# WorkForceAgent-R1: Incentivizing Reasoning Capability in LLM-based Web Agents via Reinforcement Learning

**Yuchen Zhuang**[♠*], **Di Jin**[♡*], **Jiaao Chen**[♠], **Wenqi Shi**[†], **Hanrui Wang**[♡], **Chao Zhang**[♠]
[♠] Georgia Tech [†] UT Southwestern Medical Center [♡] Eigen AI
{yczhuang,jiaaochen,chaozhang}@gatech.edu
wenqi.shi@utsouthwestern.edu, {di,ryan}@eigenai.com

## Abstract

Large language models (LLMs)-empowered web agents enables automating complex, real-time web navigation tasks in enterprise environments. However, existing web agents relying on supervised fine-tuning (SFT) often struggle with generalization and robustness due to insufficient reasoning capabilities when handling the inherently dynamic nature of web interactions. In this study, we introduce `WorkForceAgent-R1`[1], an LLM-based web agent trained using a rule-based R1-style reinforcement learning framework designed explicitly to enhance single-step reasoning and planning for business-oriented web navigation tasks. We employ a structured reward function that evaluates both adherence to output formats and correctness of actions, enabling `WorkForceAgent-R1` to implicitly learn robust intermediate reasoning without explicit annotations or extensive expert demonstrations. Extensive experiments on the WorkArena benchmark demonstrate that `WorkForceAgent-R1` substantially outperforms SFT baselines by 10.26-16.59%, achieving competitive performance relative to proprietary LLM-based agents (`gpt-4o`) in workplace-oriented web navigation tasks.

## 1 Introduction

Large language models (LLMs) have emerged as powerful agents capable of executing complex tasks across diverse domains (Yao et al., 2023; Liu et al., 2023), particularly through integration with external environments or APIs (Song et al., 2023; Wang et al., 2024; Zhuang et al., 2025; Sun et al., 2023; Liao et al., 2024; Gu et al., 2024; Zhuang et al., 2024a). Web agents, a specialized class of LLM-powered autonomous systems, navigate and interact with websites to perform tasks

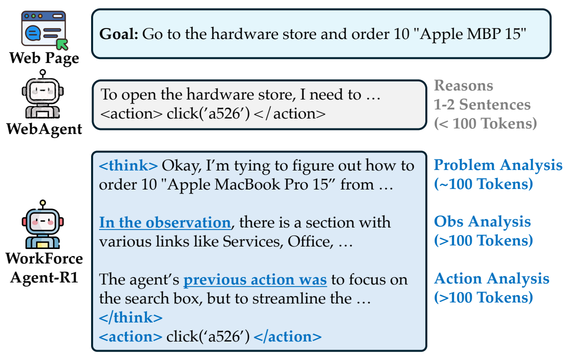

Web Page
Goal: Go to the hardware store and order 10 "Apple MBP 15"
WebAgent
To open the hardware store, I need to …
<action> click('a526') </action>
Reasons
1-2 Sentences
(< 100 Tokens)
WorkForce Agent-R1
<think> Okay, I'm tying to figure out how to order 10 "Apple MacBook Pro 15" from …
In the observation, there is a section with various links like Services, Office, …
The agent's previous action was to focus on the search box, but to streamline the …
</think>
<action> click('a526') </action>
Problem Analysis
(~100 Tokens)
Obs Analysis
(>100 Tokens)
Action Analysis
(>100 Tokens)

Figure 1: Example of reasoning capabilities that support service catalog in workplace with `WorkForceAgent-R1`.

ranging from partial assistance (*e.g.*, form completion) to complete process automation (*e.g.*, order management) (Furuta et al., 2024; Le Sellier De Chezelles et al., 2024; Chezelles et al., 2024; Drouin et al., 2024). Unlike structured API environments, web interfaces present unique challenges, including noisy HTML structures, dynamic content, and inconsistent element identification (He et al., 2024b,a), demanding robust context-aware interactions. Business-oriented web agents face additional complexity due to enterprise software prioritizing functionality over user experience (Drouin et al., 2024; Xu et al., 2024), resulting in complex workflows, unintuitive interfaces, and repetitive tasks. These challenges necessitate advanced reasoning and effective interaction strategies to reliably automate dynamic and context-rich web tasks in workplace environments.

Existing web agents mainly enhance proprietary LLMs through meticulously crafted prompts or predefined workflows (He et al., 2024a; Zheng et al., 2024; Ma et al., 2023), which pose affordability and privacy concerns for workplace implementations (Sun et al., 2024; Li et al., 2024; Zhuang et al., 2024b). On the other hand, significant performance gaps persist between proprietary and open-source (OSS) LLMs (Drouin et al., 2024; Levy

[*] Equal contribution. Correspondence to: Chao Zhang.
[1] The code of `WorkForceAgent-R1` is available at: `https://github.com/night-chen/WorkForceAgent-R1`.

et al., 2024; Pan et al., 2024b; Yoran et al., 2024), especially with inadequate post-training on complex web interactions. Recent efforts to improve web agents via supervised fine-tuning (SFT) (Lai et al., 2024; Pan et al., 2024a) often narrowly optimize individual web actions without adequately addressing the inherently interactive and dynamic nature of agent-environment interactions, thus failing to guide advanced reasoning processes. Attempts to distill reasoning trajectories from advanced models (Deng et al., 2023) frequently result in superficial "pseudo reasoning", encouraging models merely to intimate surface-level action patterns without genuinely internalizing underlying reasoning mechanisms (Adler et al., 2024).

Web navigation inherently involves complex planning, challenging LLMs to execute sequential decision making, retain memory across multiple turns, and dynamically adapt to environmental feedback (He et al., 2024a,b). Observations typically consist of *entire* HTML web pages, complicating multi-turn interactions between agents and their environments. Compared to static environments with simpler state representations, HTML data make maintaining comprehensive action-observation histories across interactions prohibitively costly and inefficient (Gur et al., 2023; Ye et al., 2025). Furthermore, the information-rich observations following each action limit the effectiveness of multi-step planning, as agents cannot reliably predict future states without direct environmental engagement. Simulations employing outcome-supervised reward models (ORM) to generate multi-turn trajectories (Qi et al., 2025) often lack sufficient precision and comprehensiveness to serve effectively as oracle guidance for complex real-world interactions. Consequently, developing robust web agents capable of accurate *single-step planning* emerges as a computationally efficient and reliable solution for effective web automation deployments.

In this work, we introduce `WorkForceAgent-R1`, a web agent trained with rule-based R1-style (Guo et al., 2025) reinforcement learning (RL) that improves reasoning-driven navigation for dynamic web environments in workplaces. We propose a progressive reward function that evaluates both format adherence and action correctness, incentivizing web agents to implicitly learn intermediate reasoning steps without relying on explicit reasoning annotations or expensive expert demonstrations. By decomposing multi-step web interactions into discrete planning and emphasizing explicit reasoning at each action step, `WorkForceAgent-R1` effectively generalizes across diverse web interfaces while naturally balancing exploration and exploitation during training. Experiments on the WorkArena benchmark (Drouin et al., 2024) demonstrate that `WorkForceAgent-R1` significantly outperforms SFT baselines by up to 16.59%; in particular, `WorkForceAgent-R1` (14B) achieves competitive performance against proprietary models (`gpt-4o` and `gpt-4.1-mini`) in workplace-oriented web navigation tasks. We summarize our main contributions as follows:

- We propose `WorkForceAgent-R1`, a rule-based RL framework specifically designed to enhance *single-step reasoning and planning* of LLM agents in web navigation tasks.
- We introduce a progressive reward function that evaluates both action correctness and adherence to structured output formats, enabling robust reasoning capabilities without explicit reasoning annotations in information-dense web environments.
- Extensive experiments on WorkArena demonstrate that `WorkForceAgent-R1` achieves superior performance over SFT baselines competitive results relative against proprietary web agents in workplace environments.

## 2 Related Works

**LLM-Empowered Web Agents.** LLM-based web agents enable autonomous website navigation, dynamic content interpretation, and execution of user interactions (Zhou et al., 2023; Deng et al., 2023; Cheng et al., 2024; Yao et al., 2022; Jang et al., 2024). Early efforts primarily rely on SFT and imitation learning (Lai et al., 2024; Pan et al., 2024a; Furuta et al., 2024; Yao et al., 2022; Nakano et al., 2021) struggle with generalization to complex and varied web scenarios. More recent studies, such as WebRL (Qi et al., 2025) and OpenWebVoyager (He et al., 2024b), have utilized RL frameworks with curriculum-driven or iterative feedback loops, achieving significant improvements in complex and dynamic web navigation tasks. Despite these advancements, existing methods focus on optimizing *individual actions* without adequately addressing the inherent complexity and interactive nature of web interactions, often relying heavily on expensive and less private-preserving proprietary models and extensive human annotations or expert demonstrations, limiting their scalability and practicality, particularly for workplace automation.

**RL for Improving Reasoning in LLM Agents.** Recent research has increasingly prioritized enhancing the reasoning capabilities of LLM agents with test-time scaling (Snell et al., 2024; Muennighoff et al., 2025; Adler et al., 2024). DeepSeek-R1 (Guo et al., 2025) further demonstrates that simple rule-based RL effectively induces strong reasoning behaviors by rewarding only the correctness of the final output, thereby implicitly guiding intermediate reasoning steps. In response, `WorkForceAgent-R1` improves open-source LLM-driven web agents through R1-style RL, which facilitates robust single-step reasoning and adaptive interaction for automating web tasks in workplace environments.

## 3 Preliminary

Web agents performing autonomous tasks on dynamic websites face inherently complex and partially observable environments. Formally, we define each web navigation task as a Partially Observable Markov Decision Process (POMDP) represented by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R})$. Here, $\mathcal{S}$ denotes the state space comprising all possible configurations of the web environment, $\mathcal{A}$ is the set of permissible actions executable by the agent (*e.g.*, clicking elements, entering text, navigating URLs), and $\mathcal{O}$ is the set of observations accessible to the agent. The transition dynamics $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ specify the evolution of states based on executed actions, and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \{0, 1\}$ represents a binary reward function indicating successful task completion. Due to partial observability, the true state $s_t \in \mathcal{S}$ is not directly observable; instead, the agent receives observations $o_t \in \mathcal{O}$, typically represented as raw HTML web pages.

At each time step $t$, given a user instruction $I$, the LLM-based web agent must select an appropriate action $a_t \in \mathcal{A}$. Actions are defined as structured operations, each consisting of an operation name $n_i$ (*e.g.*, "`click`", "`type`", *etc.*) and associated parameters $b_i$, primarily corresponding to unique element identifiers (*e.g.*, `bid` attributes). The agent's decision-making relies upon a policy $\pi$, which maps historical context and current observations to a probability distribution over actions. Specifically, at step $t + 1$, the agent's policy $\pi$ conditions on three elements: (1) the historical context $c_t = \{a_0, a_1, \ldots, a_t\}$, representing previously executed web operations, (2) the latest observation $o_t$, reflecting the most recent HTML state of the webpage, and (3) the current action set $\mathcal{A}$. The policy is formally expressed as:

$$\pi(a_{t+1}|c_t, o_t, \mathcal{A}) = \pi(a_t|\{a_{\leq t}\}, o_t, \mathcal{A}), \quad a_t \in \mathcal{A}.$$

The objective is to learn a generalized policy $\pi$ that robustly addresses diverse user instructions $I$, producing sequences of action-observation pairs $\{(a_t, o_t)\}$ that consistently lead to successful task outcomes in complex web environments.

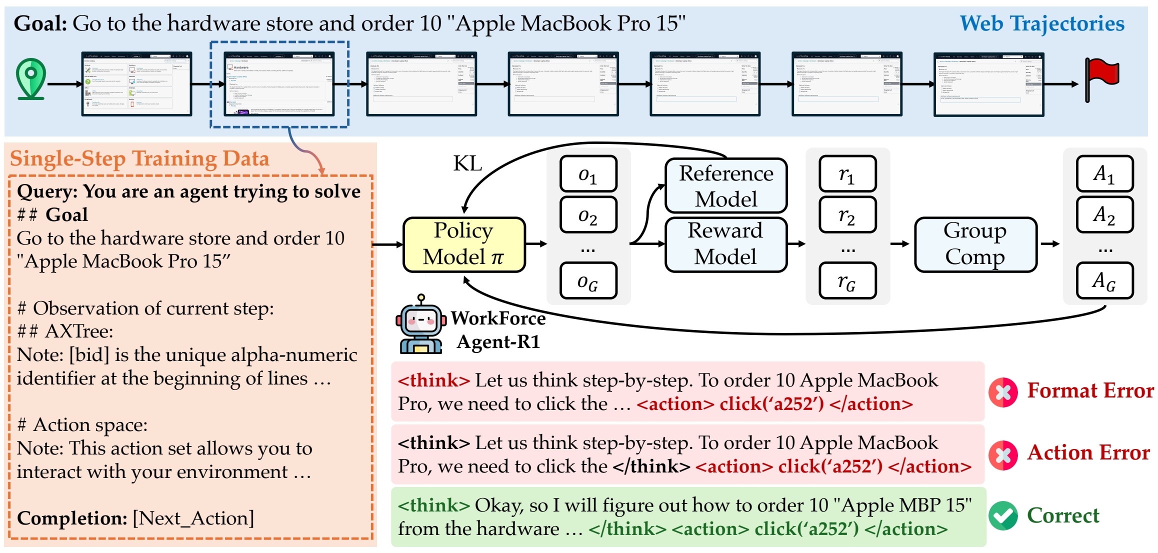
## 4 `WorkForceAgent-R1`

We introduce `WorkForceAgent-R1`, an LLM-driven web agent specifically optimized for robust single-step reasoning in workplace-oriented web navigation tasks (Figure 2). `WorkForceAgent-R1` combines behavior cloning through SFT with Group Relative Policy Optimization (GRPO), utilizing structured reward signals designed to enhance both reasoning and precise action execution.

### 4.1 Data Preparation

Previous research has extensively explored SFT for enhancing the browsing and problem-solving capabilities of LLM-based web agents (Furuta et al., 2024; Lai et al., 2024; Pan et al., 2024a). These approaches typically rely on datasets comprising natural language queries $Q$, paired with sequences of web operation actions and observations, denoted as $(a_0, o_0, \cdots, a_t, o_t)$. Despite their efficacy, SFT methods frequently exhibit limited generalization due to overfitting to memorized trajectories, thereby failing to induce robust intrinsic reasoning abilities in LLMs (Qi et al., 2025). Furthermore, collecting extensive SFT data demands significant human annotation efforts or costly API queries to advanced models (*e.g.*, `gpt-4.1`, `o3-mini`) for expert trajectory generation.

**Data Configuration, Splitting, and Augmentation.** We utilize the BrowserGym environment (Le Sellier De Chezelles et al., 2024) to generate scalable web agent interactions and employ the WorkArena benchmark (Drouin et al., 2024) to evaluate agent performance rigorously. BrowserGym provides standardized interaction environments with clearly defined observation and action spaces, while WorkArena offers a remote-hosted evaluation framework comprising 33 distinct web navigation tasks, such as form filling, knowledge base searching, and dashboard navigation. Each



Goal: Go to the hardware store and order 10 "Apple MacBook Pro 15"
Web Trajectories
Single-Step Training Data
Query: You are an agent trying to solve
## Goal
Go to the hardware store and order 10 "Apple MacBook Pro 15"
# Observation of current step:
## AXTree:
Note: [bid] is the unique alpha-numeric identifier at the beginning of lines …
# Action space:
Note: This action set allows you to interact with your environment …
Completion: [Next_Action]
KL
Policy Model π
$o_1$ $o_2$ … $o_G$
Reference Model
Reward Model
$r_1$ $r_2$ … $r_G$
Group Comp
$A_1$ $A_2$ … $A_G$
WorkForce Agent-R1
<think> Let us think step-by-step. To order 10 Apple MacBook Pro, we need to click the … <action> click('a252') </action>
Format Error
<think> Let us think step-by-step. To order 10 Apple MacBook Pro, we need to click the </think> <action> click('a252') </action>
Action Error
<think> Okay, so I will figure out how to order 10 "Apple MBP 15" from the hardware … </think> <action> click('a252') </action>
Correct

Figure 2: Overview of `WorkForceAgent-R1`.

task in WorkArena is defined by a configuration file specifying target states, relevant webpage elements, and expected outcomes. To ensure data integrity and prevent information leakage, we allocate separate training and testing data configurations. Specifically, we reserve 10 distinct configuration files per task, totaling 330 training configurations. We further mitigate potential data leakage by leveraging advanced LLMs (`o3-mini`) to perturb the training configurations, ensuring uniqueness relative to BrowserGym and WorkArena.

**Ground-Truth Trajectories.** BrowserGym incorporates heuristic-based `cheat()` functions employing Playwright[2] scripts to autonomously generate oracle-like ground-truth trajectories for web interactions. Utilizing this functionality, we systematically sample ground-truth web navigation trajectories $(a_0, o_0, \cdots, a_T, o_T)$ for each training configuration. Due to occasional inconsistencies and redundancies in heuristic-generated actions, we enforce data standardization by removing samples with invalid actions or unsuccessful outcomes, resulting in high-quality training trajectories.

**Single-Step Reasoning.** While recent methods (Wang et al., 2025; Ouyang et al., 2025; Qian et al., 2025) emphasize interleaving reasoning with environmental interactions, their application is constrained to simple or simulated settings. Web browsing scenarios present unique challenges: (1) observations, derived from interactions with webpages, are dynamically generated and thus complicate multi-step forward planning; (2) web observations, typically comprising extensive HTML content, impose computational and latency constraints when sequentially appended to actions. Consequently, we reformulate the original multi-step ground-truth trajectories $(a_0, o_0, \cdots, a_T, o_T)$ into single-step reasoning sequences:

$$\{\tau_t\}_{t=1}^{T} := \{(a_0, a_1, \cdots, a_{t-1}, o_{t-1})\}_{t=1}^{T}.$$

## 4.2 Thinking Template

Following Guo et al. (2025), we introduce a structured prompting template to facilitate explicit intermediate reasoning within LLM-generated actions. This template clearly delineates intermediate reasoning with `<think>...</think>` tags and final web actions with `<action>...</action>` tags, as illustrated in Figure 3. Such design aims to balance structural guidance and flexibility, reducing overfitting to specific prompt patterns and promoting robust generalization across diverse web interaction scenarios (Yao et al., 2023).

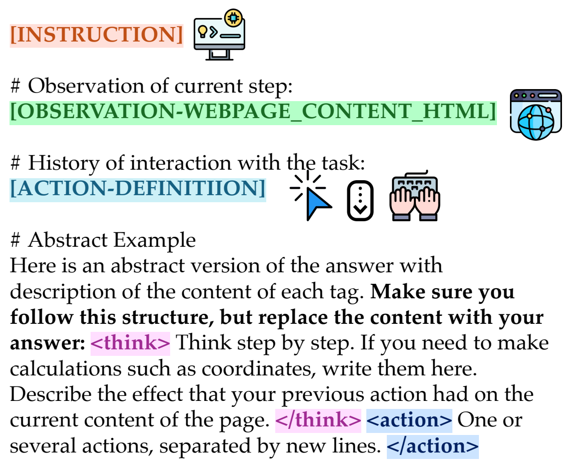
## 4.3 Web Agent Training

### 4.3.1 Behavior Cloning via SFT

We employ behavior cloning (BC) to initialize the web agent policy, leveraging SFT on action trajectories generated by heuristic methods within WorkArena. Specifically, we train the LLM to predict the correct action given the observation and context, using the standard cross-entropy loss:

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{SFT}}}\left[\sum_{l=1}^{L} \log f_\theta(y_l|y_{<l}, x)\right].$$

[2] https://playwright.dev/


[INSTRUCTION]

# Observation of current step:
[OBSERVATION-WEBPAGE_CONTENT_HTML]

# History of interaction with the task:
[ACTION-DEFINITIION]

# Abstract Example
Here is an abstract version of the answer with description of the content of each tag. **Make sure you follow this structure, but replace the content with your answer:** <think> Think step by step. If you need to make calculations such as coordinates, write them here. Describe the effect that your previous action had on the current content of the page. </think> <action> One or several actions, separated by new lines. </action>

Figure 3: Example of a thinking template with an agent prompt in `WorkForceAgent-R1`.

where $x$ denotes the input context and observation, and $y$ represents the ground-truth action with reasoning steps. This initial SFT step enables the agent to acquire baseline capabilities for effective web interaction.

### 4.3.2 Reinforcement Learning

To endow web agents with robust reasoning capabilities, we adopt the GRPO framework (Shao et al., 2024). GRPO is advantageous for web agent training due to its omission of explicit value function estimation, reduced memory footprint, and precise reward-targeting mechanisms (Ma et al., 2025).

Formally, we decompose each web navigation task into a series of independent single-step decision problems. Specifically, at each decision step, given the current observation (webpage HTML) and historical context (previous actions), the agent policy at the $(m)$-th iteration generates a group of $G$ action candidates $\{c_1, c_2, \cdots, c_G\}$ from the previous policy $\pi_{\theta(m-1)}$. Each candidate action is evaluated using a carefully defined composite reward function (Section 4.3.3). GRPO calculates group-relative advantages and updates the policy parameters to preferentially select actions exhibiting higher relative performance within each group. The GRPO objective is formalized as:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\{c_i\}_{i=1}^{G}\sim\pi_{\theta(m-1)}(y|x)} \Bigg[\frac{1}{G}\sum_{i=1}^{G}\Big(\min\left(r_i^{\text{ratio}}, \text{clip}\left(r_i^{\text{ratio}}, 1-\epsilon, 1+\epsilon\right)\right)A_i - \beta\mathbb{D}_{\text{KL}}\left(\pi_{\theta(m)}||\pi_{\text{ref}}\right)\Big)\Bigg],$$

where $r_i^{\text{ratio}} = \frac{\pi_{\theta(m)}(c_i|x)}{\pi_{\theta(m-1)}(c_i|x)}$ denotes the importance sampling ratio, quantifying the relative likelihood of candidate actions $c_i$ under updated versus previous policies $\pi_{\theta(m)}$ compared to $\pi_{\theta(m-1)}$; $A_i$ represents the group-relative advantage; the clipping operator constrains policy updates; $\epsilon$ and $\beta$ are hyperparameters governing update magnitude and KL-divergence regularization, respectively; and $\pi_{\text{ref}}$ is the reference policy.

### 4.3.3 Reward Function Design

We design a structured and progressive reward mechanism comprising three distinct components: (1) format correctness, (2) action correctness, and (3) penalty constraints. This multilayered approach comprehensively guides the training of web agent toward generating accurate and contextually appropriate web interactions.

**Format Reward $R_f$.** The agent is incentivized to strictly adhere to a predefined structured output format, encapsulating reasoning within `<think>···</think>` tags and actions within `<action>···</action>` tags. Additionally, each generated action must be a valid operation defined by the environment. The format reward is formally defined as:

$$R_f = \begin{cases} 0.1, & \text{if output format is correct,} \\ 0, & \text{otherwise.} \end{cases}$$

**Success Reward $R_s$.** Action accuracy is crucial for successful web navigation. To evaluate correctness comprehensively, we employ exact matching against ground-truth actions, distinguishing between action types (*e.g.*, click, fill) and associated parameters (*e.g.*, element identifiers). Due to practical constraints in real-time environmental feedback, exact matching serves as an effective surrogate for real-world verification. The success reward is progressively structured as:

$$R_s = \begin{cases} 1, & \text{if action and paras are correct,} \\ 0.1, & \text{if only action type is correct,} \\ 0, & \text{otherwise.} \end{cases}$$

**Penalty Reward $R_p$.** To discourage undesirable token generation and mitigate reward hacking behaviors, particularly the generation of extraneous tokens beyond defined action boundaries, we introduce a penalty term. Specifically, tokens generated after the termination tag `</action>` invalidate the action, significantly reducing the overall reward to emphasize adherence to strict completion conditions. Formally, the penalty reward is expressed

Table 1: Main experimental results on WorkArena (Drouin et al., 2024). **Bold** indicates the best performance under each task. Underline indicates the second best. Notations are consistent across tables.

| **Baselines (↓) / Tasks (→)** | **Dashboard** | **Form** | **Knowledge** | **Filter** | **Sort** | **Menu** | **Service** | **Overall** | **Δ** |
|---|---|---|---|---|---|---|---|---|---|
| *Base to Large Size Open-Source (OSS) LLMs* | | | | | | | | | |
| *Base Models* | | | | | | | | | |
| `Qwen2.5-3B-Instruct` (Yang et al., 2024) | 5.00 | 1.00 | 5.00 | 0.00 | 9.17 | 1.50 | 0.00 | 2.62 | - |
| `Qwen2.5-7B-Instruct` (Yang et al., 2024) | 12.50 | 10.00 | **30.00** | 0.00 | 0.00 | 20.00 | 15.60 | 9.42 | - |
| `Qwen3-8B` (Qwen, 2025) | 15.00 | 10.00 | 20.00 | 5.00 | 5.00 | 10.00 | 11.20 | 9.44 | - |
| `Qwen2.5-14B-Instruct` (Yang et al., 2024) | 10.00 | 28.57 | 20.00 | 0.00 | 7.61 | 45.56 | 48.97 | 23.79 | - |
| *+Supervised Fine-Tuning (SFT)* | | | | | | | | | |
| `Qwen2.5-3B-Instruct-sft` | 10.00 | 10.00 | 12.00 | 6.00 | 13.00 | 34.00 | 67.80 | 26.59 | (+23.97) |
| `Qwen2.5-7B-Instruct-sft` | 15.00 | 13.80 | 20.00 | 1.30 | 24.17 | 54.00 | 56.15 | 27.32 | (+17.90) |
| `Qwen2.5-14B-Instruct-sft` | 15.00 | 13.20 | 20.00 | 8.00 | 18.00 | 56.00 | 64.40 | 30.20 | (+6.41) |
| *+Reinforcement Learning (RL): Reasoning Models* | | | | | | | | | |
| `WorkForceAgent-R1 (3B)` | **20.00** | **36.00** | **30.00** | 10.00 | 15.00 | 64.82 | 82.12 | 36.85 | (+34.23) |
| `WorkForceAgent-R1 (7B)` | **20.00** | 19.80 | **30.00** | 17.00 | **37.00** | **82.00** | 69.81 | 39.56 | (+30.14) |
| `WorkForceAgent-R1 (14B)` | **20.00** | 30.60 | **30.00** | **25.17** | 31.33 | 75.00 | **89.81** | **46.79** | (+23.00) |
| *API-based Proprietary LLMs and XL OSS LLMs (for reference only)* | | | | | | | | | |
| `gpt-3.5-turbo` | 20.00 | 2.00 | 0.00 | 0.00 | 8.30 | 5.00 | 5.60 | 6.10 | - |
| `gpt-4o` (Hurst et al., 2024) | 62.50 | 40.00 | 80.00 | 0.00 | 10.00 | 60.00 | 77.80 | 42.65 | - |
| `gpt-4o-V` (Hurst et al., 2024) | 72.50 | 34.00 | 70.00 | 0.00 | 13.30 | 90.00 | 65.60 | 41.80 | - |
| `gpt-4.1-mini` (OpenAI, 2025a) | 25.00 | 41.80 | 25.00 | 0.33 | 7.00 | 96.50 | 95.38 | 43.27 | - |
| `gpt-4.1` (OpenAI, 2025a) | 50.00 | 52.40 | 60.00 | 15.17 | 9.52 | 97.99 | 80.00 | 48.19 | - |
| `o4-mini` (OpenAI, 2025b) | 44.00 | 78.00 | 54.00 | 16.00 | 27.00 | 88.60 | 84.80 | 55.78 | - |
| `Llama3-70B` (Dubey et al., 2024) | 37.50 | 32.00 | 30.00 | 0.00 | 1.70 | 0.00 | 26.70 | 17.90 | - |

as:

$$R_p = \begin{cases} -0.9, & \text{if tokens appears after } \texttt{<action>}, \\ 0, & \text{otherwise.} \end{cases}$$

Finally, the total reward $R$ for each action candidate is the summation of the three reward components:

$$R = R_f + R_s + R_p.$$

## 5 Experiments

### 5.1 Experiments Setup

**Tasks and Datasets.** We conduct experiments on the WorkArena (Drouin et al., 2024) benchmark, covering 7 categories of tasks in workplaces: dashboards, forms, knowledge bases, list filtering, list sorting, menus, and service. Detailed descriptions are in appendix A.

**Baselines.** We consider the following baselines for comparison: (1) API-based proprietary LLMs for reference, including `gpt-3.5-turbo`, `gpt-4o`, `gpt-4o-V`, `gpt-4.1-mini`, `gpt-4.1`, and `o4-mini`; (2) small- to XL-size OSS LLMs, including `Qwen2.5-Instruct`, `Qwen3`, and `LLaMA-3-70B`; and (3) SFT of OSS LLMs.

**Evaluation Metrics.** Following Drouin et al. (2024), we adopt *success rate (SR)* as the primary evaluation metric. The overall score represents the *weighted average* of the success rates across various task categories, adjusted according to the number of tasks in each category.

**Backbones.** We leverage varying sizes of OSS LLMs, including `Qwen2.5-Instruct` (Yang et al., 2024) (3B/7B/14B) as backbone LLMs.

**Implementation Details.** All RL experiments are conducted using the open-source framework VeRL (Sheng et al., 2024). The SFT (warm-up) phase is executed for 1 epoch with a batch size of 32 and a learning rate of $1 \times 10^{-4}$ on 1,000 randomly selected samples from the training dataset. Subsequently, RL training is performed with a batch size of 128 and a learning rate of $1 \times 10^{-5}$. The temperature parameter during model rollout is consistently set to 0.6. Throughout training, the coefficient for the KL divergence regularization term is fixed at $\beta = 1 \times 10^{-3}$. All experiments are run on 8 NVIDIA H200 GPUs, each equipped with 141GB of memory.

### 5.2 Main Experiment Results

Table 1 compares `WorkForceAgent-R1` with base models and SFT baselines. We summarize our main observations: (1) **Superior Empirical Per-**

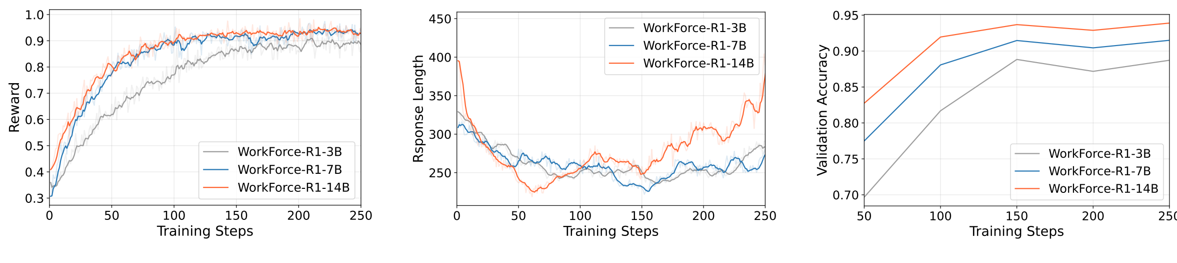


Figure 4: Learning curves across training steps.

**formance via RL:** `WorkForceAgent-R1` consistently outperforms existing OSS models by an average margin of 29.13% when employing similar model sizes. Remarkably, the 14B-parameter variant of `WorkForceAgent-R1` surpasses the proprietary state-of-the-art model `GPT-4o` by 4.99%, underscoring its effectiveness in workplace-oriented web navigation tasks. (2) **Balanced Performance Across Tasks:** Models trained via SFT exhibit significant performance imbalance, particularly evident in achieving less than 10% accuracy on list-filtering tasks while nearing 60% accuracy on service catalog tasks. In contrast, `WorkForceAgent-R1`, benefiting from RL's enhanced generalization, achieves a more balanced performance distribution across diverse task categories. (3) **Emergent Capabilities in Larger Models:** We observe emergent capabilities correlated with increased model sizes. Specifically, the 14B variants across all training strategies exhibit substantially improved performance over their smaller counterparts, indicating that larger parameter capacities facilitate more effective RL updates.

## 5.3 Training Recipes

**Training Dynamics.** Figure 4 presents training dynamics of `WorkForceAgent-R1`. The consistent improvements in average reward (Figure 4(a)) and validation accuracy (Figure 4(c)) demonstrate stable learning trajectories across training steps. Larger models attain higher rewards at an accelerated pace compared to smaller variants. Notably, Figure 4(b) indicates that the 14B-parameter model initially reduces response length, subsequently stabilizing and then increasing to accommodate richer reasoning, suggesting that longer reasoning chains contribute positively to improved performance.

**Effect of Long Reasoning.** Figure 5 compares the effectiveness of different training methodologies. Baseline models are directly adopted from `Qwen2.5-Instruct` models, while SFT models use the `o3-mini`-annotated trajectories from BrowserGym. For SFT-L models, we employ the `deepseek-ai/DeepSeek-R1-Distill-Llama-70B` to inject long-chain reasoning explicitly. The results clearly demonstrate the advantage of incorporating explicit reasoning into training, with SFT-L substantially outperforming standard SFT.

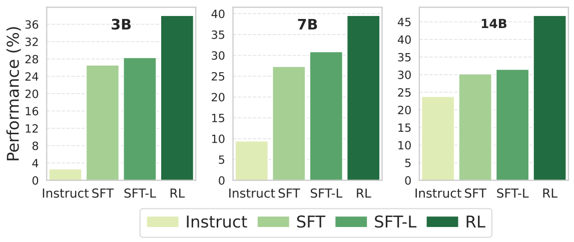


Figure 5: Performance comparison among different training strategies. "-L" denotes long reasoning models.

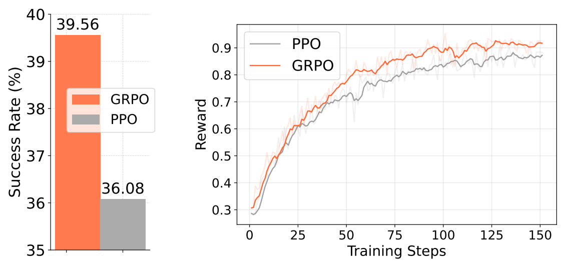


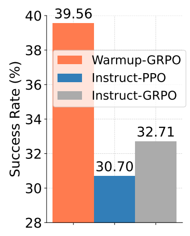
Figure 6: Comparison between PPO and GRPO with `Qwen2.5-7B-Instruct` as backbone LLM.

**Effect of RL Strategies.** Figure 6 illustrates the comparative analysis between the GRPO and Proximal Policy Optimization (PPO) algorithms. GRPO consistently achieves higher average rewards and demonstrates more rapid convergence, confirming its effectiveness in group-based relative optimization compared to the standard PPO algorithm.

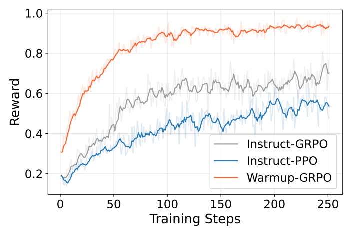
**Effect of Initial Checkpoint for RL.** Figure 7 evaluates the impact of initial model

Table 2: `LLaMA-3.1-8B` results on WorkArena (Drouin et al., 2024).

| **Baselines (↓) / Tasks (→)** | **Dashboard** | **Form** | **Knowledge** | **Filter** | **Sort** | **Menu** | **Service** | **Overall** | **Δ** |
|---|---|---|---|---|---|---|---|---|---|
| `Llama-3.1-8B-Instruct` (Dubey et al., 2024) | 5.00 | 10.00 | 20.00 | 0.00 | 0.00 | 12.00 | 18.60 | 8.57 | - |
| + SFT | 18.00 | 12.00 | 20.00 | 4.00 | 16.00 | 36.00 | 37.80 | 20.48 | (+11.91) |
| + SFT-L | **24.00** | <u>16.00</u> | <u>24.00</u> | <u>6.00</u> | <u>20.00</u> | <u>41.00</u> | <u>42.35</u> | <u>24.56</u> | (+15.99) |
| + `WorkForceAgent-R1` | <u>20.00</u> | **24.00** | **32.00** | **16.00** | **32.00** | **63.00** | **48.00** | **32.41** | (+23.84) |

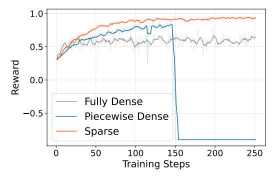


(a) Performance

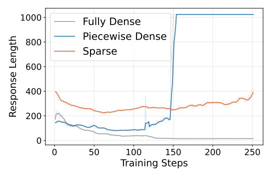


(b) Average Reward

Figure 7: Comparison between warmup model and instruct model as the initial checkpoint for RL training.



(a) Average Reward



(b) Average Response Length

Figure 8: Ablation on reward granularity, comparing our "Sparse" loss with fine-grained reward designs, "Fully Dense" and "Piecewise Dense" rewards.

(`Qwen2.5-7B-Instruct`) checkpoints on RL performance. The models initialized with an additional warm-up stage of SFT on a subset of $1,000$ training samples achieve better performance and higher average rewards compared to direct instruct-model initialization. This outcome indicates the critical role of preliminary SFT in effectively conditioning models for subsequent RL.

## 5.4 Ablation Study

**Effect of Reward Design.** We systematically investigate the influence of reward granularity on `WorkForceAgent-R1` training effectiveness. We compare three distinct schemes: (1) **Sparse Reward**, used by `WorkForceAgent-R1`, providing discrete positive feedback for correct formats and actions; (2) **Fully Dense Reward**, employing text similarity to assign partial credit continuously; and (3) **Piecewise Dense Reward**, applies sparse reward on function name and dense reward on parameters. Figure 8 demonstrates that the sparse reward scheme yields superior training stability and model performance. Conversely, fully and piecewise dense rewards lead to reward-hacking, evidenced by rapid yet unstable reward increments and drastically reduced response lengths, indicating superficial pattern memorization rather than meaningful reasoning.

**Dense Rewards Cause Reward Hacking.** Under the fully dense reward scheme, the model quickly achieves an average reward near $0.6$ but subsequently exhibits significant oscillations accompanied by a severe reduction in response length, approaching zero. Empirical analysis reveals reward hacking behavior, wherein the model repeatedly generates the most common action (*e.g.*, `click('a324')`) observed in prompts to consistently obtain partial rewards. Similarly, the piecewise dense reward scheme encourages reward hacking through infinite action generation following the terminating token `</action>`. Overly granular rewards incentivize superficial matching behaviors rather than genuine task-relevant reasoning.

**Effect of Backbone LLMs.** Table 2 summarizes the outcomes, demonstrating that although `LLaMA-3.1-8B-Instruct` performs relatively lower than similarly sized Qwen-2.5 models, applying the R1-style RL consistently enhances reasoning performance across diverse backbones.

## 6 Conclusion

In this paper, we present `WorkForceAgent-R1`, a rule-based RL framework designed to enhance the reasoning and planning capabilities of LLM-powered web agents, specifically tailored for workplace web navigation tasks. By introducing a progressive reward function that evaluates both format adherence and action accuracy, `WorkForceAgent-R1` enables LLM agents to implicitly develop robust reasoning skills without requiring explicit annotations of reasoning processes. Extensive experiments validate that `WorkForceAgent-R1` significantly surpasses SFT baselines and delivers competitive performance against state-of-the-art proprietary models.

WorkForceAgent-R1 highlights the effectiveness of RL in addressing the inherent complexity and dynamic nature of web interactions, paving the way for more generalizable and efficient web automation solutions in enterprise environments.

## Limitations

While WorkForceAgent-R1 demonstrates substantial improvements in workplace-oriented web navigation tasks, it has several inherent limitations. Firstly, there remains a performance gap between our open-source approach and API-based commercial LLMs, primarily due to disparities in scale, data quality, and fine-tuning methodologies. Secondly, WorkForceAgent-R1 explicitly focuses on workplace web navigation, which may limit its generalizability and effectiveness in other contexts or more diverse web browsing scenarios. Lastly, the reward function design, though carefully structured, may still incentivize superficial reward hacking behaviors, underscoring the need for continued refinement of reward mechanisms to better align agent incentives with genuine task execution correctness and reasoning depth.

## Privacy and Ethical Statement

We strictly adhere to ethical standards and privacy considerations. Our training and evaluation datasets are constructed exclusively from synthetic, heuristic-generated trajectories within controlled benchmarking environments, avoiding the use of sensitive or private data from real users. Consequently, we do not identify that WorkForceAgent-R1 poses any potential risks associated with data privacy or user confidentiality.

## References

Bo Adler, Niket Agarwal, Ashwath Aithal, Dong H Anh, Pallab Bhattacharya, Annika Brundyn, Jared Casper, Bryan Catanzaro, Sharon Clay, Jonathan Cohen, and 1 others. 2024. Nemotron-4 340b technical report. *arXiv preprint arXiv:2406.11704*.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024. Seeclick: Harnessing gui grounding for advanced visual gui agents. *arXiv preprint arXiv:2401.10935*.

De Chezelles, Thibault Le Sellier, Maxime Gasse, Alexandre Lacoste, Alexandre Drouin, Massimo Caccia, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, and 1 others. 2024. The browsergym ecosystem for web agent research. *arXiv preprint arXiv:2412.05467*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. 2024. Workarena: How capable are web agents at solving common knowledge work tasks? In *Forty-first International Conference on Machine Learning*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Hiroki Furuta, Kuang-Huei Lee, Ofir Nachum, Yutaka Matsuo, Aleksandra Faust, Shixiang Shane Gu, and Izzeddin Gur. 2024. Multimodal web navigation with instruction-finetuned foundation models. *The Twelfth International Conference on Learning Representations*.

Yu Gu, Yiheng Shu, Hao Yu, Xiao Liu, Yuxiao Dong, Jie Tang, Jayanth Srinivasa, Hugo Latapie, and Yu Su. 2024. Middleware for LLMs: Tools are instrumental for language agents in complex environments. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7646–7663, Miami, Florida, USA. Association for Computational Linguistics.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. 2023. A real-world webagent with planning, long context understanding, and program synthesis. *arXiv preprint arXiv:2307.12856*.

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

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Hongming Zhang, Tianqing Fang, Zhenzhong Lan, and Dong Yu. 2024b. Openwebvoyager: Building multimodal web agents via iterative real-world exploration, feedback and optimization. *arXiv preprint arXiv:2410.19609*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Lawrence Jang, Yinheng Li, Dan Zhao, Charles Ding, Justin Lin, Paul Pu Liang, Rogerio Bonatti, and Kazuhito Koishida. 2024. Videowebarena: Evaluating long context multimodal agents with video understanding web tasks. *arXiv preprint arXiv:2410.19100*.

Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, and 1 others. 2024. Autowebglm: A large language model-based web navigating agent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5295–5306.

Thibault Le Sellier De Chezelles, Maxime Gasse, and 1 others. 2024. The browsergym ecosystem for web agent research. *arXiv e-prints*, pages arXiv–2412.

Ido Levy, Ben Wiesel, Sami Marreed, Alon Oved, Avi Yaeli, and Segev Shlomov. 2024. St-webagentbench: A benchmark for evaluating safety and trustworthiness in web agents. *arXiv preprint arXiv:2410.06703*.

Changhao Li, Yuchen Zhuang, Rushi Qiang, Haotian Sun, Hanjun Dai, Chao Zhang, and Bo Dai. 2024. Matryoshka: Learning to drive black-box llms with llms. *arXiv preprint arXiv:2410.20749*.

Yusheng Liao, Shuyang Jiang, Yanfeng Wang, and Yu Wang. 2024. Reflectool: Towards reflection-aware tool-augmented clinical agents. *arXiv preprint arXiv:2410.17657*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, and 1 others. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Kaixin Ma, Hongming Zhang, Hongwei Wang, Xiaoman Pan, Wenhao Yu, and Dong Yu. 2023. Laser: Llm agent with state-space exploration for web navigation. *arXiv preprint arXiv:2309.08172*.

Peixian Ma, Xialie Zhuang, Chengjin Xu, Xuhui Jiang, Ran Chen, and Jian Guo. 2025. Sql-r1: Training natural language to sql reasoning model by reinforcement learning. *arXiv preprint arXiv:2504.08600*.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, and 1 others. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

OpenAI. 2025a. Introducing gpt-4.1 in the api. *OpenAI Blog*.

OpenAI. 2025b. Openai o3 and o4-mini system card. *OpenAI Blog*.

Jie Ouyang, Ruiran Yan, Yucong Luo, Mingyue Cheng, Qi Liu, Zirui Liu, Shuo Yu, and Daoyu Wang. 2025. Training powerful llm agents with end-to-end reinforcement learning.

Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. 2024a. Autonomous evaluation and refinement of digital agents. *arXiv preprint arXiv:2404.06474*.

Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, and 1 others. 2024b. Webcanvas: Benchmarking web agents in online environments. *arXiv preprint arXiv:2406.12373*.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, and 1 others. 2025. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *The Thirteenth International Conference on Learning Representations*.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*.

Qwen. 2025. Qwen3: Think deeper, act faster. *Qwen Blog*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. 2023. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2998–3009.

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. 2023. Adaplanner: Adaptive planning from feedback with language models. *Advances in neural information processing systems*, 36:58202–58245.

Haotian Sun, Yuchen Zhuang, Wei Wei, Chao Zhang, and Bo Dai. 2024. Bbox-adapter: Lightweight adapting for black-box large language models. In *ICML*.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*.

Zihan Wang, Kangrui Wang, Qineng Wang, and 1 others. 2025. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *Preprint*, arXiv:2504.20073.

Frank F Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, and 1 others. 2024. Theagentcompany: benchmarking llm agents on consequential real world tasks. *arXiv preprint arXiv:2412.14161*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Suyu Ye, Haojun Shi, Darren Shih, Hyokun Yun, Tanya Roosta, and Tianmin Shu. 2025. Realwebassist: A benchmark for long-horizon web assistance with real-world users. *arXiv preprint arXiv:2504.10445*.

Ori Yoran, Samuel Joseph Amouyal, Chaitanya Malaviya, Ben Bogin, Ofir Press, and Jonathan Berant. 2024. Assistantbench: Can web agents solve realistic and time-consuming tasks? *arXiv preprint arXiv:2407.15711*.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. 2024. Gpt-4v (ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, and 1 others. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

Yuchen Zhuang, Xiang Chen, Tong Yu, Saayan Mitra, Victor Bursztyn, Ryan A. Rossi, Somdeb Sarkhel, and Chao Zhang. 2024a. Toolchain*: Efficient action space navigation in large language models with a* search. In *The Twelfth International Conference on Learning Representations*.

Yuchen Zhuang, Haotian Sun, Yue Yu, Rushi Qiang, Qifan Wang, Chao Zhang, and Bo Dai. 2024b. HYDRA: Model factorization framework for black-box LLM personalization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Yuchen Zhuang, Jingfeng Yang, Haoming Jiang, and 1 others. 2025. Hephaestus: Improving fundamental agent capabilities of large language models through continual pre-training. *arXiv preprint arXiv:2502.06589*.

## A Task and Data Details

WorkArena (Drouin et al., 2024) is a benchmark designed to evaluate the capabilities of web agents performing common knowledge-worker tasks within enterprise software environments. Specifically, it comprises 33 diverse tasks derived from interactions on the widely-used ServiceNow platform, generating a total of $19,912$ unique task instances. Each task explicitly states a natural-language goal, ensuring clarity and consistency. The tasks span seven main categories, capturing representative workplace activities and workflows typically encountered by knowledge workers:

- **Lists (12 tasks, 6,900 instances)**: Tasks involve **filtering** and **sorting** data presented in tabular forms. Agents must construct complex filters with multiple conditional clauses or sort lists according to specified criteria. These operations require precise interaction with UI elements such as hidden menus and filter forms.
- **Forms (5 tasks, 5,000 instances)**: These tasks involve creating new entries through forms that vary significantly in complexity. Agents must accurately complete fields, manage dynamic auto-completion fields, navigate through hidden tabs, and handle date pickers. Task completion is validated via querying backend databases.
- **Knowledge Bases (1 task, 1,000 instances)**: Agents search through an enterprise knowledge base to retrieve specific answers to clearly stated questions. This requires keyword-based searches followed by careful navigation and textual extraction from multiple result pages. Validation checks whether the returned information matches the accepted formats.
- **Service Catalogs (9 tasks, 3,550 instances)**: Tasks require agents to navigate enterprise

product catalogs, select items with specific configurations, and complete order forms. Validation involves checking that orders placed by agents include the correct items with precise specifications.

- **Dashboards (4 tasks, 1,862 instances)**: Tasks focus on data retrieval from visual dashboards, requiring the agent to interpret numerical data from graphical charts and possibly perform simple reasoning tasks (e.g., identifying minimum or maximum values). Validation ensures the correctness of the extracted numeric values and labels.

- **Menus (2 tasks, 1,600 instances)**: Tasks require agents to navigate complex hierarchical menus or impersonate different user profiles to achieve specific goals. Validation includes confirming successful navigation to target locations or verifying user impersonation.

Each task includes carefully constructed validation functions, which provide immediate feedback by identifying errors, such as incorrectly filled fields or invalid selections. Additionally, tasks contain oracle functions implemented using Playwright browser automation scripts. These scripts not only demonstrate task feasibility but also supply ground-truth solutions for training and benchmarking purposes. The ServiceNow platform introduces unique challenges through its dynamic UI elements, non-standard and proprietary HTML implementations, and large-scale DOM structures, often ranging from 40k to 500k tokens after cleaning. Consequently, WorkArena tasks demand sophisticated context-awareness and robust generalization capabilities from the web agents.

## B Prompt Details

### B.1 Qwen-Style Prompt Template

**Qwen-Style Prompt Template**

```
<|im_start|>system
You are a helpful assistant. You first
     thinks about the reasoning
    process in the mind and then
    provides the user with the answer
    .<|im_end|>
<|im_start|>user
(prompt)<|im_end|>
<|im_start|>assistant
Let me solve this step by step.
<think>
```

### B.2 WorkArena Prompt Template – Instruction

**Instruction Templates**

```
You are an agent trying to solve a web
     task based on the content of the
    page and
user instructions. You can interact
    with the page and explore, and
    send messages to the user. Each
    time you
submit an action it will be sent to
    the browser and you will receive a
     new page.
# Instructions
Review the current state of the page
    and all other information to find
    the best
possible next action to accomplish
    your goal. Your answer will be
    interpreted
and executed by a program, make sure
    to follow the formatting
    instructions.

## Goal:
(User Query)
```

### B.3 WorkArena Prompt Template – Observation

**Observation Templates**

```
# Observation of current step:

## AXTree:
Note: [bid] is the unique alpha-
    numeric identifier at the
    beginning of lines for each
    element in the AXTree. Always use
    bid to refer to elements in your
    actions.

Note: You can only interact with
    visible elements. If the "visible"
     tag is not
present, the element is not visible on
     the page.

RootWebArea 'Classic | Unified
    Navigation App | ServiceNow'
[47] generic, live='assertive', atomic
    , relevant='additions text'
[48] generic, live='polite', atomic,
    relevant='additions text'
[53] generic, live='polite', atomic,
    relevant='all'
[56] navigation 'Global skip links'
(Web Page AXTree)
```

### B.4 WorkArena Prompt Template – Action

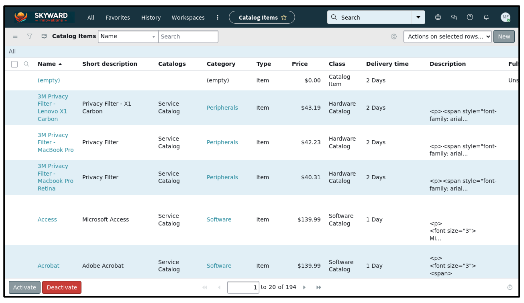
**Observation Templates**

```
# History of interaction with the task
    :
(Action History)

# Action space:
(Action Definition)

(Notes)

(Example)
```

## C Case Study

Figure 9 and Figure 10 illustrate a detailed example of `WorkForceAgent-R1` solving a `List-Sorting` task within WorkArena, where the agent must identify specific features and perform appropriate sorting actions. At each step, `WorkForceAgent-R1` generates a comprehensive reasoning sequence, including target identification, detailed observation analysis, and precise action prediction. These extensive reasoning steps culminate in accurately executed actions, clearly demonstrating `WorkForceAgent-R1`'s effectiveness in systematically navigating complex web interactions.

Figure 11 provides an additional case study on a `Service Catalog` task. This example underscores a critical characteristic of web-browsing tasks: each action step significantly transforms the webpage content and corresponding observations (HTML and AXTree structures), rendering multi-step forward planning practically infeasible. The unpredictable and dynamic nature of the observed changes after each interaction highlights the necessity of robust, single-step reasoning capabilities. Consequently, this insight validates our approach of emphasizing single-step decision-making offering valuable guidance for designing future web agent strategies in similarly dynamic environments.



| Name | Short description | Catalogs | Category | Type | Price | Class | Delivery time | Description | Ful |
|---|---|---|---|---|---|---|---|---|---|
| (empty) | | | (empty) | Item | $0.00 | Catalog Item | 2 Days | | Uns |
| 3M Privacy Filter - Lenovo X1 Carbon | Privacy Filter - X1 Carbon | Service Catalog | Peripherals | Item | $43.19 | Hardware Catalog | 2 Days | <p><span style="font-family: arial... | |
| 3M Privacy Filter - MacBook Pro | Privacy Filter | Service Catalog | Peripherals | Item | $42.23 | Hardware Catalog | 2 Days | <p><span style="font-family: arial... | |
| 3M Privacy Filter - Macbook Pro Retina | Privacy Filter | Service Catalog | Peripherals | Item | $40.31 | Hardware Catalog | 2 Days | <p><span style="font-family: arial... | |
| Access | Microsoft Access | Service Catalog | Software | Item | $139.99 | Software Catalog | 1 Day | <p> <font size="3"> Mi... | |
| Acrobat | Adobe Acrobat | Service Catalog | Software | Item | $139.99 | Software Catalog | 1 Day | <p> <font size="3"> <span> | |

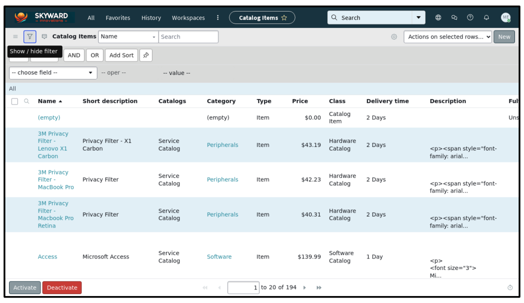
Okay, so I'm trying to figure out the next step to sort the "catalog items" list by Access Type in descending order. Let me look at the current state of the page and the history of interactions to understand where I am.

From the AXTree, I can see that the focused element is bid '44', but I'm not sure what that element is. Regardless, my goal is to sort the list based on the given field.

To do this, the most effective way is to use the sorting functionality in the list. In the AXTree, I notice that there's a button with the label "Show / hide filter" and the bid 'a46'. By clicking this button, the sorting options should appear, allowing me to set the primary sorting field to Access Type in the desired order.

action:
click('a46')

---



| Name | Short description | Catalogs | Category | Type | Price | Class | Delivery time | Description | Ful |
|---|---|---|---|---|---|---|---|---|---|
| (empty) | | | (empty) | Item | $0.00 | Catalog Item | 2 Days | | Uns |
| 3M Privacy Filter - Lenovo X1 Carbon | Privacy Filter - X1 Carbon | Service Catalog | Peripherals | Item | $43.19 | Hardware Catalog | 2 Days | <p><span style="font-family: arial... | |
| 3M Privacy Filter - MacBook Pro | Privacy Filter | Service Catalog | Peripherals | Item | $42.23 | Hardware Catalog | 2 Days | <p><span style="font-family: arial... | |
| 3M Privacy Filter - Macbook Pro Retina | Privacy Filter | Service Catalog | Peripherals | Item | $40.31 | Hardware Catalog | 2 Days | <p><span style="font-family: arial... | |
| Access | Microsoft Access | Service Catalog | Software | Item | $139.99 | Software Catalog | 1 Day | <p> <font size="3"> Mi... | |

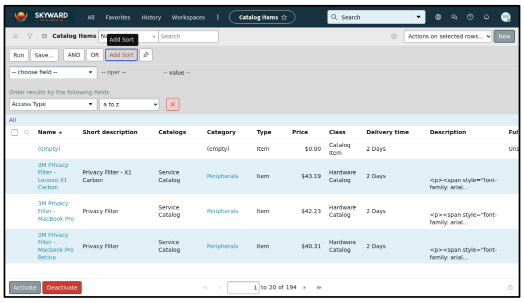
Okay, so I'm trying to figure out the next step to sort the "catalog items" list by Access Type in descending order. Let me look at the current state of the page and the history of interactions to understand where I am.

From the AXTree, I can see that the focused element is bid 'a46', which is the filter section. In the history of interactions, the last action was clicking on 'a46', which opened the filter section.

Now, to sort the list, I need to use the sorting feature. The correct action is to click the "Add Sort" button to open the sorting dialog. This will allow me to specify the field to sort by, which in this case is Access Type in descending order.

action:
click('a1029')

---



| Name | Short description | Catalogs | Category | Type | Price | Class | Delivery time | Description | Ful |
|---|---|---|---|---|---|---|---|---|---|
| (empty) | | | (empty) | Item | $0.00 | Catalog Item | 2 Days | | Uns |
| 3M Privacy Filter - Lenovo X1 Carbon | Privacy Filter - X1 Carbon | Service Catalog | Peripherals | Item | $43.19 | Hardware Catalog | 2 Days | <p><span style="font-family: arial... | |
| 3M Privacy Filter - MacBook Pro | Privacy Filter | Service Catalog | Peripherals | Item | $42.23 | Hardware Catalog | 2 Days | <p><span style="font-family: arial... | |
| 3M Privacy Filter - Macbook Pro Retina | Privacy Filter | Service Catalog | Peripherals | Item | $40.31 | Hardware Catalog | 2 Days | <p><span style="font-family: arial... | |

Okay, so I'm trying to figure out the next step to sort the "catalog items" list by Access Type in descending order. Let me look at the current state of the page and the history of interactions to understand where I am.

From the AXTree, I can see that the focused element is bid 'a1029', which is the "Add Sort" button. In the history of interactions, the last action was clicking on 'a1029', which opened the sorting dialog.

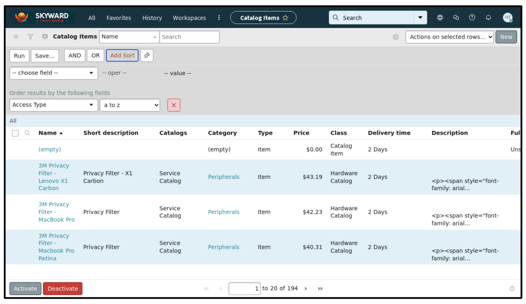
Now, I need to set the primary sorting field to Access Type in descending order. To do this, I need to select the correct field from the sorting options.

In the combobox for sorting, I need to choose the primary field to sort by. The combobox is labeled with the fields it can sort by. To sort by Access Type, I need to select that field from the sorting options.

So, the logical next step is to select the primary sorting field from the dropdown. This will apply the sorting as required.

action:
select_option('a1231', 'Access Type')

Figure 9: Case study of successful steps 1-3 of `WorkForceAgent-R1` solving a task of `List-Sorting`.



Catalog Items | Name | Search | Actions on selected rows...

-- choose field -- | -- oper -- | -- value --

Order results by the following fields

Access Type | a to z

All

| Name ▲ | Short description | Catalogs | Category | Type | Price | Class | Delivery time | Description | Ful |
|---|---|---|---|---|---|---|---|---|---|
| (empty) | | | (empty) | Item | $0.00 | Catalog Item | 2 Days | | Uns |
| 3M Privacy Filter - Lenovo X1 Carbon | Privacy Filter - X1 Carbon | Service Catalog | Peripherals | Item | $43.19 | Hardware Catalog | 2 Days | <p><span style="font-family: arial... | |
| 3M Privacy Filter - MacBook Pro | Privacy Filter | Service Catalog | Peripherals | Item | $42.23 | Hardware Catalog | 2 Days | <p><span style="font-family: arial... | |
| 3M Privacy Filter - Macbook Pro Retina | Privacy Filter | Service Catalog | Peripherals | Item | $40.31 | Hardware Catalog | 2 Days | <p><span style="font-family: arial... | |

1 to 20 of 194

Okay, so I'm trying to figure out the next step to sort the "catalog items" list by Access Type in descending order. Let me look at the current state of the page and the history of interactions to understand where I am.

From the AXTree, I can see that the focused element is bid 'a1029', which is the "Add Sort" button. In the history of interactions, the last action was clicking on 'a1029', which opened the sorting dialog.

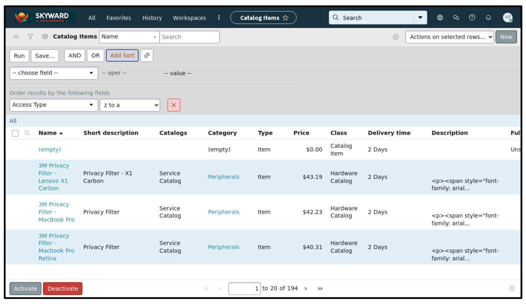
Now, I need to set the primary sorting field to Access Type in descending order. To do this, I need to select the correct field from the sorting options.

In the combobox for sorting, I need to choose the primary field to sort by. The combobox is labeled with the fields it can sort by. To sort by Access Type, I need to select that field from the sorting options.

So, the logical next step is to select the primary sorting field from the dropdown. This will apply the sorting as required.

action:
select_option('a1231', 'Access Type')

---



Catalog Items | Name | Search | Actions on selected rows...

-- choose field -- | -- oper -- | -- value --

Order results by the following fields

Access Type | z to a

All

| Name ▲ | Short description | Catalogs | Category | Type | Price | Class | Delivery time | Description | Ful |
|---|---|---|---|---|---|---|---|---|---|
| (empty) | | | (empty) | Item | $0.00 | Catalog Item | 2 Days | | Uns |
| 3M Privacy Filter - Lenovo X1 Carbon | Privacy Filter - X1 Carbon | Service Catalog | Peripherals | Item | $43.19 | Hardware Catalog | 2 Days | <p><span style="font-family: arial... | |
| 3M Privacy Filter - MacBook Pro | Privacy Filter | Service Catalog | Peripherals | Item | $42.23 | Hardware Catalog | 2 Days | <p><span style="font-family: arial... | |
| 3M Privacy Filter - Macbook Pro Retina | Privacy Filter | Service Catalog | Peripherals | Item | $40.31 | Hardware Catalog | 2 Days | <p><span style="font-family: arial... | |

1 to 20 of 194

Okay, so I'm trying to figure out the next step to sort the "catalog items" list by Access Type in descending order. Let me look at the current state of the page and the history of interactions to understand where I am.

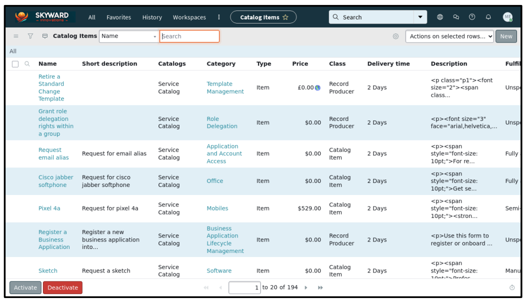
From the AXTree, I can see that the focused element is bid 'a1029', which is the "Add Sort" button. In the history of interactions, the last action was selecting 'Access Type' from the sorting fields.

Now, I need to set the sort order for Access Type to descending. To do this, I need to change the combobox for Access Type from the default order to z to a.

So, the correct action is to select the option in the combobox that corresponds to descending order. This will apply the sorting as required.

action:
select_option('a1335', 'descending')

---



Catalog Items | Name | Search | Actions on selected rows...

All

| Name | Short description | Catalogs | Category | Type | Price | Class | Delivery time | Description | Fulfil |
|---|---|---|---|---|---|---|---|---|---|
| Retire a Standard Change Template | | Service Catalog | Template Management | Item | £0.00 | Record Producer | 2 Days | <p class="p1"><font size="2"><span class... | Unsp |
| Grant role delegation rights within a group | | Service Catalog | Role Delegation | Item | $0.00 | Record Producer | 2 Days | <p><font size="3" face="arial,helvetica,... | Unsp |
| Request email alias | Request for email alias | Service Catalog | Application and Account Access | Item | $0.00 | Catalog Item | 2 Days | <p><span style="font-size: 10pt;">For re... | Fully |
| Cisco jabber softphone | Request for cisco jabber softphone | Service Catalog | Office | Item | $0.00 | Catalog Item | 2 Days | <p><span style="font-size: 10pt;">Get se... | Fully |
| Pixel 4a | Request for pixel 4a | Service Catalog | Mobiles | Item | $529.00 | Catalog Item | 2 Days | <p><span style="font-size: 10pt;"><stron... | Semi |
| Register a Business Application | Register a new business application into... | Service Catalog | Business Application Lifecycle Management | Item | $0.00 | Record Producer | 2 Days | <p>Use this form to register or onboard ... | Unsp |
| Sketch | Request a sketch | Service Catalog | Software | Item | $0.00 | Catalog Item | 2 Days | <p><span style="font-size: | Manu |

1 to 20 of 194

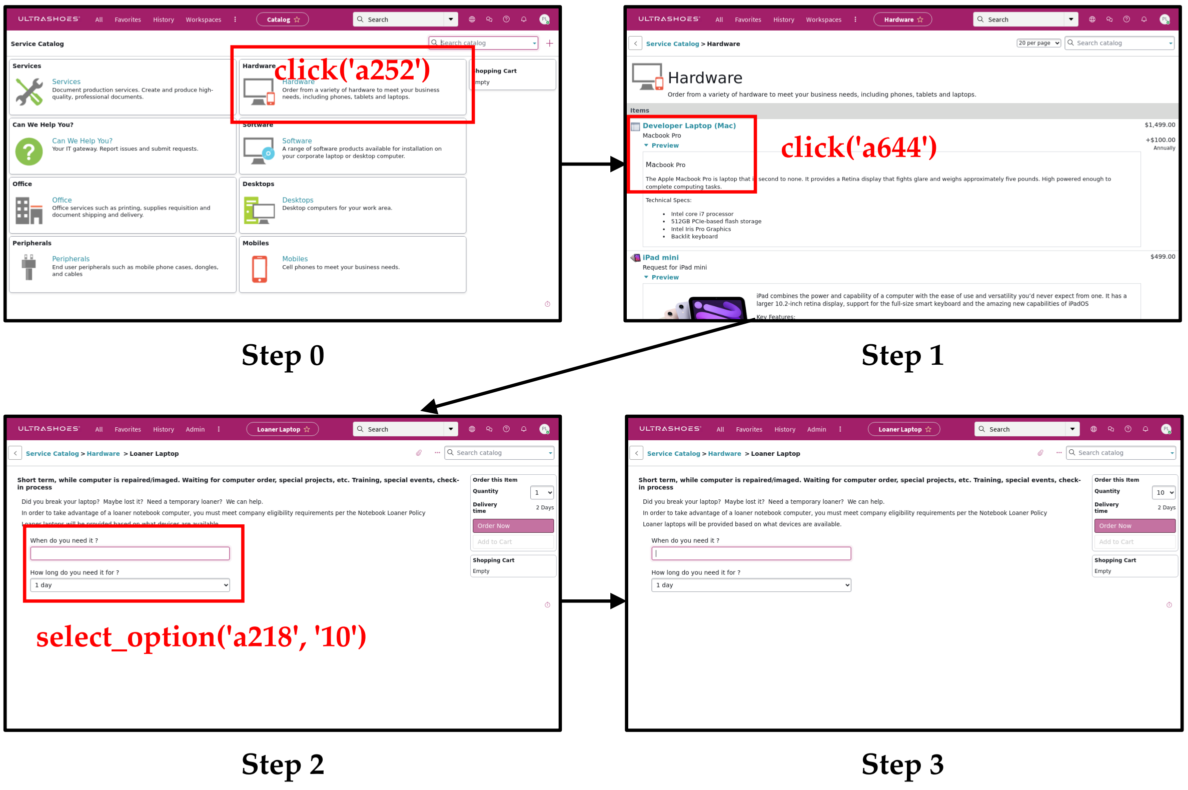
Okay, so I'm trying to figure out the next step to sort the "catalog items" list by Access Type in descending order. Let me look at the current state of the page and the history of interactions to understand where I am.

From the AXTree, I can see that the focused element is bid 'a1029', which is the "Add Sort" button. In the history of interactions, the last action was selecting 'descending' for the sorting field, which means the sorting is now correctly set to Access Type in the desired order.

Now, I need to apply these sorting settings to see the sorted list. The correct action is to run the sorting to execute the changes and see the results. This will likely refresh the list with the items sorted as required.

action:
click('a1024')

Figure 10: Case study of successful steps 4-6 of `WorkForceAgent-R1` solving a task of `List-Sorting`.



Service Catalog

Services
Services
Document production services. Create and produce high-quality, professional documents.

Hardware
Hardware
Order from a variety of hardware to meet your business needs, including phones, tablets and laptops.

Shopping Cart
Empty

Can We Help You?
Can We Help You?
Your IT gateway. Report issues and submit requests.

Software
Software
A range of software products available for installation on your corporate laptop or desktop computer.

Office
Office
Office services such as printing, supplies requisition and document shipping and delivery.

Desktops
Desktops
Desktop computers for your work area.

Peripherals
Peripherals
End user peripherals such as mobile phone cases, dongles, and cables

Mobiles
Mobiles
Cell phones to meet your business needs.

Service Catalog > Hardware

Hardware
Order from a variety of hardware to meet your business needs, including phones, tablets and laptops.

Items

Developer Laptop (Mac) $1,499.00
Macbook Pro +$100.00 Annually
Preview

Macbook Pro

The Apple Macbook Pro is laptop that is second to none. It provides a Retina display that fights glare and weighs approximately five pounds. High powered enough to complete computing tasks.

Technical Specs:

- Intel core i7 processor
- 512GB PCIe-based flash storage
- Intel Iris Pro Graphics
- Backlit keyboard

iPad mini $499.00
Request for iPad mini
Preview

iPad combines the power and capability of a computer with the ease of use and versatility you'd never expect from one. It has a larger 10.2-inch retina display, support for the full-size smart keyboard and the amazing new capabilities of iPadOS

Key Features:

Service Catalog > Hardware > Loaner Laptop

**Short term, while computer is repaired/imaged. Waiting for computer order, special projects, etc. Training, special events, check-in process**

Did you break your laptop? Maybe lost it? Need a temporary loaner? We can help.

In order to take advantage of a loaner notebook computer, you must meet company eligibility requirements per the Notebook Loaner Policy

Loaner laptops will be provided based on what devices are available.

When do you need it ?

How long do you need it for ?
1 day

| | |
|---|---|
| Order this Item | |
| Quantity | 1 |
| Delivery time | 2 Days |

Shopping Cart
Empty

Service Catalog > Hardware > Loaner Laptop

**Short term, while computer is repaired/imaged. Waiting for computer order, special projects, etc. Training, special events, check-in process**

Did you break your laptop? Maybe lost it? Need a temporary loaner? We can help.

In order to take advantage of a loaner notebook computer, you must meet company eligibility requirements per the Notebook Loaner Policy

Loaner laptops will be provided based on what devices are available.

When do you need it ?

How long do you need it for ?
1 day

| | |
|---|---|
| Order this Item | |
| Quantity | 10 |
| Delivery time | 2 Days |

Shopping Cart
Empty

Figure 11: Case study of successful steps 0-3 of `WorkForceAgent-R1` solving a task of `Service Catalog`.