Appendix

A. Broader Impacts

Although the results presented in this paper are only on a research benchmark, if we extrapolate forward the capabilities of these models and methods, we anticipate vast broader impacts that have the potential to revolutionize numerous industries. By allowing LLMs to execute tasks on computers, our approach can enhance the capabilities of AI assistants and automation tools. This could lead to increased efficiency, reduced labor costs, and improved user experiences across any sector which uses computers to do work. We are most excited about gains in productivity in science and education, including AI research, which will lead to even faster development of new beneficial technologies and treatments.

However, there are many potential misuses and unintended consequences associated with allowing these models to take actions in the world. Malicious actors may leverage LLMs to automate cyber-attacks, manipulate information, or propagate disinformation on a large scale. Additionally, the potential loss of jobs due to widespread automation could lead to economic disruption and increased income inequality. There are also obvious security risks of running LLMs on computers (or even virtual machines) such as prompt injection attacks. Perhaps most alarming, future LLMs taking actions on computers may lead to catastrophic runaway chains of events, especially if LLMs are integrated widely in the economy.

To mitigate these risks, it is crucial for researchers, policymakers, and industry leaders to work together to establish regulations and ethical guidelines that govern the development and deployment of such technologies. Ensuring transparency, accountability, and fairness in AI systems will be vital in harnessing the benefits while minimizing potential harm. We also believe that the time has come where we as a research community must discuss possible ways to coordinate to slow down the pace of developing highly-disruptive technology, if necessary.

B. Related Works

B.1. Automated computer tasks

The automation of computer tasks is an important topic for both information retrieval and natural language processing (Nogueira & Cho, 2016; Pasupat & Liang, 2014; Pasupat et al., 2018; Iki & Aizawa, 2022; Srivastava et al., 2020). Recent efforts have focused on the development of reinforcement learning agents that interact with websites using raw mouse and keyboard actions (Shi et al., 2017). MiniWoB, a benchmark proposed in (Shi et al., 2017), has been extended in MiniWoB++ (Liu et al., 2018), which has become a widely-used platform for studying models for computer tasks. Reinforcement learning and imitation learning have been employed in several studies to tackle MiniWoB++ tasks (Liu et al., 2018; Gur et al., 2019; Jia et al., 2019; Gur et al., 2021). However, achieving human-level performance requires a significant amount of expert demonstration data (6,300 hours), as demonstrated in (Humphreys et al., 2022). Recent work (Gur et al., 2022; Furuta et al., 2023) has suggested the use of large language models (LLMs) to comprehend HTML code and vision transformer (Dosovitskiy et al., 2020) to extract screenshot image features, with a few-shot in-context approach showing promising results without extensive RL exploration. Nevertheless, significant amounts of expert demonstration data are still required to finetune LLMs. On the contrary, the agent we suggest needs less than two demonstrations per task on average and doesn't necessitate any finetuning. WebGPT (Nakano et al., 2021) and WebShop (Yao et al., 2022) show that LLMs can automate some web-based tasks by introducing a handful of custom commands such as `Search <query>` and `Next Page`. As a result, these methods are limited in scope and do not work on general computer tasks which require keyboard strokes and mouse clicks. In contrast, our approach can tackle open-domain tasks at scale.

B.2. LLMs with actions

In recent years, there have been significant advancements in large language models (LLMs), leading to new possibilities for utilizing natural language for decision-making tasks. One approach involves augmenting LLMs with executable actions (Mialon et al., 2023). Huang et al. (Huang et al., 2022a) showed that LLMs can be used to plan and achieve simple household tasks, utilizing a method for grounding the actions generated by LLMs by comparing their embeddings with a predefined list of admissible actions. However, their work did not consider state grounding. Another study by Ahn et al. (Ahn et al., 2022) proposed SayCan, which grounded the actions by multiplying each candidate action's probability under FLAN (Wei et al., 2022a) with the action's value function, serving as an indicator for the suitability of actions. Huang et al. (Huang et al., 2022b) proposed an extension to the SayCan model called Inner Monologue, which incorporates a feedback loop for state grounding. However, Inner Monologue still requires a pre-trained language-conditioned robot policy with underlying reasoning capabilities that are not free-formed and flexible, thereby hindering generalization to diverse task domains. Similarly, Zeng et al.(Zeng et al., 2023) employed a combination of LLMs with a visual-language model (VLM) and a pre-trained language-conditioned robot policy (Shridhar et al., 2022) to perform open vocabulary pick-and-place robotic tasks. Meanwhile, Dasgupta et al.(Dasgupta et al., 2022) used Chinchilla(Hoffmann et al., 2022) as a planner for an agent in the PycoLab environment, but their actor module requires pre-training with reinforcement learning (RL) to follow natural language instructions. In a related line of research, Carta et al. (Carta et al., 2023) employed online RL fine-tuning to achieve functional grounding of LLMs in the BabyAI-Text environment. In contrast to these previous approaches, our method does not rely on additional model components beyond LLMs for grounding actions. Instead, we propose the use of RCI prompting, which enables LLMs to update their actions to be grounded autonomously. As a result, our approach can scale to a wider range of action spaces, including keyboard and mouse actions. Furthermore, prior approaches have been limited by the need for fine-tuning. In contrast, our RCI prompting method is a zero-shot approach that overcomes these limitations. More recently, an approach to improve the efficacy of LLMs involves their integration with APIs, allowing them to use external tools such as information retrieval systems, code interpreters, and web browsers (Schick et al., 2023; Thoppilan et al., 2022; Menick et al., 2022; Glaese et al., 2022). Notably, these external tools necessitate manual engineering and may be constrained in their functionality. In contrast, our agent is equipped with a general computer interface, enabling it to access a wide range of functionalities offered by computers.

B.3. LLMs with reasoning

Recent research has also demonstrated that large language models (LLMs) exhibit enhanced performance in compositional tasks when they produce traces of the underlying reasoning process along with the final answer, as evidenced by studies such as (Wei et al., 2022b; Nye et al., 2022; Kojima et al., 2022; Press et al., 2022a). This discovery has led to the emergence of a new line of research where reasoning capabilities are used to address tasks beyond reasoning (Yao et al., 2023; Huang et al., 2022b), or enhance reasoning proficiency (Kojima et al., 2022; Liu et al., 2023; Wang et al., 2023a; Madaan & Yazdanbakhsh, 2022; Zelikman et al., 2022; Press et al., 2022b; Yang et al., 2022; Sun et al., 2023; Dohan et al., 2022). Furthermore, various reasoning architectures have been proposed, expanding from naive prompting, such as Selection-Inference (Creswell et al., 2022), Least-to-Most (Zhou et al., 2023), and Faithful reasoning (Creswell & Shanahan, 2022). In the existing literature, a work closely related to our research is ReAct (Yao et al., 2023) which interleaves reasoning and action for resolving the issue of hallucination and error propagation as well as helping the model induce, track, and update actions plans. An alternative method, Reflexion (Shinn et al., 2023), extends ReAct by improving its performance by allowing LLMs to consider previous trial and error experiences. Nevertheless, due to the necessity of multiple rounds of explicit task-specific success feedback from trial and error, this approach may not scale as effortlessly as ours because it requires task-specific success feedback. Similarly, Corrective Re-prompting, as proposed by Raman et al. (Raman et al., 2022) necessitates the establishment of task-specific preconditions. RCI pertains to an extended reasoning architecture where LLMs are instructed to find errors in their outputs and improve them accordingly, which can further be used to ground actions generated from LLMs in decision-making problems. Saunders et al. (Saunders et al., 2022) used a similar approach to ours by utilizing the self-critiquing ability of LLMs to generate critical feedback on summaries produced by LLMs. The aim is to accelerate the human evaluation process by uncovering possible errors in the generated summaries. Likewise, Ganguli et al. (Ganguli et al., 2023), employed LLMs to morally self-correct their outputs to prevent the generation of harmful content. The most recent work that is in the same vein with RCI is Self-Refine (Madaan et al., 2023) which uses localized and aspect-based feedback to iteratively refine outputs from LLMs. However, our work is, to the best of our knowledge, the first to demonstrate the self-critiquing capability of LLMs only with implicit feedback (*e.g.,* "Find problems with this plan") in enhancing reasoning proficiency.

C. Experimental setup

C.1. Language models

In our evaluation, various pre-trained language models were used. RCI prompting on reasoning tasks is evaluated using *gpt-3.5-turbo*, which is presented in Table 1 and Table 2. Our primary evaluation on MiniWoB++ tasks is conducted using *text-davinci-003*, as shown in Figure 4(a) and Figure 10. We also used *davinci*, *text-davinci-002*, and *gpt-3.5-turbo* for our ablation study on MiniWoB++ tasks. For all model usage, a maximum token length of 128 and a temperature value of 0, indicating greedy decoding, are used. All models are accessed through the OpenAI API between January 2023 and March 2023.

Language model	# of parameters	Max. tokens	API provider	API name
GPT-3	175 B$(*1)$	2,049	OpenAI API	davinci
InstructGPT-3	$-(*2)$	4,097	OpenAI API	text-davinci-002
InstructGPT-3 + RLHF	$-(*2)$	4,097	OpenAI API	text-davinci-003
InstructGPT-3 + RLHF	$-(*2)$	4,096	OpenAI API	gpt-3.5-turbo
InstructGPT-3 + RLHF	$-(*2)$	8,192	OpenAI API	gpt-4

Table 3: Description of language models. $(*1)$ We identify the model size of GPT-3 by referring to the official document that OpenAI provides (https://beta.openai.com/docs/model-index-for-researchers). $(*2)$ The size of InstructGPT-based models remains undisclosed by its provider.

C.2. Reasoning tasks

We conducted an evaluation of RCI prompting on eight datasets, encompassing two categories of reasoning tasks: arithmetic and commonsense. In the domain of arithmetic reasoning, we considered six datasets: SingleEq (Koncel-Kedziorski et al., 2015), AddSub (Hosseini et al., 2014), MultiArith (Roy & Roth, 2016), AQuA (Ling et al., 2017), GSM8K (Cobbe et al., 2021), and SVAMP (Patel et al., 2021). For commonsense reasoning, we utilized the CommonsenseQA dataset (Talmor et al., 2019) and the StrategyQA dataset (Geva et al., 2021). To ensure a fair comparison with baselines, we specifically selected tasks that were employed in the work of Kojima et al. (Kojima et al., 2022). In the experiment on reasoning tasks, we enable RCI loop to get implicit feedback to correct outputs. We fix the maximum number of loops to 2. Following previous works (Madaan et al., 2023; Shinn et al., 2023; Welleck et al., 2022), we use the correct label to decide when to stop the RCI loop. In our setting, we can consider the correct label to be another source of feedback (label feedback).

C.3. MiniWoB++ task selection

In order to ensure a fair and comprehensive evaluation, a subset of MiniWoB++ tasks we use in the evaluation is selected from the evaluation of *WebN-T5-3B* (Gur et al., 2022), the most recent work on MiniWoB++ tasks, which employs LLMs. However, certain tasks such as book-flight, choose-date-easy, choose-date-medium, choose-date, and click-pie have been excluded from our evaluation due to their HTML code exceeding the maximum context length of language models. On the other hand, some of the challenging tasks such as terminal and simple-algebra have been included in the evaluation. The choice of these tasks is determined by the suboptimal performance of *CC-Net* (Humphreys et al., 2022), which currently represents the state-of-the-art model in the field. The purpose of this inclusion is to showcase the potential of leveraging LLMs in computer tasks, in contrast to the conventional approaches of Supervised Learning (SL) and Reinforcement Learning (RL). While our agent has not been evaluated on tasks that necessitate additional actions, such as drag and copy & paste, we posit that their inclusion can be readily achieved through the expansion of the actions space specification within the prompts.

C.4. MiniWoB++ task selection for ablation studies

In ablation studies, we categorize the tasks based on the success rate achieved by our full agent. We select a subset of tasks from three levels of difficulty, as depicted in Table 4.

easy [0.9, 1]	click-shape	0.98
	click-widget	0.98
	enter-date	0.96
medium [0.6, 0.9)	click-checkboxes-soft	0.72
	click-collapsible-2	0.62
	click-tab-2	0.74
hard [0, 0.6)	click-tab-2-hard	0.56
	count-shape	0.4
	guess-number	0.2

Table 4: The tasks used in the ablation study are classified according to their level of difficulty.

C.5. Modifications on MiniWoB++ tasks

In Table 5, we outline several modifications that were incorporated into the MiniWoB++ benchmark for the purpose of our evaluation with language models that have a limited context length.

Tasks	Modifications
social-media-all social-media social-media-some	We constrain the quantity of media components ranging from three to six.
email-inbox-forward-nl-turk email-inbox-forward-nl email-inbox-nl-turk	The quantity of randomly generated emails has been restricted to a range of three to six.

Table 5: Modifications on MiniWoB++ tasks.

D. Prompts for MiniWoB++ tasks

We have an autonomous computer control agent that can perform a set of instructions to control computers. First, given the instruction that matches the regular expression, <type regex>, it can type a list of characters via the keyboard. This instruction should specify the target keyboard input for the agent to type. Before this typing instruction, you should first locate the cursor by clicking the input box with the click instruction. Second, given the instruction that matches the regular expression, <press regex>, it can press a specific key on the keyboard. Third, given the instruction that matches the regular expression, <clickoption regex>, it can click an option HTML element in a list with an XPath that is visible on the webpage. The target of this instruction should be a valid XPath. Fourth, given the instruction that matches the regular expression, <movemouse regex>, it can move the mouse cursor on an HTML element with an XPath that is visible on the webpage. Lastly, given the instruction that matches the regular expression, <clickxpath regex>, it can click an HTML element with an XPath that is visible on the webpage. The target of this instruction should be a valid XPath.

Table 6: Agent specification.

```
<type regex> = "^type\s.{1,}$"
<press regex> = "^press\s(enter|arrowleft|arrowright|arrowup|arrowdown|backspace)$"
<clickoption regex> = "^clickoption\s.{1,}$"
<movemouse regex> = "^movemouse\s.{1,}$"
<clickxpath regex> = "^clickxpath\s.{1,}$"
```

Listing 1: Regular expressions for specifying the admissible actions.

Based on the current plan and task, the next proper instruction should be `

Table 7: Task grounding.

Considering the output on the webpage, the specific instruction should be `

Table 8: State grounding.

This action does not match the regular expressions. The updated instruction that matches one of the regular expressions is `

Table 9: Agent grounding.

E. Examples of cross grounding

task: click on the "ok" button on the webpage.
plan:
1. type "sit arcu nullam:"
2. clickxpath //button[@data-wob_ref="6"]

We have a history of instructions that have been already executed by the autonomous agent so far.
No instruction has been executed yet.

Based on the current plan and the task, the first instruction should be `type "sit arcu nullam:"`. Therefore, considering the output on the webpage, this instruction for solving the task should be `clickxpath //button[@data-wob_ref="6"]`.

Table 10: The state grounding optimizes the action plan. In the initial plan, the inclusion of the type instruction is deemed superfluous for accomplishing the task.

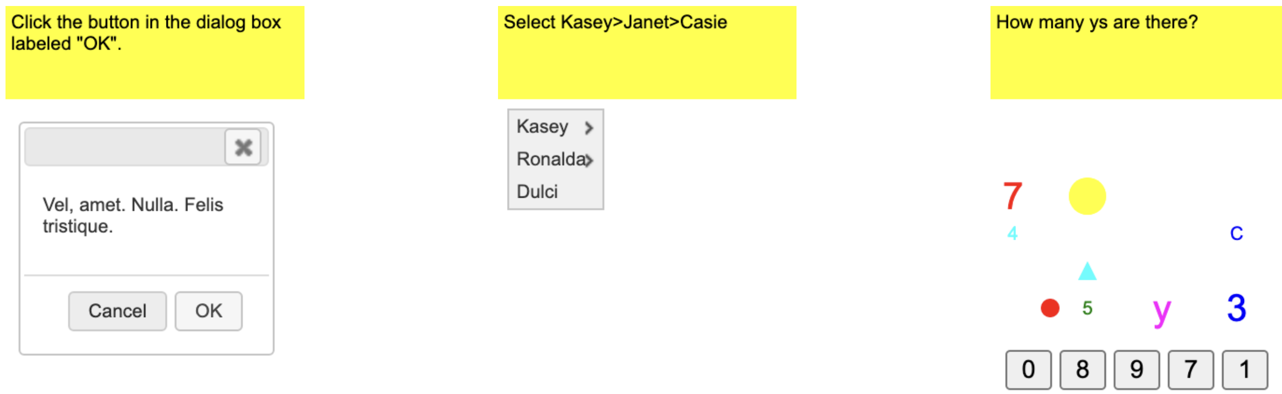
task: Find the email by Clarinda and reply to them with the text "Vel id at.".
plan:
1. clickxpath //div[@class='email-sender' and text()='Clarinda']
2. clickxpath //span[@class='email-reply']
3. clickxpath //*[@id="reply-text"]
4. type Vel id at.
5. clickxpath //*[@id="send-reply"]

We have a history of instructions that have been already executed by the autonomous agent so far.
0: clickxpath //div[@class='email-sender' and text()='Clarinda']
1: clickxpath //span[@class='email-reply']
2: clickxpath //*[@id="reply-text"]

Based on the current plan and the task, the next proper instruction after the history should be `type Vel id at`. Therefore, considering the output on the webpage, this instruction should be `type Vel id at.`

Table 11: The state grounding corrects actions to be grounded in the task. Although the target of typing action in the action plan is "Vel id at." the generated task-grounded action lacked the last punctuation mark. Nonetheless, the state grounding update was successful in remedying this discrepancy, resulting in the state-grounded action being "Vel id at." Consequently, the state-grounded action now matches the typing action specified in the action plan.

F. Failure modes on MiniWoB++ tasks



(a) click-dialog-2 (b) click-menu (c) count-shape

Figure 7: Screenshots.

F.1. click-dialog-2

The state-grounding process is occasionally inadequate for maintaining task-grounding of actions. Specifically, in the click-dialog-2 task, grounding actions in the current state results in task-irrelevant actions. Consequently, we opt to exclude the state grounding step for this particular task.

---

task: click the button in the dialog box labeled "x".
plan:
1. Clickxpath //div[@class='ui-dialog-titlebar ui-corner-all ui-widget-header ui-helper-clearfix ui-draggable-handle']/button[@title='Close']

We have a history of instructions that have been already executed by the autonomous agent so far.
No instruction has been executed yet.

Based on the current plan and task, the first instruction should be `clickxpath //div[@class='ui-dialog-titlebar ui-corner-all ui-widget-header ui-helper-clearfix ui-draggable-handle']/button[@title='Close']`. Therefore, considering the output on the webpage, this instruction should be ` clickxpath //div[@class='ui-dialog-buttonset']/button[@title='OK']` to click the button in the dialog box labeled "x".

Table 12: Example trajectory on click-dialog-2. The state grounding updates the task-grounded actions to be irrelevant to the task.

F.2. click-menu

We find that including explanatory trajectories in the few-shot examples can significantly reduce the number of required few-shot examples. Specifically, in the click-menu task, the selection of the target component necessitates the mouse hovering over the expandable menu. Therefore, we provided explanations regarding the mouse hovering technique in the few-shot examples.

F.3. count-shape

This task necessitates an agent to visually perceive rendered images on a canvas HTML element and enumerate the relevant shapes. Unfortunately, our agent encounters difficulties in tackling such tasks which entail the need for perceiving visual representations, as it solely depends on HTML observation.

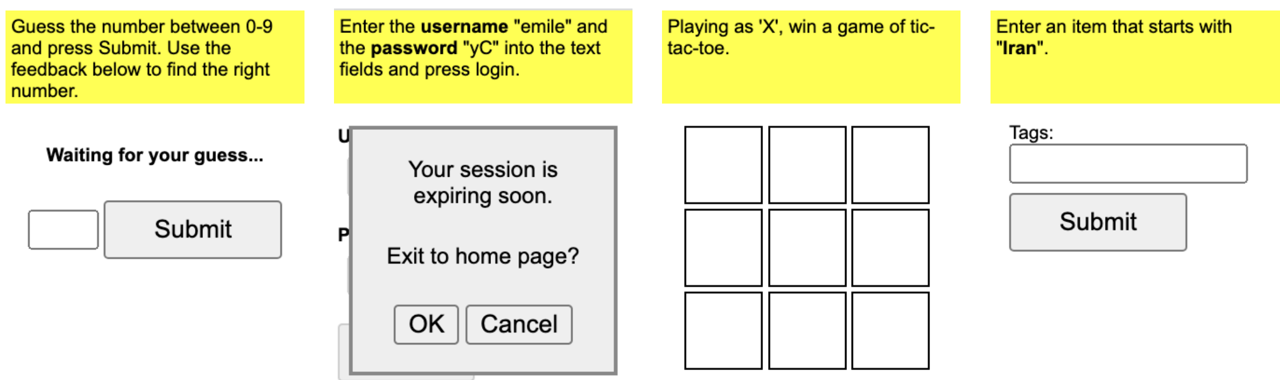
task: select Leonelle > Jane > Amabel from the HTML code provided.
plan:
1. movemouse //*[text()="Leonelle"] // we need to hover the mouse to expand the menu
2. movemouse //*[text()="Jane"] // we need to hover the mouse to expand the menu
3. clickxpath //*[text()="Amabel"]

task: select the option "Joye > Phylis" from the HTML code provided.
plan:
1. movemouse //*[text()="Joye"] // we need to hover the mouse to expand the menu
2. clickxpath //*[text()="Phylis"]

Table 13: Example trajectory on click-menu. Providing explanatory trajectory enhances the sample-efficiency of few-shot examples.

task: answer the question "How many large items are there?".
plan:
1. Clickxpath //button[text()="2"]

Table 14: Example trajectory on count-shape. The agent struggle to solve tasks that requires visual rendering of HTML.



(a) guess-number (b) login-user-popup (c) tic-tac-toe (d) use-autocomplete

Figure 8: Screenshots.

F.4. guess-number

When considering decision-making that involves a long-term perspective, our agent encounters difficulties in generating actions that are grounded in the current task. For instance, in this guess-number task, the plan generated by our agent involves the sequential typing of numbers from 0 to 9 in order to solve the task. However, due to the high number of steps required (over 50), our agent occasionally produces actions that deviate from the original plan, ultimately resulting in irreparable failure.

F.5. login-user-popup

This task involves the identification of an optimal strategy for an agent to handle an unpredictable pop-up window that emerges during a task. Due to the unexpected nature of the pop-up window, pre-planning the closure of the window is not feasible. Our agent is designed to adapt the agent's actions to the current state, so it should generate an appropriate instruction to close the pop-up window in the state-grounding step. Nevertheless, there are instances where it is unsuccessful in executing the pop-up window closure, leading to task failure.

task: Guess the number between 0-9 and press Submit. Use the feedback below to find the right number.
plan:
1. clickxpath //*[@id="tt"]
2. type 0
3. clickxpath //*[@id="subbtn"]
4. clickxpath //*[@id="tt"]
5. press backspace
6. type 1
7. clickxpath //*[@id="subbtn"]
8. clickxpath //*
9. press backspace
10. type 2
11. clickxpath //*[@id="subbtn"]
12. clickxpath //*[@id="tt"]
13. press backspace
14. type 3
15. clickxpath //*[@id="subbtn"]
16. clickxpath //*[@id="tt"]
17. press backspace
18. type 4
19. clickxpath //*[@id="subbtn"]
20. clickxpath //*[@id="tt"]
21. press backspace
22. type 5
23. clickxpath //*[@id="subbtn"]
24. clickxpath //*[@id="tt"]
25. press backspace
26. ...

We have a history of instructions that have been already executed by the autonomous agent so far.
0: clickxpath //*[@id="tt"]
1: type 0
2: clickxpath //*[@id="subbtn"]
3: press backspace
4: type 1
5: clickxpath //*[@id="subbtn"]

Based on the current plan and the task, the next proper instruction should be `type 2`.

Table 15: Example trajectory on guess-number. The agent fails to generate task-grounded action when long-horizon decision making is required.

F.6. tic-tac-toe

We also examine the causes of the poor success rate in playing tic-tac-toe. Two distinct factors are identified as responsible for its failure. The first factor is that our agent is unable to adopt a defensive strategy when its opponent is only one move away from victory and there is no immediate opportunity to win the game. The second factor relates to the inability to consider the possibility of its attack being blocked by the opponent. A case in point is the bottom-left illustration in Figure 9, where our agent's fourth move is ineffective since its diagonal direction is already obstructed by the opponent.

F.7. use-autocomplete

In use-autocomplete task, our agent demonstrated an ability to select words beginning with specific characters. However, it struggles when it comes to selecting words that ended with particular characters. This difficulty arose from the need to

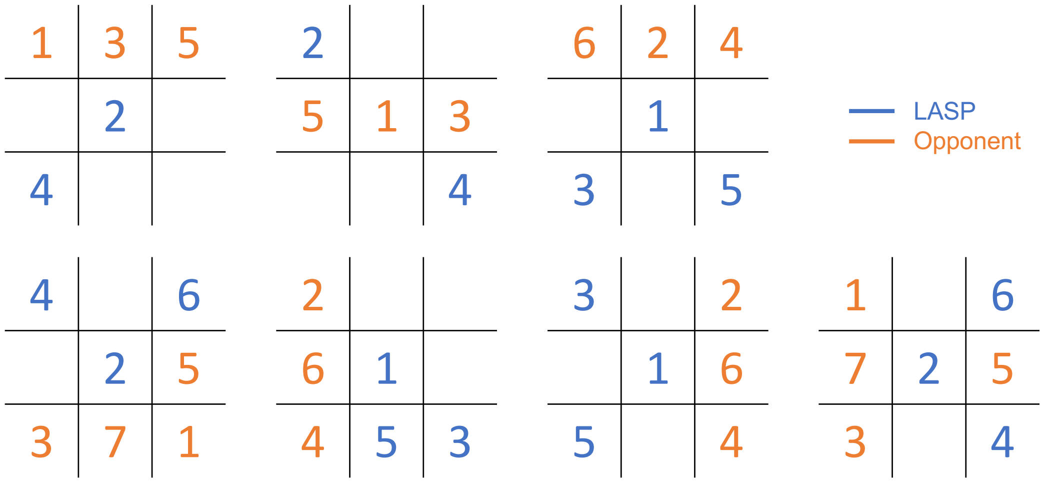


Figure 9: Failure examples of tic-tac-toe task.

identify the correct word from the autocomplete list, a process that required the agent to press the down arrow button the exact number of times needed to locate the desired word within the list.

---

task: Enter an item that starts with "Se" and ends with "ia".
plan:
1. click the input field (e.g., clickxpath //*[@id="tags"])
2. type the starting word (e.g., type Se)
3. press the down arrow key to select the word ends with "ia" (e.g., press arrowdown)
5. select the word (e.g., press enter)
6. click the submit button (e.g., clickxpath //*[@id="subbtn"])

Table 16: Example trajectory on use-autocomplete. The agent fails to identify how many times it has to press the down-arrow key.

G. Additional results

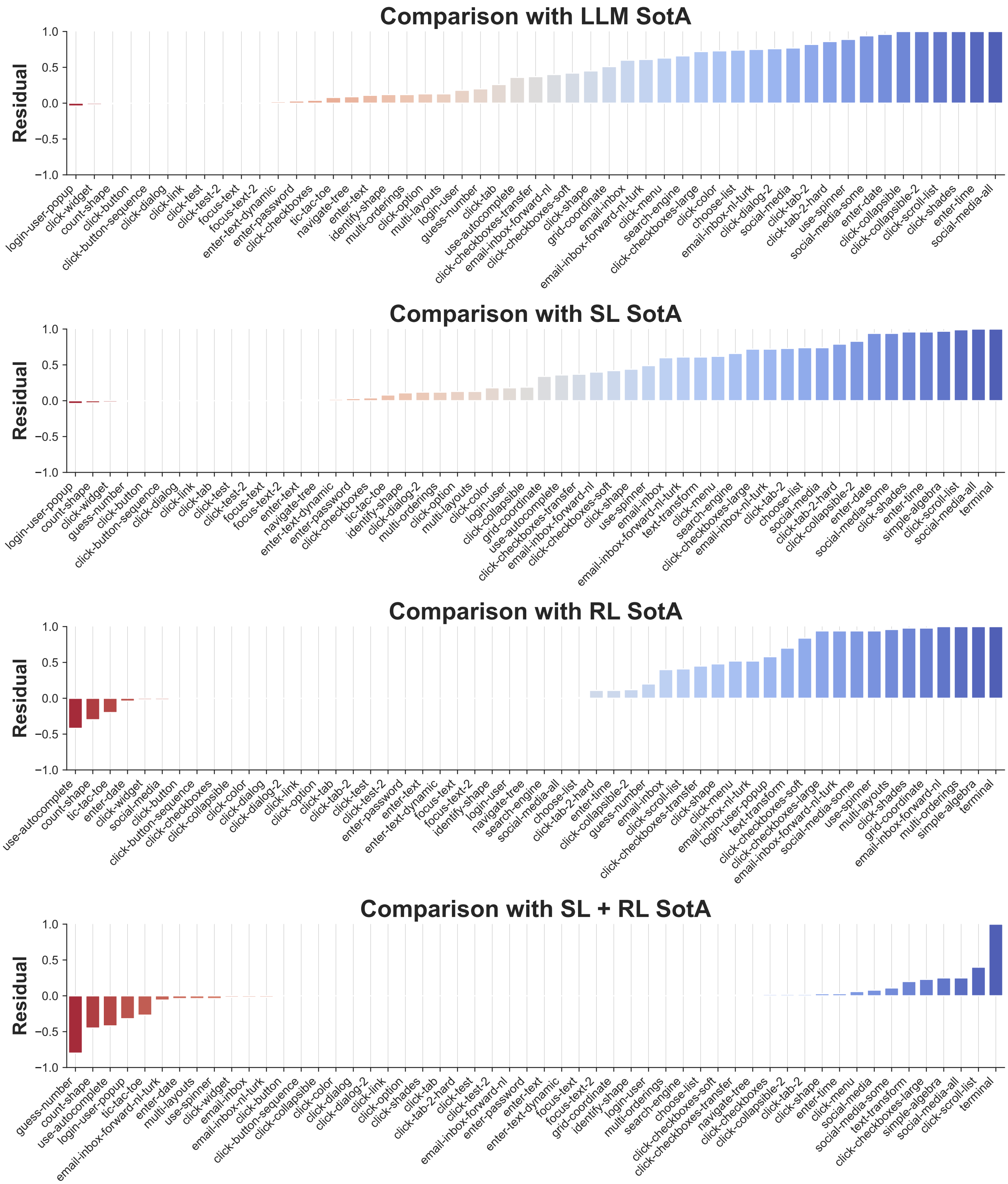


Figure 10: The task-level performance comparison with the state-of-the-art (SotA) baselines. The y-axis represents the residual values, which are obtained by subtracting the performance of SotA from our agent's performance.

TASK	Ours	Ours (w/ GPT-4)	WebN-T5-3B	CC-Net (SL + RL)	CC-Net (RL)	CC-Net (SL)	Others (SL + RL)	SotA (SL)	SotA (RL)	SotA (SL + RL)
bisect-angle	n/a	n/a	n/a	0.97	1.00	0.29	0.80	0.29	1.00	0.97
book-flight	n/a	n/a	0.00	0.87	0.00	0.00	1.00	0.00	1.00	0.87
chase-circle	n/a	n/a	n/a	0.93	0.93	0.80	1.00	0.80	0.93	1.00
choose-date-easy	n/a	n/a	0.03	0.99	0.99	0.42	n/a	0.42	0.99	0.99
choose-date-medium	n/a	n/a	0.00	0.99	0.02	0.26	n/a	0.26	0.02	0.99
choose-date	n/a	n/a	0.00	0.97	0.01	0.12	1.00	0.12	1.00	0.97
choose-list	1.00	1.00	0.26	0.99	0.99	0.19	0.26	0.26	0.99	0.99
circle-center	n/a	n/a	n/a	0.97	1.00	0.36	0.98	0.36	1.00	0.98
click-button-sequence	1.00	1.00	1.00	1.00	1.00	0.47	1.00	1.00	1.00	1.00
click-button	1.00	1.00	1.00	1.00	0.80	0.78	1.00	1.00	1.00	1.00
click-checkboxes-large	0.94	0.94	0.22	0.71	0.00	0.00	0.84	0.22	0.00	0.71
click-checkboxes-soft	0.72	0.96	0.54	0.95	0.12	0.04	0.94	0.54	0.12	0.95
click-checkboxes-transfer	1.00	1.00	0.63	0.99	0.55	0.36	0.64	0.63	0.55	0.99
click-checkboxes	1.00	1.00	0.96	0.98	0.45	0.32	1.00	0.96	1.00	0.98
click-collapsible-2	0.62	1.00	0.00	0.98	0.88	0.17	0.99	0.17	0.88	0.98
click-collapsible	1.00	1.00	0.00	1.00	1.00	0.81	1.00	0.81	1.00	1.00
click-color	1.00	1.00	0.27	1.00	1.00	0.82	1.00	0.82	1.00	1.00
click-dialog-2	1.00	1.00	0.24	1.00	1.00	0.88	1.00	0.88	1.00	1.00
click-dialog	1.00	1.00	1.00	1.00	1.00	0.95	1.00	1.00	1.00	1.00
click-link	1.00	1.00	1.00	0.99	0.94	0.59	1.00	1.00	1.00	1.00
click-menu-2	n/a	n/a	n/a	0.83	0.96	0.52	0.16	0.52	0.96	0.83
click-menu	1.00	1.00	0.37	0.94	0.48	0.22	0.13	0.38	0.48	0.94
click-option	1.00	1.00	0.87	0.99	0.78	0.21	1.00	0.87	1.00	1.00
click-pie	n/a	n/a	0.51	0.97	0.92	0.15	1.00	0.51	1.00	0.97
click-scroll-list	1.00	1.00	0.00	0.60	0.59	0.01	0.07	0.01	0.59	0.60
click-shades	1.00	1.00	0.00	1.00	0.02	0.04	0.99	0.04	0.02	1.00
click-shape	0.98	0.98	0.53	0.95	0.50	0.11	0.64	0.54	0.50	0.95
click-tab-2-easy	n/a	n/a	n/a	0.99	0.94	0.61	n/a	0.61	0.94	0.99
click-tab-2-hard	0.76	0.98	0.12	0.98	0.87	0.19	n/a	0.19	0.87	0.98
click-tab-2-medium	n/a	n/a	n/a	0.99	0.96	0.54	n/a	0.54	0.96	0.99
click-tab-2	0.74	1.00	0.18	0.98	0.91	0.27	1.00	0.27	1.00	0.98
click-tab	1.00	1.00	0.74	1.00	1.00	0.95	1.00	1.00	1.00	1.00
click-test-2	1.00	1.00	1.00	1.00	1.00	0.95	1.00	1.00	1.00	1.00
click-test-transfer	n/a	n/a	n/a	1.00	1.00	0.94	n/a	0.94	1.00	1.00
click-test	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
click-widget	0.98	0.98	1.00	1.00	1.00	0.56	1.00	1.00	1.00	1.00
copy-paste-2	n/a	n/a	n/a	0.63	0.00	0.01	0.00	0.01	0.00	0.63
copy-paste	n/a	n/a	n/a	0.79	0.00	0.04	0.00	0.04	0.00	0.79
count-shape	0.40	0.40	0.41	0.85	0.70	0.21	0.76	0.43	0.70	0.85
count-sides	n/a	n/a	n/a	1.00	1.00	0.74	0.30	0.74	1.00	1.00
drag-box	n/a	n/a	n/a	1.00	0.19	0.61	0.31	0.61	0.19	1.00
drag-cube	n/a	n/a	n/a	0.79	0.95	0.23	0.18	0.23	0.95	0.79
drag-item	n/a	n/a	n/a	1.00	0.00	0.61	n/a	0.61	0.00	1.00
drag-items-grid	n/a	n/a	n/a	0.98	0.00	0.05	0.01	0.05	0.00	0.98
drag-items	n/a	n/a	n/a	0.99	0.00	0.13	0.41	0.13	0.00	0.99
drag-shapes	n/a	n/a	n/a	0.99	0.23	0.26	0.92	0.26	0.23	0.99
drag-sort-numbers	n/a	n/a	n/a	0.97	0.00	0.11	0.66	0.11	0.00	0.97
email-inbox-delete	n/a	n/a	n/a	1.00	1.00	0.22	1.00	0.22	1.00	1.00
email-inbox-forward-nl-turk	0.94	0.94	0.33	1.00	0.00	0.00	n/a	0.33	0.00	1.00

email-inbox-forward-nl	1.00	1.00	0.60	1.00	0.00	0.00	n/a	0.60	0.00	1.00
email-inbox-forward	n/a	n/a	n/a	1.00	0.00	0.01	n/a	0.01	0.00	1.00
email-inbox-important	n/a	n/a	n/a	1.00	1.00	0.30	n/a	0.30	1.00	1.00
email-inbox-nl-turk	0.98	0.98	0.23	1.00	0.46	0.05	0.93	0.26	0.46	1.00
email-inbox-noscroll	n/a	n/a	n/a	1.00	0.48	0.13	n/a	0.13	0.48	1.00
email-inbox-reply	n/a	n/a	n/a	1.00	0.00	0.00	n/a	0.00	0.00	1.00
email-inbox-star-reply	n/a	n/a	n/a	1.00	0.47	0.11	n/a	0.11	0.47	1.00
email-inbox	0.98	0.98	0.38	1.00	0.58	0.09	0.99	0.38	0.58	1.00
enter-date	0.96	0.96	0.00	1.00	1.00	0.02	1.00	0.02	1.00	1.00
enter-password	1.00	1.00	0.97	1.00	0.01	0.02	1.00	0.97	1.00	1.00
enter-text-2	n/a	n/a	n/a	0.98	0.00	0.04	0.00	0.04	0.00	0.98
enter-text-dynamic	1.00	1.00	0.98	1.00	1.00	0.39	1.00	0.98	1.00	1.00
enter-text	1.00	1.00	0.89	1.00	1.00	0.35	1.00	0.99	1.00	1.00
enter-time	1.00	1.00	0.00	0.97	0.89	0.04	0.90	0.04	0.89	0.97
find-midpoint	n/a	n/a	n/a	0.97	0.97	0.35	0.31	0.35	0.97	0.97
find-word	n/a	n/a	n/a	0.88	0.00	0.05	0.00	0.05	0.00	0.88
focus-text-2	1.00	1.00	1.00	1.00	1.00	0.96	1.00	1.00	1.00	1.00
focus-text	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00
grid-coordinate	1.00	1.00	0.49	1.00	0.02	0.66	1.00	0.66	0.02	1.00
guess-number	0.20	0.20	0.00	1.00		0.21	0.20	0.21	0.00	1.00
highlight-text-2	n/a	n/a	n/a	1.00	0.34	0.40	0.13	0.40	0.34	1.00
highlight-text	n/a	n/a	n/a	1.00	1.00	0.51	0.90	0.51	1.00	1.00
identify-shape	0.76	1.00	0.88	1.00	1.00	0.68	1.00	0.89	1.00	1.00
login-user-popup	0.68	0.68	0.72	1.00	0.10	0.02	n/a	0.72	0.10	1.00
login-user	1.00	1.00	0.82	1.00	0.00	0.00	1.00	0.82	1.00	1.00
moving-items	n/a	n/a	n/a	0.88	0.69	0.13	0.78	0.13	0.69	0.88
multi-layouts	0.72	0.96	0.83	1.00	0.00	0.00	1.00	0.83	0.00	1.00
multi-orderings	1.00	1.00	0.88	1.00	0.00	0.00	1.00	0.88	0.00	1.00
navigate-tree	0.86	1.00	0.91	0.99	0.94	0.32	1.00	0.99	1.00	0.99
number-checkboxes	n/a	n/a	n/a	0.99	0.00	0.00	0.16	0.00	0.00	0.99
read-table-2	n/a	n/a	n/a	0.94	0.00	0.00	0.00	0.00	0.00	0.94
read-table	n/a	n/a	n/a	0.97	0.00	0.01	0.00	0.01	0.00	0.97
resize-textarea	n/a	n/a	n/a	1.00	0.68	0.27	0.11	0.27	0.68	1.00
right-angle	n/a	n/a	n/a	0.98	0.98	0.26	0.38	0.26	0.98	0.98
scroll-text-2	n/a	n/a	n/a	1.00	1.00	0.88	0.96	0.88	1.00	1.00
scroll-text	n/a	n/a	n/a	0.96	0.00	0.04	0.00	0.04	0.00	0.96
search-engine	1.00	1.00	0.34	1.00	0.01	0.15	1.00	0.34	1.00	1.00
simon-says	n/a	n/a	n/a	0.00	0.00	0.02	0.28	0.02	0.00	0.28
simple-algebra	1.00	1.00	n/a	0.75	0.00	0.03	0.04	0.03	0.00	0.75
simple-arithmetic	n/a	n/a	n/a	0.86	0.00	0.38	0.07	0.38	0.00	0.86
social-media-all	1.00	1.00	0.00	0.75	0.00	0.00	1.00	0.00	1.00	0.75
social-media-some	0.90	0.96	0.02	0.85	0.02	0.01	0.42	0.02	0.02	0.85
social-media	0.98	0.98	0.21	0.90	0.02	0.03	1.00	0.24	1.00	0.90
terminal	1.00	1.00	n/a	0.00	0.00	0.00	0.00	0.00	0.00	0.00
text-editor	n/a	n/a	n/a	0.98	0.00	0.11	0.01	0.11	0.00	0.98
text-transform	0.80	0.80	n/a	0.60	0.10	0.19	0.00	0.19	0.10	0.60
tic-tac-toe	0.56	0.56	0.48	0.83	0.76	0.32	0.47	0.48	0.76	0.83
unicode-test	n/a	n/a	n/a	1.00	1.00	0.86	n/a	0.86	1.00	1.00
use-autocomplete	0.58	0.58	0.22	1.00	1.00	0.07	0.98	0.22	1.00	1.00
use-colorwheel-2	n/a	n/a	n/a	0.95	0.85	0.38	1.00	0.38	0.85	1.00
use-colorwheel	n/a	n/a	n/a	0.98	0.82	0.68	1.00	0.68	0.82	1.00
use-slider-2	n/a	n/a	n/a	0.95	0.00	0.03	0.15	0.03	0.00	0.95
use-slider	n/a	n/a	n/a	0.91	0.47	0.18	0.51	0.18	0.47	0.91
use-spinner	0.88	0.96	0.07	1.00	0.02	0.47	0.17	0.47	0.02	1.00

visual-addition	n/a	n/a	n/a	0.99	0.00	0.36	0.01	0.36	0.00	0.99

Table 17: Comprehensive task-level success rate evaluation of baseline models in MiniWoB++ tasks. *Ours (w/ GPT-4)* depicts the performance outcomes obtained through the use of the GPT-4 model for some tasks, which are visually highlighted in the color blue. The performance of baseline models has been sourced from prior studies (Humphreys et al., 2022; Gur et al., 2022). The average success rates of the tasks highlighted with violet color are shown in Figure 3. The state-of-the-art (SotA) in supervised learning (SL) is represented by the works of (Humphreys et al., 2022; Gur et al., 2022) while the SotA in reinforcement learning (RL) includes the studies of (Humphreys et al., 2022; Gur et al., 2019; Jia et al., 2019). Furthermore, the SotA in the combined application of SL and RL consists of the contributions of (Humphreys et al., 2022; Shi et al., 2017; Liu et al., 2018). Combined result of models proposed prior to CC-Net (Humphreys et al., 2022) is denoted as *Others*, which include (Shi et al., 2017; Liu et al., 2018; Gur et al., 2019; Jia et al., 2019). This corresponds to *Aggregated SotA (Augmented)* baseline in previous works (Humphreys et al., 2022). We generously estimate the performance of *CC-Net (RL)* based on their figures.