LEARNING TO CONTEXTUALIZE WEB PAGES FOR ENHANCED DECISION MAKING BY LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in large language models (LLMs) have led to a growing interest in developing LLM-based agents for automating web tasks. However, these agents often struggle with even simple tasks on real-world websites due to their limited capability to understand and process complex web page structures. In this work, we introduce LCoW, a framework for **L**earning language models to **Co**ntextualize complex **W**eb pages into a more comprehensible form, thereby enhancing decision making by LLM agents. LCoW decouples web page understanding from decision making by training a separate contextualization module to transform complex web pages into comprehensible format, which are then utilized by the decision-making agent. We demonstrate that our contextualization module effectively integrates with LLM agents of various scales to significantly enhance their decision-making capabilities in web automation tasks. Notably, LCoW improves the success rates of closed-source LLMs (e.g., Gemini-1.5-flash, GPT-4o, Claude-3.5-Sonnet) by an average of 15.6%, and demonstrates a 23.7% average improvement in success rates for open-source LMs (e.g., Llama-3.1-8B, Llama-3.1-70B) on the WorkArena benchmark. Moreover, the Gemini-1.5-flash agent with LCoW achieves state-of-the-art results on the WebShop benchmark, outperforming human experts.

1 INTRODUCTION

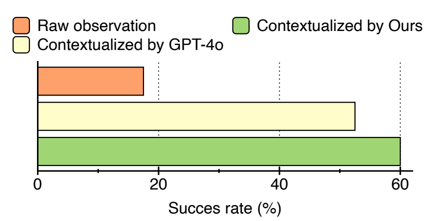

Figure 1: Success rate of the Gemini-1.5-flash agent on 40 WorkArena tasks. We selected a subset of 40 tasks by simply choosing the first 40 tasks based on the task indices. When the agent leverages observations contextualized by GPT-4o (yellow), its success rate improves by 31%, with further improvements achieved with our method (green).

Large language models (LLMs) have demonstrated strong potential in automating web tasks by treating web browsing as a sequential decision-making process, where web pages serve as observations and user interactions, such as clicking and typing, function as actions (Yao et al., 2022a;b). Various approaches have been developed to enhance the performance of LLM agents in these tasks. One such method involves fine-tuning open-source LLMs using demonstration data from web browsing tasks (Furuta et al., 2023; Lai et al., 2024). While promising, this approach requires extensive data collection and significant computational resources for effective fine-tuning. Alternatively, several studies have utilized advanced closed-source LLMs, such as GPT-4o (OpenAI, 2024), with carefully designed prompting techniques (Drouin et al., 2024; Zhou et al., 2023a; Sodhi et al., 2024; Pan et al., 2024). By leveraging the general world knowledge and reasoning capabilities of these models, the methods enhance web automation but at the cost of reduced controllability.

However, despite the advancements, state-of-the-art LLM agents often struggle to process complex raw web content, such as HTML and accessibility trees, posing significant challenges for their effective use in web task automation. While LLMs excel in tasks that require detailed reasoning, such as solving mathematical problems or coding, we hypothesize that their underperformance in seemingly

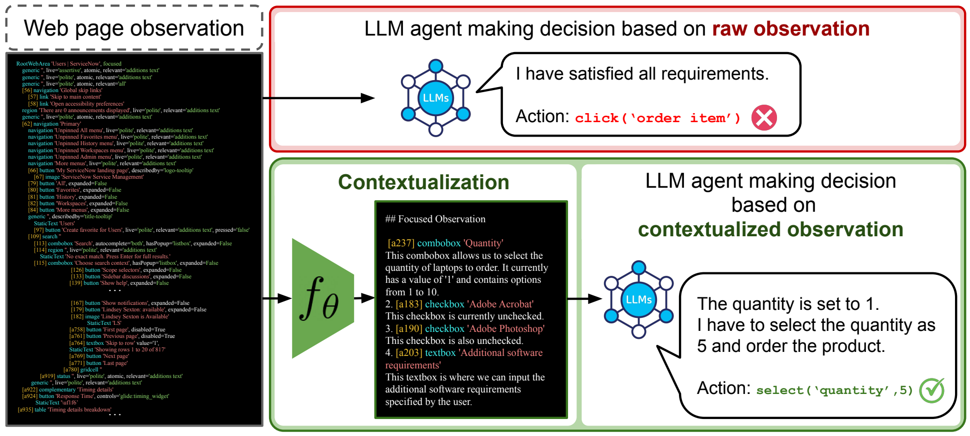

Figure 2: **(Top)** In the conventional pipeline, LLM agents decide on the next action based on raw, complex web page observations (e.g., HTML, accessibility trees), which often hinder accurate decision making. **(Bottom)** In our proposed pipeline, a contextualization module transforms these complex web page observations into a more comprehensible format, thereby enabling LLM agents to make more accurate decisions by enhancing their understanding of the web page.

simple decision-making tasks like web browsing is not due to a lack of decision-making capabilities but rather to difficulties in understanding and processing complex web page observations.

To validate our hypothesis, we conducted an initial experiment that demonstrated an LLM agent based on Gemini-1.5-flash can achieve substantial improvements in web browsing tasks when equipped with a module designed to contextualize complex web page observations (i.e., contextualization module). This module enhances task performance by removing irrelevant UI elements and highlighting key components with explanations, thereby simplifying the decision-making process. We evaluated the performance of this agent on 40 tasks from WorkArena (Drouin et al., 2024), a benchmark designed to assess web agents on real-world, enterprise-related websites. As shown in Figure 1, utilizing GPT-4o as the contextualization module (yellow) resulted in a 31% absolute improvement in the agent's success rate compared to direct processing of raw observations (red). These results support our hypothesis that the difficulty in understanding web pages is a major bottleneck for LLM-based web agents.

In this work, we propose LCoW, a framework that includes a contextualization module and a training algorithm to fine-tune this module to enhance the decision-making capabilities of LLM agents in web automation. As illustrated in Figure 2, the contextualization module transforms complex web page observations into a comprehensible format, enabling LLM agents to make more accurate decisions. Furthermore, to enable the contextualization module to provide context more grounded in real websites, we propose an iterative algorithm designed to train the contextualization module. Our training algorithm consists of three phases: (i) trajectory collection, (ii) sampling contextualized observations, and (iii) updating the contextualization module. Notably, the proposed method does not depend on manually curated data for training; instead, it gathers data through the agent's interactions within the web browsing environment. For each observation from the collected trajectories, we generate multiple contextualized observations using the current contextualization module. Each observation is then assigned a reward based on whether a set of LLM agents can accurately predict the correct action given the contextualized observation. Finally, we select the one with the highest reward as the target and train the contextualization module to maximize the likelihood of the target given the original raw observation.

As demonstrated in our initial experiment using the Gemini-1.5-flash (see Figure 1), LCoW significantly enhances the decision-making capabilities of LLM agents, even beyond the improvements seen with state-of-the-art LLMs like GPT-4o used as a contextualization module. In our experiments, we conduct comprehensive evaluations of our proposed approach on WebShop (Yao et al., 2022a), WorkArena (Drouin et al., 2024), and WebArena-Lite (Liu et al., 2024), popular benchmarks for evaluating agent performance in web environments. First, we demonstrate that LCoW significantly enhances the overall performance of LLM agents with varying scales (Llama-3.1-8B, Llama-3.1-

70B, Gemini-1.5-flash, GPT-4o, Claude-3.5-Sonnet) in WebShop and WorkArena, and also effective in WebArena-Lite when integrated to GPT-4o. Second, we analyze how the contextualization module refines complex web pages and how the contextualization enhances the decision-making of LLM agents.

2 BACKGROUND

In this section, we describe the formulation of web browsing as a sequential decision-making problem and the use of LLMs as decision-making agents.

Web browsing can be formulated as a Partially Observable Markov Decision Process (POMDP), defined by $\langle \mathcal{S}, \mathcal{O}, \mathcal{A}, T, \mathcal{R} \rangle$. The state $s_t \in \mathcal{S}$ represents the internal configuration of the web browser at time step t, which is only partially observable. The observation $o_t \in \mathcal{O}$ corresponds to the web page rendered by the web browser given s_t, which can take various forms (e.g., screenshot, HTML, accessibility trees). The action space $\mathcal{A}$ is the set of all possible interactions with the UI elements (e.g., clicking, typing). The state transition function T defines the probability of transitioning from state s to state s' after performing an action $a_t \in \mathcal{A}$, such as clicking a link or scrolling. While transitions are typically deterministic, occasional stochastic events (e.g., pop-ups, network errors) can occur. The reward function $\mathcal{R}$ assesses the functional correctness, evaluating whether the resulting state s_t satisfies pre-defined criteria for successful task completion.

Leveraging their ability to interpret web pages and generate actions in text form, LLMs are increasingly employed as agents for automating web-based tasks. In this framework, an LLM agent π generates an action a_t to interact with UI elements at each time step t, based on the user instruction `[TASK]`, the current web page observation o_t, and the history of previous actions $a_{<t}$. The objective of the agent is to complete the given task in order to maximize the reward.

3 METHOD

In this section, we present LCoW, a framework for enhancing the capability of LLM agents by contextualizing complex web pages. Section 3.1 outlines the concept of the contextualization module and its integration with LLM agents for web automation tasks. Section 3.2 introduces an iterative algorithm for training the contextualization modules to improve decision making of LLM agents.

3.1 CONTEXTUALIZATION MODULE

In this work, we decouple web page understanding from decision making of LLM agents. Our hypothesis is that while LLM agents possess strong decision-making capabilities, their performance can significantly degrade when the observations they rely on are lengthy and non-contextualized, such as HTML and accessibility trees. To address this limitation, we introduce a *contextualization module*, a separate language model designed to enhance LLM agents by contextualizing complex web page observations into a form that is more easily processed and comprehensible. Intuitively, a proper contextualization of observations can enhance the agent's understanding of web content and its decision making based on that understanding (see Figure 3 for an example of the input and output of the module). Formally, given a web page observation o_t at time step t, the objective of our contextualization module f_θ is to generate a contextualized observation o_t^{co} that serves as input to the LLM agent π to enhance its decision making. Specifically, f_θ uses the task instruction `[TASK]`, the previous actions of the agent $a_{<t}$, and the current web page observation o_t to generate a contextualized observation:

$$o_t^{\text{co}} = f_\theta(\texttt{[TASK]}, a_{<t}, o_t).$$

The LLM agent π then predicts the next action based on the contextualized observation:

$$a_t = \pi(\texttt{[TASK]}, a_{<t}, o_t^{\text{co}}).$$

While an arbitrary language model can serve as a contextualization module f_θ, it is important for the module to learn from experience in the web environment to provide more grounded context for decision making, such as role of a particular button or interaction with specific UI elements.

Role: Your task is to generate a "Reasoning" and a "Refined observation" based on the provided information.

Task: "Purchase me an iPad pro with silver color and ..."
Action history:

```
1. click [a32] ''Hardware"
2. click [a45] ''iPad pro"
```

Web page observation:

```
RootWebArea 'Catalog | ServiceNow'
    generic '', live='assertive', ...
    generic '', live='polite', ...
    [55] navigation 'Global skip links'
    ...
    [a157] rowgroup ''
        [a158] row ''
            [a159] gridcell ''
            ...
    [a401] LayoutTableRow ''
        [a402] LayoutTableCell 'Shopping Cart'
            [a403] LayoutTableCell 'Empty'
```

Reasoning: We have navigated to the Hardware store and clicked on the iPad pro link. The actions needed to accomplish the user instruction are to choose color option and disk option.

Refined observation: I can focus on following elements:

```
[a192] combobox 'Quantity' value='1', checked=true,
It confirms that 1 is already selected as a quantity.

[a195] radiogroup 'Mandatory'
    [a196] radio '\\uf Silver', checked=True
    [a197] radio '\\uf Space Gray'
This group contains the color options.

[a198] radiogroup 'Mandatory'
    [a200] radio '\\uf137 128 GB', checked=True
    [a203] radio '\\uf137 256 GB [add $100.00]'
This group contains the storage options.

[a206] button 'Order Now'
This button allows to finalize the process.
```

Figure 3: An example of a input of contextualization module including lengthy web page observation **(left)** and an observation contextualized by the contextualization module trained using LCoW **(right)**. The module converts raw observations into a more concise form to enhance decision making in agents. The prompt used is provided in Appendix A.4.

3.2 Algorithm for training the contextualization module

We now describe an iterative algorithm for training the contextualization module f_θ to enhance decision making of LLM agents. In a nutshell, the algorithm involves an iterative process of collecting paired input-output data for training the contextualization module and subsequently updating the module based on the collected data. For data collection, we begin by gathering trajectories of successfully completed tasks from the web browsing environment. For each observation o_t in the collected trajectories, we sample multiple candidate contextualized observations, and select the one that best provides the relevant context for multiple LLM agents to accurately predict the next action a_t. Based on the chosen target observations, we update f_θ via supervised fine-tuning. We now outline a single iteration of LCoW, followed by a detailed explanation of the design of the reward used for evaluating the candidate contextualized observations.

Single iteration A single iteration of LCoW starts with the contextualization module $f_{\theta^{(i)}}$ and aims to update this module to $f_{\theta^{(i+1)}}$. This process consists of three phases:

> **Step 1 (Trajectory collection).** Given a set of training tasks, we roll out the LLM agent π in the web environment to collect trajectory data. Specifically, the agent determines the next action based on the contextualized observation produced by $f_{\theta^{(i)}}$ until the episode terminates. We collect only those trajectories that end in the successful completion of the tasks.
>
> **Step 2 (Sampling optimal contextualization).** As illustrated in Figure 4, we start by sampling multiple candidates o_t^{co} from the current contextualization module $f_{\theta^{(i)}}$ (i.e., $o_t^{\text{co}} \sim f_{\theta^{(i)}}(\cdot \mid \texttt{[TASK]}, a_{<t}, o_t)$) for each web page observation o_t in the collected trajectories. Each candidate is then assigned a reward based on whether a set of LLM agents can accurately predict the ground-truth action a_t given o_t, with the candidate receiving the maximum reward selected as the optimal contextualized observation. If all candidates receive a zero reward, we retry the sampling process with the ground-truth action a_t provided as additional context to $f_{\theta^{(i)}}$ (i.e., $o_t^{\text{co}} \sim f_{\theta^{(i)}}(\cdot \mid \texttt{[TASK]}, a_{<t}, o_t, a_t)$) to guide the generation of valid contextualized observations.
>
> **Step 3 (Model update).** We update the current module $f_{\theta^{(i)}}$ by fine-tuning it with the optimal contextualized observations collected in Step 2. With the updated module $f_{\theta^{(i+1)}}$, we return to Step 1 and repeat the process.

Algorithm 1 outlines the steps for a single iteration of the training algorithm. Starting with an initial contextualization module $f_{\theta^{(0)}}$, we iteratively train the module through M iterations. After each

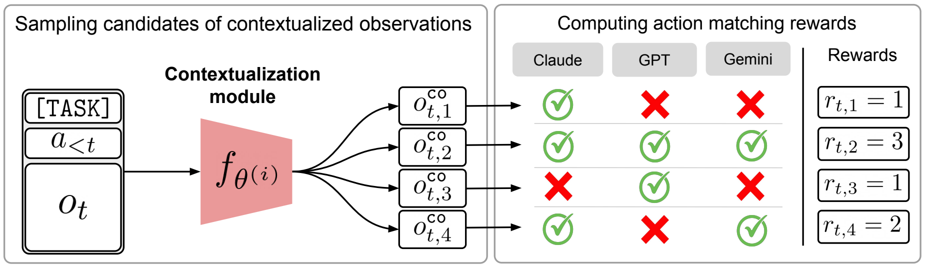

Figure 4: Illustration of sampling optimal contextualization. First, we sample multiple candidates of contextualized observations, given user instruction `[TASK]`, previous actions $a_{<t}$, and observation o_t. Subsequently, multiple LLM agents predict the next action based on each candidate, and the reward for each candidate is computed according to how many LLM agents correctly predict the ground-truth action a_t. In the figure, $o^{\text{co}}_{t,2}$ receives the highest reward and is therefore used as the target data for updating $f_{\theta^{(i)}}$.

Algorithm 1 One iteration of LCoW

Require: a contextualization module $f_{\theta^{(i)}}$, an LLM agent π, a set of LLM agents for computing the reward $\Pi = \{\pi_i\}_{i=1}^{K}$, a trajectory buffer $\mathcal{T}$, an empty data buffer $\mathcal{D}$, and a set of training tasks $\mathcal{G}_{tr}$

// Trajectory collection
for `[TASK]` $\in \mathcal{G}_{tr}$ **do** ▷ Collect successful trajectories from training environment
    $\tau, R \sim \pi(\cdot \mid \texttt{[TASK]}, f_{\theta^{(i)}})$
    if $R = 1.0$ **then**
        $\mathcal{T}$.append(τ)
// Sampling optimal contextualizations
for (o_t, a_t) in $\mathcal{T}$ **do**
    for $n \leftarrow 1$ to N **do** ▷ Sample N candidate contextualized observations and assign rewards
        $o^{\text{co}}_{t,n} \sim f_{\theta^{(i)}}(\cdot \mid \texttt{[TASK]}, a_{<t}, o_t)$
        $r_{t,n} = \sum_{\pi \in \Pi} \text{ActionMatchingScore}(\pi(\texttt{[TASK]}, a_{<t}, o^{\text{co}}_{t,n}), a_t)$
    if $\max_n(r_{t,n}) = 0$ **then**
        for $n \leftarrow 1$ to N **do** ▷ Retry the sampling if rewards are zero for all candidates
            $o^{\text{co}}_{t,n} \sim f_{\theta^{(i)}}(\cdot \mid \texttt{[TASK]}, a_{<t}, o_t, a_t)$
            $r_{t,n} = \sum_{\pi \in \Pi} \text{ActionMatchingScore}(\pi(\texttt{[TASK]}, a_{<t}, o^{\text{co}}_{t,n}), a_t)$
    $o^{\text{co}}_{t,*} = \arg\max_n(r_{t,n})$
    $\mathcal{D}$.append$([(\texttt{[TASK]}, a_{<t}, o_t), o^{\text{co}}_{t,*}])$
// Parameter update
$\theta^{(i+1)} := \arg\max_{\theta^{(i)}} \mathbb{E}_{(\texttt{[TASK]}, o^{\text{co}}_{t,*}, a_{<t}, o_t) \sim \mathcal{D}}[f_{\theta^{(i)}}(o^{\text{co}}_{t,*} \mid \texttt{[TASK]}, a_{<t}, o_t)]$
return $f_{\theta^{(i+1)}}$

iteration, the module updates from $f_{\theta^{(i)}}$ to $f_{\theta^{(i+1)}}$ until reaching the final $f_{\theta^{(M)}}$.[1] Additionally, $\mathcal{T}$ can be initialized as a set of human demonstrations in order to facilitate the training process.

Reward for contextualized observations The reward for the contextualized observation o^{co}_t is defined as the sum of the action-matching scores computed using multiple LLM agents. Each action matching score evaluates whether an LLM agent π correctly predicts the ground-truth action a_t given the contextualized observation o^{co}_t. By leveraging multiple LLM agents to compute the reward, we ensure that the module produces contextualized observations that generalize across a diverse set of agents, preventing overfitting to the behavior of any single agent and enhancing the module's adaptability to arbitrary LLM agents. Section 4.2 presents empirical results on the generalization capabilities of the contextualization module.

[1]In the initial iteration, we assume a limited set of seed demonstrations and use relatively strong LLMs to sample candidate contextualized observations to accelerate training.

4 EXPERIMENTS

We design our experiments to investigate the following questions:

- How effective is LCoW in training a contextualization module for improving decision making of LLM agents? (Section 4.2)
- Can the contextualization module trained with LCoW generalize to arbitrary LLMs with varying scales? (Section 4.2)
- What form do the web page observations take after contextualization and how the contextualized observation aid the decision making of LLM agents? (Section 4.3)
- Can the contextualization module trained with LCoW generalize to unseen task types or website domain? (Appendix A.1, A.2)

4.1 EXPERIMENTAL SETUP

Benchmarks We consider the three benchmarks, WebShop (Yao et al., 2022a), WorkArena (Drouin et al., 2024), and WebArena (Zhou et al., 2023a), which are popular benchmarks designed to evaluate the capabilities of web agents in completing various web tasks. WebShop provides a simulated online shopping environment with real-world product data, consisting of 500 evaluation tasks and 5,500 training tasks, each defined by natural language instructions to purchase products that meet specific criteria. During each decision-making step, the agent receives accessibility tree input to predict the next action, and at the end of an episode, it receives a reward ranging from 0 to 1 based on how well the attributes of the purchased item align with the intended product criteria. WorkArena, on the other hand, focuses on assessing web agents' ability to complete enterprise-related tasks, such as creating user accounts and ordering products from a service catalog, on real-world websites. This benchmark comprises 33 task types, with each type containing up to 1,000 individual task instances, where agents must navigate websites by following natural language instructions and receive rewards based on successful task completion. WebArena, although similar to WorkArena in being based on a realistic web environment, consists of 812 diverse tasks spanning over 6 websites (Shopping, Gitlab, Reddit, Map, Wikipedia, and content management systems). Additionally, WebArena-Lite (Liu et al., 2024) comprises 165 evaluation tasks filtered from original 812 tasks in WebArena, which we utilize for evaluation.

Evaluation Details In WebShop, we train the contextualization module using LCoW on environments associated with 500 of the 5,500 training tasks in WebShop. After training, we evaluate the module on 500 evaluation tasks, measuring both the success rate and average reward, with a task considered successful if the reward equals 1. To prompt the agent, we utilize a one-shot exemplar as described in Yao et al. (2022b). The maximum number of state transitions is limited to 10, and episodes are terminated if the same action is repeated more than twice. In the WorkArena, we utilized 5 task instances from each 33 task types (i.e., 165 evaluation tasks as a total) for evaluation tasks, and used disjoint 15 task instances for each 33 task types (i.e., 495 tasks as a total) for training tasks. In the WebArena, we utilized 165 tasks provided by WebArena-Lite as a evaluation tasks, while utilizing remaining 647 tasks in original WebArena as a training tasks. Here, for evaluation in both WorkArena and WebArena, the maximum number of state transitions is set to 20, and episodes are also terminated if the same action is repeated more than twice. For prompting the LLM agent, we use the prompt provided by BrowserGym (Drouin et al., 2024), a unified framework for the development and evaluation of web agents. The complete prompts used can be found in Appendix A.4, and training details are described in Appendix A.6.

4.2 MAIN RESULTS

We first demonstrate the effectiveness of LCoW on the WebShop and WorkArena benchmarks and show that the contextualization module trained via LCoW enhances performance even for an LLM agent that was not involved in the LCoW training process, such as Llama-3.1-70B. We evaluate against two baselines: (i) decision making based solely on raw observations without the contextualization module (i.e., base prompt), and (ii) decision making based on the observation contextualized by the LLM agent itself (i.e., self-contextualization). Additionally, we provide comparison to deci-

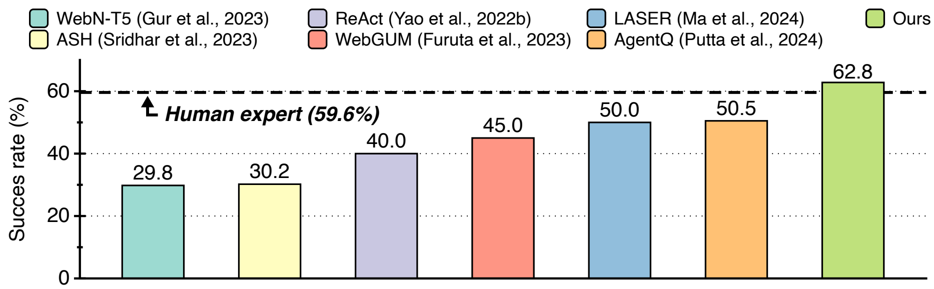

Figure 5: Success rate on 500 evaluation tasks from WebShop. Average human performance and expert human performance are 50% and 59.6%, respectively (Yao et al., 2022a). The Gemini-1.5-flash agent with the contextualization module trained for three iterations achieves a state-of-the-art success rate of 62.8%, outperforming the human expert performance, as well as previous baselines (Yao et al., 2022b; Furuta et al., 2023; Putta et al., 2024; Sridhar et al., 2023; Ma et al., 2024; Gur et al., 2023).

	GPT-4o		Gemini-1.5-flash		Claude-3.5-Sonnet		Llama-3.1-70B (Unseen)	
	Success	Reward	Success	Reward	Success	Reward	Success	Reward
Base prompt	34.8%	0.496	43.6%	0.693	26.6%	0.336	34.2%	0.590
Self-ctx	26.2%	0.459	46.4%	0.608	12.4%	0.146	40.2%	0.547
LCoW (iter 1)	27.8%	0.545	46.4%	0.705	39.4%	0.600	39.2%	0.666
LCoW (iter 2)	46.0%	0.647	58.2%	0.796	58.8%	**0.780**	55.0%	0.781
LCoW (iter 3)	**50.6%**	**0.666**	**62.8%**	**0.803**	**59.8%**	0.771	**59.6%**	**0.803**

Table 1: We investigate the efficacy of LCoW across multiple LLM agents in WebShop. For all LLM agents, LCoW consistently improves both success rate and reward over iterations, surpassing self-contextualization (self-ctx) and even human expert-level success rate by the third iteration. Additionally, LCoW is also effective when combined with Llama-3.1-70B, which was not used for computing the action-matching reward (i.e., unseen) during training the contextualization module.

sion making based on observations parsed by Reader-LM, an language model specialized to convert HTML into comprehensible format, in Appendix A.3.

Effectiveness of LCoW As shown in Table 1, both the GPT-4o and Claude-3.5-Sonnet agents perform poorly on the WebShop benchmark, frequently struggling to complete the tasks. Instead of selecting an item from the search results and reviewing its details, the agents often browse through multiple options in an attempt to find an exact match to the instructions, which frequently results in unsuccessful episode terminations. Although self-contextualization improves the success rate with Gemini-1.5-flash and Llama-3.1-70B, the performance of both the Claude-3.5-Sonnet and GPT-4o agents declines. In contrast, both the success rate and the average reward achieved by all three LLM agents improve substantially when integrated with the contextualization module trained with LCoW. Particularly, both the Gemini-1.5-flash and Claude-3.5-Sonnet agents surpass the average human performance of 50.5% when combined with the contextualization module trained for 2 iterations. When the contextualization module is trained for 3 iterations, the agents exceed the expert human-level performance of 59.6%. As illustrated in Figure 5, the Gemini-1.5-flash agent combined with LCoW achieves state-of-art performance on the WebShop benchmark, outperforming prior methods in success rate by more than 12%. Notably, the Claude-3.5-Sonnet agent, lower performing than the other agents on WebShop, achieves superhuman performance at 59.8% when integrated with our contextualization module, demonstrating that LCoW can effectively enhance the decision-making capabilities of LLM agents.

As shown in Table 2, on WorkArena, the Claude-3.5-Sonnet agent outperforms the GPT-4o agent, achieving a success rate of 44.8% in our evaluation setup. When integrated with LCoW, Claude-3.5-Sonnet agent achieves an even higher success rate of 55.8%, which is higher than all baselines evaluated. GPT-4o and Gemini-1.5-flash also improve 5% and 30%, respectively, when combined with LCoW. We describe the experiment result on WebArena-Lite benchmark in Appendix A.1.

	GPT-4o	Gemini-1.5-flash	Claude-3.5-Sonnet	Llama-3.1-70B (Unseen)	Llama-3.1-8B (Unseen)
Base prompt	38.2%	11.5%	44.8%	26.1%	1.2%
Self-ctx	43.0%	12.7%	50.3%	29.1%	7.3%
LCoW (iter 1)	**44.2%**	**41.2%**	**55.8%**	**40.0%**	**37.0%**

Table 2: We evaluate the success rate of five LLM agents with varying scales on 165 tasks in the WorkArena benchmark, which is based on realistic web environment.

Generalization to arbitrary LLM agents We evaluate the Llama-3.1-8B agent and Llama-3.1-70B agent integrated with LCoW to assess its effectiveness with LLM agents not involved in the training process (i.e., those not used for computing action-matching rewards). As shown in Table 1, the Llama-3.1-70B agent combined with the contextualization module trained for two iterations outperforms average human performance (50.0%) on WebShop and achieves the success rate of human experts (59.6%) when trained for three iterations. Furthermore, Table 2 shows that the contextualization module also enhances Llama-3.1-8B and Llama-3.1-70B agent on WorkArena, improving the success rate by approximately 36% and 13%, respectively. It is noteworthy that a relatively small LLM agent (i.e., Llama 3.1-8B) struggles to perform tasks when given raw observations, but when combined with LCoW, its success rate rises on significant margin. This result further supports our hypothesis that the bottleneck in LLM agents performing web automation tasks lies in their observation understanding capability rather than decision-making capability. It also demonstrates that LCoW can elicit a significant level of decision-making capability even from smaller models at the 8B scale.

4.3 Analysis

Can LCoW optimize the contextualization module for specific LLM agents? We demonstrate that LCoW effectively optimizes the contextualization module to generate context that leads to more accurate decision making by LLM agents. In the WebShop benchmark, we used the contextualization module checkpoint from each LCoW training iteration to generate contextualized observations for the 1,372 raw observations present in the 397 seed demonstrations. We then calculated the action-matching reward by comparing the actions predicted from these contextualized observations with the ground-truth actions derived from the demonstrations. Figure 6 shows the average action-matching reward across three iterations, demonstrating that with each round of training, the contextualization module learns to increasingly produce contextualized observations that enhance decision making of the LLM agents used in data collection (i.e., GPT-4o, Gemini-1.5-flash, Claude-3.5-Sonnet). This shows the effectiveness of LCoW in optimizing the contextualization module for specific LLM agents. Furthermore, it suggests the potential to extend LCoW as a method for indirectly tuning the behavior of closed-source LLM agents, where direct optimization is not feasible, across a wide range of text-based decision-making tasks.

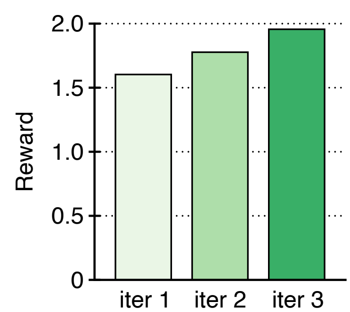

Figure 6: Average action-matching rewards across three iterations on WebShop show a consistent increase, suggesting that the contextualization module is optimized to generate observations that enhance decision-making in LLM agents.

How web pages are contextualized? We qualitatively analyze how the contextualization module, trained via LCoW, processes real-world web page observations and how the contextualization enhance decision making of LLM agents on web tasks. We observe that the contextualization module simplifies web page content by verbally explaining web page observation, extracting UI elements relevant to the given task and providing clear descriptions of their functionalities and interaction methods. These contextualized explanations aid LLM agents significantly in making more accurate decisions, while reducing the selection of inadmissible or redundant actions, which is the main failure mode of the LLM agent for web browsing, as shown in the Figure 7. Additional examples are provided in the Appendix A.5. Such knowledge about the functionality and interaction of UI elements ground to the real website appears to have been learned during the LCoW training process,

Contextualized observation:

```
The AXTree observation shows we are on the Shipping
Report page. The page contains a filter section with date
range inputs, order status options, and other report
parameters. There's also an export function and a table
to display the report results.
                            ...
Following elements are crucial for completing the user
instruction:
        [753] LabelText ''
            StaticText 'From'
            StaticText '*'
        [756] textbox 'From *' value='08/05/2022'
        [757] button 'undefined \ue627'
        [760] LabelText ''
            StaticText 'To'
            StaticText '*'
        [763] textbox 'To *' value='03/01/2023', focused
        [764] button 'undefined \ue627'
This section contains the filter parameters for the
shipping report. The date range inputs (From and To) have
already been filled with the specified dates (08/05/2022
and 03/01/2023, respectively).
```

Contextualized observation:

```
The AXTree observation shows a list of forums on a
website called Postmill. Each forum is presented as an
article with its name, number of subscribers, and number
of submissions. The forums are sorted by the number of
submissions in descending order.
                            ...
Additionally, for context, we can include the first forum
in the list to show where we are currently:
[141] article ''
    [143] heading 'AskReddit — AskReddit'
        [144] link 'AskReddit — AskReddit'
    [149] paragraph ''
    [152] Section ''
        [154] button 'Subscribe No subscribers'
            StaticText '0'
    [160] paragraph ''
        StaticText '10,041 submissions'
This extraction shows the first forum in the list, which
is "AskReddit" with 10,041 submissions. It confirms that
we are currently on the first page of forums and need to
navigate to the next page to continue our search for the
UpliftingNews subreddit.
```

Figure 7: Examples of how the contextualization module refines complicated web pages into comprehensible format. As indicated by blue color, the contextualization module provides comprehensible context by verbally explaining the web page and UI elements relevant to the given task.

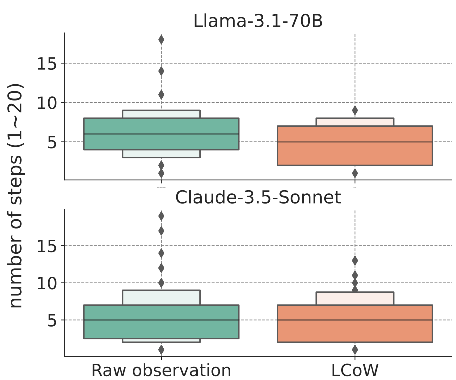

Figure 8: Comparison of the number of steps required to complete tasks in WorkArena. We only consider tasks that LCoW and baseline both succeed. Claude-3.5-Sonnet and Llama-3.1-70B agent both demonstrate efficient decision making by avoiding erroneous or redundant actions when the LCoW is utilized.

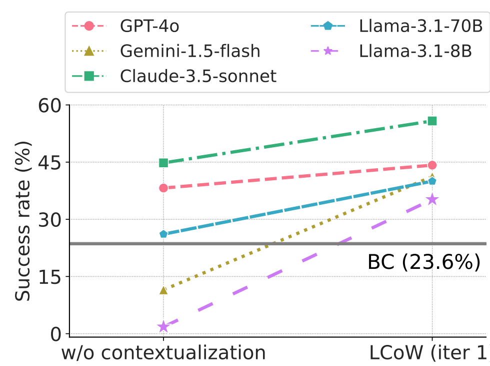

Figure 9: Comparison to training LLM agent via behavior cloning (BC). We fine-tune Llama-3.1-8B with demonstrations as a BC baseline. LLM agents combined with LCoW achieves much higher performance compared to BC baseline on 165 tasks in WorkArena

where the contextualization module explores various observation contextualizations and learns to generate contextualized observations that lead to more accurate decision making by LLM agents. Figure 8 shows that, with the contextualization module, the average number of decision-making steps by the agent decreases in both Claude-3.5-Sonnet and Llama-3.1-70B agent, allowing tasks to be completed more efficiently. More detailed examples are also provided in the Appendix A.5.

Comparison to directly training LLM agents One might argue that, given available demonstrations, it would be more straightforward to train the LLM agent directly rather than using the demonstrations to train a contextualization module. To demonstrate the superiority of training the contextualization module with an equivalent amount of demonstration data, we conduct comparative experiments against directly training LLM agent. In this analysis, we fine-tune Llama-3.1-8B with 264 seed demonstrations using behavior cloning (BC) as the baseline. To ensure a fair comparison in terms of model scale, we define a Llama-3.1-8B agent that performs tasks based on the output of a contextualization module trained using LCoW as the direct comparison group. As shown in

Figure 9, both smaller LLM agents, such as Llama-3.1-8B, and larger LLM agents achieve higher success rates when using the LCoW-trained contextualization module compared to the BC baseline.

Limitations While the contextualization module trained using LCoW has shown to enhance LLM agents by converting web pages into a comprehensible format and generalizes to unseen task types and website, it was hard to expect generalization to unseen UI elements too specific to certain functionalities, as detailed in Appendix A.1 and Appendix A.2. Although our experiment utilized small scale self-generated dataset based on handful of seed demonstrations for training the contextualization module, we believe that scaling up LCoW (e.g., utilizing large scale demonstrations covering wide range of web pages as seed demonstrations) can be an interesting future direction. Additionally, LCoW induces latency in decision making process due to computational cost required to generate the contextualized observation. Considering that most tokens of the contextualized observation corresponds to extraction from the raw observation, we expect LCoW can be integrated with efficient decoding strategy (e.g., speculative decoding) for resolving the latency problem.

5 Related work

LLM agents for web automation As LLMs continue to improve and demonstrate remarkable performance across diverse domains, developing web automation agents based on LLMs is gaining growing interest (Zhou et al., 2023a; Drouin et al., 2024). Recent works have explored various methods to utilize LLMs as agents for automating real-world web tasks. Pan et al. (2024) propose to apply a self-refine mechanism (Madaan et al., 2023) to improve decision making of agents through self-generated feedback. Sodhi et al. (2024) enhance web automation by using LLMs to manage low-level workflows handcrafted by humans, while Wang et al. (2024) extend this approach by introducing agent workflow memory, which extracts reusable routines from past experiences.

Similar to our work, a promising direction for enhancing web automation involves integrating a summarization module that condenses web page observations, enabling agents to predict actions based on these summarized inputs. For example, Deng et al. (2024) propose MindAct, which consists of an HTML extraction module and an action prediction module. The HTML extraction module ranks individual HTML elements using a ranking language model trained to assess relevance based on user instructions and past actions. Additionally, Gur et al. (2024) introduce HTML-T5, a specialized language model for web page understanding pre-trained on a large corpus of HTML documents and fine-tuned for extractive summarization. In contrast, we propose training language models to contextualize complex web page observations to enhance decision making in LLM agents.

Automated prompting for closed-source models The quality of outputs from closed-source LLMs heavily depends on the prompts used, leading to a substantial body of research dedicated to prompt engineering to elicit more effective responses from these models. (Kojima et al., 2022; Yao et al., 2022b; Lightman et al., 2024). For example, several recent studies have explored automating the discovery of more effective prompting formats (Shin et al., 2020; Zhou et al., 2023b). Mañas et al. (2024) demonstrate that prompt refinement can yield more consistent responses from closed-source models, especially with text-to-image models. Also, Xu et al. (2024) have suggested compressing the input content for LLMs, focusing on the task of retrieval-augmented generation. In this work, we focus on contextualizing the raw observations for agents performing sequential decision making and introduce a novel training algorithm.

6 Conclusion

In this work, we introduce LCoW, a novel approach to enhancing LLM agents in completing web tasks by leveraging language models to contextualize complex web pages into a more comprehensible form. Our approach separates the understanding of web content from the decision-making process by training a specialized module that generates contextualized representations of complex web pages, which are utilized by LLM agents for enhanced decision making. Through extensive experiments, we demonstrate that this contextualization module significantly improves the decision-making capabilities of LLM agents of varying scales.

Ethics statement

We introduce LCoW, a framework for improving decision-making capability of LLM agents for web automation. We caution that LLM agents may cause safety issues such as cybersecurity or risks regarding private information, while we believe that LCoW can be valuable for guiding the agents from potential mistakes by providing more contextualized information on the UI elements. Additionally, we believe the improved efficiency and capability of LLM agents with our methods can provide social opportunities to improve user interactions of using digital devices for those with disabilities.

Reproducibility statement

For the reproducibility of our results, we have provided a detailed description of our methods and experimental setups in Section 4.1. Additionally, to further facilitate the reproduction, we will release our codes and the model checkpoints in the final version.

References

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36, 2024.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, et al. Workarena: How capable are web agents at solving common knowledge work tasks? *arXiv preprint arXiv:2403.07718*, 2024.

Hiroki Furuta, Kuang-Huei Lee, Ofir Nachum, Yutaka Matsuo, Aleksandra Faust, Shixiang Shane Gu, and Izzeddin Gur. Multimodal web navigation with instruction-finetuned foundation models. *arXiv preprint arXiv:2305.11854*, 2023.

Izzeddin Gur, Ofir Nachum, Yingjie Miao, Mustafa Safdari, Austin Huang, Aakanksha Chowdhery, Sharan Narang, Noah Fiedel, and Aleksandra Faust. Understanding html with large language models, 2023.

Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. In *International Conference on Learning Representations*, 2024.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, et al. Autowebglm: Bootstrap and reinforce a large language model-based web navigating agent. *arXiv preprint arXiv:2404.03648*, 2024.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024.

Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Yifan Xu, Xixuan Song, Shudan Zhang, Hanyu Lai, Xinyi Liu, Hanlin Zhao, et al. Visualagentbench: Towards large multimodal models as visual foundation agents. *arXiv preprint arXiv:2408.06327*, 2024.

Kaixin Ma, Hongming Zhang, Hongwei Wang, Xiaoman Pan, Wenhao Yu, and Dong Yu. Laser: Llm agent with state-space exploration for web navigation, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023.

Oscar Mañas, Pietro Astolfi, Melissa Hall, Candace Ross, Jack Urbanek, Adina Williams, Aishwarya Agrawal, Adriana Romero-Soriano, and Michal Drozdzal. Improving text-to-image consistency via automatic prompt optimization. *arXiv preprint arXiv:2403.17804*, 2024.

OpenAI. https://openai.com/index/hello-gpt-4o/, 2024.

Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. Autonomous evaluation and refinement of digital agents. In *First Conference on Language Modeling*, 2024.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. Agent q: Advanced reasoning and learning for autonomous ai agents. *arXiv preprint arXiv:2408.07199*, 2024.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2020.

Paloma Sodhi, S. R. K. Branavan, Yoav Artzi, and Ryan McDonald. Step: Stacked llm policies for web actions, 2024.

Abishek Sridhar, Robert Lo, Frank F. Xu, Hao Zhu, and Shuyan Zhou. Hierarchical prompting assists large language model on web navigation, 2023.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory, 2024.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. Recomp: Improving retrieval-augmented lms with compression and selective augmentation. In *International Conference on Learning Representations*, 2024.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022b.

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

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *International Conference on Learning Representations*, 2023b.

A Appendix

A.1 WebArena-Lite

In this section, we provide experimental results in WebArena-Lite (Liu et al., 2024), consisting of diverse tasks spanning over 6 websites. WebArena-Lite is a benchmark composed of 165 tasks filtered from original WebArena benchmark (Zhou et al., 2023a). Out of entire 812 tasks in WebArena, we consider 165 tasks in WebArena-Lite as an evaluation split, and remaining 647 tasks as a training split. As a seed demonstration, we collect 363 successful trajectories from the training split via utilizing GPT-4o and Claude-3-5-sonnet. As a next step, we train the contextualization module via LCoW, and evaluate it on WebArena-Lite benchmark. As shown in the Table 3, LCoW results in remarkably better performance compared to the baseline, implying LCoW is effective in learning contextualization in broad range of websites.

WebArena-Lite (165 tasks)	GPT-4o
Raw observation	29.7%
LCoW	**35.8%**

Table 3: We evaluate the success rate of GPT-4o-2024-08-06 and LCoW + GPT-4o-2024-08-06 on 165 tasks in WebArena-Lite benchmark. Contextualizing web page observation via LCoW results in improved success rate.

Generalization to unseen task types Additionally, we analyze whether LCoW effectively generalizes to task types that were unseen during the training. Specifically, the 812 tasks in the WebArena benchmark are all different but are created based on 190 task templates. Therefore, tasks created from the same task template are different but similar. For example, “What is the top-1 best-selling product in 2022” and “What is the top-3 best-selling product in 2023” are from the same task template. Our research question is whether LCoW can generalize to unseen task templates that are not included in the 363 successful trajectories used for training. Among 165 WebArena-Lite evaluation tasks, 48 tasks belongs to unseen task templates, while 117 tasks belongs to seen task templates. Based on the information, we evaluate success rates on 117 tasks corresponding to seen task templates and 48 tasks corresponding to unseen task templates, respectively. As shown in Table 4, LCoW also generalizes to tasks corresponding to completely unseen task templates, which implies further improvement over LCoW iterations. We believe this is possible because contextualization module learns to explain the UI elements in the web pages via LCoW, which can be transferred to unseen types of tasks.

	GPT-4o	
	Seen-task-template (117 tasks)	Unseen-task-template (48 tasks)
Raw observation	35.9%	14.5%
LCoW	**41.9%**	**20.8%**

Table 4: LCoW also generalizes to unseen types of tasks, demonstrating more than 6% improvement in success rate on 48 tasks corresponding to the unseen-task-template, as well as 117 tasks corresponding to the seen-task-template.

Generalization to unseen websites Finally, we conduct experiment to confirm whether the contextualization module trained within certain set of websites generalizes to unseen website. The WebArena benchmark includes tasks across 6 websites (GitLab, CMS, Reddit, Map, Wikipedia, and Shopping). Therefore, we train contextualization module using data from 5 of these websites, excluding Shopping, and then tested on the 46 tasks corresponding to the Shopping website. As shown in the Table 5, contextualization module trained via LCoW generalizes to unseen websites, Shopping, demonstrating better performance compared to the baseline. Although websites are different, there are UI elements commonly used across websites. For example, UI elements for searching in Map or Gitlab is not different from search functionalities in the shopping website, and UI elements for date-based filtering in content management services or Reddit is also similar to that in the shopping website, which enables LCoW to generalize across different websites.

(Unseen website)	GPT-4o
Raw observation	17.4%
LCoW	**21.7%**

Table 5: LCoW+GPT-4o agent improves performance on tasks corresponding to the unseen websites, compared to the GPT-4o agent. Although websites are different, there are UI elements commonly used across websites (e.g., UI elements for searching in Map and UI elements for filtering results based on date in CMS), which enables LCoW to generalize across different website.

A.2 Generalization at different levels

In this section, we conduct systematic evaluation of LCoW's generalization capabilities in WorkArena. WorkArena features a two-level task hierarchy: categories at the top level and types within each category. WorkArena provides 7 categories of tasks (i.e., `Dashboard`, `Menus`, `Service catalog`, `Knowledge base`, `Forms`, `Sort list`, and `Filter list`). Although `Sort list` and `Filter list` are sub-category under the List task category, we determine to consider them as a discrete separate task category in order to avoid confusion. For instance, "Form" is a task *category*, and within it, "creating and submitting an incident report" and "creating new user information" are task *types*. We consider two levels of generalization: 1) Unseen-type tasks, tasks of a different type within the same category (i.e., medium generalization) and 2) Unseen-category tasks, tasks of a different type and category (i.e., hard generalization). Detailed information about seen and unseen tasks are provided in Table 6. In our experiments, we trained the contextualization module on 13 different tasks types and evaluated its performance on 100 individual tasks corresponding to 14 unseen-type tasks and 6 unseen-category tasks. As shown in the Table 6, LCoW demonstrated strong generalization to unseen-type tasks, achieving a 22.6% improvement when using Gemini 1.5-flash as the LLM agent. For example, knowledge learned from tasks of writing change requests and submitting can be generalized to creating new user account. However, we found LCoW struggles to generalize to unseen-category tasks (i.e., `Filter-list`) that contains UI elements (e.g., UI for utilizing hidden menu), highlighting the need for greater task diversity in training or enhanced contextual reasoning to address completely new task types.

	GPT-4o		Gemini-1.5-flash	
	Unseen-type	Unseen-category	Unseen-type	Unseen-category
Raw observation	35.7%	0.0%	14.5%	0.0%
LCoW	42.9%	0.0%	37.1%	0.0%

Table 6: LCoW shown to be generalized to unseen task types within the same task category on both GPT-4o and Gemini-1.5-flash backbone, but it struggles to generalize to tasks corresponding to unseen category.

A.3 Comparison with LLM-based parser

In this section, we provide experimental results comparing pre-trained LLM-based HTML parser and LCoW. As a LLM-based HTML parser, we utilize Reader-LM-1.5B, an LLM pre-trained to generate concise markdown given complicated HTML. Although Reader-LM generates reasonable summary when given non-lengthy HTML input (i.e., less than 2048 tokens), it tends to generate non-meaningful continuation given HTML longer than 20K tokens, which is prevalent in WorkArena benchmark. Therefore, we utilized accessibility tree observation, a textual web page representation with reduced noisy and enhanced readability. However, Reader-LM tend to repeat the given accessibility tree observations, rather than summarizing or rephrasing them, until it reaches maximum output token limit (i.e., 2048). As a result, as shown in 7 Reader-LM degrades success rates compared to using raw observation directly. Additionally, typical failure cases of Reader-LM are described in 10.

	GPT-4o
Raw observation	38.2%
Reader-LM	9.7%
LCoW (iter 1)	**44.2%**

Table 7: We compare the LCoW and Reader-LM, an LLM pre-trained to summarize web page into markdown format in WorkArena benchmark. However, we observed that Reader-LM struggles to summarize web page observations in WorkArena benchmark.

Task type	Seen / Unseen
Single-chart-value-retrieval	seen
Single-chart-minmax retrieval	seen
Multi-chart-value retrieval	unseen-type
Multi-chart-minmax retrieval	unseen-type
Create change request	seen
Create problem	seen
Create incident	unseen-type
Create hardward asset	unseen-type
Create user	unseen-type
Knowledge base search	seen
Sort user list	seen
Sort hardware list	seen
Sort asset list	unseen-type
Sort change request list	unseen-type
Sort incident list	unseen-type
Sort service-catalog list	unseen-type
All menu	seen
Impersonation	unseen-type
Order developer laptop	seen
Order iPad mini	seen
Order iPad pro	seen
Order apple watch	seen
Order standard laptop	seen
Order sales laptop	unseen-type
Order apple Macbook pro	unseen-type
Order development laptop PC	unseen-type
Order loander laptop	unseen-type
Filter asset list	unseen-category
Filter change request list	unseen-category
Filter hardware list	unseen-category
Filter incident list	unseen-category
Filter service catalog item list	unseen-category
Filter user list	unseen-category

Table 8: Task type split for systematic evaluation of generalization at different levels in WorkArena benchmark.

```
RootWebArea 'Search Results | iPad mini | ServiceNow', focused
        \[46\] generic, live='assertive', atomic, relevant='additions text'
        \[55\] navigation 'Global skip links'
                \[56\] link 'Skip to main content'
                \[57\] link 'Open accessibility preferences'
        \[234\] region 'There are 0 announcements displayed', live='polite',
relevant='additions text'
        \[58\] generic, live='polite', atomic, relevant='additions text'
        \[61\] navigation 'Primary'
                navigation 'Unpinned All menu', live='polite', relevant='additions text'
* * *
*   [AX Tree](https://www.ax.com/ax-tree)
*   [Accessibility Tree](https://www.ax.com/accessibility-tree)

*   [Accessibility Tree](https://www.ax.com/accessibility-tree)

*   [Accessibility Tree](https://www.ax.com/accessibility-tree)

*   [Accessibility Tree](https://www.ax.com/accessibility-tree)

*   [Accessibility Tree](https://www.ax.com/accessibility-tree)

*   [Accessibility Tree](https://www.ax.com/accessibility-tree)

*   [Accessibility Tree](https://www.ax.com/accessibility-tree)

*   [Accessibility Tree](https://www.ax.com/accessibility-tree)

*   [Accessibility Tree](https://www.ax.com/accessibility-tree)

*   [Accessibility Tree](https://www.ax.com/accessibility-tree)

*   [Accessibility Tree](https://www.ax.com/accessibility-tree)

*   [Accessibility Tree](https://www.ax.com/accessibility-tree)

*   [Accessibility Tree](https://www.ax.com/accessibility-tree)

*   [Accessibility Tree](https://www.ax.com/accessibility-tree)

*   [Accessibility Tree](https://www.ax.com/accessibility-tree)
```

```
RootWebArea 'Users | ServiceNow', focused
        \[47\] 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'
                \[57\] link 'Skip to main content'
                \[58\] link 'Open accessibility preferences'
        \[236\] region 'There are 0 announcements displayed', live='polite',
relevant='additions text'
        \[59\] generic, live='polite', atomic, relevant='additions text'
        \[62\] navigation 'Primary'
                navigation 'Unpinned All menu', live='polite', relevant='additions text'
                navigation 'Unpinned Favorites menu', live='polite', relevant='additions text'
                navigation 'Unpinned History menu', live='polite', relevant='additions text'
                navigation 'Unpinned Workspaces menu', live='polite', relevant='additions
text'
                navigation 'Unpinned Admin menu', live='polite', relevant='additions text'
                navigation 'More menus', live='polite', relevant='additions text'
                \[66\] button 'My ServiceNow landing page', describedby='logo-tooltip'
                        \[67\] image 'ServiceNow Service Management'
                \[79\] button 'All', expanded=False
                \[80\] button 'Favorites', expanded=False
                \[81\] button 'History', expanded=False
                \[82\] button 'Workspaces', expanded=False
                \[84\] button 'More menus', expanded=False
                generic, describedby='title-tooltip'
                        StaticText 'Users'
                        \[97\] button 'Create favorite for Users', live='polite',
relevant='additions text', pressed='false'
                \[109\] search ''
                        \[113\] combobox 'Search', autocomplete='both', hasPopup='listbox',
expanded=False
                        \[114\] region '', live='polite', relevant='additions text'
                                StaticText 'No exact match. Press Enter for full results.'
                        \[115\] combobox 'Choose search context', hasPopup='listbox',
expanded=False
                \[126\] button 'Scope selectors', expanded=False
                \[133\] button 'Sidebar discussions', expanded=False
                \[139\] button 'Show help', expanded=False
```

Figure 10: Reader-LM occasionally generates meaningless summarization of given web pages (Left), and mainly re-generates given web pages until the maximum output token limit has reached (Right).

A.4 Entire Prompts

In this section, we provide entire prompts used across entire experiments in WorkArena and WebShop.

A.4.1 WorkArena & WebArena

Prompt for contextualization module

```
<system prompt>
You are an agent tasked with extracting and refining a subset of the
webpage's observations based on the content of the page and user
instructions.

<main prompt>
You are currently on the {domain_info} website.
Your task is to generate a "Reasoning" and a "Refined observation"
based on the provided inputs.

First, review the "User instruction" and "History of interactions"
and, then, generate the "Reasoning".
Analyze the progress made so far, and provide a rationale for the
next steps needed to efficiently accomplish the user instruction on
the {domain_info} website.

Second, refine the "AXTree observation at the current time step"
into a "Refined observation".
Select a subset of the AXTree observation that is essential for
completing the user instruction and provide explanations for the
corresponding elements in the selected subset.

[Information source]
# User instruction
{goal}

# History of interactions
{history}

# AXTree observation at the current time step
{observation}
```

Prompt for contextualization module in retry phase

```
<system prompt>
You are an agent tasked with extracting and refining a subset of the
webpage's observations based on the content of the page and user
instructions.

<main prompt>
You are currently on the {domain_info} website.
Your task is to generate a "Reasoning" and a "Refined observation"
based on the provided inputs.

First, review the "User instruction" and "History of interactions"
and, then, generate the "Reasoning".
Analyze the progress made so far, and provide a rationale for the
next steps needed to efficiently accomplish the user instruction on
the {domain_info} website.

Second, refine the "AXTree observation at the current time step"
into a "Refined observation".
Select a subset of the AXTree observation that is necessary for
completing the user instruction.
```

```
You may refer to the Hints, which consists of the ground truth next
action, but do not explicitly mention these hints in your output.

[Information source]
# User instruction
{goal}

# History of interactions
{history}

# AXTree observation at the current time step
{observation}

# Hint
Ground-truth next action: {action}
```

Prompt for self-contextualization

```
<system prompt>
You are an agent tasked with extracting and refining a subset of the
webpage's observations based on the content of the page and user
instructions.

<main prompt>
[General instructions]
You are currently on the {domain_info} website.
Your task is to generate a "Reasoning" and a "Refined observation"
based on the provided inputs.

First, review the "User instruction" and "History of interactions" and,
then, generate the "Reasoning".
Analyze the progress made so far, and provide a rationale for the next
steps needed to efficiently accomplish the user instruction on
the {domain_info} website.

Second, refine the "AXTree observation at the current time step"
into a "Refined observation".
Extract a subset of the AXTree observation (e.g., chart, table,
menu items) that contains necessary information for completing
the user instruction, and explain the extracted elements.
Ensure that the information on the elements (e.g., numeric element ID)
are correctly included.

Please follow the format in the [Reasoning & Refinement example]
carefully.

[Information source]
# User instruction
{goal}

# History of interactions
{history}

# AXTree observation at the current time step
{observation}

[Reasoning & Refinement example]
# Abstract example
Here is an abstract version of the answer, describing
the content of each tag.
Make sure you follow this structure, but replace the
content with your own answer:
```

```
<reasoning>
Think step by step. Based on the "User instruction,",
"History of interaction," and "AXTree observation at the
current time step":
1. Provide a high-level description of the "AXTree observation at the
current time step."
2. Based on the "User instruction" and "History of interaction,"
track your progress and provide your reasoning on the next action
needed to accomplish the "User instruction."
</reasoning>

<extraction>
Based on your reasoning, identify the elements
(e.g., links, buttons, static text, table row, chart) to focus on.
Then, explain the semantics and functionalities of
each extracted element.
Ensure that:
You do not alter the structure of the AXTree observation.
You extract the element ID (id in [ ]) accurately without any errors.
When extracting chart or table, you must extract the entire chart
or table to avoid any confusion or loss of information.
</extraction>
```

Prompt for LLM agent

```
<system prompt>
You are an agent trying to solve a web task based on the content of
the page and a user instructions.
You can interact with the page and explore.
Each time you submit an action it will be sent to the browser and you
will receive a new page.

<main prompt>
# 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:
{goal}

{history}

# Refined observation of current step:
{refined observation}

# Action space:
13 different types of actions are available.
noop(wait_ms: float = 1000)
    Description: Do nothing, and optionally wait for
    the given time (in milliseconds).
    Examples:
        noop()
        noop(500)

send_msg_to_user(text: str)
    Description: Send a message to the user.
    You should send a short answer as a message and
    do not ask questions through message.
    Examples:
        send_msg_to_user(\'the city was built in 1751.\')
        send_msg_to_user(\'Yes\')
        send_msg_to_user(\'No\')
```

```
        send_msg_to_user(\'31112\')
        send_msg_to_user(\'Yoshua Bengio\')

scroll(delta_x: float, delta_y: float)
    Description: Scroll horizontally and vertically.
    Amounts in pixels, positive for right or down scrolling,
    negative for left or up scrolling. Dispatches a wheel event.
    Examples:
        scroll(0, 200)
        scroll(-50.2, -100.5)

fill(bid: str, value: str)
    Description: Fill out a form field.
    It focuses the element and triggers an input event
    with the entered text. It works for <input>,
    <textarea> and [contenteditable] elements.
    Examples:
        fill('237', 'example value')
        fill('45', 'multi-line\nexample')
        fill('a12', 'example with "quotes"')

select_option(bid: str, options: str | list[str])
    Description: Select one or multiple options in a <select> element.
    You can specify option value or label to select.
    Multiple options can be selected.
    Examples:
        select_option('48', 'blue')
        select_option('48', ['red', 'green', 'blue'])

click(bid: str, button: Literal['left', 'middle', 'right'] = 'left',
    modifiers: list[typing.Literal['Alt', 'Control', 'Meta', 'Shift']]
    = [])
    Description: Click an element.
    Examples:
        click('51')
        click('b22', button='right')
        click('48', button='middle', modifiers=['Shift'])

dblclick(bid: str, button: Literal['left', 'middle', 'right'] =
'left', modifiers: list[typing.Literal['Alt', 'Control', 'Meta',
'Shift']] = [])
    Description: Double click an element.
    Examples:
        dblclick('12')
        dblclick('ca42', button='right')
        dblclick('178', button='middle', modifiers=['Shift'])

hover(bid: str)
    Description: Hover over an element.
    Examples:
        hover('b8')

press(bid: str, key_comb: str)
    Description: Focus the matching element and press a combination of
    keys.
    It accepts the logical key names that are emitted in the
    keyboardEvent.
    key property of the keyboard events: Backquote, Minus, Equal,
    Backslash, Backspace, Tab, Delete, Escape, ArrowDown, End, Enter,
    Home, Insert, PageDown, PageUp, ArrowRight, ArrowUp, F1 - F12,
    Digit0 - Digit9, KeyA - KeyZ, etc. You can alternatively specify a
    single character you'd like to produce such as "a" or "#".
    Following modification shortcuts are also supported: Shift,
    Control, Alt, Meta.
```

```
    Examples:
        press('88', 'Backspace')
        press('a26', 'Control+a')
        press('a61', 'Meta+Shift+t')

focus(bid: str)
    Description: Focus the matching element.
    Examples:
        focus('b455')

clear(bid: str)
    Description: Clear the input field.
    Examples:
        clear('996')

drag_and_drop(from_bid: str, to_bid: str)
    Description: Perform a drag & drop.
    Hover the element that will be dragged.
    Press left mouse button.
    Move mouse to the element that will receive the drop.
    Release left mouse button.
    Examples:
        drag_and_drop('56', '498')

upload_file(bid: str, file: str | list[str])
    Description: Click an element and wait for a "filechooser" event,
    then select one or multiple input files for upload.
    Relative file paths are resolved relative to the current working
    directory.
    An empty list clears the selected files.
    Examples:
        upload_file('572', 'my_receipt.pdf')
        upload_file('63', ['/home/bob/Documents/image.jpg',
        '/home/bob/Documents/file.zip'])

Only a single action can be provided at once. Example:
fill('a12', 'example with "quotes"')
Multiple actions are meant to be executed sequentially without any
feedback from the page.
Don't execute multiple actions at once if you need feedback from the
page.

# 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 single action to be executed.
You can only use one action at a time.
</action>

# Concrete Example
Here is a concrete example of how to format your answer.
Make sure to follow the template with proper tags:
```

```
<think>
My memory says that I filled the first name and last name, but I can't
see any content in the form.
I need to explore different ways to fill the form.
Perhaps the form is not visible yet or some fields are disabled.
I need to replan.
</think>

<action>
fill('a12', 'example with "quotes"')
</action>
```

A.4.2 WebShop

Prompt for contextualization module

```
<system prompt>
You are an agent tasked with extracting and rephrasing a subset of
the webpage's observations based on the content of the page and user
instructions.

<main prompt>
You are currently on the online shopping website.
Your task is to generate a "Reasoning" and a "Refined observation"
based on the provided inputs.

First, review the "User instruction" and "History of interactions"
and, then, generate the "Reasoning".
Analyze the progress made so far, and provide a rationale for the
next steps needed to efficiently accomplish the user instruction on
the online shopping website.

Second, rephrase the "AXTree observation at the current time step"
into a "Rephrased observation".
Select a subset of the AXTree observation that is essential
for completing the user instruction and provide explanations
for the corresponding elements in the selected subset.

[Information source]
# User instruction
{goal}

# History of interactions
{previous_actions}

# AXTree observation at the current time step
{obs}
```

Prompt for self-contextualization

```
<system prompt>
You are an agent tasked with extracting and rephrasing a subset of
the webpage's observations based on the content of the page and user
instructions.

<main prompt>
The current webpage on the web shopping site is described
in the observation.
Evaluate the current progress based on previous actions
and current observation.
Determine the next action by reasoning based on
goal and progress.
Condense the observation into a concise format, highlighting
```

```
clickable buttons indicated by [].
Ensure the summary includes only elements relevant to the
goal and not already covered in previous actions.
Focus on clickable buttons indicated as [].

Here are a few examples.

**goal**: i would like a 3 ounce bottle of bright citrus
deodorant for sensitive skin, and price lower than 50.00 dollars
**previous actions**:
1. search[3 ounce bright citrus deodorant sensitive skin]
**current observation**:
[ Back to Search ]
Page 1 (Total results: 50)
[ Next > ]
[ B078GWRC1J ]
Bright Citrus Deodorant by Earth Mama | Natural and Safe
for Sensitive Skin, Pregnancy and Breastfeeding,
Contains Organic Calendula 3-Ounce
$10.99
[ B078GTKVXY ]
Ginger Fresh Deodorant by Earth Mama | Natural and Safe for
Sensitive Skin, Pregnancy and Breastfeeding, Contains Organic
Calendula 3-Ounce
$10.99
[ B08KBVJ4XN ]
Barrel and Oak - Aluminum-Free Deodorant, Deodorant for Men,
Essential Oil-Based Scent, 24-Hour Odor Protection, Cedar &
Patchouli Blend, Gentle on Sensitive Skin (Mountain Sage,
2.7 oz, 2-Pack)
$15.95

**rephrased observation**:
Progress: I searched the keyword '3 ounce bright citrus deodorant
sensitive skin' to see the relvant items, And now I am looking at
the item list.
Reasoning: the next step is to choose an item satisfying the
specification of bright citrus deodorant.
I can focus on:
[B078GWRC1J]
Bright Citrus Deodorant by Earth Mama | Natural and Safe for
Sensitive Skin, Pregnancy and Breastfeeding, Contains Organic
Calendula 3-Ounce
$10.99


**goal**: i would like a 3 ounce bottle of bright citrus deodorant
for sensitive skin, and price lower than 50.00 dollars
**previous actions**:
1. search[3 ounce bright citrus deodorant sensitive skin]
2. click[B078GWRC1J]
**current observation**:
[ Back to Search ]
[ < Prev ]
size
[ travel set (4-pack) ]
[ 3 ounce (pack of 1) ]
[ 3-ounce (2-pack) ]
scent
[ assorted scents ]
[ bright citrus ]
[ calming lavender ]
[ ginger fresh ]
```

```
[ simply non-scents ]
Bright Citrus Deodorant by Earth Mama | Natural and Safe for
Sensitive Skin, Pregnancy and Breastfeeding, Contains Organic
Calendula 3-Ounce
Price: $10.99
Rating: N.A.
[ Description ]
[ Features ]
[ Reviews ]
[ Buy Now ]

**rephrased observation**:
Progress: I searched and and clicked the item seems to be most
relevant to the goal specification. I am looking at the option list.
Reasoning: As the goal requires 3-ounce bottle, I can focus
on the size option.
I can focus on:
size
[ travel set (4-pack) ]
[ 3 ounce (pack of 1) ]
[ 3-ounce (2-pack) ]


**goal**: i would like a 3 ounce bottle of bright citrus deodorant
for sensitive skin, and price lower than 50.00 dollars
**previous actions**:
1. search[3 ounce bright citrus deodorant sensitive skin]
2. click[B078GWRC1J]
3. click[3 ounce (pack of 1)]
**current observation**:
You have clicked 3 ounce (pack of 1).
[ Back to Search ]
[ < Prev ]
size
[ travel set (4-pack) ]
[ 3 ounce (pack of 1) ]
[ 3-ounce (2-pack) ]
scent
[ assorted scents ]
[ bright citrus ]
[ calming lavender ]
[ ginger fresh ]
[ simply non-scents ]
Bright Citrus Deodorant by Earth Mama | Natural and Safe for
Sensitive Skin, Pregnancy and Breastfeeding, Contains Organic
Calendula 3-Ounce
Price: $10.99
Rating: N.A.
[ Description ]
[ Features ]
[ Reviews ]
[ Buy Now ]

**rephrased observation**:
Progress: I searched and and clicked the item id.
Among the option list, and I clicked size option.
Reasoning: According to the progress, I have to focus
on the scent option as a next step.
I can focus on:
scent
[ assorted scents ]
[ bright citrus ]
[ calming lavender ]
```

```
[ ginger fresh ]
[ simply non-scents ]


**goal**: i would like a 3 ounce bottle of bright citrus
deodorant for sensitive skin, and price lower than 50.00 dollars
**previous actions**:
1. search[3 ounce bright citrus deodorant sensitive skin]
2. click[B078GWRC1J]
3. click[3 ounce (pack of 1)]
4. click[bright citrus]
**current observation**:
You have clicked 3 ounce (pack of 1).
You have clicked bright citrus.
[ Back to Search ]
[ < Prev ]
size
[ travel set (4-pack) ]
[ 3 ounce (pack of 1) ]
[ 3-ounce (2-pack) ]
scent
[ assorted scents ]
[ bright citrus ]
[ calming lavender ]
[ ginger fresh ]
[ simply non-scents ]
Bright Citrus Deodorant by Earth Mama | Natural and Safe
for Sensitive Skin, Pregnancy and Breastfeeding,
Contains Organic Calendula 3-Ounce
Price: $10.99
Rating: N.A.
[ Description ]
[ Features ]
[ Reviews ]
[ Buy Now ]

**rephrased observation**:
Progress: Based on **observation** and **previous actions**,
I clicked size option and scent option.
Reasoning: As there is no more options to select and I met
all requirements specified in the goal, next step is to buy the item.
I can focus on:
[ Buy Now ]

Here is the task.

**goal**:
{goal}
**previous actions**:
{previous_actions}
**current observation**:
{obs}

**rephrased observation**:
```

Prompt for LLM agent

```
Webshop
Instruction:
i would like a 3 ounce bottle of bright citrus deodorant for sensitive
skin, and price lower than 50.00 dollars
[ Search ]
```

```
Action: search[3 ounce bright citrus deodorant sensitive skin]

Observation:

Progress: I searched the keyword '3 ounce bright citrus deodorant
sensitive skin' to see the relvant items, And now I am looking at the
item list.
Reasoning: Based on the Progress and current observation, the
next step is to choose an item satisfying the specification.
I can focus on:
[B078GWRC1J]
Bright Citrus Deodorant by Earth Mama | Natural and Safe for Sensitive
Skin, Pregnancy and Breastfeeding, Contains Organic Calendula 3-Ounce
$10.99
[B078GTKVXY]
Ginger Fresh Deodorant by Earth Mama | Natural and Safe for Sensitive
Skin, Pregnancy and Breastfeeding, Contains Organic Calendula 3-Ounce
$10.99

Action: click[B078GWRC1J]

Observation:

Progress: I searched and and clicked the item seems to be most
relevant to the goal specification.
I am looking at the option list.
Reasoning: As the goal requires 3-ounce bottle, I can focus on
the size option.
I can focus on:
size
[ travel set (4-pack) ]
[ 3 ounce (pack of 1) ]
[ 3-ounce (2-pack) ]

Action: click[3-ounce (pack of 1)]

Observation:

Progress: I searched and and clicked the item id.
Among the option list, and I clicked size option.
Reasoning: According to the progress, I have to focus on the
scent option as a next step.
I can focus on:
scent
[ assorted scents ]
[ bright citrus ]
[ calming lavender ]
[ ginger fresh ]
[ simply non-scents ]

Action: click[bright citrus]

Observation:

Progress: Based on **observation** and **previous actions**, I clicked
size option and scent option.
Reasoning: As there is no more options to select and I met all
requirements specified in the goal, next step is to buy the item.
I can focus on:
[ Buy Now ]

Action: click[Buy Now]

Now Here is the task.
```

```
Instruction:
{instruction}

{History of observations and actions}

Observation:
{observation}

Action:
```

A.4.3 Action matching evaluation prompt

Prompt for model-based evaluation of action matching

```
<system prompt>
Your task is to evaluate whether the given two action commands are
semantically aligned.

<main prompt>
You will be given
1). **reference action** which indicates an correct action.
2). **predicted action** which is predicted by assistant agent

Your task is to assess whether the message in **predicted action**
is semantically aligned with message in the **reference action**.
Please make sure you read and understand these instructions
carefully.
Please keep this document open while reviewing, and refer to it as
needed.

Evaluation Criteria:
Alignment = 1: the predicted action is semantically aligned with the
reference action.
    send_msg_to_user('30%') and send_msg_to_user('The percentage of
    amount of pending orders among entire orders is 30%') are
    semantically aligned.
    click('a34') and click('a34', button='left') is semantically
    aligned.

Alignment = 0: the predicted action is semantically not aligned with
the reference action.
    send_msg_to_user('$25') and send_msg_to_user('The requested
    value is $29') are not semantically aligned.
    click('a34') and click('a34', button='left') are semantically
    aligned.

Evaluation Steps:
1. Write a simple feedback that assess whether the predicted action
is semantically aligned with the reference action.
2. After writing a feedback, write a score that is 0 or 1. You
should refer to the Evaluation Criteria.
3. The output format should look as follows: "Feedback: (write a
feedback for criteria) [RESULT] (an integer number among 0 or 1)"
4. Please do not generate any other opening, closing, and
explanations.

**reference action**: {ref_action}
**predicted action**: {pred_action}

Feedback:
```

A.5 Qualitative Analysis

In this section, we provide 1) concrete examples of webpages contextualized by LCoW and 2) examples of action sequence of LLM agents with and without LCoW.

A.5.1 Web page contextualization

As demonstrated in the following examples, contextualization model trained via LCoW provides contexts (e.g., role of the UI elements specific to the website) that enables more accurate decision making of LLM-based web agent.

Example 1

Instruction: Sort the "users" list by the following fields:- Updates (descending)

Contextualized web page observation:

```
[a183] grid 'Users.', multiselectable=False
This is the main table containing the list of users.
We need to focus on this element to find ways to interact
with the column header.
[a186] rowgroup ''
  [a187] row ''
    [a188] columnheader 'Select All'
    [a193] columnheader '\uf1e4 Show column search row'
    [a197] columnheader 'User ID User ID column options'
    [a204] columnheader 'Name Name column options'
    [a211] columnheader 'Email Email column options'
    [a218] columnheader 'Avatar Avatar column options'
    [a225] columnheader 'Title Title column options'
    [a232] columnheader 'Company Company column options'
    [a239] columnheader 'Department Department column options'
    [a246] columnheader 'Location Location column options'
    [a253] columnheader 'Time zone Time zone column options'
    [a260] columnheader 'Business phone Business phone column options'
These are the column headers in the users table.
We need to focus on one of these elements to trigger
the sorting action.
However, the AXTree does not show any direct options to
sort these columns.

[a77] button '\uf13e Personalize List'
This button might provide options to customize
the list view, potentially including sorting options.
```

Example 2

Instruction: Go to the hardware store and order 10 "Developer Laptop (Mac)" with configuration 'Additional software requirements': 'Trello, Zoom, Microsoft Office 365, HubSpot, Adobe Creative Cloud', 'Adobe Acrobat': True, 'Adobe Photoshop': True, 'Eclipse IDE': True

Contextualized web page observation:

```
[a184] group ''
    [a187] Legend ''
        [a189] heading 'Optional Software'
    [a194] LayoutTable ''
        [a195] checkbox 'Adobe Acrobat', checked='true'
        [a196] LabelText ''
    [a201] LayoutTable ''
        [a202] checkbox 'Adobe Photoshop', checked='false'
        [a203] LabelText ''
    [a208] LayoutTable ''
        [a209] checkbox 'Eclipse IDE', checked='true'
        [a210] LabelText ''
```

```
This section contains checkboxes for optional software.

The "Adobe Acrobat" checkbox [a195] is already checked,
which aligns with the user's requirements.

The "Adobe Photoshop" checkbox [a202] is currently unchecked,
and we need to check it to meet the user's specifications.

The "Eclipse IDE" checkbox [a209] is already checked, which
is correct according to the user's requirements.

The LabelText elements [a196], [a203], and [a210] associated
with each checkbox are likely the clickable areas to toggle
the checkbox state. We should interact with the LabelText for
"Adobe Photoshop" to check the box.
```

Example 3

Instruction: What is the value of ”broken” in the chart (in percent)?

Contextualized web page observation:

```
we should focus on the pie chart elements.
The relevant section is:

[a962] region 'Pie on Change Request table by Reason
created by Henry Turner: 2024-09-26 05:41:16.
Highcharts interactive chart.'
 [a963] region 'Chart screen reader information,
        Pie on Change Request table by Reason created by
        Henry Turner:
    2024-09-26 05:41:16.'
  StaticText 'Pie chart with 2 slices.'
 generic '', live='polite', relevant='additions text'
 generic '', live='assertive', relevant='additions text'
 generic '', live='polite', relevant='additions text'
 [a973] SvgRoot 'Interactive chart'
  group ''
   group ''
    image '1. (empty), 30.'
    image '2. broken, 3.'
 [a977] region 'Toggle series visibility,
    Pie on Change Request table by Reason created by Henry
    Turner: 2024-09-26 05:41:16'
  [a978] list ''
   [a979] listitem ''
    [a980] button 'Show (empty)', pressed='true'
   [a981] listitem ''
    [a982] button 'Show broken', pressed='true'
 [a984] button 'View chart menu, Pie on Change Request
    table by Reason created by Henry Turner:
    2024-09-26 05:41:16', expanded=False
 StaticText 'End of interactive chart.'

This section contains the pie chart. The chart has two
slices: "(empty)" with a value of 30, and "broken" with a
value of 3.
The "broken" slice is the one we need to focus on to answer
the user's question.
```

Example 4

Instruction: Sort the "users" list by the following fields: - Language (ascending)

Contextualized web page observation:

```
we should focus on the following elements:

1. [a1053] button '\uf135 Add selected options to the Selected listbox'
   This button is used to add the selected Language field to
   the Selected list in the personalization dialog.

2. [a1108] button 'OK'
   This button is used to close the personalization dialog
   and apply the changes.

3. [a246] columnheader 'Location Location column options'
   [a248] button 'Location'
   [a252] button 'Location column options', hasPopup='menu'
   These elements represent the sorting control for the
   Location column, which might include the Language
   information.
   We'll need to interact with this to sort the list after
   closing the personalization dialog.

Note: The Language column is not directly visible in the
current AXTree observation, likely because it hasn't been
added to the list yet.
After adding it and closing the dialog, we would need to find
the newly added Language column header to sort it.
```

Example 5

Instruction: Create a new user with a value of "8273378363827782819" for field "User ID", a value of "Webster-Richard" for field "Last name", a value of "Lynn-David" for field "First name", a value of "lynn-david.webster-richard@workarena.com" for field "Email", a value of "false" for field "Internal Integration User", a value of "Outlook" for field "Calendar integration", a value of "(514) 357-8997" for field "Mobile phone", a value of "false" for field "Password needs reset", a value of "English" for field "Language", and a value of "false" for field "Web service access only".

Contextualized web page observation:

```
we should focus on the following elements:

[a298] LabelText ''
    [a299] note ''
    StaticText 'Internal Integration User'
[a305] checkbox 'Internal Integration User', checked='false'

This element represents the "Internal Integration User" field.
It's a checkbox that is currently unchecked (false), which
aligns with the user instruction.
No action is needed for this field.

[a353] LabelText ''
    [a354] note ''
    StaticText 'Calendar integration'
[a359] combobox 'Calendar integration' value='Outlook',
hasPopup='menu', expanded=False
    [a360] menuitem '-- None --', selected=False
    [a361] menuitem 'Outlook', selected=True

This element represents the "Calendar integration" field.
It's a combobox that is already set to "Outlook", which
```

```
aligns with the user instruction.
No action is needed for this field.

[a429] LabelText ''
    [a430] note ''
    StaticText 'Mobile phone'
[a435] textbox 'Mobile phone'

This element represents the "Mobile phone" field.
It's a textbox where we need to enter "(514) 357-8997" as per
the user instruction.

[a327] LabelText ''
    [a328] note ''
    StaticText 'Language'
[a333] combobox 'Language' value='-- None --',
                        hasPopup='menu', expanded=False
    [a334] menuitem '-- None --', selected=True
    [a335] menuitem 'English', selected=False

This element represents the "Language" field.
It's a combobox where we need to select "English" as per the
user instruction.

[a285] LabelText ''
    [a286] note ''
    StaticText 'Web service access only'
[a292] checkbox 'Web service access only', checked='false'
```

A.5.2 Action sequence

In this subsection, we provide the action sequences obtained by rolling out the LLM agent on evaluation tasks in Figure 11. For a given instruction, we compare the rollouts with and without the application of LCoW, highlighting the differences in decision-making behavior and task completion efficiency.

A.6 Experimental details

WebShop For the WebShop benchmark, we fine-tune Phi-3-mini-Instruct as a contextualization module, setting the learning rate, warmup ratio, and batch size to 1e-5, 1e-2, and 32, respectively. The module is trained for a single epoch over the collected data for each LCoW iteration, and we utilize demonstrations corresponding to 397 individual tasks provided in the WebShop benchmark as seed demonstrations. In the WebShop environment, we employ Gemini-1.5-flash as the LLM agent, combined with the contextualization module during the trajectory collection phase. It is also utilized for sampling optimal contextualization at the initial iteration of LCoW.

WorkArena & WebArena For WorkArena and WebArena, we fine-tune Llama-3.1-8B-Instrcut as an observation contextualization module, and we set the learning rate, warm-up ratio, and batch size as 1e-5, 1e-1, and 128. Additionally, we set the training epochs as 4 and 3 for WorkArena and WebArena, respectively. Since no demonstrations are provided in both benchmarks, In WorkArena, we collected 264 seed demonstrations (i.e., successful trajectories) among 495 training tasks across 33 task types using Claude-3.5-Sonnet and GPT-4o, while no successful trajectories were collected for the 10 task types. Summary statistics of the collected seed demonstrations across task types can be found in Table 9. In WebArena, we collected 363 seed demonstrations (i.e., successful trajectories) among 647 training tasks (excluding 165 evaluation tasks in WebArena-Lite from 812 WebArena tasks) using Claude-3.5-Sonnet and GPT-4o. Additionally, as determination of action matching based on parsing is infeasible due to open-ended actions (e.g., sending message to user), we exploit GPT-4o as an action-matching evaluator. Detailed prompt for the action-matching evaluator is provided in Appendix A.4. For WorkArena and WebArena, we use Claude-3.5-Sonnet as the agent LLM for trajectory collection and we utilize the same LLM for the sampling optimal contextualization at the initial iteration.

Instruction:
Create a new problem with a value of "Email system is down again" for field "Problem statement", a value of "" for field "Service offering", a value of "" for field "Service", a value of "3 - Low" for field "Urgency", a value of "Software" for field "Category", a value of "pompier Glathsheim subscribe putty bionomist" for field "Description", a value of "" for field "Configuration item", a value of "" for field "Assignment group", and a value of "3 - Low" for field "Impact"

Llama-3.1-70B

1. fill Email system is down again in [a487] textbox '\uf1ddProblem statement'
2. select 'Software' from [a225] combobox 'Category'
3. fill pompier Glathsheim subscribe putty bionomist in [a504] textbox 'Description'
4. fill Email system is down again in [a487] textbox 'Field value has changed since last update Problem statement'
5. select '3 - Low' from [a410] combobox 'Urgency'
6. fill Email system is down again in [a487] textbox 'Field value has changed since last update Problem statement'
7. fill Email system is down again in [a487] textbox 'Field value has changed since last update Problem statement'
8. fill pompier Glathsheim subscribe putty bionomist in [a504] textbox 'Description'
9. click [a1350] button 'Submit'

LCoW + Llama-3.1-70B

1. fill Email system is down again in [a487] textbox '\uf1ddProblem statement'
2. select 'Software' from [a225] combobox 'Category'
3. fill pompier Glathsheim subscribe putty bionomist in [a504] textbox 'Description'
4. click [a90] button 'Submit'

Instruction:
Go to the hardware store and order 9 "iPad pro" with configuration {'Choose the colour': 'Silver', 'Choose the storage': '128'}

Llama-3.1-70B

1. click [a252] link 'Hardware'
2. click [a268] link 'iPad pro'
3. select 'Silver' from [a198] radiogroup 'Mandatory'
4. click [a184] radio '\uf137 Silver'
5. hover the cursor on [a184] radio '\uf137 Silver'
6. click [a185] LabelText ''
7. select '9' from [a244] combobox 'Quantity'
8. click [a285] button 'Order Now'

LCoW + Llama-3.1-70B

1. click [a252] link 'Hardware'
2. click [a268] link 'iPad pro'
3. click [a184] radio '\uf137 Silver'
4. click [a185] LabelText ''
5. select '9' from [a244] combobox 'Quantity'
6. click [a285] button 'Order Now'

Figure 11: Llama-3.1-70B agent making decisions based on raw observations **(left)** completes the task but exhibits inefficient behavior, such as repeating actions it has already performed or deciding on redundant actions, which are indicated as a red color. In contrast, the Llama-3.1-70B agent making decisions based on contextualized observations via LCoW **(right)** completes the task efficiently, with few or no mistakes and minimal repetition of redundant actions.

Task type	Number of seed demonstrations
Multi-chart-value retrieval	8
Multi-chart-minmax retrieval	12
Single-chart-value-retrieval	12
Single-chart-minmax retrieval	11
Create change request	8
Create incident	0
Create hardward asset	0
Create problem	14
Create user	11
Knowledge base search	12
Filter asset list	0
Filter change request list	0
Filter hardware list	0
Filter incident list	0
Filter service catalog item list	0
Filter user list	0
Sort asset list	0
Sort change request list	0
Sort hardware list	3
Sort incident list	3
Sort service-catalog list	5
Sort user list	4
All menu	15
Impersonation	12
Order developer laptop	15
Order iPad mini	14
Order iPad pro	15
Order sales laptop	15
Order standard laptop	15
Order apple watch	15
Order apple Macbook pro	15
Order development laptop PC	15
Order loander laptop	15
Total	264

Table 9: We collected trajectories from 15 individual tasks for each 33 task types in WorkArena benchmark using GPT-4o-0806 and Claude-3.5-sonnet agent, thereby collecting 264 successful trajectories. We utilized them as a seed demonstrations for LCoW.