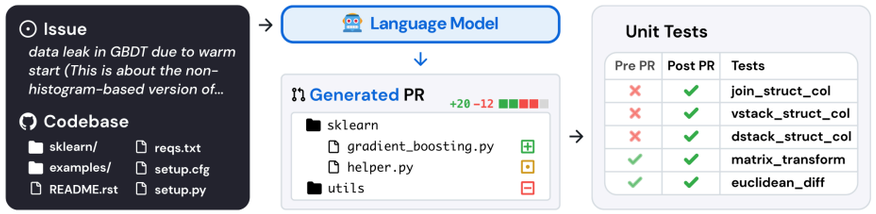
Scaling Long-Horizon LLM Agent via Context-Folding

https://context-folding.github.io

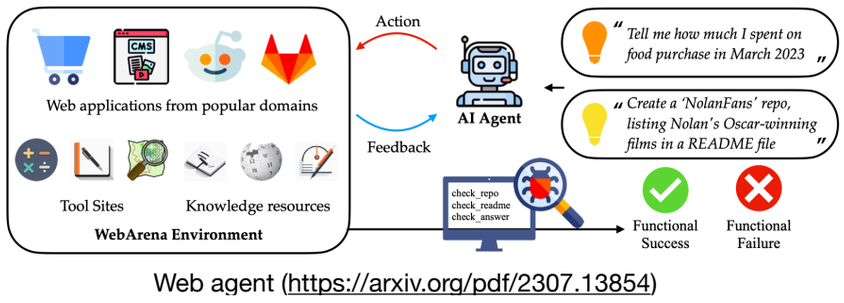
Joint work by Weiwei Sun*, Miao Lu*, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, Jiecao Chen

* Work down during internship at ByteDance Seed

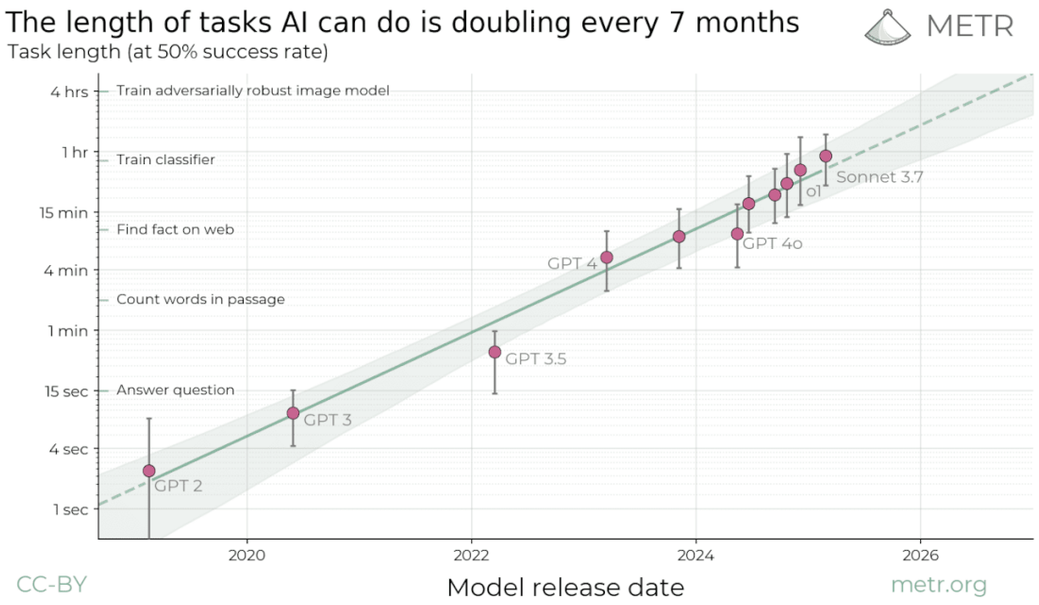
- LLM agents have shown remarkable capability in various complex tasks:
  - Deep research agent, Coding agent, Web agent, …
- A key bottleneck: Exploding context length
  - "Length of tasks AI can do is growing exponentially, with a doubling time of 7 months"



Coding agent (https://arxiv.org/pdf/2310.06770)



Web agent (https://arxiv.org/pdf/2307.13854)

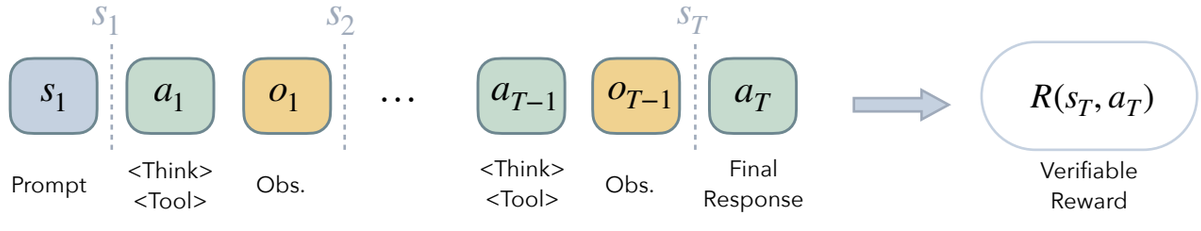


- Challenges in the long context length regime:
  - Context window limit by the backend LLM
  - Degrading instruction-following and reasoning capabilities
  - Quadratic attention cost, KV-cache overhead

- LLM agents have shown remarkable capability in various complex tasks:
  - Deep research agent, Coding agent, Web agent, …
- A key bottleneck: Exploding context length
  - "Length of tasks AI can do is growing exponentially, with a doubling time of 7 months"
- Existing approaches:
  - *Post-hoc context compression*: e.g., summarization, short-term memory
    - Cons: abrupt the agent's work flow; does not utilize the task structure
  - *Multi-agent systems*: split tasks among agents.
    - Cons: mostly the roles of agents are handcrafted; unknown how to optimize end2end
- Can we use RL to train long-horizon LLM agents that can actively manage their context?
  - This work: **Context-Folding** mechanism and **FoldGRPO**

Multi-Turn LLM Agent: ReAct Agent [Yao et al., 2022]

- Let's begin with the De facto modeling of LLM agents, known as ReAct agent [Yao et al., 2022]
- Formulation: at step/turn t
  - a_t: action (reasoning & tool call) at step t
  - o_t: observation at step t
  - s_t: agent state, the key quantity to formulate an MDP for doing RL
- ReAct agent:
  - s_t: the working context; $a_t \sim \pi_\theta(\cdot \mid s_t)$, where $s_t = (q, a_1, o_1, \cdots, a_{t-1}, o_{t-1})$
  - Appends entire history to the working context, i.e., $s_{t+1} = \delta\big(\cdot\,; (s_t, a_t, o_t)\big)$

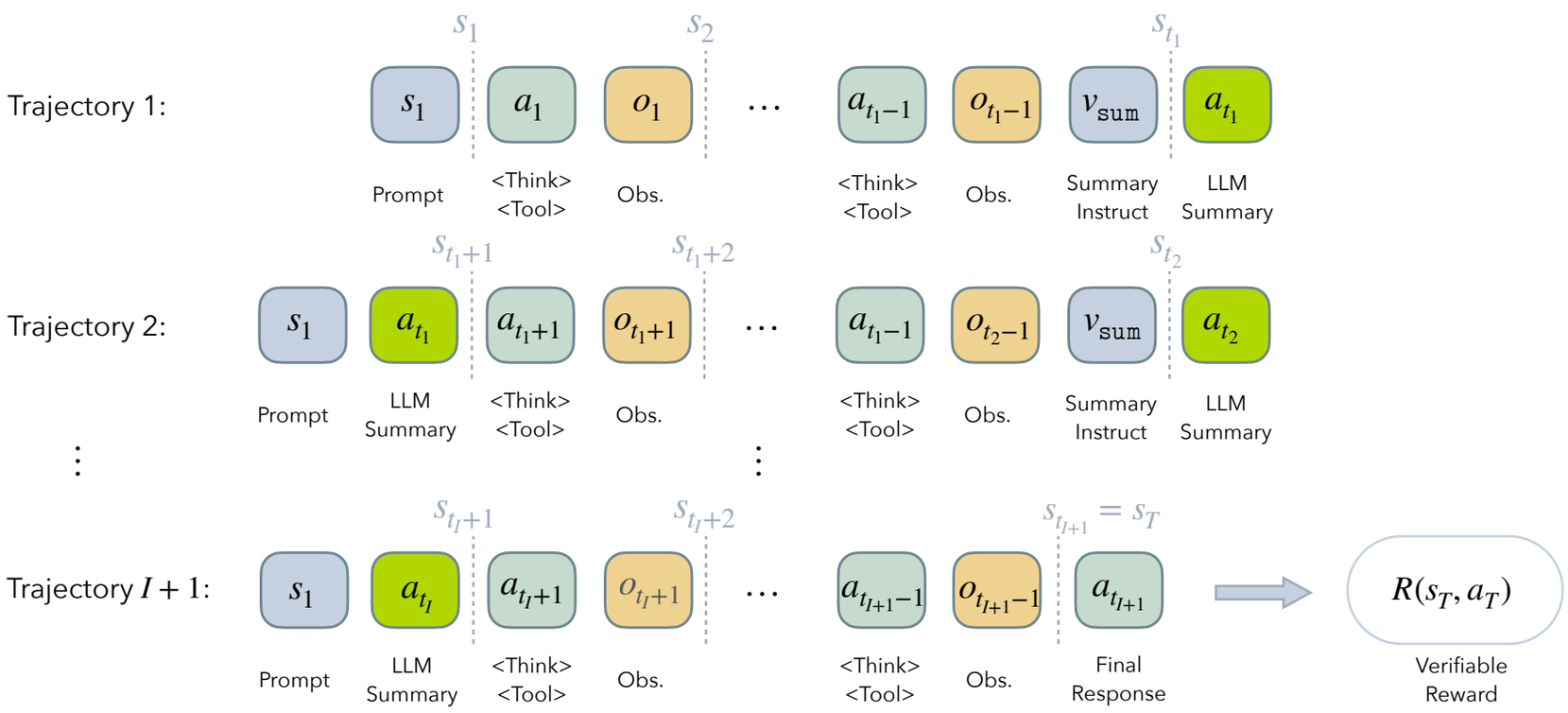


Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., & Cao, Y. (2022, October). React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Multi-Turn LLM Agent: Summary Agent [Lu et al., 2025]

- Summary agent:
  - s_t: the working context; $a_t \sim \pi_\theta(\cdot \mid s_t)$; Summarize the working context upon context limit

$$s_{t+1} := \begin{cases} (s_t, a_t, o_t) & \text{if } v_{\texttt{sum}} \notin s_t \text{ and } |(s_t, a_t, o_t)| < L, \\ (s_t, a_t, o_t, v_{\texttt{sum}}) & \text{if } v_{\texttt{sum}} \notin s_t \text{ and } |(s_t, a_t, o_t)| \geq L, \\ (s_1, a_t) & \text{if } v_{\texttt{sum}} \subseteq s_t\,. \end{cases}$$



Lu, M., Sun, W., Du, W., Ling, Z., Yao, X., Liu, K., & Chen, J. (2025). Scaling LLM Multi-turn RL with End-to-end Summarization-based Context Management. *arXiv preprint arXiv:2510.06727*.

Multi-Turn LLM Agent: Context-Folding Agent [Sun et al., 2025]

- Context-Folding agent:
  - Idea: let the agent itself decide its context
  - Method: maintain two different types of agent state: planning state (P) and execution state (E)
  - Two special actions: Branch and Return, to switch between P and E

- Branch (description, prompt)
  - Branch from main thread (planning state) to use a separate working context to complete a sub-task (execution state), i.e., P to E

- Return (message)
  - Fold the context generated in the branch (execution state) and return to the main thread (planning state), i.e., E to P

- Planning state (P): high-level reasoning, decompose the task, decide when to Branch

- Execution state (E): complete assigned token-intensive sub-tasks, Return upon completion

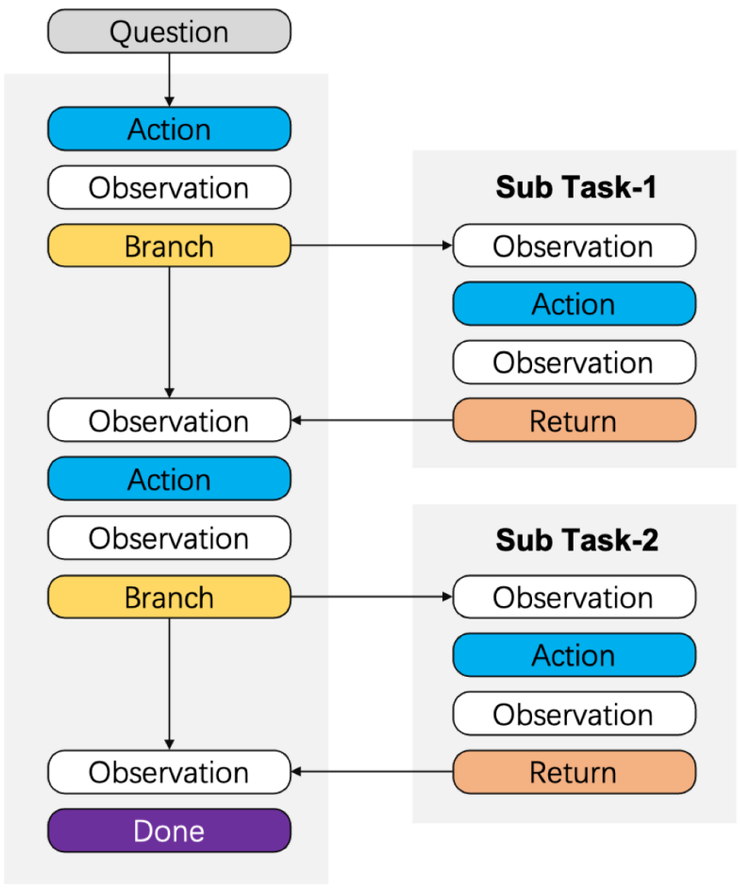
- Mathematically, $a_t \sim \pi_\theta(\cdot \mid s_t)$, where $s_t = (q, \mathcal{F}(\tau_{<t}))$ and $\mathcal{F}(\cdot)$ folds the previous E state history.

$$\mathcal{F}\big(a_1, o_1, a_2, \underbrace{o_2, a_3, o_3, a_4}_{\text{branch 1}}, o_4, a_5, \underbrace{o_5, a_6, o_6, a_7, o_7, a_8}_{\text{branch 2}}, o_8, a_9, o_9, a_{10}, o_{10}\big)$$
$$\rightarrow \big(a_1, o_1, a_2, o_4, a_5, o_8, a_9, o_9, a_{10}, o_{10}\big)$$

Sun, W., Lu, M., Ling, Z., Liu, K., Yao, X., Yang, Y., & Chen, J. (2025). Scaling Long-Horizon LLM Agent via Context-Folding. *arXiv preprint arXiv:2510.11967*.

Multi-Turn LLM Agent: Context-Folding Agent [Sun et al., 2025]

- Context-Folding agent:
  - Idea: let the agent itself decide its context
  - Method: maintain two different types of agent state: planning state (P) and execution state (E)
  - Two special actions: Branch and Return, to switch between P and E



  - At state E, uses the same prefix as corresponding P state
    - The context length in state P is compact
    - KV-cache friendly
  - When is Return called, KV-cache rolls back to the corresponding Branch position
  - We disable creating new branches in an active E state

Sun, W., Lu, M., Ling, Z., Liu, K., Yao, X., Yang, Y., & Chen, J. (2025). Scaling Long-Horizon LLM Agent via Context-Folding. *arXiv preprint arXiv:2510.11967*.

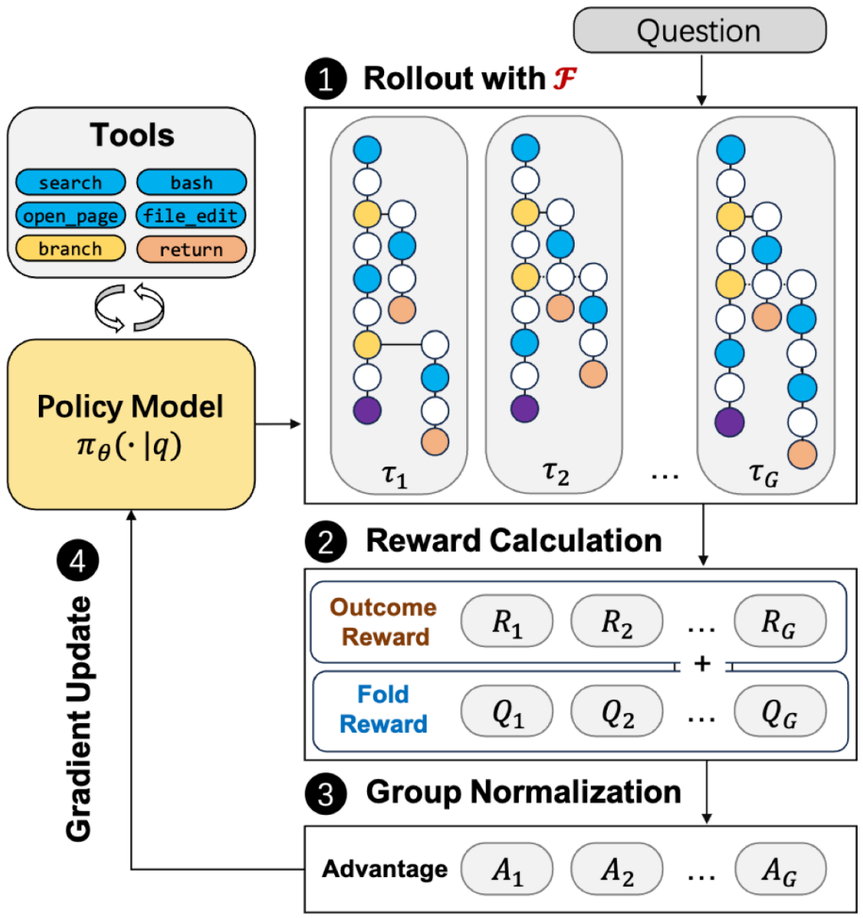
Multi-Turn LLM Agent: Context-Folding Agent [Sun et al., 2025]

- Context-Folding agent:
  - Idea: let the agent itself decide its context
  - Method: maintain two different types of agent state: planning state (P) and execution state (E)
  - Two special actions: Branch and Return, to switch between P and E
- Why context-folding?
  - Compact and efficient working context
  - Naturally preserves reasoning and task flow
  - Active management, ready for e2e RL to autonomously discover good agent state patterns

Sun, W., Lu, M., Ling, Z., Liu, K., Yao, X., Yang, Y., & Chen, J. (2025). Scaling Long-Horizon LLM Agent via Context-Folding. *arXiv preprint arXiv:2510.11967*.

Scaling Long-horizon Agent with RL: FoldGRPO

- We aim to advance the context-folding agent via RL. An overview:



- Learning objective (FoldGRPO):

$$\mathbb{E}_{\substack{q\sim\mathcal{D},\\ \{\tau_i\}_{i=1}^{G}\sim\pi_{\text{old}}(\cdot\,|q)}}\left[\frac{1}{\sum_{i=1}^{G}|\tau_i|}\sum_{i=1}^{G}\sum_{t=1}^{|\tau_i|}\min\left\{r_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}\big(r_{i,t}(\theta),1-\epsilon_{\text{low}},1+\epsilon_{\text{high}}\big)\hat{A}_{i,t}\right\}\right],$$

- Importance sampling ratio & Group advantage estimator

$$r_{i,t}(\theta)=\frac{\pi_\theta(\tau_{i,t}\mid q,\mathcal{F}(\tau_{i,<t}))}{\pi_{\theta_{\text{old}}}(\tau_{i,t}\mid q,\mathcal{F}(\tau_{i,<t}))}\cdot\mathbf{1}^{\text{LLM}}_{\tau_{i,t}},\quad \hat{A}_{i,t}=\frac{\text{clip}(R_i+Q_{i,t},0,1)-\text{mean}(\{R_i\}_{i=1}^{G})}{\text{std}(\{R_i\}_{i=1}^{G})}.$$

- Two key components of FoldGRPO:
  - Context-folding $\mathcal{F}(\cdot)$ during rollout and optimization
  - Process reward signal $Q_{i,t}$ to efficiently guide c-f behavior

Sun, W., Lu, M., Ling, Z., Liu, K., Yao, X., Yang, Y., & Chen, J. (2025). Scaling Long-Horizon LLM Agent via Context-Folding. *arXiv preprint arXiv:2510.11967*.

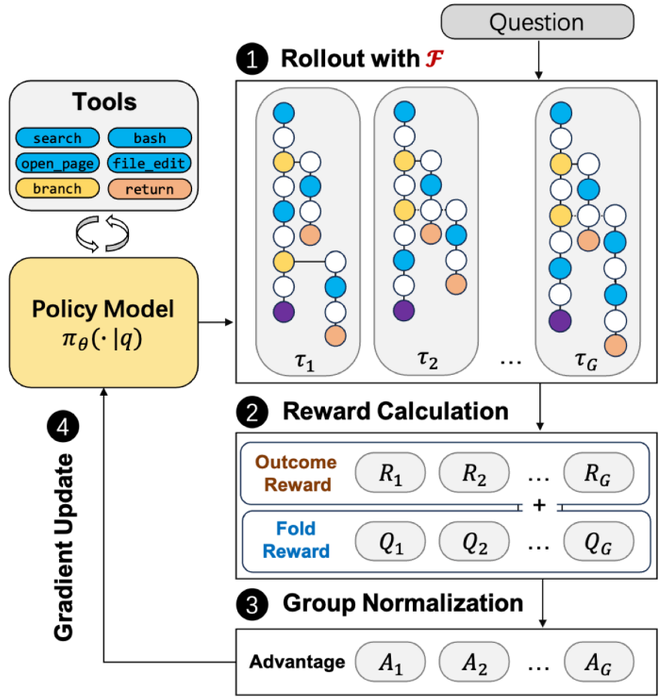
Key Component 1: Context-Folding

- The mechanism is as introduced previously. But how to implement and optimize in the RL pipeline?

- Rollout: each Branch action introduces a separate trajectory. Rollout i of task k results in $N_{k,i}$= 1+#Branch trajectories

  - The main thread (P states) is one trajectory with all the E states context folded

  - Each new E state is a separate trajectory with previous E states context folded



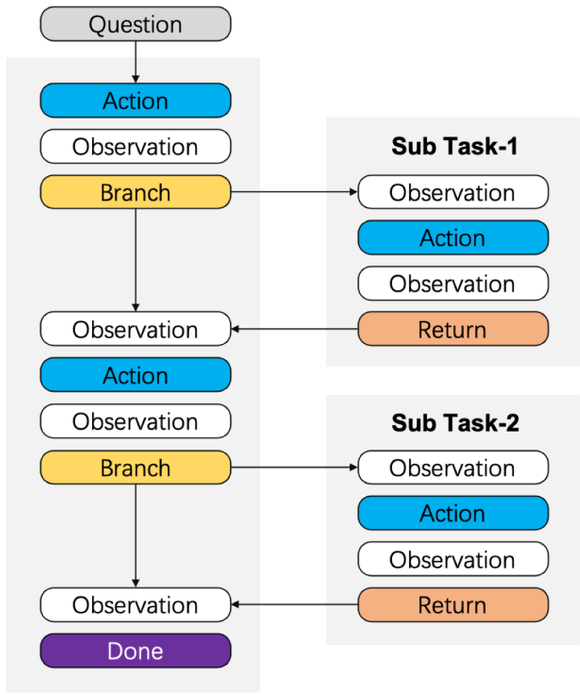
$$\mathbb{E}_{\substack{q\sim\mathcal{D},\\ \{\tau_i\}_{i=1}^{G}\sim\pi_{\text{old}}(\cdot|q)}}\left[\frac{1}{\sum_{i=1}^{G}|\tau_i|}\sum_{i=1}^{G}\sum_{t=1}^{|\tau_i|}\min\left\{r_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}\big(r_{i,t}(\theta),1-\epsilon_{\text{low}},1+\epsilon_{\text{high}}\big)\hat{A}_{i,t}\right\}\right],$$

$$r_{i,t}(\theta)=\frac{\pi_\theta(\tau_{i,t}\mid q,\mathcal{F}(\tau_{i,<t}))}{\pi_{\theta_{\text{old}}}(\tau_{i,t}\mid q,\mathcal{F}(\tau_{i,<t}))}\cdot\mathbf{1}^{\text{LLM}}_{\tau_{i,t}},\quad \hat{A}_{i,t}=\frac{\text{clip}(R_i+Q_{i,t},0,1)-\text{mean}(\{R_i\}_{i=1}^{G})}{\text{std}(\{R_i\}_{i=1}^{G})}.$$

- After rollout, the $N_{k,i}$ trajectory are returned to the training infra. Thus a rollout stage results in $N=\sum_{k=1}^{\text{batchsize}}\sum_{i=1}^{G}N_{k,i}$ trajectories

- The $N_{k,i}$ trajectory share the same reward $R_{k,i}$, and the group mean is averaged over G rollouts (instead of $\sum_{i=1}^{G}N_{k,i}$)

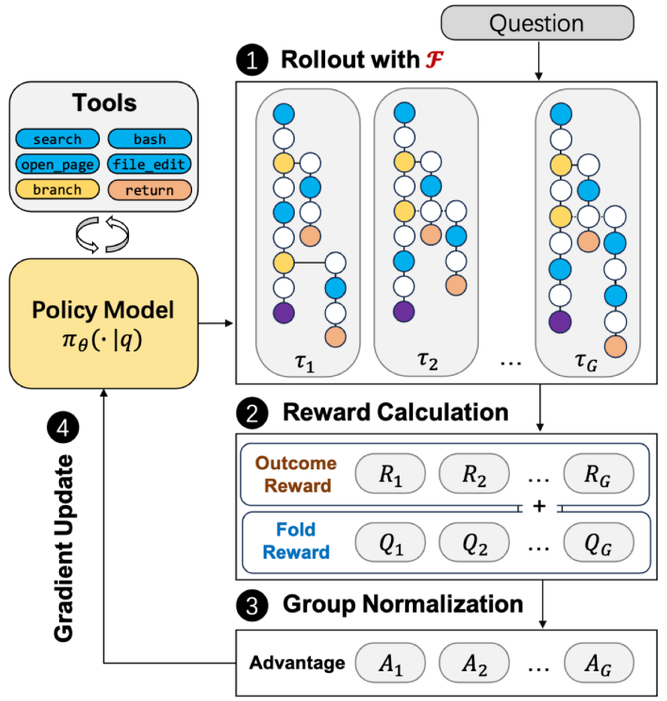
- We pad N with dummy samples to multipliers of mini_batchsize



Sun, W., Lu, M., Ling, Z., Liu, K., Yao, X., Yang, Y., & Chen, J. (2025). Scaling Long-Horizon LLM Agent via Context-Folding. *arXiv preprint arXiv:2510.11967*.

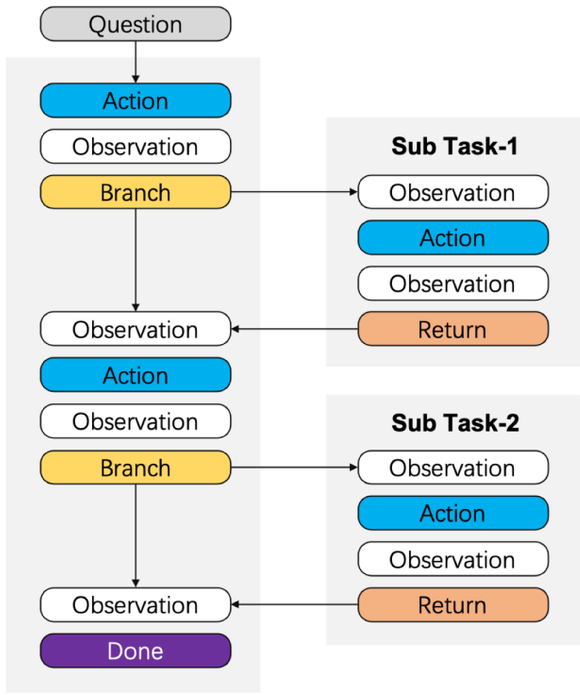
Key Component 2: Process Reward Signal

- At each token t of rollout i, we introduce signal $Q_{i,t}$
- Idea: $Q_{i,t}$ aims to penalize two bad behaviors: (i) unfolded token penalty; (ii) out-of-scope penalty.
  - Unfolded token penalty: $Q_{i,t} = -1$ to all the tokens in the main thread when len(main thread) > 0.5 * context limit
  - Out-of-scope penalty: $Q_{i,t} = -0.2$ to all the tokens in a branch to penalize out of scope behavior
  - We also apply $Q_{i,t} = -1$ to all the tokens in a failed tool call turn
  - In all other cases $Q_{i,t} = 0$



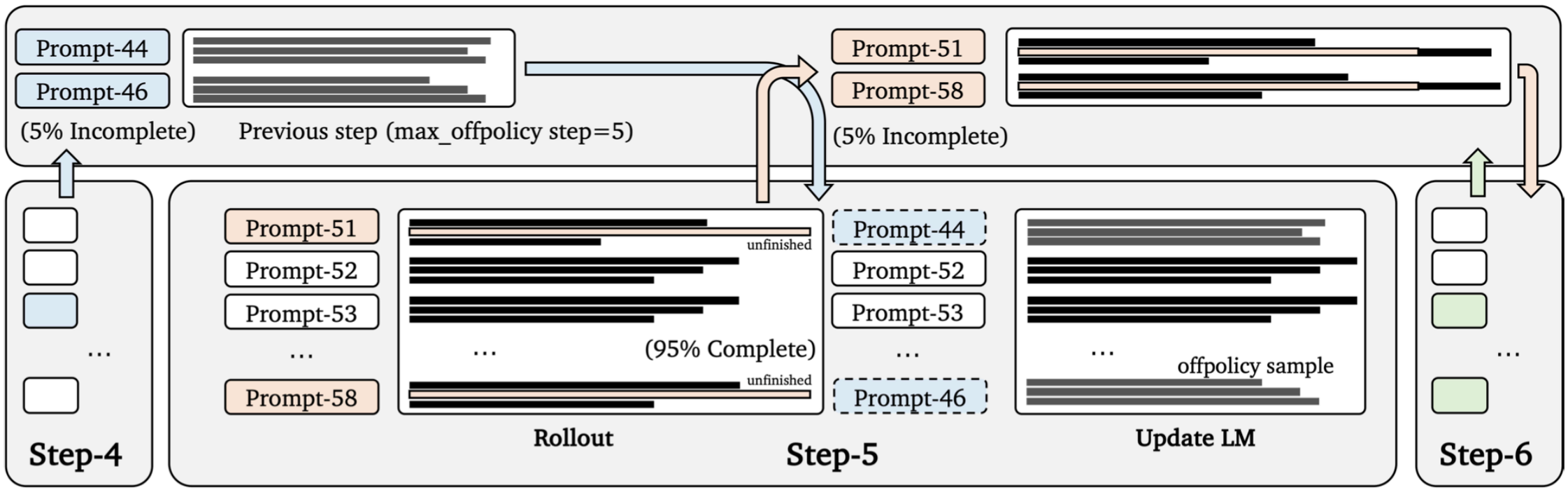
$$\mathbb{E}_{\substack{q\sim\mathcal{D},\\ \{\tau_i\}_{i=1}^{G}\sim\pi_{\text{old}}(\cdot|q)}}\left[\frac{1}{\sum_{i=1}^{G}|\tau_i|}\sum_{i=1}^{G}\sum_{t=1}^{|\tau_i|}\min\left\{r_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}\big(r_{i,t}(\theta),1-\epsilon_{\text{low}},1+\epsilon_{\text{high}}\big)\hat{A}_{i,t}\right\}\right],$$

$$r_{i,t}(\theta)=\frac{\pi_\theta(\tau_{i,t}\mid q,\mathcal{F}(\tau_{i,<t}))}{\pi_{\theta_{\text{old}}}(\tau_{i,t}\mid q,\mathcal{F}(\tau_{i,<t}))}\cdot\mathbf{1}^{\text{LLM}}_{\tau_{i,t}},\quad \hat{A}_{i,t}=\frac{\text{clip}(R_i+Q_{i,t},0,1)-\text{mean}(\{R_i\}_{i=1}^{G})}{\text{std}(\{R_i\}_{i=1}^{G})}.$$



Sun, W., Lu, M., Ling, Z., Liu, K., Yao, X., Yang, Y., & Chen, J. (2025). Scaling Long-Horizon LLM Agent via Context-Folding. *arXiv preprint arXiv:2510.11967*.

Other Components: Standalone Rollout Node



- The main rollout process stops once it completes 95% of the tasks in the batch
- Remaining jobs are handled by the standalone process
- At each optimization step, the data come from (i) 95% of current batch; (ii) completed rollout from previous step
- We setup a maximum off-policy step 5

Sun, W., Lu, M., Ling, Z., Liu, K., Yao, X., Yang, Y., & Chen, J. (2025). Scaling Long-Horizon LLM Agent via Context-Folding. *arXiv preprint arXiv:2510.11967*.

Experiments: Setups

- Base LLM: Seed-OSS-36B-Instruct
- Tasks: BrowseComp, SWE-Bench
  - Uses corpus from BrowseComp-Plus: 680 instances for training and 150 hold-out instances for val.
  - Use SWE-Bench Verified as evaluation set. Collected a subset of size 740 from SWE-Gym and SWE-Rebench as the training set.
- Methods compared:
  - ReAct Agent
  - Summary Agent
  - Context-Folding Agent

Sun, W., Lu, M., Ling, Z., Liu, K., Yao, X., Yang, Y., & Chen, J. (2025). Scaling Long-Horizon LLM Agent via Context-Folding. *arXiv preprint arXiv:2510.11967*.

Experiments: Main Results

Model	Peak Length	Max #Token	BrowseComp-Plus Pass@1	BrowseComp-Plus Tool Calls	SWE-Bench Verified Pass@1	SWE-Bench Verified Tool Calls
ReAct Agent with 100B+ LLM						
GPT-5	327K	327K	0.793	14.2	0.718	42.6
GPT-4.1	327K	327K	0.640	5.6	0.486	28.7
DeepSeek-V3.1	327K	327K	0.613	10.6	0.610	53.2
GLM-4.5-Air	327K	327K	0.566	11.1	0.576	51.2
Qwen3-235B-A22B	327K	327K	0.560	12.8	0.344	32.1
ReAct Agent						
Seed-OSS-36B	32K	32K	0.286 (-19.2)	3.8	0.436 (-11.6)	25.8
+ RL (GRPO)	32K	32K	0.446 (-3.2)	5.5	0.480 (-7.2)	27.8
Seed-OSS-36B$^{\psi}$	327K	327K	0.478 (+0.0)	10.8	0.552 (+0.0)	49.5
+ RL (GRPO)	327K	327K	0.540 (+6.2)	10.2	0.574 (+2.2)	55.4
Summary Agent						
Seed-OSS-36B	32K	32K × 10	0.386 (-9.2)	17.4	0.488 (-6.4)	77.0
+ RL (GRPO)	32K	32K × 10	0.527 (+4.9)	18.0	0.550 (-0.2)	74.9
Folding Agent (Ours)						
Seed-OSS-36B	32K	32K × 10	0.420 (-5.8)	12.9	0.492 (-6.0)	72.8
+ RL (GRPO)	32K	32K × 10	0.567 (+8.9)	16.0	0.564 (+1.2)	79.5
+ RL (FoldGRPO)	32K	32K × 10	**0.620** (+14.2)	19.2	**0.580** (+2.8)	96.5

Sun, W., Lu, M., Ling, Z., Liu, K., Yao, X., Yang, Y., & Chen, J. (2025). Scaling Long-Horizon LLM Agent via Context-Folding. *arXiv preprint arXiv:2510.11967*.

Experiments: Further Studies 1

	BrowseComp-Plus				SWE-Bench Verified			
	Finish	Main Len	Scope	# Branch	Finish	Main Len	Scope	# Branch
Folding Agent (Seed-OSS-36B)	0.806	12,195	0.774	3.51	0.781	47,475	0.473	3.05
+ RL (GRPO)	0.738	22,285	0.762	3.88	0.612	48,908	0.419	3.80
+ RL (FoldGRPO)	0.935	7,752	0.895	4.98	0.962	8,885	0.754	5.90

Table 2 Model behavior statistics of different optimization methods. FoldGRPO encourages focused branching and condensed main context, boosting both scope accuracy and finish rate.

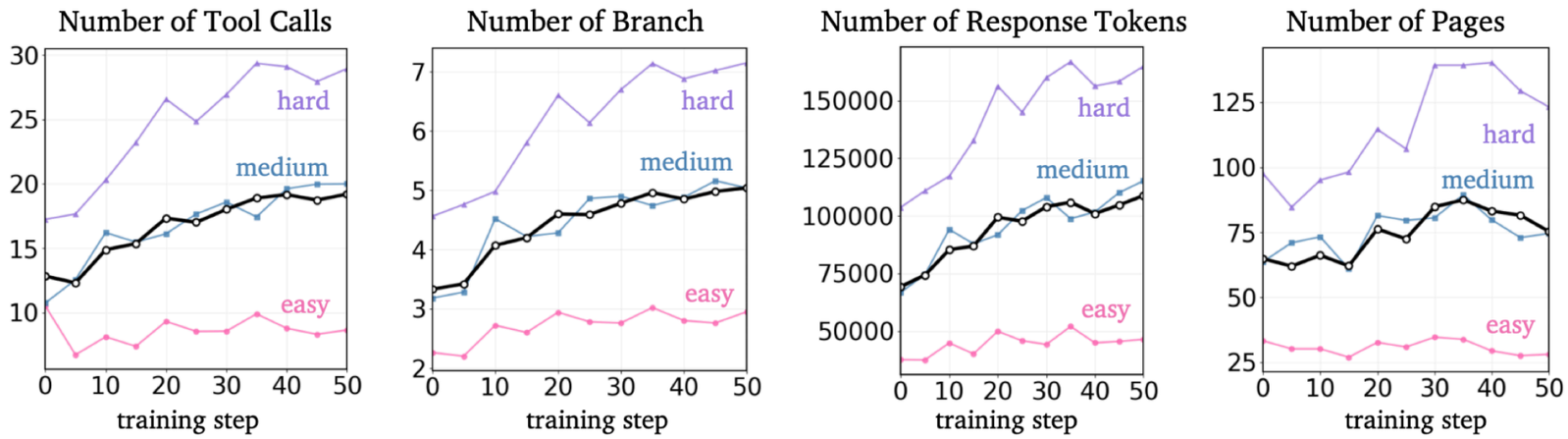


Figure 4 With RL training, we observe an increase in the number of tool calls, branching behavior, total number of tokens, and the number of searched pages.

Sun, W., Lu, M., Ling, Z., Liu, K., Yao, X., Yang, Y., & Chen, J. (2025). Scaling Long-Horizon LLM Agent via Context-Folding. *arXiv preprint arXiv:2510.11967*.

Experiments: Further Studies 2

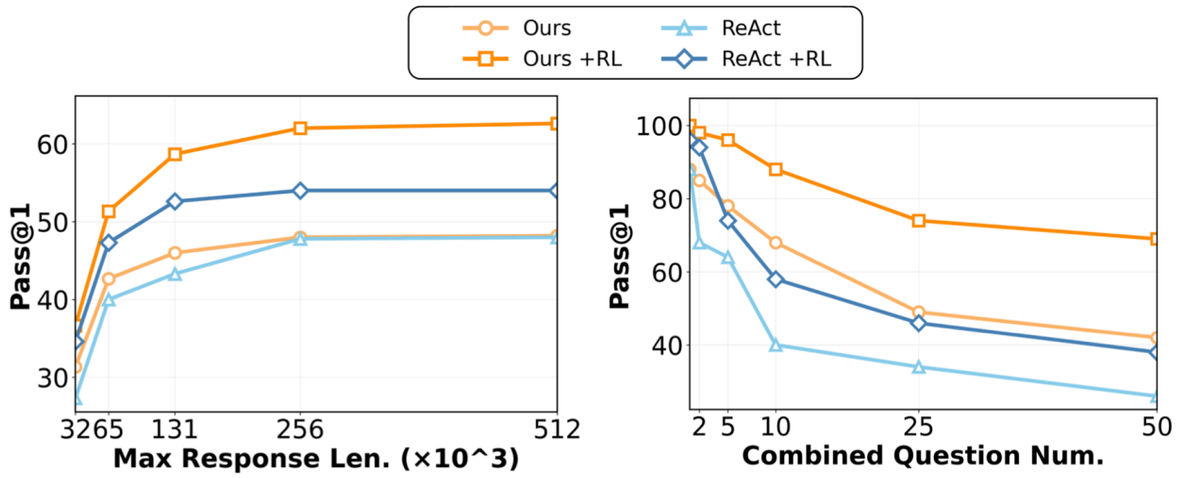


Figure 5 Left: Pass@1 vs. agent max context length. **Right:** Pass@1 vs. number of combined questions. Multiple easy questions are combined into a single harder question to increase problem complexity; a higher number of combined questions indicates more required actions and a longer context to answer them correctly. See Section 4.4.2 for details.

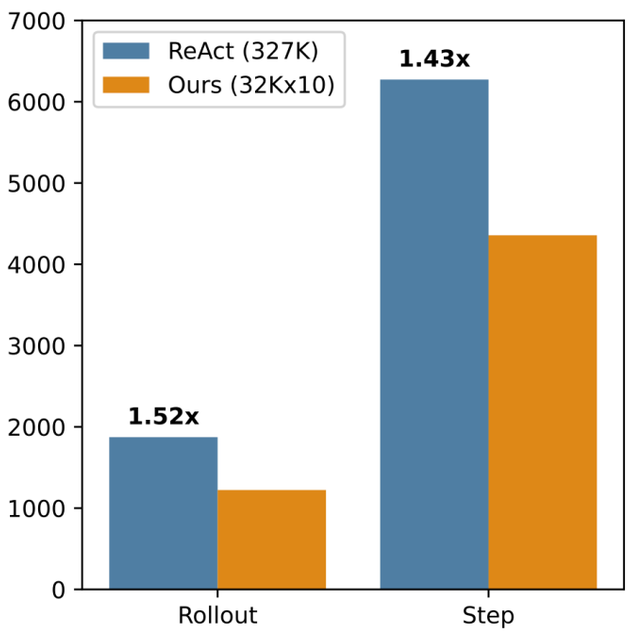


Figure 8 Training time cost. The figure shows the stepwise average time for rollout and for each training step.

Sun, W., Lu, M., Ling, Z., Liu, K., Yao, X., Yang, Y., & Chen, J. (2025). Scaling Long-Horizon LLM Agent via Context-Folding. *arXiv preprint arXiv:2510.11967*.

Experiments: Further Studies 3

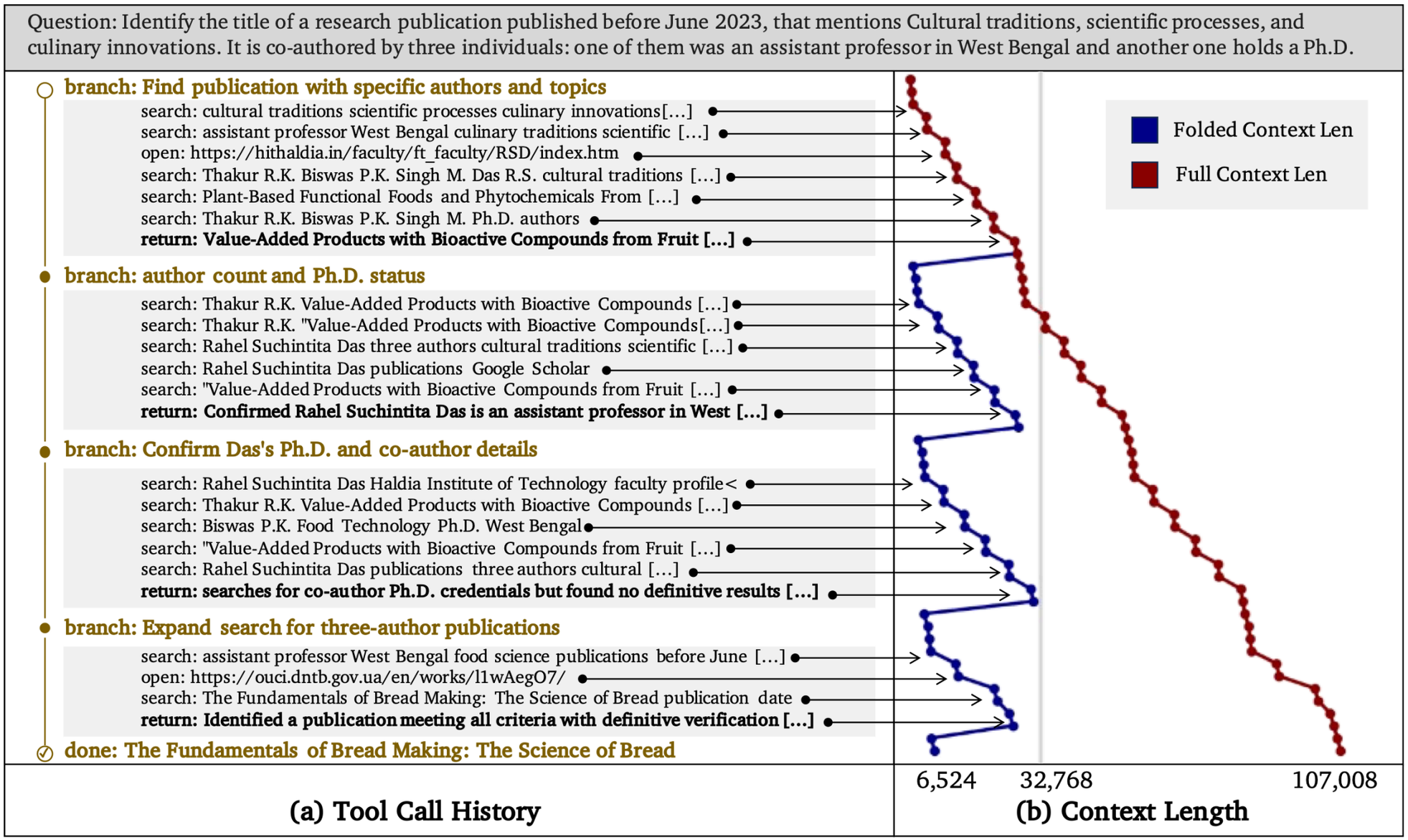


Figure 7 Example of an agent's tool call history and context length. on BrowseComp-Plus.

Sun, W., Lu, M., Ling, Z., Liu, K., Yao, X., Yang, Y., & Chen, J. (2025). Scaling Long-Horizon LLM Agent via Context-Folding. *arXiv preprint arXiv:2510.11967*.

Future Works

- Hierarchical Context-Folding Mechanisms
- Training with Proximal Policy Optimization (PPO)
- …

Thank you for your attention!