$$\mathcal{L}_{\text{rl}}(\theta^{\text{S}}) = -\mathbb{E}_{i,n,t,k}\left[\min(\rho_{i,n,t,k}(\theta)A_{i,n}, \text{clip}(\rho_{i,n,t,k}(\theta), 1-\epsilon, 1+\epsilon)A_{i,n})\right] + \beta_{\text{KL}}D_{\text{KL}}, \tag{10}$$

where ϵ and β_{KL} control policy clipping and KL regularization, respectively.

The full objective of the consultation-free stage in Sibyl is:

$$\mathcal{L}_{\text{confree}} = \mathcal{L}_{\text{rl}} + \lambda_{\text{opd}}\mathcal{L}_{\text{opd}}. \tag{11}$$

Here, $\mathcal{L}_{\text{rl}}$ improves task-level performance from environment feedback, while $\mathcal{L}_{\text{opd}}$ supplies dense hindsight supervision on the same on-policy trajectories, with λ_{opd} controlling the OPD signal strength. After the complete self-evolving RL in this section, we freeze the resulting checkpoint as $\pi_{\bar{\theta}^{\text{S}}}$, which serves as the pre-consultation snapshot for subsequent training with active consultation and consultation-feedback internalization.

4.3 Consultation Cold Start

This section introduces `call_cloud` as a new action for $\pi_{\theta^{\text{S}}}$ (Eq. 3) and provides an explicit cold start for its use.

Decisive-Disagreement State Mining. The pre-consultation policy $\pi_{\bar{\theta}^{\text{S}}}$ is first rolled out on each training task x_i to collect the failed-trajectory set:

$$\mathcal{D}^{-} = \{\tau_i \mid R(\tau_i) = 0,\ \tau_i \sim \pi_{\bar{\theta}^{\text{S}}}(\cdot \mid x_i)\}, \qquad \tau_i = \{(s_{i,t}, a^{\text{S}}_{i,t})\}_{t=1}^{T_i}. \tag{12}$$

For each $s_{i,t}$ visited in $\mathcal{D}^{-}$, the environment instance, action history, current observation, and admissible action set are recorded and fed to the cloud policy $\pi_{\theta^{\text{L}}}$ to perform a one-step action:

$$a^{\text{L}}_{i,t} \sim \pi_{\theta^{\text{L}}}(\cdot \mid s_{i,t}, x_i). \tag{13}$$

If $a^{\text{L}}_{i,t} = a^{\text{S}}_{i,t}$, the cloud and local policies agree on this step; otherwise, a disagreement branch is formed by replacing $a^{\text{S}}_{i,t}$ with $a^{\text{L}}_{i,t}$ at state $s_{i,t}$, after which the subsequent steps are executed by $\pi_{\bar{\theta}^{\text{S}}}$:

$$\widetilde{\tau}_i = \{(s_{i,1}, a^{\text{S}}_{i,1}), \ldots, (s_{i,t-1}, a^{\text{S}}_{i,t-1}), (s_{i,t}, a^{\text{L}}_{i,t}), (s_{i,t+1}, \widetilde{a}_{i,t+1}), \ldots, (s_{i,\widetilde{T}_{i,t}}, \widetilde{a}_{i,\widetilde{T}_{i,t}})\}, \tag{14}$$

where $\widetilde{a}_{i,u} \sim \pi_{\bar{\theta}^{\text{S}}}(\cdot \mid s_{i,u}, x_i)$ for $u > t$. We define the set of *decisive-disagreement* states as:

$$\mathcal{S}_{\text{D}} = \left\{s_{i,t} \mid a^{\text{L}}_{i,t} \neq a^{\text{S}}_{i,t},\ R(\widetilde{\tau}_i) - R(\tau_i) = 1\right\},$$

identifying critical decision points where a single cloud intervention can reverse the outcome, thereby providing precise supervision for when to ask for cloud assistance.

Consultation Cold-Start Fine-Tuning. The mined decisive-disagreement states are converted into supervised trajectories for initializing the consultation behavior of $\pi_{\theta^{\text{S}}}$. For each $s_{i,t} \in \mathcal{S}_{\text{D}}$, the original action history $\{(s_{i,1}, a^{\text{S}}_{i,1}), \ldots, (s_{i,t-1}, a^{\text{S}}_{i,t-1})\}$ is retained, and the target action at step t is replaced by `call_cloud`. The resulting trajectory with consultation is:

$$\tau^{\text{call}}_i = \{(s_{i,1}, a^{\text{S}}_{i,1}), \ldots, (s_{i,t-1}, a^{\text{S}}_{i,t-1}), (s_{i,t}, \texttt{call_cloud})\}. \tag{15}$$

These consultation trajectories are mixed with successful consultation-free trajectories sampled from $\pi_{\bar{\theta}^{\text{S}}}$ to fine-tune $\pi_{\theta^{\text{S}}}$, cold-starting selective cloud consultation at decisive states while preserving its standalone acting capability.

4.4 RL WITH CONSULTATION AND INTERNALIZATION

After consultation cold start, we continue training $\pi_{\theta^{\mathrm{S}}}$ with on-policy RL to jointly learn when to consult and internalize cloud guidance, enabling it to handle similar states independently while retaining consultation for cases beyond its capability.

Consultation-aware credit assignment. We retain the group-based RL objective $\mathcal{L}_{\mathrm{rl}}$ from § 4.2 to optimize $\pi_{\theta^{\mathrm{S}}}$ over the augmented action space. However, its trajectory-level advantage cannot disentangle the contribution of a consultation from subsequent actions. We therefore introduce consultation-aware credit by comparing cloud-assisted and local continuations from matched states.

Specifically, we construct an exact-state key $g_{i,n,t}$ for each visited state from its current observation and admissible action set. For each state $s_{i,n,t}$, we associate it with the matched-state set indexed by its key, formed over all states in the N trajectories sampled for task x_i:

$$\mathcal{I}_{g_{i,n,t}} = \{s_{i,m,u} \mid 1 \le m \le N,\ 1 \le u \le T_{i,m},\ g_{i,m,u} = g\}, \qquad g = g_{i,n,t}. \tag{16}$$

Let $\mu_{i,n,t}$ denote the mean trajectory return of the states in $\mathcal{I}_{g_{i,n,t}}$. We define the matched-state advantage as $A^{\mathrm{step}}_{i,n,t} = R(\tau_{i,n}) - \mu_{i,n,t}$.

To further isolate the contribution of consultation, for $q \in \{\mathrm{call}, \mathrm{local}\}$, we define the action-conditioned mean return within $\mathcal{I}_g$ as:

$$\overline{R}^{q}_{g} = \frac{\sum_{s_{i,m,u} \in \mathcal{I}_g} \mathbb{I}_q(a_{i,m,u}) R(\tau_{i,m})}{\sum_{s_{i,m,u} \in \mathcal{I}_g} \mathbb{I}_q(a_{i,m,u})}, \tag{17}$$

where $\mathbb{I}_{\mathrm{call}}(a) = 1$ if $a = \texttt{call_cloud}$, and $\mathbb{I}_{\mathrm{local}}(a) = 1$ if $a \in \mathcal{A}(s)$. For each consultation state $s_{i,n,t}$, we define:

$$A^{\mathrm{cf}}_{i,n,t} = \overline{R}^{\mathrm{call}}_{g_{i,n,t}} - \overline{R}^{\mathrm{local}}_{g_{i,n,t}}. \tag{18}$$

Notably, $A^{\mathrm{cf}}_{i,n,t} = 0$ for local actions or when no matched local continuation exists. Combining this consultation-specific term with the trajectory- and matched-state-level advantages, we define

$$A^{Sibyl}_{i,n,t} = A_{i,n} + \alpha_{\mathrm{step}} A^{\mathrm{step}}_{i,n,t} + \alpha_{\mathrm{cf}} A^{\mathrm{cf}}_{i,n,t}, \tag{19}$$

where α_{step} and α_{cf} control the contributions of matched-state and consultation-specific credit, respectively. This formulation preserves trajectory-level optimization while assigning finer-grained credit to matched states and beneficial consultations.

Cloud-guidance internalization. To further reduce cloud cost while preserving task success, we internalize guidance from on-policy cloud-assisted trajectories, enabling $\pi_{\theta^{\mathrm{S}}}$ to handle similar states independently whenever possible while retaining consultation for cases beyond its capability.

We reuse the frozen trajectory-skill supervisor $\mathcal{T}_{\theta^{\mathrm{S}}}$ from § 4.2. After each rollout, $\mathcal{T}_{\theta^{\mathrm{S}}}$ analyzes the trajectory $\tau_{i,n}$, including consultation decisions, executed cloud actions, subsequent observations, and the final outcome, to extract a hindsight skill:

$$z_{i,n} = \mathcal{T}_{\theta^{\mathrm{S}}}\big(x_i, \tau_{i,n}, R(\tau_{i,n})\big). \tag{20}$$

The skill-conditioned $\mathcal{L}_{\mathrm{opd}}$ from § 4.2 (Eq. 9) is then applied to distill behaviors favored under $z_{i,n}$ into the ordinary policy. Since $z_{i,n}$ is extracted from on-policy trajectories, cloud guidance internalization tracks the evolving state distribution.

Combining the above components, the final-stage training objective of Sibyl is:

$$\mathcal{L}_{Sibyl} = \mathcal{L}_{\mathrm{rl}}\big(\theta^{\mathrm{S}}; A^{Sibyl}\big) + \lambda_{\mathrm{opd}} \mathcal{L}_{\mathrm{opd}}, \tag{21}$$

where λ_{opd} controls the strength of guidance internalization. At inference time, the trajectory-skill supervisor and skill-augmented branch are removed.

Through the full three-stage training pipeline, Sibyl enables SLM agents to seek cloud assistance when beneficial and internalize reusable guidance to strengthen local capability, thereby further reducing cloud cost while preserving task success. Algorithm A details the training procedure. Together, matched-state credit and guidance internalization indirectly pursue the success–cost objective in Eq. 4 by focusing consultations on consequential states and reducing repeated cloud use. Tuning α_{cf} provides an implicit control over the performance–cost balance represented by λ in Eq. 4, producing the empirical Pareto frontier shown in § 5 and Figure 3.

Table 1: Results on ALFWorld and WebShop, averaged over three runs. ♡, ♢, ♣, and ♠: device-only, cloud-only, cloud-assisted training, and collaborative inference, respectively. The Qwen3-1.7B results for GRPO and SEED are from (Wu et al., 2026).

Method	ALFWorld							WebShop	
	Pick	Look	Clean	Heat	Cool	Pick2	All	Score	Succ.
Closed-Source Model									
♢ Qwen3.7-Max	$88.6_{\pm 2.9}$	$100.0_{\pm 0.0}$	$80.3_{\pm 7.7}$	$66.7_{\pm 3.6}$	$64.0_{\pm 4.0}$	$95.8_{\pm 4.2}$	$82.4_{\pm 0.8}$	$52.7_{\pm 0.2}$	$46.4_{\pm 0.6}$
♢ Gemini 3.1 Pro	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$96.3_{\pm 3.7}$	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$95.8_{\pm 4.1}$	$98.6_{\pm 1.4}$	$65.2_{\pm 0.2}$	$60.0_{\pm 0.4}$
Qwen3-0.6B assisted by Qwen3.7-Max									
♡ GRPO	$12.4_{\pm 1.7}$	$25.6_{\pm 19.4}$	$6.2_{\pm 5.7}$	$4.2_{\pm 3.6}$	$6.7_{\pm 2.3}$	$4.2_{\pm 4.2}$	$9.1_{\pm 2.3}$	$67.1_{\pm 0.1}$	$16.0_{\pm 0.4}$
♡ GiGPO	$54.3_{\pm 2.9}$	$59.0_{\pm 4.4}$	$46.9_{\pm 7.7}$	$27.1_{\pm 13.0}$	$33.3_{\pm 14.1}$	$1.4_{\pm 2.4}$	$37.4_{\pm 3.2}$	$78.6_{\pm 7.4}$	$33.4_{\pm 6.8}$
♣ $\text{GRPO}_{w/\text{SFT}}$	$100.0_{\pm 0.0}$	$87.2_{\pm 4.4}$	$100.0_{\pm 0.0}$	$79.2_{\pm 3.6}$	$78.7_{\pm 6.1}$	$88.9_{\pm 2.4}$	$90.7_{\pm 1.2}$	$85.9_{\pm 1.1}$	$79.8_{\pm 0.9}$
♣ $\text{GiGPO}_{w/\text{SFT}}$	$100.0_{\pm 0.0}$	$82.1_{\pm 11.7}$	$98.8_{\pm 2.1}$	$85.4_{\pm 13.0}$	$82.7_{\pm 4.6}$	$90.3_{\pm 6.4}$	$91.7_{\pm 0.4}$	$86.7_{\pm 1.3}$	$80.6_{\pm 0.6}$
♣ SEED	$84.8_{\pm 4.3}$	$89.7_{\pm 4.4}$	$92.6_{\pm 6.4}$	$93.8_{\pm 6.2}$	$85.3_{\pm 12.9}$	$87.5_{\pm 4.2}$	$88.3_{\pm 0.4}$	$67.6_{\pm 0.1}$	$49.2_{\pm 1.4}$
♠ RouteLLM	$77.1_{\pm 2.9}$	$84.6_{\pm 7.7}$	$51.9_{\pm 6.4}$	$62.5_{\pm 6.3}$	$44.0_{\pm 4.0}$	$79.2_{\pm 4.2}$	$65.7_{\pm 1.9}$	$24.6_{\pm 1.2}$	$22.8_{\pm 0.8}$
♠ Hera	$85.7_{\pm 2.9}$	$71.8_{\pm 4.4}$	$33.3_{\pm 3.7}$	$60.4_{\pm 7.2}$	$65.3_{\pm 12.9}$	$69.4_{\pm 2.4}$	$65.0_{\pm 2.1}$	$32.7_{\pm 1.7}$	$26.2_{\pm 1.2}$
♠ **Sibyl**	$100.0_{\pm 0.0}$	$92.3_{\pm 0.0}$	$100.0_{\pm 0.0}$	$89.6_{\pm 3.6}$	$92.0_{\pm 0.0}$	$91.7_{\pm 4.2}$	$95.2_{\pm 0.8}$	$90.7_{\pm 0.3}$	$80.4_{\pm 0.5}$
Qwen3-1.7B assisted by Qwen3.7-Max									
♡ GRPO	71.1	41.7	36.4	40.0	31.8	31.6	46.1	67.3	38.3
♡ GIGPO	$48.6_{\pm 7.5}$	$33.3_{\pm 4.4}$	$35.8_{\pm 2.1}$	$25.0_{\pm 6.2}$	$21.3_{\pm 9.2}$	$5.6_{\pm 2.4}$	$29.8_{\pm 2.8}$	$53.9_{\pm 0.1}$	$52.8_{\pm 0.4}$
♣ $\text{GRPO}_{w/\text{SFT}}$	$100.0_{\pm 0.0}$	$87.2_{\pm 8.9}$	$96.3_{\pm 3.7}$	$89.6_{\pm 13.0}$	$88.0_{\pm 14.4}$	$52.8_{\pm 4.8}$	$86.7_{\pm 4.8}$	$90.0_{\pm 0.8}$	$72.2_{\pm 1.2}$
♣ $\text{GiGPO}_{w/\text{SFT}}$	$100.0_{\pm 0.0}$	$94.9_{\pm 4.4}$	$96.3_{\pm 3.7}$	$97.9_{\pm 3.6}$	$94.7_{\pm 4.6}$	$61.1_{\pm 4.8}$	$91.0_{\pm 1.1}$	$89.0_{\pm 0.5}$	$75.4_{\pm 0.4}$
♣ SEED	97.6	100.0	100.0	80.0	84.2	90.0	92.0	87.1	77.3
♠ RouteLLM	$84.8_{\pm 1.7}$	$87.2_{\pm 4.4}$	$38.3_{\pm 2.1}$	$39.6_{\pm 3.6}$	$40.0_{\pm 6.9}$	$63.9_{\pm 8.7}$	$59.3_{\pm 3.6}$	$27.2_{\pm 1.4}$	$24.4_{\pm 3.6}$
♠ Hera	$92.4_{\pm 1.7}$	$87.2_{\pm 8.9}$	$27.2_{\pm 4.3}$	$43.8_{\pm 16.5}$	$72.0_{\pm 8.0}$	$70.8_{\pm 8.3}$	$66.4_{\pm 2.1}$	$38.6_{\pm 1.8}$	$29.2_{\pm 0.6}$
♠ **Sibyl**	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$92.0_{\pm 4.0}$	$93.1_{\pm 2.4}$	$97.4_{\pm 0.8}$	$91.1_{\pm 0.9}$	$84.6_{\pm 0.6}$

5 EXPERIMENTS

This section empirically evaluate Sibyl across a variety of long-horizon agentic tasks, aiming to assess task performance, consultation efficiency, the contributions of each training stage, and training and inference costs.

5.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate Sibyl on challenging benchmarks including ALFWorld (Shridhar et al., 2021) and WebShop (Yao et al., 2022). ALFWorld is a text-based embodied benchmark with 3,827 instances across six household task types: Pick & Place (Pick), Examine in Light (Look), Clean & Place (Clean), Heat & Place (Heat), Cool & Place (Cool), and Pick Two & Place (Pick2). WebShop requires agents to navigate a simulated e-commerce site and purchase suitable products from over 1.1 million items based on 12K user instructions. Our detailed dataset splits are provided in Appendix B.

Training details. We train Qwen3-0.6B/1.7B (Yang et al., 2025a) as the on-device SLMs, assisted by Qwen3.7-Max (Qwen Team, 2026) and Gemini 3.1 Pro (Gemini Team, 2026) as cloud LLMs. Agent outputs follow the ReAct (Yao et al., 2023) format. Training is conducted on eight AMD Instinct MI308X GPUs. Stage 1 (§ 4.2) trains $\mathcal{T}_{\theta^{\text{S}}}$ with learning rate (lr) 5×10^{-6} and batch size (bs) 8, then performs consultation-free RL with $\pi_{\theta^{\text{S}}}$ at lr 10^{-6} with group size 8, bs 256, $\beta_{\text{opd}} = 5.0$, and $\lambda_{\text{opd}} = 0.01$. Stage 2 (§ 4.3) trains at lr 2×10^{-6} with bs 16. Stage 3 (§ 4.4) performs RL at lr 10^{-6} with group size 8, bs 256, $\lambda_{\text{opd}} = 0.01$, $\alpha_{\text{step}} = 0.5$, and $\alpha_{\text{cf}} = 0.5$. Further details are provided in Appendix C.

Baselines. We compare Sibyl against four baseline families: (1) *cloud-only methods* directly prompt the frontier closed-source Qwen3.7-Max and Gemini 3.1 Pro; (2) *device-only methods* train SLMs with agentic RL, including GRPO (Shao et al., 2024) with group-relative critic-free optimization and GiGPO (Feng et al., 2025) with hierarchical step-level credit assignment; (3) *training-time cloud-assisted methods* include SEED (Wu et al., 2026) with cloud-guided on-policy hindsight-skill distillation, and GRPO/GiGPO cold-started through supervised fine-tuning on cloud-generated trajectories; (4) *inference-time collaborative methods* include RouteLLM (Ong et al., 2025) and Hera (Zhang et al., 2026), which train routers for request-level and step-level routing, respectively.

Table 2: Task performance and test-time cloud cost on ALFWorld and WebShop. Call ratio is the percentage of executed steps delegated to the cloud, Calls/Ep. is the mean number of cloud calls per episode, and Cost is the total API cost over the full test set, calculated from cloud input/output token usage and API prices.

Method	ALFWorld				WebShop				
	Succ.↑	Call ratio↓	Calls / Ep.↓	Cost↓	Score↑	Succ.↑	Call ratio↓	Calls / Ep.↓	Cost↓
Closed-Source Model									
Qwen3.7-Max	82.4	100.0	15.2	$8.6	52.7	46.4	100.0	10.8	$48.5
Gemini-3.1 Pro	98.6	100.0	9.1	$5.8	65.2	60.0	100.0	8.4	$58.6
Qwen3-0.6B assisted by Qwen3.7-Max									
RouteLLM	65.7	64.5	12.5	$5.3	24.6	22.8	85.5	5.1	$18.3
Hera	65.0	54.6	5.3	$4.8	32.7	26.2	68.9	4.2	$12.2
Sibyl	95.2	7.7	0.8	$0.4	90.7	80.4	64.8	3.9	$16.1
Qwen3-1.7B assisted by Qwen3.7-Max									
RouteLLM	59.3	60.4	12.4	$5.1	27.2	24.4	86.6	5.7	$20.9
Hera	66.4	47.9	4.8	$3.9	38.6	29.2	67.7	4.2	$12.0
Sibyl	97.4	2.8	0.3	$0.2	91.1	84.6	3.1	0.2	$1.1

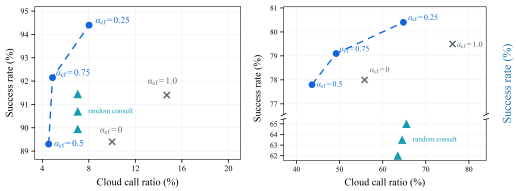
5.2 Overall Results

Table 1 shows that Sibyl improves long-horizon task success across both SLM sizes. With Qwen3-0.6B, it achieves 95.2% success on ALFWorld and 80.4% on WebShop, versus 37.4% and 33.4% for the strongest device-only baselines. Cloud-assisted training substantially improves the SLM, but Sibyl further raises ALFWorld success from 91.7% to 95.2% and achieves a higher WebShop score (90.7 versus 86.7); its WebShop success is comparable to the best training-time baseline (80.4% versus 80.6%). The higher score despite similar exact success suggests better partial progress on episodes that do not result in a complete purchase. With Qwen3-1.7B, Sibyl exceeds the strongest training-time baselines by 5.4 and 7.3 success points on the two benchmarks, indicating that selective consultation remains useful as local model capacity increases. Among methods that use cloud assistance at inference time, Sibyl achieves higher success than RouteLLM and Hera with fewer cloud calls, as detailed in the next section. Both baselines train an external router to allocate decisions between fixed device and cloud policies. Sibyl instead jointly optimizes local actions and consultation within the SLM, enabling it to seek help at consequential states and internalize cloud guidance for similar future states. Notably, Sibyl surpasses cloud-only Qwen3.7-Max on both benchmarks while consulting that same model for only a subset of steps, suggesting that it learns to combine local execution and targeted cloud intervention into more effective trajectories than cloud-only execution.

5.3 Analysis of Cloud Consultation

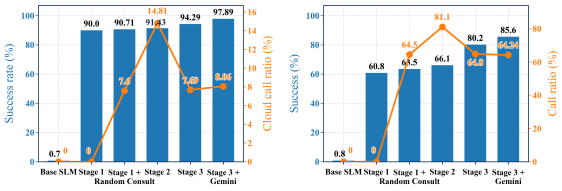
Table 2 shows that Sibyl achieves strong task success with exceptionally low cloud-invocation cost. With a 0.6B SLM, it reaches 95.2% success on ALFWorld and 80.4% on WebShop while reducing cost by 21.5× and 3.0× relative to cloud-only Qwen3.7-Max. With a 1.7B SLM, success rises to 97.4% and 84.6%, at 43× and 44× lower cost; only 2.8% and 3.1% of steps invoke the cloud. The ALFWorld result approaches cloud-only Gemini 3.1 Pro (98.6%), while the WebShop result substantially exceeds it (60.0%). Although RouteLLM and Hera use external routers to coordinate fixed device and cloud policies, Sibyl jointly learns local actions and consultation while internalizing cloud guidance. The SLM can thus handle similar states independently and reserve calls for cases beyond its capability, achieving a stronger success–cost trade-off.

Figure 3 further shows that this advantage persists across cloud budgets. These results are from a first run checkpoint, rather than the averages reported earlier. On ALFWorld, Sibyl reaches 94.3% success at an approximately 8% call ratio, or 92.1% with fewer than 5% of steps delegated; the corresponding WebShop frontier reaches 80.4% at 67% and 77.8% at 45%. These settings outperform random consultation, RouteLLM, and Hera, indicating that the gains stem from learning when to seek assistance rather than simply having access to the cloud. Moreover, neither removing consultation-specific credit ($\alpha_{\mathrm{cf}} = 0$) nor assigning it excessive weight ($\alpha_{\mathrm{cf}} = 1$) reaches the frontier. Intermediate



(a) ALFWorld (b) WebShop

Figure 3: Success rate versus cloud call ratio on (a) ALFWorld and (b) WebShop.



(a) ALFWorld (b) WebShop

Figure 4: Ablation of Sibyl with a Qwen3-0.6B SLM assisted by Qwen3.7-Max on (a) ALFWorld and (b) WebShop.

weights most effectively reward consequential consultations while suppressing unnecessary cloud calls.

5.4 Ablation Study

Figure 4 traces the contribution of each training stage on ALFWorld and WebShop. Starting from base SLM success rates of 0.7% and 0.8%, Stage 1 raises success to 90.0% and 60.8% without inference-time consultation, establishing a capable local policy. Adding random cloud calls yields only modest further gains, showing that cloud access alone is insufficient. Stage 2 improves success by cold-starting consultation at states where a one-step cloud action can reverse failure, but its higher call ratios indicate that this supervision has yet to learn when consultation can be avoided. Stage 3 effectively resolves this trade-off: success rises to 94.3% and 80.2% while call ratios fall to 7.7% and 64.8%, respectively. It also substantially outperforms random consultation at nearly the same call ratios, supporting the value of jointly optimizing consultation decisions and internalizing cloud guidance. Notably, replacing Qwen3.7-Max with the stronger Gemini 3.1 Pro only at inference improves success on both benchmarks without retraining the SLM, suggesting that the learned consultation policy can transfer across cloud models and benefit from stronger assistance.

5.5 Cost Analysis

Table 3: Training and inference costs of Sibyl on ALFWorld and WebShop.

	Training API Cost (Stage 1/2/3)	Training GPU Hours	Inference API Cost (full test set)
ALFWorld	\$10.7/\$1.7/\$92.7	149.7	\$0.4
WebShop	\$13.0/\$6.5/\$126.0	205.7	\$16.1

Table 3 reports training and inference API costs, together with training GPU-hours. Training incurs \$105.1 and \$145.5 in API charges for ALFWorld and WebShop, respectively, with Stage 3 accounting for the largest share. At inference time, Sibyl reduces the average cloud API cost from \$8.6 to \$0.4 on ALFWorld and from \$48.5 to \$16.1 on WebShop, relative to cloud-only Qwen3.7-Max, corresponding to savings of 95.4% and 66.8%. The training API expenditure is therefore offset after approximately 1,794 ALFWorld or 2,245 WebShop tasks. These break-even estimates exclude the cost of 149.7 and 205.7 AMD Instinct MI308X GPU-hours of training, respectively.

6 Conclusion

We presented Sibyl as a policy-optimization framework that trains SLM agents to selectively consult stronger cloud models and internalize their guidance. By treating consultation as a native policy action, its three-stage training jointly strengthens local capability, learns when to ask, and reduces repeated cloud dependence. Experiments on challenging long-horizon benchmarks show that Sibyl achieves strong task success while substantially reducing cloud calls, advancing the success–cost trade-off over existing agent training and routing methods. More broadly, Sibyl offers an integrated framework that unifies coordination and capability transfer across heterogeneous language-model agents; its limitations and future directions are discussed in Appendix F.

AI USE STATEMENT

We used generative AI tools to help formulate and refine the research methodology, discuss experimental design, and interpret results. We also used them to discover related work and to draft and polish parts of the manuscript. The authors reviewed and revised AI-assisted text, checked citations against the cited works, and verified reported results against experimental records. We take responsibility for all content of this paper, including text and claims produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

§ 4 specifies the three-stage training procedure and optimization objectives, while § 5.1 reports the models, baselines, and main training settings. The appendix provides further details on benchmark splits, evaluation protocols, prompts, and implementation. We will release the code and model checkpoints shortly.

REFERENCES

Antonis Antoniades, Albert Örwall, Kexun Zhang, Yuxi Xie, Anirudh Goyal, and William Yang Wang. SWE-search: Enhancing software agents with monte carlo tree search and iterative refinement. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=G7sIFXugTX.

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: Cost-efficient and quality-aware query routing. In *The Twelfth International Conference on Learning Representations*, 2024.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems*, 2025.

Gemini Team. "Gemini 3.1 Pro: A smarter model for your most complex tasks", 2026. Available: https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-1-pro/.

Kimi Team. "Kimi K3: Open Frontier Intelligence", 2026. Available: https://www.kimi.com/blog/kimi-k3.

Taeyoon Kwon, Dongwook Choi, Hyojun Kim, Sunghwan Kim, Seungjun Moon, Beong woo Kwak, Kuan-Hao Huang, and Jinyoung Yeo. Embodied agents meet personalization: Investigating challenges and solutions through the lens of memory utilization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=E5L43l5EIu.

Siteng Liao, Yuzhu Liang, Hengzhong Rao, Xizhao Luo, and Tian Wang. Alpharouter: Token-level routing between slm and llm with reinforcement learning and tree search. In *Forty-third International Conference on Machine Learning*, 2026.

Genglin Liu, Shijie Geng, Sha Li, Hejie Cui, Sarah Zhang, Xin Liu, and Tianyi Liu. Webcoach: Self-evolving web agents with cross-session memory guidance. In *ICLR 2026 Workshop on Memory for LLM-Based Agentic Systems*, 2026. URL https://openreview.net/forum?id=FDrGfjwQM5.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024.

Xinji Mai, Haotian Xu, Weinong Wang, Yingying Zhang, Wenqiang Zhang, et al. Agentic rl scaling law: Spontaneous code execution for mathematical problem solving. *Advances in Neural Information Processing Systems*, 38:7325–7340, 2026.

Avanika Narayan, Dan Biderman, Sabri Eyuboglu, Avner May, Scott Linderman, James Zou, and Christopher Re. Cost-efficient Collaboration between On-device and Cloud Language Models. In *Proc. of the 42nd ICML*, 2025.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=8sSqNntaMr.

OpenAI. "GPT-5.6: Frontier intelligence that scales with your ambition", 2026. Available: https://openai.com/index/gpt-5-6/.

Qwen Team. Qwen3.7: The agent frontier, May 2026. URL https://qwen.ai/blog?id=qwen3.7.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

SeungWon Seo, SooBin Lim, Seongrae Noh, Haneul Kim, and HyeongYeop Kang. From assumptions to actions: Turning LLM reasoning into uncertainty-aware planning for embodied agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=GODFBZhFcX.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

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

Zishen Wan, Yuhang Du, Mohamed Ibrahim, Jiayi Qian, Jason Jabbour, Yang Zhao, Tushar Krishna, Arijit Raychowdhury, and Vijay Janapa Reddi. Reca: Integrated acceleration for real-time and efficient cooperative embodied autonomous agents. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 982–997, 2025.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning, 2025. URL https://arxiv.org/abs/2504.20073.

Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. Autodroid: Llm-powered task automation in android. In *Proceedings of the 30th annual international conference on Mobile computing and networking*, pp. 543–557, 2024.

Jinyang Wu, Shuo Yang, Zhengxi Lu, Fan Zhang, Yuhao Shen, Lang Feng, Haoran Luo, Zheng Lian, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. SEED: Self-evolving on-policy distillation for agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2607.14777.

Frank Fangzheng Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, et al. Theagentcompany: benchmarking llm agents on consequential real world tasks. *Advances in Neural Information Processing Systems*, 38, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a. URL https://arxiv.org/abs/2505.09388.

Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. Agentoccam: A simple yet strong baseline for LLM-based web agents. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=oWdzUpOlkX.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

Kexun Zhang, Weiran Yao, Zuxin Liu, Yihao Feng, Zhiwei Liu, Rithesh R N, Tian Lan, Lei Li, Renze Lou, Jiacheng Xu, Bo Pang, Yingbo Zhou, Shelby Heinecke, Silvio Savarese, Huan Wang, and Caiming Xiong. Diversity empowers intelligence: Integrating expertise of software engineering agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=cKlzKs3Nnb.

Yuxin Zhang, Mengxue Hu, Zheng Lin, Xiaoyi Fan, Fan Xie, Zihan Fang, Jing Yang, Wenjun Zhu, Zhiwen Chen, Chengfei Lv, et al. Hera: Learning long-horizon coordination for device-cloud collaborative llm agents. *arXiv preprint arXiv:2605.24598*, 2026.

Zesen Zhao, Shuowei Jin, and Z Morley Mao. Eagle: Efficient training-free router for multi-llm inference. In *Proc. of the 7th Workshop on Machine Learning for Systems at NeurIPS*, 2024.

A PSEUDOCODE

Algorithm 1 Sibyl: Three-Stage Training for Step-Level Device–Cloud Collaboration

Input: Training tasks $\mathcal{X}$; SLM policy $\pi_{\theta^{\mathrm{S}}}$; fixed cloud policy $\pi_{\theta^{\mathrm{L}}}$; rollouts per task N; weights $\lambda_{\mathrm{opd}}, \alpha_{\mathrm{step}}, \alpha_{\mathrm{cf}}$.
Output: Trained SLM policy $\pi_{\theta^{\mathrm{S}}}$.
Stage 1: Consultation-free self-evolving RL
Roll out $\pi_{\theta^{\mathrm{S}}}$ without consultation and obtain hindsight skills from the cloud trajectory teacher.
Fine-tune $\pi_{\theta^{\mathrm{S}}}$ on trajectory–skill pairs to initialize the skill supervisor $\mathcal{T}_{\theta^{\mathrm{S}}}$; restore action generation through SFT on cloud-generated state–action pairs.
for each RL iteration **do**
  for all tasks x_i **do**
    Sample N local-only trajectories $\{\tau_{i,n}\}_{n=1}^{N}$.
    Extract $z_{i,n} = \mathcal{T}_{\theta^{\mathrm{S}}}(x_i, \tau_{i,n}, R(\tau_{i,n}))$ and compute group-relative advantages $A_{i,n}$.
  end for
  Update θ^{S} using $\mathcal{L}_{\mathrm{rl}} + \lambda_{\mathrm{opd}}\mathcal{L}_{\mathrm{opd}}$.
end for
Freeze the resulting policy as $\bar{\pi}^{\mathrm{S}}$ and the skill supervisor $\mathcal{T}_{\theta^{\mathrm{S}}}$.
Stage 2: Consultation cold start
Collect failed trajectories $\mathcal{D}^{-}$ and successful trajectories $\mathcal{D}^{+}$ using $\bar{\pi}^{\mathrm{S}}$.
for all states $s_{i,t}$ in trajectories $\tau_i \in \mathcal{D}^{-}$ **do**
  Sample one cloud action $a^{\mathrm{L}}_{i,t} \sim \pi_{\theta^{\mathrm{L}}}(\cdot \mid s_{i,t}, x_i)$.
  if $a^{\mathrm{L}}_{i,t} \neq a^{\mathrm{S}}_{i,t}$ **then**
    Replace $a^{\mathrm{S}}_{i,t}$ with $a^{\mathrm{L}}_{i,t}$ and complete the branch using $\bar{\pi}^{\mathrm{S}}$, obtaining $\tilde{\tau}_i$.
    if $R(\tilde{\tau}_i) - R(\tau_i) = 1$ **then**
      Add the original trajectory prefix followed by $(s_{i,t}, \texttt{call_cloud})$ to $\mathcal{D}^{\mathrm{call}}$.
    end if
  end if
end for
Fine-tune $\pi_{\theta^{\mathrm{S}}}$ on $\mathcal{D}^{\mathrm{call}} \cup \mathcal{D}^{+}$.
Stage 3: RL with consultation and internalization
for each RL iteration **do**
  for all tasks x_i **do**
    Sample N trajectories using the augmented action space; each `call_cloud` executes one cloud action before local control resumes.
    Group matching states by exact-state key g and compute trajectory advantages $A_{i,n}$ and matched-state advantages $A^{\mathrm{step}}_{i,n,t}$.
    For consultation actions, compute $A^{\mathrm{cf}}_{i,n,t} = \overline{R}^{\mathrm{call}}_{g} - \overline{R}^{\mathrm{local}}_{g}$ when both groups exist; otherwise set it to zero.
    Set $A^{\mathrm{SIBYL}}_{i,n,t} = A_{i,n} + \alpha_{\mathrm{step}} A^{\mathrm{step}}_{i,n,t} + \alpha_{\mathrm{cf}} A^{\mathrm{cf}}_{i,n,t}$.
    Extract hindsight skills $z_{i,n}$ using the frozen $\mathcal{T}_{\theta^{\mathrm{S}}}$.
  end for
  Update θ^{S} using $\mathcal{L}_{\mathrm{rl}}(A^{\mathrm{SIBYL}}) + \lambda_{\mathrm{opd}}\mathcal{L}_{\mathrm{opd}}$.
end for
return $\pi_{\theta^{\mathrm{S}}}$.

B BENCHMARK PROTOCOL AND DATA SPLITS

We evaluate the Sibyl on two interactive benchmarks, ALFWorld and WebShop, under the benchmark-specific protocols described below.

ALFWorld. ALFWorld (Shridhar et al., 2021) provides a text-based view of household tasks aligned with the embodied ALFRED environment. An agent receives a natural-language goal, observes the environment through text, and selects admissible actions until the goal is completed. The benchmark covers six task types: *Pick & Place*, *Examine in Light*, *Clean & Place*, *Heat & Place*, *Cool & Place*,

and *Pick Two & Place*. The official split contains 3,553 training games, 140 Seen games, and 134 Unseen games. Seen games use rooms encountered during training with new object configurations, whereas Unseen games use previously unseen rooms and layouts. We report the main results on the 140-task Seen split and use the 134-task Unseen split to assess out-of-distribution generalization. Each episode is limited to 30 steps, and the prompt contains the most recent 5 interaction steps. The prompt and response limits are 2,048 and 512 tokens, respectively.

WebShop. WebShop (Yao et al., 2022) is a simulated shopping website built from more than 1.1 million products and 12,087 crowdsourced instructions. Each instruction specifies requirements such as product attributes, options, and a price limit. The agent searches the catalog, navigates product and detail pages, selects options, and buys a product. The environment computes a reward from the match between the selected product and the instruction, including its type, attributes, options, and price. We report Task Score, defined as 100 times the mean reward, and Success Rate, the fraction of episodes with reward one. We evaluate all methods on 500 test samples, with a maximum of 15 steps per episode and the most recent 5 interaction steps included in the prompt. The prompt and response limits are 12,288 and 512 tokens, respectively.

C Experimental Details

C.1 Prompts

At each step, the on-device SLM receives the task instruction, current observation, recent interaction history, and exact admissible actions. The initial prompt contains no interaction history. The action set also includes `call_cloud`. For local action generation, both benchmarks use the ReAct format with one `<think>` block followed by one `<action>` block. ALFWorld actions are drawn from the admissible environment actions, whereas WebShop actions take the form `search[...]` or `click[...]`. When the SLM selects `call_cloud`, the cloud model receives the complete action–observation–result trajectory and the current available actions, then returns one executable action under the benchmark-specific prompts below.

ALFWorld.

```
You are an expert agent operating in the ALFRED Embodied Environment. Your task is to: {
    task_description}
Prior to this step, you have already taken {step_count} step(s). Below are the most recent {
    history_length} observations and the corresponding actions you took: {action_history}
You are now at step {current_step} and your current observation is: {current_observation}
Your admissible actions of the current situation are: [{admissible_actions}].

Now it's your turn to take an action.
You should first reason step-by-step about the current situation. This reasoning process MUST
    be enclosed within <think> </think> tags.
Once you've finished your reasoning, you should choose an admissible action for current step
    and present it within <action> </action> tags.
The special action 'call_cloud' delegates only the current step to a cloud model. After that
    single action, you resume control.
```

WebShop.

```
You are an expert autonomous agent operating in the WebShop e-commerce environment.
Your task is to: {task_description}.
Prior to this step, you have already taken {step_count} step(s). Below are the most recent {
    history_length} observations and the corresponding actions you took: {action_history}
You are now at step {current_step} and your current observation is: {current_observation}.
Your admissible actions of the current situation are:
[
{available_actions}
].

Now it's your turn to take one action for the current step.
You should first reason step-by-step about the current situation, then think carefully which
    admissible action best advances the shopping goal. This reasoning process MUST be
    enclosed within <think> </think> tags.
Your entire response MUST contain both the nonempty <think> block and exactly one <action>
    block, in that order.
```

```
Use this exact layout: <think>brief current-step reasoning</think><action>search[red shoes]</
    action>
The action must be either a search query or a click on one of the listed clickables. Do not
    use Markdown or add text outside the two tags.
```

When the on-device SLM selects `call_cloud`, the cloud model receives the complete interaction history and the current admissible action set. We use the following benchmark-specific prompts to constrain it to one executable action.

ALFWorld.

```
You are the one-step cloud controller for an ALFWorld episode.

Choose exactly one action for the current state.

Return only <action>ONE EXACT ADMISSIBLE ACTION</action>.

Do not return a plan, explanation, multiple actions, or call_cloud.

Task: {task}

Inventory: {inventory_or_empty}

Complete action-observation-result trajectory:

{full_action_observation_result_trajectory}

Current observation: {current_observation}

Current admissible actions:

- {admissible_action_1}
- {admissible_action_2}
...
```

WebShop.

```
You are the one-step cloud controller for a WebShop episode.

Choose exactly one action for the current state.

Return exactly one action using this format:

<action>ONE EXACT WEBSHOP ACTION</action>

Do not output <think>, reasoning, explanation, Markdown, multiple actions, or call_cloud.

The action must be either:

- search[nonempty query]

- click[one exact action from the available actions]

Shopping instruction: {task}

Complete action-observation-result trajectory:

{history}

Current observation: {current_observation}

Current available actions:

- {available_action_1}
- {available_action_2}
- {available_action_3}
...
```

C.2 TRAJECTORY–SKILL GENERATION

In Stage 1, the cloud trajectory teacher receives a task instruction x, an SLM-generated trajectory τ^{S}, and its return $R(\tau^{\mathrm{S}})$, and produces a JSON annotation containing an episode summary and a

reusable episode skill. Table 4 shows a real ALFWorld example from the annotation data. The source trajectory contains 30 steps and ends in failure; we summarize the trajectory to fit the layout and reproduce the generated fields verbatim.

Skill-generation prompt. The trajectory teacher is prompted with the following template. The `episode_skill_instruction` is selected according to the episode outcome.

```
Analyze the following agent episode and return ONLY valid JSON.

You need to complete two fields:

1. Write a concise episode_summary.
2. {episode_skill_instruction}

Important constraints:

- Write episode_skill as one short policy-facing skill, not as retrospective
  explanation of the trajectory.
- Return only these top-level fields: episode_summary, episode_skill.

Return format:
{
  "episode_summary": "string",
  "episode_skill": "string"
}

Episode context:

- Task description: {task_description}
- episode_success: {success|failure|unknown}
- Interaction trajectory:

{formatted_trajectory}

For a successful trajectory, write one episode_skill that extracts the
successful trajectory into a workflow: the core decision rule and action
ordering that made the trajectory work.

For a failed trajectory, write one episode_skill that extracts the failed
trajectory into avoidance rules: the core mistake and warning signs that the
agent should avoid.

The formatted trajectory is:

Step {step_index}

Observation: {observation}

Response: {model_response}

Action valid: {true|false}
```

C.3 Models and Training

We train Qwen3-0.6B or Qwen3-1.7B as the on-device SLMs. Qwen3.7-Max is the primary cloud model, and Gemini 3.1 Pro is used for the separate cloud-model comparison. The primary ALFWorld run uses 8 AMD Instinct MI308X GPUs.

Stage 1 first collects 180 training tasks with eight trajectories per task, yielding 1,440 trajectory–skill examples (1,296 train and 144 validation). Skill SFT uses three epochs, maximum sequence length 8,192, learning rate 5×10^{-6}, and batch size 8. We then perform action/protocol-repair SFT on 512 train and 64 validation examples for one epoch, with maximum sequence length 4,096, learning rate 2×10^{-6}, and batch size 8. Consultation-free RL runs for 160 updates at learning rate 10^{-6}, with group size 8, batch size 256, $\beta_{\text{opd}} = 5.0$, and $\lambda_{\text{opd}} = 0.01$.

Stage 2 starts from the Stage 1 policy. Failure mining uses 1,024 label-training games, 256 validation games, and 256 runtime games; each failed trajectory contributes at most three candidate states, and replay uses the exact prefix. The resulting cold-start SFT train for two epochs with maximum sequence length 4,096, learning rate 2×10^{-6}, and batch size 16.

Input


```
{
  "task_description":
    "put two candle in drawer.",
  "episode_success": "failure",
  "candidate_step_indices": [0, 1, ..., 29],
  "trajectory": [
    {
      "step_index": 0,
      "observation": "-= Welcome to TextWorld
    , ALFRED! =- ...
        Your task is to: put two candle in
    drawer.",
      "response": "<think>...</think><action>
    go to countertop 1</action>",
      "actions": ["go to countertop 1"],
      "action_valid": true
    },
    ...,
    {
      "step_index": 29,
      "observation": "Nothing happens.",
      "response": "<think>...</think><action>
    go to drawer 1</action>",
      "actions": ["go to drawer 1"],
      "action_valid": true
    }
  ]
}
```


Output


```
{
  "episode_summary": "The agent failed to
     complete the task of putting two candles
      in a drawer. It systematically searched
      drawers and countertops but never
     located any candles. During the search,
     the agent exhibited inefficient
     behaviors such as repeatedly visiting
     already-checked empty drawers, picking
     up and immediately replacing irrelevant
     items (cloth), and taking unrelated
     objects (dishsponge) without purpose.
     The episode ended with the agent still
     searching and holding an irrelevant item,
      having made no progress toward the
     actual goal.",


  "episode_skill": "When searching for target
      objects, strictly avoid interacting
     with or picking up items that do not
     match the task description; if a
     location has been confirmed empty or
     containing only irrelevant items, do not
      revisit it, and maintain focus on
     unexplored areas until the target object
      is found."
}
```


Table 4: A real ALFWorld trajectory-to-skill example from Stage 1. The input preserves the original JSON fields and selected trajectory entries; ellipses indicate omitted intermediate steps. The output is reproduced from the corresponding annotation.

Stage 3 continues in the augmented action space, where each consultation executes one cloud action before local control resumes. We train for one epoch over these rows, yielding 160 optimizer updates, and use learning rate 10^{-6}, group size 8, batch size 256, $\alpha_{\text{step}} = 0.5$, $\alpha_{\text{cf}} = 0.25$, and $\lambda_{\text{opd}} = 0.01$.

D Supplementary Results

D.1 Cross-Domain Generalization

On the ALFWorld Unseen subset, we achieve 96.3%, compared with 94.3% on Seen, for a combined score of 95.3%. The Unseen rooms and layouts were not observed during training, demonstrating strong out-of-distribution (OOD) generalization. Success is at least 94.1% on five task types, with 100% on Look and Clean (Table 5). This strong OOD performance may be due to the way the model consults the cloud, which makes the impact of OOD on its performance small.

Table 5: Unseen-split success by task type.

Task type	Success (%)
Pick	91.7
Look	100.0
Clean	100.0
Heat	95.7
Cool	95.2
Pick2	94.1
Overall	**96.3**

D.2 Analysis by Task Type

Table 6 compares success and average cloud calls per episode for each task type. On Seen, Pick and Clean achieve the highest success, whereas Heat and Pick2 are the most challenging and use the most consultations. On Unseen, success improves for every task type except Pick: Heat and Pick2 gain 8.15 and 6.62 percentage points, respectively, while Look and Clean reach 100%. The consultation pattern also changes across splits. Pick, Look, and Clean use more calls on Unseen, whereas Cool, Heat, and Pick2 use fewer. Thus, the checkpoint does

Table 6: Task-wise success and average cloud calls per episode on the Seen and Unseen splits.

Task	Seen success (%)	Seen calls/ep.	Unseen success (%)	Unseen calls/ep.
Pick	100.0	0.1	91.7	0.5
Look	92.3	0.5	100.0	0.7
Clean	100.0	0.5	100.0	0.7
Cool	92.0	1.2	95.2	0.7
Heat	87.5	1.5	95.7	0.8
Pick2	87.5	1.3	94.1	0.6

not simply consult more on tasks with lower success; its consultation demand varies with the task structure and the encountered layout.

E Cases Study

Success rates quantify the overall benefit of consultation but do not establish whether the cloud model's input was strictly necessary. To examine this question, we analyze four representative rescue cases from the evaluation traces in Table 7 under aligned comparison settings. Both ALFWorld branches use the same 30-step budget, while both WebShop branches use the same 15-step budget, start from the same restored session, and share the same frozen edge continuation, differing only in the action taken at the disputed state. We present two representative cases from each benchmark to illustrate our central finding: targeted cloud consultation can resolve decision bottlenecks and recover successful trajectories.

In all four cases, the on-device SLM makes an incorrect decision when operating alone. The cloud model is consulted at this decision bottleneck and provides an executable corrective action; the on-device SLM then resumes control and completes the task successfully. This reflects Sibyl's intended division of labor: the cloud model resolves a difficult local decision, while the edge model executes the remaining routine steps.

The two ALFWorld cases involve cloud-guided search and target disambiguation. Once the cloud model identifies the correct location or target, the on-device SLM correctly performs subsequent actions such as picking up, cleaning, and placing the object. The two WebShop cases are more localized: the on-device SLM fails to select the required product option on its own, whereas the cloud model selects the correct option. The on-device SLM then executes the same final "buy now" action, which succeeds with the correct option in place. Together, these traces provide qualitative evidence that Sibyl invokes the cloud at decision bottlenecks, corrects local edge-model errors, and delegates the remaining workflow to the on-device model.

F Limitations and Future Work

Sibyl focuses on when an on-device agent should request one-step assistance from a fixed cloud model. This formulation assumes that the cloud service is available when needed and that the agent can share sufficient task context to obtain a useful action. Privacy-sensitive applications and intermittent connectivity introduce additional constraints: the agent may need to limit the context it transmits or continue safely when a consultation fails. Future work could incorporate privacy-aware context selection and fallback behavior into the consultation policy.

The current formulation also treats the cloud as a single source of assistance. A deployed agent may instead have access to models with different capabilities, prices, and response times. Extending Sibyl to choose both when to consult and which model to consult could improve its flexibility under changing workloads and service conditions, while retaining local execution as the default.

Case	Edge-only branch	Cloud's critical action	Edge continuation and outcome
ALFWorld: clean butterknife	30 steps, no cloud calls, and failure. The on-device SLMs takes an ordinary `knife`, cleans and returns it, and never reaches the target `butterknife`.	The cloud steers the search to `diningtable 3` at step 20, where `butterknife 1` is visible.	The on-device SLMs takes the butterknife, cleans it at the sink, and moves it to `sidetable 1`. The collaborative run succeeds in 25 steps with 3 cloud calls.
ALFWorld: two credit cards	30 steps, no cloud calls, and failure while searching for the second card after placing the first.	The cloud first requests `inventory`, then chooses `diningtable 1` to locate `creditcard 1`, and finally `armchair 1` as the destination.	The on-device SLMs takes the card and executes the final move locally. The collaborative run succeeds in 14 steps with 3 cloud calls.
WebShop: random-color bath mitt	From the same product page, the forced edge action is `click[buy now]`; the purchase has empty options and scores 0.75.	The cloud selects the required option `click[random color 1pc]`.	The identical edge continuation then executes `click[buy now]`, producing the correct option and score 1.00.
WebShop: ottoman size	From the same product page, the forced edge action is `click[buy now]`; the purchase has empty options and scores 0.80.	The cloud selects the required size `click[100x40x40cm]`.	The identical edge continuation executes `click[buy now]`, producing the correct size and score 1.00.

Table 7: Representative edge–cloud rescue cases. The cloud provides the bottleneck action shown in the third column; subsequent actions are executed by the on-device SLM. The two benchmarks follow their respective evaluation protocols.