In-Context Teacher–Student Guidance for Open-Weight Browser Agents

Wéwerthon Silva Cardoso, Ricardo Marcacini

Universidade de São Paulo, Brazil
{wewerthon.cardoso,ricardo.marcacini}@usp.br

Abstract. Browser agents based on large language models can automate web tasks, but deployment remains constrained by repeated model calls, monetary cost, and privacy risks. In practical settings, agents often process page states that may expose credentials, personal data, internal records, business documents, or authenticated session information. Sending these states and interaction histories to external model APIs can increase the risk of data leakage, unauthorized retention, or unintended disclosure. This paper investigates whether reasoning traces from a stronger teacher model can improve open-weight browser agents without fine-tuning. We propose *In-Context Learning Guidance* (ICLG), a training-free teacher–student protocol in which the teacher solves representative browser tasks offline and its reasoning traces are converted into reusable prompt-level guidance for similar tasks. The student then uses this guidance during execution while reading the concrete target values from the current browser instance. We evaluate ICLG on MiniWoB++, using GPT-5.1 as the teacher and ten open-weight student variants. ICLG increases solved student-task pairs from 208 to 271 and improves 7 out of 10 students. Ablation results suggest that these gains are not due only to retrying or generic prompt refinement. Overall, the results show that reusable teacher guidance can improve lower-cost browser agents and reduce dependence on proprietary teacher APIs during deployment.

CCS Concepts: • **Computing methodologies** → **Intelligent agents**.

Keywords: browser agents, teacher-student guidance, knowledge transfer, web task automation

1. INTRODUCTION

Large language models (LLMs) are used as controllers for agents that reason, plan, and interact with tools [Wei et al. 2022; Yao et al. 2023; Schick et al. 2023]. Browser agents are relevant because workflows are executed through web interfaces, including form completion, search, dashboard exploration, administrative processing, and data entry. Unlike static question answering, browser automation requires sequential decisions over changing page states. The agent must observe the interface, interpret the instruction, select valid actions, and revise behavior after each transition.

Recent browser-agent benchmarks range from controlled interaction suites such as MiniWoB++ to more realistic environments such as WebArena and WorkArena [Liu et al. 2018; Zhou et al. 2024; Drouin et al. 2024]. Existing methods improve browser interaction through representation, grounding, planning, and learning [Lai et al. 2024; Yang et al. 2025; He et al. 2024; Qi et al. 2025; Shen et al. 2024; Zhang et al. 2025; Gu et al. 2025].

Despite these advances, deployment remains challenging. Strong proprietary models often achieve higher success rates, but browser tasks require many calls because execution alternates among observation, reasoning, and action selection. This increases token usage and monetary cost [Chen et al. 2024; Xiao et al. 2025]. Browser automation may also involve authenticated sessions, internal dashboards, personal data, credentials, or organizational documents. Sending page states and interaction histories to external APIs can increase privacy and security exposure [He et al. 2025; Yan et al. 2025].

Open-weight or locally deployed models reduce this dependence, but often underperform in agentic environments due to weak state tracking, incorrect element grounding, invalid actions, or premature termination [Ma et al. 2024; Liu et al. 2024].

This paper investigates whether a stronger model can improve a weaker browser agent without being called during private deployment. Existing alternatives do not directly address this setting. Routing and cascading still keep the stronger model in the execution loop [Chen et al. 2024; Ong et al. 2025; Dekoninck et al. 2025]. Distillation can transfer behavior to smaller models, but usually requires training data and parameter updates [Hinton et al. 2015; Shridhar et al. 2023; Magister et al. 2023; Kang et al. 2025]. Prompting methods such as Chain-of-Thought and ReAct improve reasoning during execution, but do not provide task-specific reasoning generated by a stronger teacher [Wei et al. 2022; Yao et al. 2023].

We propose *In-Context Learning Guidance* (ICLG), a training-free protocol for improving browser agents through teacher-generated reasoning traces. In ICLG, a high-capacity teacher solves selected tasks in an offline guidance-generation phase. Its natural-language reasoning traces are then formatted as prompt-level guidance for an open-weight student model. The student parameters remain unchanged, and the teacher is not used during deployment-time browser execution.

We evaluate ICLG on MiniWoB++ using one teacher model and ten open-weight student variants. The benchmark provides a controlled environment for comparing models and prompt conditions over the same tasks. The contributions are: (i) a training-free teacher-guided protocol for browser agents based on natural-language reasoning traces; (ii) an empirical evaluation showing that task-specific teacher guidance improves several open-weight students without parameter updates; and (iii) ablation and cost analyses discussing when offline teacher guidance can support lower-cost and privacy-aware browser automation.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 presents the ICLG protocol. Section 4 covers the experiments and results. Section 5 concludes with limitations and future work.

2. RELATED WORK

In context learning allows LLMs to adapt their behavior from prompt information without parameter updates [Dong et al. 2024]. This property is relevant for agents because decisions depend on the task instruction, the current environment state, and the interaction history. Chain of Thought prompting improves multi step inference through intermediate reasoning [Wei et al. 2022], while ReAct interleaves reasoning and actions in interactive environments [Yao et al. 2023]. Other methods use feedback or experience as natural language guidance, including Reflexion [Shinn et al. 2023], Self Refine [Madaan et al. 2023], ExpeL [Zhao et al. 2023], and AdaPlanner [Sun et al. 2023]. ICLG follows this inference time perspective and uses task specific reasoning traces produced by a stronger teacher model as prompt guidance for a lower cost student model.

Browser agents apply LLM based agents to web interfaces. Each step requires observing the page, interpreting the instruction, selecting relevant elements, executing an action, and updating the plan from the resulting state. Prior work improves this loop through page representation, visual grounding, learning, and planning. AutoWebGLM adapts HTML representations and training strategies for compact web agents [Lai et al. 2024]. AgentOccam shows that simple agent designs can be competitive when observation and action spaces are aligned with LLM capabilities [Yang et al. 2025]. WebVoyager uses large multimodal models for end to end web navigation [He et al. 2024], SeeAct analyzes visual grounding for element selection [Zheng et al. 2024], and BrowserAgent studies human inspired browsing actions [Zhang et al. 2025]. WebRL, ScribeAgent, and WebDreamer improve agents through reinforcement learning, workflow supervision, or future state reasoning [Qi et al. 2025; Shen et al. 2024; Gu et al. 2025]. ICLG focuses on open weight student browser agents guided through

prompting alone, with no additional training during evaluation.

Teacher student learning is commonly implemented as knowledge distillation, where a larger teacher provides labels, logits, rationales, plans, or trajectories to train a smaller student model [Hinton et al. 2015]. Recent work extends this idea to reasoning and agent behavior with distilled explanations or interaction traces [Shridhar et al. 2023; Magister et al. 2023; Kang et al. 2025]. These methods can be effective, although they usually require training data, optimization, and validation of the resulting student. Model cascading reduces cost through routing between weaker and stronger models according to expected difficulty or quality requirements [Chen et al. 2024; Ong et al. 2025; Dekoninck et al. 2025]. These methods assume that the stronger model remains available during inference. ICLG targets private execution scenarios in which the teacher generates static reasoning guidance before deployment and only the student interacts with the browser.

3. ICLG BROWSER AGENTS

This section presents the proposed ICLG framework. Section 3.1 defines the browser-agent setting and notation, Section 3.2 describes the baseline condition, and Section 3.3 presents the two-phase ICLG procedure.

3.1 Problem Definition

In-Context Learning Guidance (ICLG) refers to the use of task-specific reasoning traces, generated by a high-capacity teacher model, as prompt-level guidance for a lower-cost student browser agent. The student model is not fine-tuned and its parameters remain unchanged. Instead, the teacher's reasoning is injected into the task prompt and used only as in-context information during execution.

Let $\mathcal{T} = \{\tau_1, \ldots, \tau_N\}$ be a set of browser-based tasks. Each task τ is defined by a natural-language instruction u_τ, an initial browser state o_0^τ, and a task-specific evaluator R_τ. At step t, the agent observes the current state o_t^τ, selects an action $a_t \in \mathcal{A}$, and the browser transitions to $o_{t+1}^\tau = F_\tau(o_t^\tau, a_t)$. The available interaction history is $h_t^\tau = (o_0^\tau, a_0, o_1^\tau, a_1, \ldots, o_{t-1}^\tau, a_{t-1}, o_t^\tau)$.

A browser agent is controlled by a language model M, which induces an action policy $a_t \sim \pi_M(a \mid q, u_\tau, h_t^\tau)$ given a prompt q, the task instruction, and the interaction history. Executing M on task τ produces a trajectory $\Gamma_{M,\tau}$, scored as $s_{M,\tau} = R_\tau(\Gamma_{M,\tau})$. A task is considered solved when $s_{M,\tau} \geq \delta$. In our experiments, $\delta = 0.5$, following the evaluation criterion adopted in the experimental setup.

3.2 Baseline Condition

In the baseline condition, the student model M_S receives only the original task prompt $q_\tau^B = \text{Prompt}(u_\tau, e_\tau)$, where e_τ contains the environment access information required to start the task, such as the target browser page. The baseline prompt was: **"`Go to {url}. Solve the task shown on the page only once following the instructions provided.`"** The baseline score is $s_{M_S,\tau}^B = R_\tau(\Gamma_{M_S,\tau}^B)$, where $\Gamma_{M_S,\tau}^B$ is the trajectory produced under q_τ^B.

We define the set of difficult tasks for a student model as $\mathcal{D}(M_S) = \{\tau \in \mathcal{T} \mid s_{M_S,\tau}^B < \delta\}$. This set identifies tasks for which additional guidance is most relevant. In deployment, the same procedure can be applied to public, synthetic, sanitized, or validation tasks before the student is used in private browser sessions.

3.3 In-Context Learning Guidance

ICLG uses a high-capacity teacher model M_T to generate task-specific guidance before deployment. Unlike knowledge distillation, the student is not trained on teacher outputs and its parameters remain unchanged. Transfer occurs only through the prompt.

For each selected task τ, the teacher executes the task and produces a natural-language reasoning trace $Z_\tau^T = (r_0^T, r_1^T, \ldots, r_L^T)$, where each r_i^T is a reasoning step generated during teacher execution. The default ICLG prompt uses only reasoning fields. Low-level teacher actions are not included, so the trace provides strategic guidance rather than an exact action demonstration.

The trace is converted into a textual guidance block $G_\tau^T = \phi(Z_\tau^T)$ and concatenated with the baseline prompt, yielding $q_\tau^G = q_\tau^B \oplus G_\tau^T$. The student then acts according to $a_t^S \sim \pi_{M_S}(a \mid q_\tau^G, u_\tau, h_t^\tau)$. An example of an ICLG prompt is provided in APPENDIX A.1. During deployment, browser execution is performed only by the student. The teacher is used only during the guidance-generation phase, before the private execution environment is accessed.

4. EXPERIMENTAL EVALUATION

This section presents the experimental evaluation of ICLG. Section 4.1 describes the benchmark and task environment, Section 4.2 details the experimental setup, Section 4.3 reports and discusses the results and ablations, and Section 4.4 analyzes deployment cost and privacy.

4.1 Benchmark and Task Environment

Several benchmarks evaluate browser agents under different levels of realism and control [Zhou et al. 2024; Deng et al. 2023; Yao et al. 2022; Drouin et al. 2024; de Chezelles et al. 2025]. MiniWoB++ contains 130 controlled web-interaction tasks covering clicking, form filling, navigation, selection, visual reasoning, and sequential manipulation [Liu et al. 2018]. Its stable interfaces and local execution allow us to isolate the effect of teacher guidance from website updates, network instability, and dynamic content. We served all tasks locally using a Python HTTP server.

4.2 Evaluation Setup

We evaluated GPT-5.1 as the teacher model and ten open-weight student variants: Gemma-3-4B, Gemma-3n-e4B, Llama-3.1-8B, Phi-4, Mistral-Nemo, Ministral-3B-2512, Ministral-8B-2512, Nemotron-Nano-9B-V2, GPT-OSS-20B, and Qwen3.5-9B.

The evaluation considers three conditions. The baseline evaluates each student without teacher guidance. The topline evaluates the teacher under the same browser-agent protocol and is used only as a reference. The ICLG condition evaluates each student with task-specific teacher reasoning appended to the prompt.

The experimental protocol is as follows. First, each student is evaluated on the 130 MiniWoB++ tasks using the baseline prompt. Second, the teacher is executed in the offline guidance-generation phase to produce reasoning traces for the selected tasks. Third, the traces are converted into guidance blocks and appended to the student prompt. Fourth, each student is evaluated again under the ICLG condition. The teacher is not called during student execution in the ICLG condition. Each browser-agent execution was limited to 5 minutes and 25 interaction steps to prevent infinite loops and ensure comparability. A task was considered solved when its final benchmark score was at least 0.5. For a model M and prompt condition q, success rate is defined as

$$\mathrm{Acc}(M, q) = |\mathcal{T}|^{-1} \sum_{\tau \in \mathcal{T}} \mathrm{success}(M, \tau \mid q). \tag{1}$$

Since each student is evaluated under baseline and ICLG conditions, we also report the number of solved student-task pairs:

$$S(q) = \sum_{M_S} \sum_{\tau \in \mathcal{T}} \mathrm{success}(M_S, \tau \mid q). \tag{2}$$

4.3 Results and Discussion

Table I reports the number of tasks solved by each model. GPT-5.1 solved 91 out of 130 tasks, corresponding to a success rate of 70.0%. This result is a topline reference for the tested execution protocol, not a theoretical upper bound.

Table I. Overall performance on MiniWoB++.

Model	**Baseline**	**ICLG**	**Gain**
GPT-5.1, teacher topline	91 (70.0%)	–	–
Gemma-3-4B	0 (0.0%)	4 (3.1%)	+4
Gemma-3n-e4B	0 (0.0%)	1 (0.8%)	+1
GPT-OSS-20B	44 (33.8%)	74 (56.9%)	+30
Llama-3.1-8B	6 (4.6%)	16 (12.3%)	+10
Ministral-3B-2512	13 (10.0%)	12 (9.2%)	-1
Ministral-8B-2512	41 (31.5%)	40 (30.8%)	-1
Mistral-Nemo	18 (13.8%)	24 (18.5%)	+6
Nemotron-Nano-9B-V2	23 (17.7%)	26 (20.0%)	+3
Phi-4	2 (1.5%)	1 (0.8%)	-1
Qwen3.5-9B	61 (46.9%)	73 (56.2%)	+12

Across the ten student variants, ICLG increased the total number of solved student-task pairs from 208 to 271. According to Equation 2, this corresponds to 63 additional successful executions. In total, 7 out of 10 student variants improved with teacher guidance.

The largest gains occurred for GPT-OSS-20B (+30), Qwen3.5-9B (+12), and Llama-3.1-8B (+10). Smaller improvements occurred for four additional models, while three students regressed by one task, suggesting that teacher traces can increase prompt complexity or induce action plans that some students cannot execute reliably. Thus, ICLG appears capacity-dependent rather than universally beneficial.

To limit experimental cost, we evaluate the ablations on the three students with the largest ICLG gains: GPT-OSS-20B, Qwen3.5-9B, and Llama-3.1-8B. The *retry* condition reruns the student with the baseline prompt, while the *refined prompt* condition uses stronger fixed instructions without teacher reasoning. Table II reports the number of solved MiniWoB++ tasks. An example of the *refined* prompt is in APPENDIX A.2.

Table II. Ablation results on MiniWoB++.

Model	**Base.**	**Retry**	**Refined**	**ICLG**
GPT-OSS-20B	44	49	62	74
Qwen3.5-9B	61	51	54	73
Llama-3.1-8B	6	4	8	16

Retry does not explain the ICLG gains, since it improves only GPT-OSS-20B slightly and reduces performance for Qwen3.5-9B and Llama-3.1-8B. The refined prompt helps two models, but remains below ICLG in all cases. These results indicate that task-specific teacher reasoning provides information beyond repeated execution and generic prompting.

4.4 Cost and Privacy Aware Deployment

ICLG is not intended to reduce cost for a single isolated run, since teacher guidance also requires inference. Its deployment value comes from generating teacher traces offline on public, synthetic, sanitized, or validation tasks, and then executing private sessions with the student only.

Table III shows that ICLG does not uniformly reduce student workload. It reduces Llama-3.1-8B from 114.3k tokens and 22.5 calls to 88.6k tokens and 13.1 calls per solved task, but increases total tokens for Qwen3.5-9B and GPT-OSS-20B. The main advantage is therefore deployment-oriented: guided student execution is much cheaper than teacher execution per solved task, with reductions of 38.1× for Llama-3.1-8B, 13.5× for Qwen3.5-9B, and 26.1× for GPT-OSS-20B. From a privacy perspective, the teacher is not called during private browser execution, so private browser states are not sent to the teacher model. This benefit is strongest when the student runs in a self-hosted environment.

Table III. Average workload and reference API cost per solved task.

Model	Cond.	Input	Output	Total	Calls	Cost
Llama-3.1-8B	Base.	109.4k	5.0k	114.3k	22.5	$0.00234
Llama-3.1-8B	ICLG	85.3k	3.2k	88.6k	13.1	$0.00180
Qwen3.5-9B	Base.	32.9k	4.6k	37.5k	7.3	$0.00398
Qwen3.5-9B	ICLG	43.9k	4.6k	48.5k	7.4	$0.00508
GPT-OSS-20B	Base.	40.6k	10.1k	50.7k	9.7	$0.00259
GPT-OSS-20B	ICLG	45.9k	9.3k	55.3k	7.9	$0.00264
GPT-5.1	Teacher	29.5k	3.2k	32.7k	5.2	$0.06873

5. CONCLUSION AND FUTURE WORK

This article proposed *In-Context Learning Guidance* (ICLG), a training-free teacher–student protocol for deployment-aware browser agents. The central idea is to use a teacher model in an offline guidance-generation phase, rather than as the default deployment-time executor. The teacher produces task-specific reasoning traces for selected tasks, and these traces are reused as in-context guidance for open-weight student models. The student parameters remain unchanged, and private browser execution can be performed without calling the teacher.

The results on MiniWoB++ show that ICLG can improve lower-cost browser agents without fine-tuning. Across ten student variants, the number of solved student-task pairs increased from 208 to 271, with improvements in 7 out of 10 models. Ablations indicate that these gains are not explained only by retrying the same prompt or by adding stronger generic instructions.

These findings position ICLG between two common deployment extremes: always using an expensive proprietary model and relying only on a weaker local model. Limitations include single-run evaluation, MiniWoB++'s limited realism and performance dependence on the student's ability to interpret teacher guidance. Future work will investigate more compact guidance traces, automatic trace selection, repeated-run evaluation, and experiments in more realistic browser environments.

Acknowledgments

The first author thanks Instituto de Pesquisas Eldorado for providing institutional support and time to conduct and present this research. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

APPENDIX A. PROMPT AND GUIDANCE EXAMPLES

APPENDIX A.1 ICLG Reasoning-Guided Prompt Excerpt

```
1. Go to {url}. Solve the task shown on the page only once following the instructions provided.
2.  Teacher guidance for this MiniWoB++ task type: Open the given MiniWoB++ URL and wait until the task interface is
visible. If a START button is shown, click it once and then read the task instruction carefully.
3. Inspect the current page state and identify the required interaction, such as clicking, selecting checkboxes,
typing, choosing an option, or submitting a form. Use only the target values displayed in the current task instance.
4. For selection tasks, choose exactly the elements requested by the visible instruction. Then click the appropriate
confirmation control, such as Submit, OK, or Done.
5. Avoid repeating actions that do not change the page state. Call done() only after the visible task objective
appears completed.
6. This guidance describes the strategy only. Exact target values must always be read from the current MiniWoB++ page.
```

APPENDIX A.2 Refined Fixed Prompt Excerpt

```
Go to {url}. Solve the task shown on the page only once following the instructions provided.

MiniWoB Benchmark Task Instructions

WORKFLOW:
1. Navigate to the given MiniWoB task URL if the page is not already open.
2. If a START button is present and the task has not started yet, click START exactly once.
3. Carefully read the task instruction and identify the exact objective before acting.
[...]
10. Avoid repeating an action if it does not change the page state.
11. Do not call done() until the task objective appears to be completed.
12. When the task objective is completed, call done().
```

REFERENCES

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

de Chezelles, T. L. S., Gasse, M., Lacoste, A., Caccia, M., Drouin, A., Boisvert, L., Thakkar, M., Marty, T., Assouel, R., Shayegan, S. O., Jang, L. K., Lù, X. H., Yoran, O., Kong, D., Xu, F. F., Reddy, S., Neubig, G., Cappart, Q., Salakhutdinov, R., and Chapados, N. The browsergym ecosystem for web agent research. *Transactions on Machine Learning Research*, 2025.

Dekoninck, J., Baader, M., and Vechev, M. A unified approach to routing and cascading for llms. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Deng, X., Gu, Y., Zheng, B., Chen, S., Stevens, S., Wang, B., Sun, H., and Su, Y. Mind2web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*. Vol. 36, 2023.

Dong, Q., Li, L., Dai, D., Zheng, C., Ma, J., Li, R., Xia, H., Xu, J., Wu, Z., Liu, T., Chang, B., Sun, X., and Sui, Z. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 1107–1128, 2024.

Drouin, A., Gasse, M., Caccia, M., Laradji, I. H., Del Verme, M., Marty, T., Vazquez, D., Chapados, N., and Lacoste, A. Workarena: How capable are web agents at solving common knowledge work tasks? In *Proceedings of the 41st International Conference on Machine Learning*. Proceedings of Machine Learning Research, vol. 235. PMLR, pp. 11642–11662, 2024.

Gu, Y., Zhang, K., Ning, Y., Zheng, B., Gou, B., Xue, T., Chang, C., Srivastava, S., Xie, Y., Qi, P., Sun, H., and Su, Y. Is your llm secretly a world model of the internet? model-based planning for web agents. *Transactions on Machine Learning Research*, 2025.

He, F., Zhu, T., Ye, D., Liu, B., Zhou, W., and Yu, P. S. The emerged security and privacy of llm agent: A survey with case studies. *ACM Computing Surveys* 58 (6): 1–36, 2025.

He, H., Yao, W., Ma, K., Yu, W., Dai, Y., Zhang, H., Lan, Z., and Yu, D. Webvoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, L.-W. Ku, A. Martins, and V. Srikumar (Eds.). Association for Computational Linguistics, pp. 6864–6890, 2024.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network, 2015.

Kang, M., Jeong, J., Lee, S., Cho, J., and Hwang, S. J. Distilling llm agent into small models with retrieval and code tools, 2025.

Lai, H., Liu, X., Iong, I. L., Yao, S., Chen, Y., Shen, P., Yu, H., Zhang, H., Zhang, X., Dong, Y., and Tang, J. Autowebglm: A large language model-based web navigating agent, 2024.

Liu, E. Z., Guu, K., Pasupat, P., Shi, T., and Liang, P. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations*, 2018.

Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., Gu, Y., Ding, H., Men, K., Yang, K., Zhang, S., Deng, X., Zeng, A., Du, Z., Zhang, C., Shen, S., Zhang, T., Su, Y., Sun, H., Huang, M., Dong, Y., and Tang, J. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, 2024.

Ma, C., Zhang, J., Zhu, Z., Yang, C., Yang, Y., Jin, Y., Lan, Z., Kong, L., and He, J. Agentboard: An analytical evaluation board of multi-turn llm agents. In *Advances in Neural Information Processing Systems*. Vol. 37, 2024.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Gupta, S., Majumder, B. P., Hermann, K., Welleck, S., Yazdanbakhsh, A., and Clark, P. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*. Vol. 36, 2023.

Magister, L. C., Mallinson, J., Adamek, J., Malmi, E., and Severyn, A. Teaching small language models to reason. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 1773–1781, 2023.

Ong, I., Almahairi, A., Wu, V., Chiang, W.-L., Wu, T., Gonzalez, J. E., Kadous, M. W., and Stoica, I. Routellm: Learning to route llms with preference data. In *International Conference on Learning Representations*, 2025.

Qi, Z., Liu, X., Iong, I. L., Lai, H., Sun, X., Zhao, W., Yang, Y., Yang, X., Sun, J., Yao, S., Zhang, T., Xu, W., Tang, J., and Dong, Y. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning, 2025.

Schick, T., Dwivedi-Yu, J., Dessi, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*. Vol. 36. pp. 68539–68551, 2023.

Shen, J., Jain, A., Xiao, Z., Amlekar, I., Hadji, M., Podolny, A., and Talwalkar, A. Scribeagent: Towards specialized web agents using production-scale workflow data, 2024.

Shinn, N., Cassano, F., Berman, E., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*. Vol. 36, 2023.

Shridhar, K., Stolfo, A., and Sachan, M. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics: ACL 2023*. Association for Computational Linguistics, pp. 7059–7073, 2023.

Sun, H., Zhuang, Y., Kong, L., Dai, B., and Zhang, C. Adaplanner: Adaptive planning from feedback with language models, 2023.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*. Vol. 35. pp. 24824–24837, 2022.

Xiao, Y.-A., Gao, P., Peng, C., and Xiong, Y. Reducing cost of llm agents with trajectory reduction. *arXiv e-prints*, 2025.

Yan, B., Li, K., Xu, M., Dong, Y., Zhang, Y., Ren, Z., and Cheng, X. On protecting the data privacy of large language models (llms) and llm agents: A literature review. *High-Confidence Computing* 5 (2): 100300, 2025.

Yang, K., Liu, Y., Chaudhary, S., Fakoor, R., Chaudhari, P., Karypis, G., and Rangwala, H. Agentoccam: A simple yet strong baseline for llm-based web agents. In *International Conference on Learning Representations*, 2025.

Yao, S., Chen, H., Yang, J., and Narasimhan, K. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*. Vol. 35, 2022.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Zhang, Z., Lyu, Z., Gong, J., Yi, H., Wang, X., Zhou, Y., Yang, J., Nie, P., Huang, Y., and Chen, W. Browseragent: Building web agents with human-inspired web browsing actions, 2025.

Zhao, A., Huang, D., Xu, Q., Lin, M., Liu, Y.-J., and Huang, G. Expel: Llm agents are experiential learners, 2023.

Zheng, B., Gou, B., Kil, J., Sun, H., and Su, Y. Gpt-4v(ision) is a generalist web agent, if grounded. In *Proceedings of the 41st International Conference on Machine Learning*. Proceedings of Machine Learning Research, vol. 235. PMLR, pp. 61349–61385, 2024.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, T., Bisk, Y., Fried, D., Alon, U., and Neubig, G. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024.