

# 000 TOWARDS INTERNET-SCALE TRAINING FOR AGENTS

001  
002  
003 **Anonymous authors**

004 Paper under double-blind review

## 005 006 007 ABSTRACT

008  
009 The predominant approach for training web navigation agents is to gather human  
010 demonstrations for a set of popular websites and hand-written tasks, but it is  
011 becoming clear that human data is an inefficient resource. We develop a pipeline to  
012 facilitate internet-scale training for agents without laborious human annotations.  
013 In the first stage, an LLM annotates 150k sites with agentic tasks. In the next  
014 stage, LLM agents complete tasks and produce trajectories. In the final stage, an  
015 LLM filters trajectories by judging their success. Language models are powerful  
016 data curation tools, identifying harmful content with an accuracy of 97%, judging  
017 successful trajectories with an accuracy of 82.6%, and producing effective data.  
018 We train agents based on *Qwen 3 1.7B* that are competitive with frontier LLMs as  
019 web agents, while being smaller and faster. Our top agent reaches a success rate of  
020 56.9%, outperforming the data collection policy *Qwen 3 235B*, a 235 times larger  
021 *Llama 4 Maverick*, and reaching 94.7% of the performance of *Gemini 2.5 Flash*.  
022 We will be releasing code, models and data to reproduce the entire pipeline.

## 023 1 INTRODUCTION

024 The predominant approach for training LLM web navigation agents is to collect human demonstrations  
025 for a set of manually curated websites and tasks (Deng et al., 2023; Zhou et al., 2024b; Putta et al.,  
026 2024; Koh et al., 2024a; Liu et al., 2024; Lù et al., 2024; Rawles et al., 2023). Human data can  
027 be laborious to collect, and becomes costly to scale as the breadth of skills that users require from  
028 language model agents grows. There are more than 300M sites on the western internet according  
029 to The Common Crawl Foundation (2025), and the range of sites that researchers have annotated  
030 represents a tiny fraction of the vast available data. And crucially, the existing human data is *static*.  
031 There is a growing need to automate pipelines for training the next generation of language model  
032 agents in a *dynamic internet-scale environment*. This paper addresses the core challenge of building  
033 this environment—reducing dependence on human annotations. We develop an automatic pipeline  
034 that aims to facilitate internet-scale training for agents, which we refer to as the InSTA pipeline.

035 The InSTA pipeline has three stages. In the first stage, we employ a language model task proposer  
036 that annotates 150k sites with live web navigation tasks for agents to perform. Existing works are  
037 limited to 200 popular websites (He et al., 2024; Lù et al., 2024; Rawles et al., 2023; Deng et al.,  
038 2023; Zhou et al., 2024c; Murty et al., 2025) that researchers have annotated manually, and a handful  
039 of synthetic websites (Zhou et al., 2024b; Koh et al., 2024a; Yao et al., 2023a). Our first goal in this  
040 paper is to improve coverage of real-world sites. To accomplish this, we cast a wide net. Starting  
041 from the top 1M sites on the internet ranked by popularity, our task proposer filters down to 150k  
042 sites that have safe content. Safety is critical when building autonomous agents, and our task proposer  
043 succeeds at detecting harmful content with an accuracy of 97%. Tasks are generated for sites marked  
044 as safe by the task proposer, and we run language model agents to complete the generated tasks.  
045 Agent progress is then fed back to the task proposer, which reviews trajectories and judge evaluations  
046 in order to assign a harder task based on the latest content of the website, closing the loop.

047 Scaling the task generation loop, we annotate 150k diverse sites with challenging agentic tasks, and  
048 release an official huggingface dataset: *data-for-agents/insta-150k-v2*. Motivated by the importance  
049 of internet data to progress in modern deep learning, our second goal in this paper is an internet-scale  
050 data flywheel for training LLM agents. We approach this by harnessing LLMs as data curation tools.  
051 After generating tasks, the pipeline employs pretrained LLMs as agents to complete tasks and produce  
052 trajectories, which are filtered by a judge to select the best data. Agents control a virtual web browser  
053 and produce reasoning traces that contain function calls to interact with and navigate live webpages.

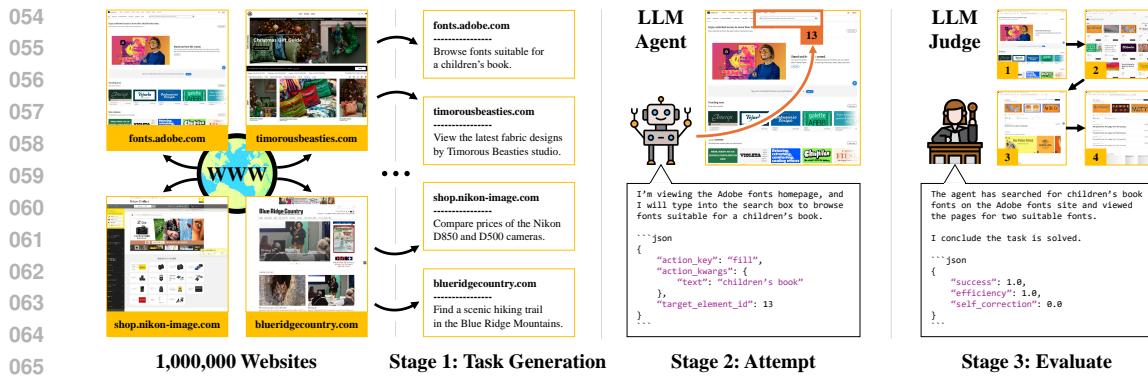


Figure 1: **Overview of the InSTA pipeline.** Our work unlocks a dynamic internet-scale environment that allows training small models to match frontier LLMs as agents, on a fraction of the budget. Starting from the top 1M sites on the internet, we annotate 150k sites with challenging agentic tasks, and release the entire pipeline, including code, models and an official huggingface dataset, on our website: [anonymous-instapipeline.github.io](https://anonymous-instapipeline.github.io).

The judge produces a reasoning trace that considers whether a trajectory is successful, and scores the agent on a continuous scale from 0 to 1. We scale the pipeline to create a large reasoning dataset for multimodal agents, with 2.2M screenshots, 2.2M traces for actions, and 150k traces for the judge.

Our data unlocks great potential in small language models as agents. We train a series of models based on *Qwen 3 1.7B* on varying scales of data from the InSTA pipeline, and match the performance of frontier LLM agents, on a fraction of the budget. Our top agent has a success rate of 56.9%, outperforming the data collection policy *Qwen 3 235B*, beating a 235 times larger *Llama 4 Maverick*, and reaching 94.7% of the performance of *Gemini 2.5 Flash*. To share our progress, we are releasing the entire pipeline, including code, models and data, on our website: [anonymous-instapipeline.github.io](https://anonymous-instapipeline.github.io).

## 2 RELATED WORKS

**Language Model Agents.** There is an emerging paradigm in modern NLP using language models (Radford et al., 2019; Brown et al., 2020; Touvron et al., 2023a;b) as the backbone for agents (Andreas, 2022). These models show impressive reasoning capabilities (Bubeck et al., 2023; Zhong et al., 2024; Valmeekam et al., 2024) that allow them to generalize to downstream applications, such as web navigation, where observations differ from LLM training data. Search algorithms provide a secondary axis to improve the reasoning capabilities of language model agents (Yao et al., 2023b; Besta et al., 2024; Koh et al., 2024b; Zhou et al., 2024a) by providing an explicit algorithmic scaffold and allowing test-time compute to improve reasoning steps (Snell et al., 2024; Zhong et al., 2024). Although most recent work focuses on running language models as zero-shot agents, fine-tuning language models to improve their effectiveness as agents is becoming popular (Putta et al., 2024; Zeng et al., 2023; Zhang et al., 2023; Hong et al., 2023; Xie et al., 2024; Wang et al., 2024; Zhou et al., 2024c; Murty et al., 2025) as target benchmarks are becoming more difficult to solve zero-shot.

**Agent Pipelines.** There are a growing number of agent pipelines aimed at fine-tuning language models to improve their effectiveness as agents (Mitra et al., 2024; Zeng et al., 2023; Putta et al., 2024; Chen et al., 2023; Ou et al., 2024; Zhou et al., 2024c; Murty et al., 2025). However, driven by the limited data available, many such works train on data with significant overlap with their test environment, either with different tasks for the same environment configuration as the test setting (Deng et al., 2023; Zhou et al., 2024c; Murty et al., 2025), or even the same tasks (Putta et al., 2024). We consider a setting where tasks and environment configurations (i.e. websites) are entirely separate between training and testing, creating a strong train-test split. This presents a challenge: human data for training LLM agents is limited (Deng et al., 2023; Lù et al., 2024). We address this challenge by reducing dependence on human data in agent pipelines. We employ LLMs as data curation tools to automatically design challenging tasks, and select the best training data. Our pipeline allows us to train small models that match frontier LLMs on Web Voyager (He et al., 2024), without using any data from Web Voyager. Contrast this with methods that train primarily on data from Web Voyager (Zhou et al., 2024c; Murty et al., 2025), and may not transfer to other benchmarks (Xue et al., 2025).

108 **Agent Datasets.** Datasets for training web navigation agents typically rely on human annotators to  
 109 create tasks (Zhou et al., 2024b; Koh et al., 2024a; Rawles et al., 2023), and provide demonstrations  
 110 for tasks (Deng et al., 2023; Lù et al., 2024; Rawles et al., 2023; Shen et al., 2024). However, the  
 111 amount of data researchers have annotated represents a tiny fraction of the available internet data.  
 112 There are more than 300M sites on the internet according to The Common Crawl Foundation (2025),  
 113 yet existing datasets are limited to 200 popular sites that human annotators are familiar with (Deng  
 114 et al., 2023; Lù et al., 2024; Shen et al., 2024). Human data can be laborious to collect, and becomes  
 115 costly to scale as the capabilities users require from agents grows. And crucially, human data is *static*.  
 116 Our work moves away from fixed datasets for training agents, and towards a dynamic internet-scale  
 117 environment that grows with an ever-changing internet. We are not the first to build an environment  
 118 (Zhou et al., 2024b; Koh et al., 2024a; Yao et al., 2023a; He et al., 2024), nor are we the first to  
 119 consider synthetic data (Gandhi et al., 2024; Ou et al., 2024; Setlur et al., 2024; Tajwar et al., 2024),  
 120 but we have solved a key challenge that unlocks the internet as the largest environment for agents.

121 **Language Model Judges.** Using LLMs to judge the correctness of responses is becoming popular  
 122 to refine LLM predictions (Li et al., 2024), including to verify reasoning steps (Zhang et al., 2024),  
 123 rejection sample (Snell et al., 2024; Sun et al., 2024), prioritize frontier nodes in search algorithms  
 124 (Zhou et al., 2024a; Koh et al., 2024b), filter out harmful responses (Inan et al., 2023), provide  
 125 feedback for response improvement (Madaan et al., 2023; Paul et al., 2024; Patel et al., 2024;  
 126 Yuksekgonul et al., 2024), and provide ratings for alignment (Lee et al., 2024; Ouyang et al., 2024).  
 127 Our use of language models to evaluate agents is inspired by Generative Verifiers (Zhang et al., 2024),  
 128 and the multimodal verifier in He et al. (2024). One difference is our verifier predicts a reasoning  
 129 trace that scores the agent from 0 to 1, which helps us rank trajectories to select the best data.

### 131 3 LANGUAGE MODEL AGENTS

132 Language model agents are a class of decision-making agents represented by  $\pi_{\text{LLM}}(\mathbf{a}_t | \mathbf{s}_t, \mathbf{c}_n)$ , a  
 133 policy that processes multimodal observations  $\mathbf{s}_t$  (from a virtual web browser in our case) and predicts  
 134 textual actions  $\mathbf{a}_t$  to complete a task  $\mathbf{c}_n$ . Underneath this abstraction, a large language model (LLM)  
 135 generates actions via the next-token prediction, conditioned on a system prompt  $\mathbf{x}_{\text{agent}}$ .

$$137 \mathbf{a}_t = f^{\text{text} \rightarrow \text{act}}(\text{LLM}(\mathbf{x}_{\text{agent}}, \mathbf{c}_n, \text{Enc}(\mathbf{s}_t))) \quad (1)$$

138 Environment representations for observations and actions typically differ from the expected input  
 139 format of the language model (typically images and text), and functions are introduced that map the  
 140 observations to a multimodal prompt  $\text{Enc}(\cdot)$ , and parse actions from the language model generated  
 141 response  $f^{\text{text} \rightarrow \text{act}}(\cdot)$ . For web navigation, the environment state  $\mathbf{s}_t$  is HTML DOM, and is often  
 142 formatted as raw HTML code, an Accessibility Tree, Set-of-marks, or screenshots (Zhou et al., 2024b;  
 143 Koh et al., 2024a; Chezelles et al., 2024; Shen et al., 2024). We built a fast Markdown parser that  
 144 converts webpage observations into a compact readable format (refer to the code). Action formats  
 145 vary between works, and we build on Schick et al. (2023)’s function-calling framework, where a  
 146 language model generates code that is parsed into a function name and corresponding arguments.  
 147 Given sets of function and argument names  $L_i$ , and sets of argument values  $G_i$ , the action space  $\mathcal{A}$  is:

$$149 \mathcal{A} = L_{\text{func}} \times (L_{\text{arg1}} \times G_{\text{arg1}}) \times (L_{\text{arg2}} \times G_{\text{arg2}}) \times \cdots \times (L_{\text{argN}} \times G_{\text{argN}}) \quad (2)$$

150 where  $L_{\text{func}}$  is the set of function names on the page object in the Playwright API (Microsoft, 2024),  
 151 and function arguments have a name and value ( $L_{\text{arg1}} \times G_{\text{arg1}}$ ) corresponding to the Playwright API.  
 152 We allow the agent access to call arbitrary functions in Playwright (Microsoft, 2024), a Microsoft-  
 153 developed browser automation library that wraps a headless web browser. The agent’s goal is to  
 154 complete a web navigation task specified via a natural language instruction  $\mathbf{c} \in L$ , starting from an  
 155 initial URL, and operating the browser via function calls to the Playwright API until the desired task  
 156 is complete, after which the agent calls the `stop` function and provides a final response:

$$158 \mathbf{a}_{\text{stop}} = (\text{"stop"}, (\text{"response"}, \text{"the task has been completed."})) \quad (3)$$

159 We prompt the agent to produce a reasoning trace of a desired length (ablated in Figure 10) that  
 160 contains function call actions as JSON in a fenced code block. To parse actions from the response,  
 161 we employ a regex template that matches the first JSON code block, and a JSON decoder  $f^{\text{text} \rightarrow \text{act}}(\cdot)$

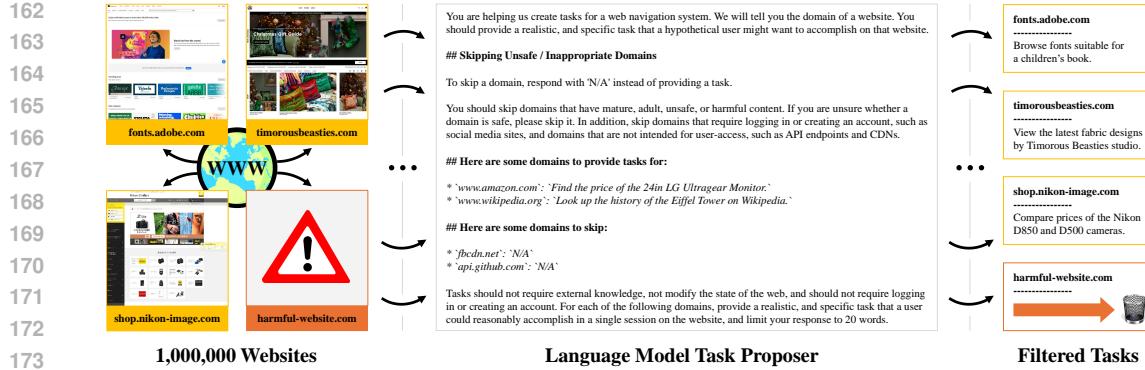


Figure 2: **Annotating 150k live sites with agentic tasks.** Starting from 1,000,000 websites, we employ a pretrained language model that marks sites as safe/unsafe for annotation, and assigns a realistic task that a hypothetical user might want to accomplish on each site. The task proposer aggressively filters out 85% of websites from the pipeline, resulting in 150k safe websites annotated with realistic tasks.

to parse the contents within the code block. When parsing fails due to invalid syntax, we allow the agent to generate a second response. Equipped with a language model agent that makes function calls with the Playwright API, we may consider the crucial task of obtaining large and diverse data.

## 4 INTERNET-SCALE TASK GENERATION

Training the next generation of LLM agents requires a large and diverse set of websites and tasks beyond what researchers have annotated so far (Deng et al., 2023; Zhou et al., 2024b; Koh et al., 2024a; Liu et al., 2024; Lù et al., 2024; Rawles et al., 2023; He et al., 2024). We develop an approach to efficiently annotate vast numbers of sites from diverse sections of the internet with agentic tasks. Our approach introduces two important desiderata: (1) it should not rely on human annotations, and (2) tasks should derive from a feedback process that deeply explores the environment.

### 4.1 LANGUAGE MODEL TASK PROPOSER

The key idea in stage one of the pipeline is a feedback loop, where a language model task proposer  $\psi_{\text{LLM}}(\cdot)$  guides exploration on a website via an initial easy task. We then run a language model agent to explore the site, conditioned on the initial task, which produces an exploratory trajectory that is fed back to the task proposer. Conditioned on a trajectory that deeply explores the website, the task proposer then creates a harder, grounded task. This process is summarized as an equation.

$$\mathbf{c}_{n+1} \sim \psi_{\text{LLM}}(\cdot | \mathbf{w}, \mathbf{c}_n, \underbrace{\mathbf{s}_1, \mathbf{a}_1, \dots, \mathbf{s}_T, \mathbf{a}_T, \mathbf{r}_T}_{\text{last trajectory}}) \text{ st } \mathbf{a}_t \sim \pi_{\text{LLM}}(\cdot | \mathbf{s}_t, \mathbf{c}_n) \quad \mathbf{s}_{t+1} \sim \mathbb{P}(\cdot | \mathbf{s}_t, \mathbf{a}_t) \quad (4)$$

The website url is  $\mathbf{w}$ , the initial task is  $\mathbf{c}_0$ , the trajectory includes states  $\mathbf{s}_t$ , actions  $\mathbf{a}_t$ , judge score  $\mathbf{r}_T$ , and a harder, grounded task  $\mathbf{c}_{n>0}$  for the next loop. Highlighted in Figure 2, we annotate 150k sites with tasks by scaling Equation 4, and release them on huggingface at: *data-for-agents/insta-150k-v2*.

**Model Details.** We utilize pretrained and frozen language models that conform to a chat interface and accept system, user, and assistant prompts. The task proposer system prompt is listed in Appendix 11, and details all cases for which sites are considered unsafe. We employ the Llama 3.1 family of LLMs from Meta (Grattafiori et al., 2024; Touvron et al., 2023b;a), the GPT family of LLMs from OpenAI, and the Gemini family of LLMs from Google. Inference is served using vLLM (Kwon et al., 2023) for the Llama series of models. We employ a sampling temperature of 0.5 and a maximum budget of 1024 new generated tokens, while all other parameters are kept as defaults in the OpenAI chat completions API, which is used to make inference calls to all LLMs.

**Prompt Details.** The task proposer operates in two phases. In an initial phase when just the url of a website is observed, the task proposer generates an initial task  $\mathbf{c}_0 \sim \psi_{\text{LLM}}(\mathbf{c}_0 | \mathbf{w})$ , and can mark a website as unsafe by setting  $\mathbf{c}_0 = \text{N/A}$ . The system prompt for this phase is listed in

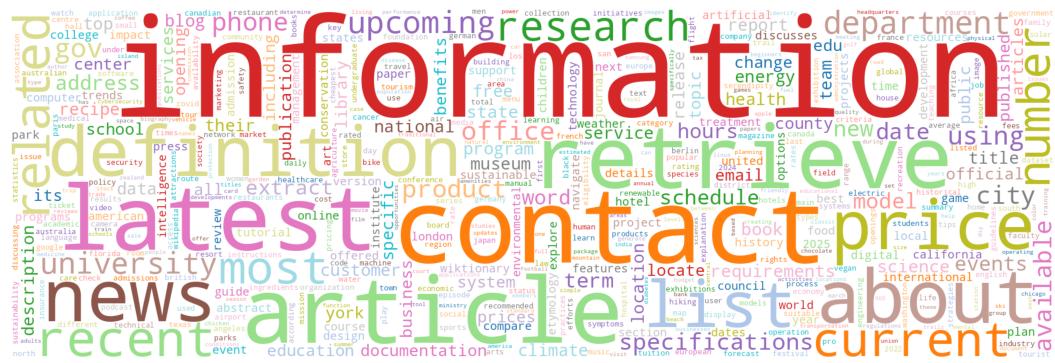


Figure 3: **Most frequent words in our tasks.** This wordcloud shows the top 500 most frequent words in tasks from the training set of our official huggingface dataset. The size of each word corresponds to its frequency in the dataset. Our tasks span diverse categories and lexicon.

Appendix 11. Agents discussed in Section 5 explore 150k sites annotated with initial tasks  $c_0$  and produce trajectories. In a second phase of task generation, we prompt the task proposer with the website url  $w$ , the initial task  $c_0$ , the trajectory  $s_1, a_1, \dots, s_T, a_T$ , and a system prompt that instructs the LLM to produce a reasoning trace that contains a harder, grounded task  $c_{n>0}$ . The system prompt for the second phase of task generation is listed in Appendix 11. The refined tasks produced by this iterative process lead to broadly capable agents, demonstrated in Section 6 by our ability to zero-shot transfer agents trained on our data to Web Voyager (He et al., 2024) and compete with frontier LLMs.

The design of the task proposer as a feedback process is important, and the full potential of this design will be realized in future work that trains agents with an on-policy reinforcement learning algorithm. In such future work, the task proposer can be used within the RL loop to generate incrementally harder tasks as agents learn. For this paper, we employ one loop of task generation.

## 4.2 SAFETY & RELIABILITY

Safety is critical when building autonomous agents. The internet contains significant amounts of content that should be removed from training data, in order to avoid agents learning harmful behaviors. To understand the robustness of our safety filter, we input 100 carefully selected websites to the task proposer, of which 50 contain harmful, or mature content. Table 1 reports the accuracy, precision, and recall of the safety filter on this data. For a variety of LLMs, the safety filter displays high accuracy—up to 97% of websites are correctly classified, and recall for detecting unsafe websites is as high as 1.0, suggesting that nearly all unsafe websites are detected by the safety filter.

Method	Acc.	Prec.	Recall
<i>Llama 3.1 70B</i>	85%	0.77	<b>1.00</b>
<i>GPT-4o</i>	95%	0.91	<b>1.00</b>
<i>Gemini 1.5 Pro</i>	<b>97%</b>	<b>0.96</b>	0.98

**Table 1: The safety filter displays a high accuracy.** We measure the accuracy, precision, and recall of the safety filter on a set of 100 websites, where 50 contain harmful, or mature content. Up to 97% of websites are correctly classified, and recall is as high as 1.0.

Method	Verifiable Rate
<i>Llama 3.1 70B</i>	75%
<i>GPT-4o</i>	85%
<i>Gemini 1.5 Pro</i>	89%

**Table 2: Generated tasks are typically achievable.**  
 We measure the rate that human workers were able to complete and verify their completion of tasks produced by the task proposer for a set of 100 websites. Up to 89% of tasks are achievable, and verifiable.

Reliability is equally important for autonomous agents. Instructions should be followed faithfully by agents, and this requires training them with tasks that are achievable, and verifiable. Table 2 reports the rate that human workers were able to complete and verify their completion of tasks produced by the task proposer in its initial phase. Up to 89% of tasks are achievable, and verifiable according to the study, which suggests the pipeline is producing reliable tasks. Together with results in Section 6, it is likely that data from the InSTA pipeline leads to agents that follow instructions faithfully.

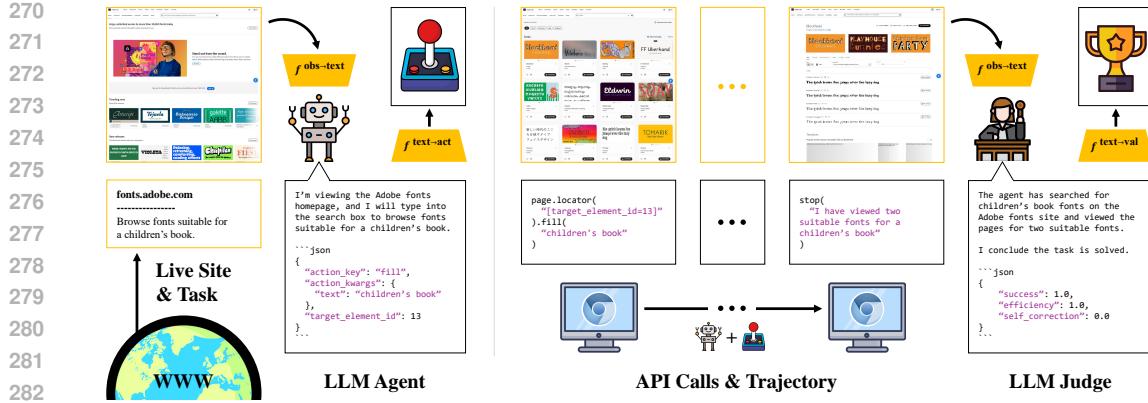


Figure 4: **Automatic evaluation for agents with language model judges.** Building on the large and diverse set of tasks generated by the pipeline, we employ pretrained language models to attempt and evaluate web navigation tasks. We dispatch language model agents to perform tasks by making calls to the Playwright API. We then employ language model judges to evaluate the trajectories.

### 4.3 SCALING TO 150,000 WEBSITES

We leverage Common Crawl for task generation. As of May 2025, the latest web graph released by The Common Crawl Foundation (2025) contains more than 300M unique hosts, which we adapt into a data source for agents. In particular, we select the top 1M sites based on their PageRank values. Common Crawl likely contain many unsafe websites, and these are filtered out by the task proposer. Each phase of task generation requires 14 hours of compute time using two 8-GPU v100 machines, and filters to 150k safe websites annotated with tasks. Statistics of tasks are shown in Figure 3.

## 5 INTERNET-SCALE ENVIRONMENT

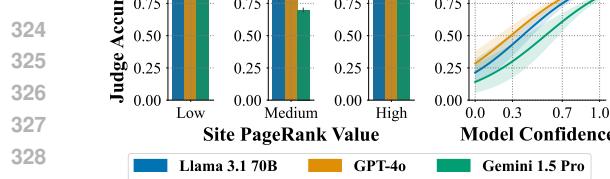
By this point, we have reached our first goal—to improve coverage of real-world sites by annotating 150k diverse sites with challenging agentic tasks. To reach our second goal, and move beyond a static dataset, towards a *dynamic internet-scale environment*, we require a robust evaluator for these tasks. Evaluation presents a subtle challenge. The web evolves constantly, and daily changes in website content may invalidate a fixed ground truth reference solution. Driven by necessity, this environment must be evaluated by a model that judges whether an agent’s solution is correct, in the latest context.

### 5.1 EVALUATION WITH LANGUAGE MODELS

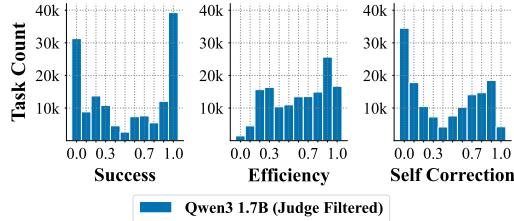
We model the process of evaluating trajectories from agents as a classification problem, where the goal is to estimate the probability  $r_T$  that a task  $c_n$  is solved, and generate  $r_T$  via next-token prediction, conditioned on a system prompt  $x_{\text{judge}}$ , a task  $c_n$ , and a trajectory  $s_1, a_1, \dots, s_T, a_T$ . The LLM is instructed to produce a reasoning trace that scores the agent on a scale from 0 to 1, and embed the score as JSON in a fenced code block. We employ a regex template that matches to the first code block in the response, and a JSON decoder to parse  $r_T$  from the response, given by  $f^{\text{text}\rightarrow\text{val}}(\cdot)$ .

$$r_T = f^{\text{text}\rightarrow\text{val}}(\text{LLM}(x_{\text{judge}}, c_n, \text{Enc}(s_1), a_1, \dots, \text{Enc}(s_T), a_T)) \quad (5)$$

**Verifying The Judge.** To understand the robustness of the judge, we measure its accuracy detecting successful trajectories that were annotated by human workers. We annotate 100 trajectories with binary success labels, and apply a threshold  $r_T > 0.5$  to obtain binary predictions from the judge. Figure 5 reports the accuracy of the judge as a function of the PageRank values of sites, and as a function of the confidence of the judge, given by  $\text{conf} = 2 \cdot |r_T - 1/2|$ . For all LLMs tested, the judge shows a high accuracy, ranging from 78.0% for *Gemini 1.5 Pro*, to 81.7% for *Llama 3.1 70B*, and 82.6% for *GPT-4o*. Accuracy is stable as PageRank falls, suggesting the judge is accurate for less popular sites that may not be well represented in the LLM’s training data. Shockingly, the confidence predicted by LLMs is highly interpretable, and correlates with accuracy to the point that trajectories where  $\text{conf} = 1$  are classified with an accuracy up to 93.1%. The emergent robustness of the judge equips us to efficiently and accurately verify agent solutions on a dynamic internet.



324  
325  
326  
327  
328  
329  
330  
331 **Figure 5: Language models are robust evaluators.**  
332 We measure the accuracy of language models for  
333 detecting successful trajectories, and find that accuracy  
334 remains stable relative to PageRank values (*left plot*).  
335 As models become more confident, their accuracy improves  
336 (*right plot*), suggesting confidence is a useful  
337 proxy for the reliability of their predictions.



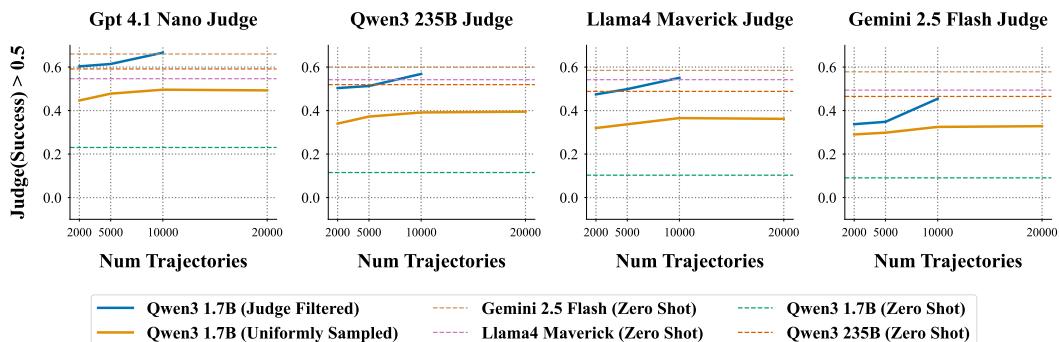
330  
331 **Figure 6: Statistics for our agent reasoning dataset.**  
332 We conduct a large data collection experiment using  
333 our top checkpoint for *Qwen3 1.7B*. Our dataset has  
334 2.2M screenshots, 2.2M reasoning traces for actions,  
335 and 150k traces for the judge. 50.0% of the trajectories  
336 are successful according to the judge, and have diverse  
337 ratings for efficiency and self-correction.

## 338 5.2 SCALING TO 150,000 AGENTS

339 With our task proposer, agent, and judge driven by pretrained language models, we have all components  
340 needed to harness internet-scale data. We conduct a large data collection experiment, running  
341 language model agents to complete tasks on 150k websites from our official dataset, producing 2.2M  
342 screenshots and 2.2M reasoning traces for actions within 150k trajectories. The judge annotates these  
343 trajectories, producing 150k reasoning traces for evaluations, and leading to the statistics in Figure 6.  
344 For this experiment, we employ a fine-tuned *Qwen3 1.7B* as the agent (refer to the next section),  
345 and *Qwen3 235B* zero-shot as the judge. Data collection requires 1,200 v100 GPU hours, and costs  
346 \$521.55 based on current AWS spot instance pricing, a fraction of the budget that industry labs are  
347 spending towards agents. If you have the right data, this small budget is sufficient and no human  
348 annotations are required to build models that compete with frontier LLMs as agents.

## 349 6 TRAINING AGENTS

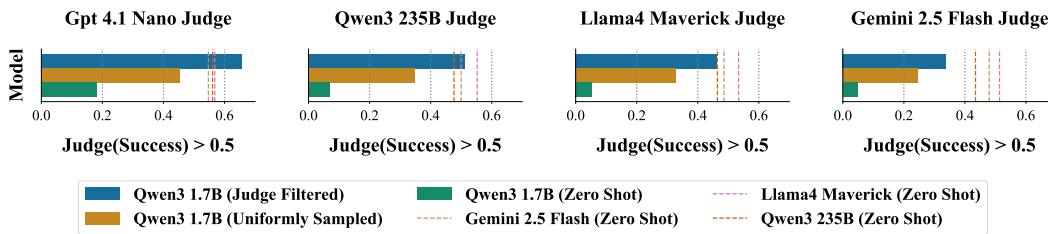
350 We've built an internet-scale data flywheel that produces trajectories annotated with scores from a  
351 judge that can help us train LLM agents. To understand the quality of data produced by the flywheel,  
352 we conduct a series of experiments training models on the data, and testing on popular benchmarks.  
353 These experiments focus on three main questions: (1) *what is the impact of increasing data scale?*  
354 (2) *do agents transfer to new domains?* (3) *do agents scale with test-time compute?*



355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377 **Figure 7: InSTA unlocks great potential in small models.** We train agents based on *Qwen 3 1.7B*  
378 using trajectories produced by a *Qwen 3 235B* data collection policy, and optionally filtered by a  
379 *Qwen 3 235B* judge (see Judge Filtered vs. Uniformly Sampled). We report success rates on a test set  
380 of 3,000 held-out websites and tasks. Before training, *Qwen 3 1.7B* has a zero-shot success rate of  
381 11.5% according to a *Qwen 3 235B* judge, and we improve this by +45.3% absolute percentage points.  
382 Our top checkpoint outperforms the *Qwen 3 235B* data collection policy, and *Llama 4 Maverick*, a  
383 frontier LLM with 400B parameters, for which our model is 235 times smaller. Notably, filtering  
384 with a *Qwen 3 235B* judge leads to agents that improve according to independent secondary judges,  
385 including *Gemini 2.5 Flash*, *Llama 4 Maverick*, and *Gpt 4.1 Nano*, suggesting it generalizes well.

378 6.1 PERFORMANCE IMPROVES WITH DATA SCALE  
379

380 The recurring lesson in deep learning is that large-scale high-quality data wins, but researchers are  
381 struggling to materialize this promise for agents (Xue et al., 2025). Our paper aims to solve the  
382 data problem blocking researchers from materializing this promise, and this experiment provides  
383 a valuable signal that scaling high-quality data allows small models to compete with strong LLMs  
384 from top industry labs. To proceed, we collect 20k trajectories using a *Qwen 3 235B* data collection  
385 policy, annotated with scores from a *Qwen 3 235B* judge. We then train agents based on *Qwen 3*  
386 *1.7B* with SFT on varying scales of the data. Results in Figure 7 show that performance improves  
387 with increasing data scale, and gains scale faster on data filtered by the judge. To filter the data, we  
388 select trajectories where  $\text{Judge}(\text{Success}) = 1$ . Our top checkpoint outperforms the *Qwen 3*  
389 *235B* data collection policy, and beats *Llama 4 Maverick*, a frontier LLM with 400B parameters, for  
390 which our model is 235 times smaller. The trend in the figure suggests there is room to scale further.  
391

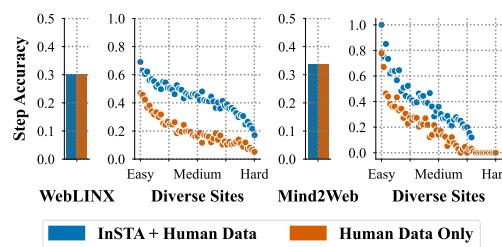


392 **Figure 8: Our agents zero-shot transfer to WebVoyager.** With no additional training or specialized  
393 data, our checkpoints for *Qwen 3 1.7B* in Section 6.1 zero-shot transfer to the WebVoyager benchmark.  
394 Trends found on our test set appear to hold for WebVoyager as well, and our top checkpoints for  
395 *Qwen 3 1.7B* continue to match frontier LLMs in performance for three of four judges.  
396

403 6.2 AGENTS TRANSFER TO NEW DOMAINS  
404

405 Statistically correct evaluation for deep learning models requires a test dataset that does not overlap  
406 with the training dataset, but researchers do not agree on how to implement this guideline for agents.  
407 Recent works train agents on the same websites they test on (Murty et al., 2025; Zhou et al., 2024c;  
408 Putta et al., 2024), which may obfuscate progress (Xue et al., 2025). Our next experiment shows that  
409 we can implement a stronger train-test split. Agents trained with our data can zero-shot transfer to  
410 WebVoyager (He et al., 2024) without any data from the benchmark. Results in Figure 8 show our  
411 *Qwen 3 1.7B* checkpoint matching strong LLMs on WebVoyager (He et al., 2024) for three of four  
412 judges, confirming trends in Section 6.1. Our ability to zero-shot transfer relatively small agents to  
413 WebVoyager (He et al., 2024) makes it likely that our pipeline leads to capable agents.  
414

415 **Static Benchmarks.** To complement the online  
416 evaluation in Figure 8, we also explore how our  
417 data impacts agents trained on static benchmarks.  
418 We first train baseline agents on human demonstra-  
419 tions from the Mind2Web (Deng et al., 2023) and  
420 WeBLINX (Lù et al., 2024) datasets. We write a  
421 preprocessor to convert our data into the expected  
422 action-observation formats these benchmarks use  
423 (which involves discarding our reasoning trace).  
424 With data converted, we compare the baseline to  
425 agents trained on a mix of 80% human data, and  
426 20% our data, and test on (1) their official test set,  
427 and (2) 500 diverse sites from our official test set.  
428 Results in Figure 9 show that agents trained with  
429 our data perform equally well on the original test  
430 sets for these benchmark, but generalize better to  
431 our harder test set. Overall, we see +149.0% for WeBLINX agents, +156.3% for Mind2Web agents,  
432 and gains in *Step Accuracy* on our test set are larger for the harder tasks. Additional experimental  
433 details are listed in Appendix H. On three popular benchmarks (WebVoyager, WeBLINX, Mind2Web),  
434 our pipeline trains capable agents, and does not overly rely on human annotations.  
435



436 **Figure 9: Our data transfers to static benchmarks.**  
437 We train agents with all human data from the We-  
438 bLINX and Mind2Web training sets, and resulting  
439 agents struggle to generalize to more diverse test data.  
440 Adding our data improves generalization by +149.0%  
441 for WebLINX, and +156.3% for Mind2Web.  
442

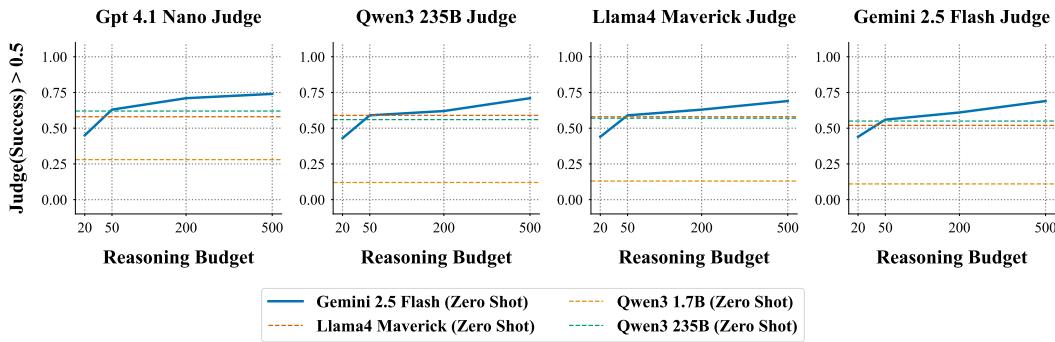


Figure 10: **Agents improve with a larger reasoning budget.** We ablate the number of tokens in the reasoning budget for the top-performing agent, and see a monotonic improvement in the success rate as the reasoning budget increases. *Gemini 2.5 Flash* has a 70% success rate with a budget of 500 reasoning tokens, up from 60% for a budget of 50 tokens. The scaling of performance with the reasoning budget highlights a promising behavior in successful web agents that can be studied.

### 6.3 PERFORMANCE SCALES WITH TEST-TIME COMPUTE

To understand the ability for web agents to scale with additional test-time compute, we ablate the number of tokens in the reasoning budget. Figure 10 shows the success rate as a function of the reasoning budget for *Gemini 2.5 Flash*, the top-performing agent we tested. There is a monotonic improvement in the success rate as the reasoning budget increases, and the trend suggests that performance may not be saturated with 500 reasoning tokens. Training language model agents to reason before taking actions is a promising path to better agents, and we are releasing a large reasoning dataset for multimodal agents to study this. Our dataset contains 2.2M screenshots, 2.2M reasoning traces for actions, 150k traces for judge evaluations, and led to the results in Section 6.1. The data will be linked on our website, alongside an official huggingface dataset for tasks.

## 7 CONCLUSION

In the spirit of deep learning, we have developed an approach to efficiently harness internet data for LLM agents, and have unlocked a *dynamic internet-scale environment*. In building this environment, we presented a method to annotate 150k diverse sites with challenging agentic tasks, and showed how training on data from our pipeline allows small models to compete with frontier LLMs as agents, on a fraction of the budget. Our pipeline consists of a task proposer, agent, and judge driven by pretrained language models that together curate high-quality data for agents, without human intervention.

Our top checkpoint for *Qwen 3 1.7B* has a success rate of 56.9% on our test environment, outperforming the data collection policy *Qwen 3 235B*, beating the 235 times larger *Llama 4 Maverick*, and reaching 94.7% of the performance of *Gemini 2.5 Flash*, while being smaller and faster than these. Our models zero-shot transfer to WebVoyager, and scale with test-time compute. We are releasing the entire pipeline, including code, models and data, so that it may serve as a foundation for researchers to build the next generation of language model agents with internet data.

### 7.1 FUTURE WORK

Our work reveals several exciting directions in future work. First, our work can be scaled further: the latest Common Crawl release contains data for more than 300M sites, suggesting another 1,000 times more data could be available for agents by scaling the pipeline. In addition, we trained agents to optimize the judge scores indirectly via filtered SFT, and the judge’s high accuracy suggests that it could be optimized via reinforcement learning instead. RL is especially promising for how it can improve reasoning capabilities in agents. Finally, while the data we collect is multimodal, we focus on textual tasks in this paper, and our pipeline could be extended to produce multimodal tasks.

486 REFERENCES  
487

488 Jacob Andreas. Language models as agent models. In Yoav Goldberg, Zornitsa Kozareva, and  
489 Yue Zhang (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp.  
490 5769–5779, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational  
491 Linguistics. doi: 10.18653/v1/2022.findings-emnlp.423. URL <https://aclanthology.org/2022.findings-emnlp.423>.

492

493 Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi,  
494 Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefer. Graph  
495 of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI  
496 Conference on Artificial Intelligence*, 38(16):17682–17690, Mar. 2024. doi: 10.1609/aaai.v38i16.  
497 29720. URL <https://ojs.aaai.org/index.php/AAAI/article/view/29720>.

498 Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal,  
499 Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel  
500 Herbert Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler,  
501 Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott  
502 Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya  
503 Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL <https://arxiv.org/abs/2005.14165>.

504

505 Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar,  
506 Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio  
507 Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4,  
508 2023. URL <https://arxiv.org/abs/2303.12712>.

509

510 Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact:  
511 Toward language agent fine-tuning, 2023. URL <https://arxiv.org/abs/2310.05915>.

512 Thibault Le Sellier De Chezelles, Maxime Gasse, Alexandre Drouin, Massimo Caccia, Léo Boisvert,  
513 Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang,  
514 Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Quentin Cappart, Graham  
515 Neubig, Ruslan Salakhutdinov, Nicolas Chapados, and Alexandre Lacoste. The browsergym  
516 ecosystem for web agent research, 2024. URL <https://arxiv.org/abs/2412.05467>.

517

518 Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and  
519 Yu Su. Mind2web: Towards a generalist agent for the web, 2023. URL <https://arxiv.org/abs/2306.06070>.

520

521 Saumya Gandhi, Ritu Gala, Vijay Viswanathan, Tongshuang Wu, and Graham Neubig. Better  
522 synthetic data by retrieving and transforming existing datasets, 2024. URL <https://arxiv.org/abs/2404.14361>.

523

524 Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, and et al. The llama 3 herd of models, 2024.  
525 URL <https://arxiv.org/abs/2407.21783>.

526

527 Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong  
528 Lan, and Dong Yu. WebVoyager: Building an end-to-end web agent with large multimodal  
529 models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd  
530 Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp.  
531 6864–6890, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi:  
532 10.18653/v1/2024.acl-long.371. URL <https://aclanthology.org/2024.acl-long.371>.

533

534 Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan  
535 Wang, Yuxuan Zhang, Juanzi Li, Bin Xu, Yuxiao Dong, Ming Ding, and Jie Tang. Cogagent: A  
536 visual language model for gui agents, 2023. URL <https://arxiv.org/abs/2312.08914>.

537

538 Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael  
539 Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama guard: Llm-based  
input-output safeguard for human-ai conversations, 2023. URL <https://arxiv.org/abs/2312.06674>.

540 Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham  
 541 Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating  
 542 multimodal agents on realistic visual web tasks, 2024a. URL <https://arxiv.org/abs/2401.13649>.

543

544 Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. Tree search for language  
 545 model agents, 2024b. URL <https://arxiv.org/abs/2407.01476>.

546

547 Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E.  
 548 Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model  
 549 serving with pagedattention, 2023. URL <https://arxiv.org/abs/2309.06180>.

550

551 Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton  
 552 Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. Rlaif vs. rlhf:  
 553 Scaling reinforcement learning from human feedback with ai feedback, 2024. URL <https://arxiv.org/abs/2309.00267>.

554

555 Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita  
 556 Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, Kai Shu, Lu Cheng, and Huan Liu. From  
 557 generation to judgment: Opportunities and challenges of llm-as-a-judge. *arXiv preprint arXiv:*  
 558 [2411.16594](https://arxiv.org/abs/2411.16594), 2024.

559

560 Junpeng Liu, Yifan Song, Bill Yuchen Lin, Wai Lam, Graham Neubig, Yuanzhi Li, and Xiang  
 561 Yue. Visualwebbench: How far have multimodal llms evolved in web page understanding and  
 562 grounding?, 2024. URL <https://arxiv.org/abs/2404.05955>.

563

564 Xing Han Lù, Zdeněk Kasner, and Siva Reddy. Weblinx: Real-world website navigation with  
 565 multi-turn dialogue, 2024. URL <https://arxiv.org/abs/2402.05930>.

566

567 Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon,  
 568 Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder,  
 569 Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative  
 570 refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing  
 Systems*, 2023. URL <https://openreview.net/forum?id=S37h0erQLB>.

571

572 Microsoft. Playwright. <https://github.com/microsoft/playwright>, 2024.

573

574 Arindam Mitra, Luciano Del Corro, Guoqing Zheng, Shweta Mahajan, Dany Rouhana, Andres Codas,  
 575 Yadong Lu, Wei ge Chen, Olga Vrousgos, Corby Rosset, Fillipe Silva, Hamed Khanpour, Yash  
 576 Lara, and Ahmed Awadallah. Agentinstruct: Toward generative teaching with agentic flows, 2024.  
 577 URL <https://arxiv.org/abs/2407.03502>.

578

579 Shikhar Murty, Hao Zhu, Dzmitry Bahdanau, and Christopher D. Manning. Nnetnav: Unsupervised  
 580 learning of browser agents through environment interaction in the wild, 2025. URL <https://arxiv.org/abs/2410.02907>.

581

582 Tianyue Ou, Frank F. Xu, Aman Madaan, Jiarui Liu, Robert Lo, Abishek Sridhar, Sudipta Sengupta,  
 583 Dan Roth, Graham Neubig, and Shuyan Zhou. Synatra: Turning indirect knowledge into direct  
 584 demonstrations for digital agents at scale, 2024. URL <https://arxiv.org/abs/2409.15637>.

585

586 Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong  
 587 Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton,  
 588 Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and  
 589 Ryan Lowe. Training language models to follow instructions with human feedback. In *Proceedings  
 590 of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red  
 591 Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9781713871088.

592

593 Ajay Patel, Markus Hofmarcher, Claudiu Leoveanu-Condrei, Marius-Constantin Dinu, Chris Callison-  
 Burch, and Sepp Hochreiter. Large language models can self-improve at web agent tasks, 2024.  
 594 URL <https://arxiv.org/abs/2405.20309>.

594 Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West,  
 595 and Boi Faltings. REFINER: Reasoning feedback on intermediate representations. In Yvette  
 596 Graham and Matthew Purver (eds.), *Proceedings of the 18th Conference of the European Chapter  
 597 of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1100–1126,  
 598 St. Julian’s, Malta, March 2024. Association for Computational Linguistics. URL <https://aclanthology.org/2024.eacl-long.67>.

600 Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and  
 601 Rafael Rafailev. Agent q: Advanced reasoning and learning for autonomous ai agents, 2024. URL  
 602 <https://arxiv.org/abs/2408.07199>.

603

604 Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language  
 605 models are unsupervised multitask learners, 2019. URL <https://api.semanticscholar.org/CorpusID:160025533>.

606

607 Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Android in  
 608 the wild: A large-scale dataset for android device control, 2023. URL <https://arxiv.org/abs/2307.10088>.

609

610

611 Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke  
 612 Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach  
 613 themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*,  
 614 2023. URL <https://openreview.net/forum?id=Yacmpz84TH>.

615

616 Amrith Setlur, Saurabh Garg, Xinyang Geng, Naman Garg, Virginia Smith, and Aviral Kumar. RI  
 617 on incorrect synthetic data scales the efficiency of llm math reasoning by eight-fold, 2024. URL  
 618 <https://arxiv.org/abs/2406.14532>.

619

620 Junhong Shen, Atishay Jain, Zedian Xiao, Ishan Amlekar, Mouad Hadji, Aaron Podolny, and Ameet  
 621 Talwalkar. Scribeagent: Towards specialized web agents using production-scale workflow data,  
 622 2024. URL <https://arxiv.org/abs/2411.15004>.

623

624 Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally  
 625 can be more effective than scaling model parameters, 2024. URL <https://arxiv.org/abs/2408.03314>.

626

627 Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang,  
 628 Peter Bartlett, and Andrea Zanette. Fast best-of-n decoding via speculative rejection. In *The  
 629 Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL <https://openreview.net/forum?id=348hfcpUs>.

630

631 Fahim Tajwar, Anikait Singh, Archit Sharma, Rafael Rafailev, Jeff Schneider, Tengyang Xie, Stefano  
 632 Ermon, Chelsea Finn, and Aviral Kumar. Preference fine-tuning of llms should leverage suboptimal,  
 633 on-policy data, 2024. URL <https://arxiv.org/abs/2404.14367>.

634

635 The Common Crawl Foundation. Common crawl, 2025. URL <https://commoncrawl.org/>.

636

637 Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée  
 638 Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand  
 639 Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language  
 640 models, 2023a. URL <https://arxiv.org/abs/2302.13971>.

641

642 Hugo Touvron, Louis Martin, Kevin Stone, and et al. Llama 2: Open foundation and fine-tuned chat  
 643 models, 2023b. URL <https://arxiv.org/abs/2307.09288>.

644

645 Brandon Trabucco, Kyle Doherty, Max A Gurinas, and Ruslan Salakhutdinov. Effective data  
 646 augmentation with diffusion models. In *The Twelfth International Conference on Learning  
 647 Representations*, 2024. URL <https://openreview.net/forum?id=ZWzUA9zeAg>.

648

649 Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. Llms still can’t plan; can lrms? a  
 650 preliminary evaluation of openai’s o1 on planbench, 2024. URL <https://arxiv.org/abs/2409.13373>.

648 Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai  
 649 Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on  
 650 large language model based autonomous agents. *Frontiers of Computer Science*, 18(6), March  
 651 2024. ISSN 2095-2236. doi: 10.1007/s11704-024-40231-1. URL <http://dx.doi.org/10.1007/s11704-024-40231-1>.

653 Junlin Xie, Zhihong Chen, Ruifei Zhang, Xiang Wan, and Guanbin Li. Large multimodal agents: A  
 654 survey, 2024. URL <https://arxiv.org/abs/2402.15116>.

656 Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Song, Huan Sun, and  
 657 Yu Su. An illusion of progress? assessing the current state of web agents. 2025. URL <https://arxiv.org/abs/2504.01382>.

659 Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable  
 660 real-world web interaction with grounded language agents, 2023a. URL <https://arxiv.org/abs/2207.01206>.

663 Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik  
 664 Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023b.  
 665 URL <https://arxiv.org/abs/2305.10601>.

666 Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and  
 667 James Zou. Textgrad: Automatic "differentiation" via text, 2024. URL <https://arxiv.org/abs/2406.07496>.

669 Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning:  
 670 Enabling generalized agent abilities for llms, 2023. URL <https://arxiv.org/abs/2310.12823>.

672 Chi Zhang, Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu.  
 674 Appagent: Multimodal agents as smartphone users, 2023. URL <https://arxiv.org/abs/2312.13771>.

676 Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal.  
 677 Generative verifiers: Reward modeling as next-token prediction, 2024. URL <https://arxiv.org/abs/2408.15240>.

680 Tianyang Zhong, Zhengliang Liu, Yi Pan, and et al. Evaluation of openai o1: Opportunities and  
 681 challenges of agi, 2024. URL <https://arxiv.org/abs/2409.18486>.

682 Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language  
 683 agent tree search unifies reasoning acting and planning in language models, 2024a. URL <https://arxiv.org/abs/2310.04406>.

686 Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng,  
 687 Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic  
 688 web environment for building autonomous agents, 2024b. URL <https://arxiv.org/abs/2307.13854>.

690 Yifei Zhou, Qianlan Yang, Kaixiang Lin, Min Bai, Xiong Zhou, Yu-Xiong Wang, Sergey Levine,  
 691 and Erran Li. Proposer-agent-evaluator(pae): Autonomous skill discovery for foundation model  
 692 internet agents, 2024c. URL <https://arxiv.org/abs/2412.13194>.

693  
 694  
 695  
 696  
 697  
 698  
 699  
 700  
 701

702 **A LIMITATIONS & SAFEGUARDS**  
703

704  
 705 Language model agents present unique challenges and risks when applied to live tasks on the internet.  
 706 For instance, agents visiting shopping sites can influence the statistics produced by analytics tools,  
 707 which can impact prices on products, and product decisions from companies. Furthermore, agents  
 708 seeing harmful content on the web can add that content to datasets inadvertently, and propagate  
 709 harmful behaviors to future agents. We mitigate these risks in the design of the task proposal stage.  
 710 We consider the risks posed to analytics tools by limiting the engagement between agents and sites.  
 711 We generate only one task per website, and we limit agents to just 30 actions per site, which includes  
 712 clicks, typing, dropdown selection, and more. By limiting the interaction between agents and sites, the  
 713 change in website traffic generated by the InSTA pipeline is minimal (just 90 seconds of interaction  
 714 per site on average). By utilizing data from the InSTA pipeline in an offline fashion, as in Section 6  
 715 of the main paper, no additional web traffic is generated when training agents. To ensure that agents  
 716 do not modify the state of the web (i.e. avoid attempting to make purchases, avoid leaving comments  
 717 on posts, avoid making accounts, etc), we provide instruct the task proposer (see Figure 2) to avoid  
 718 writing tasks that require the agent to interact with personal data, or user accounts.

719 The task proposer is instructed via the system prompt to filter out sites with harmful content, sites  
 720 not intended for user access, and sites that require making an account to operate, including social  
 721 media, and forum sites. There is likely a manner to safely train agents to operate user accounts, but  
 722 we leave this task to future researchers. We explore the performance of the task proposer at filtering  
 723 out unsuitable sites in Section 4.2, and find that all models detect unsuitable sites with a recall from  
 724 0.98 to 1.0, and accuracy up to 97%, suggesting our filter is reliable. Sites used to benchmark the  
 725 performance of the safety filter are discussed in Appendix E, and thoroughly test the safety filter.

726 To remove Personally Identifiable Information (PII) from the data used for training agents, we include  
 727 `scrubadub`, an industry standard PII removal tool for python developed by Leap Beyond, a data  
 728 consultancy based in the European Union. Our pipeline has an argument that toggles the usage of  
 729 `scrubadub` to remove PII from all website data, and we recommend this option be set.

730 **B ETHICAL CONSIDERATIONS**  
731

732  
 733 One important ethical consideration when harnessing internet data is to carefully handle copyrighted,  
 734 private, and sensitive materials. The internet contains vast amounts of PII, which should be avoided  
 735 when training models. We address this in two ways. First, we instruct the task proposer to filter out  
 736 sites that may contain PII, including social media websites, and forums. Second, our pipeline has  
 737 an argument that toggles the usage of `scrubadub` to remove PII, and we recommend this option  
 738 be set. To prevent copyrighted materials from being used for training, we do not allow the agent to  
 739 copy any books, documents, videos, audio, or files. The information present in the HTML DOM is  
 740 publically available, and is converted into a sparse Markdown summary that focuses on interactive  
 741 elements, and significantly transforms the underlying webpage in a manner that is not reversible.

742 These steps significantly reduce, but do not completely eliminate the risk that private, and sensitive  
 743 materials are shown to the agent, and methods for detecting, replacing, and removing such materials  
 744 from text and images remains an important task for researchers working on safety.

745 **C BROADER IMPACTS**  
746

747  
 748 As their capabilities broaden, language models are being used to interface with real-world systems.  
 749 This shift comes with several benefits and risks. Agents that operate your computer to aid in work  
 750 tasks can significantly boost productivity for certain workers, but can displace others whose jobs have  
 751 been fully automated. Agents that operate browsers to complete personal tasks provide convenience,  
 752 but expose a new attack vector where rogue agents perform unintended actions. Certain risks can be  
 753 mitigated with proper safeguards, such as post-processing data to prevent jail-breaking, but other risks  
 754 are existential, harder to address purely with new agent research, and may require policy changes.

You are helping us create tasks for a web navigation system. We  
will tell you the domain of a website. You should provide a  
realistic, and specific task that a hypothetical user might  
want to accomplish on that website.

## Skipping Unsafe / Inappropriate Domains

To skip a domain, respond with 'N/A' instead of providing a task.

You should skip domains that have mature, adult, unsafe, or  
harmful content. If you are unsure whether a domain is safe,  
please skip it. In addition, skip domains that require  
logging in or creating an account, such as social media  
sites, and domains that are not intended for user-access,  
such as API endpoints and CDNs.

## Here are some domains to provide tasks for:

- \* 'www.amazon.com': 'Find the price of the 24in LG Ultragear  
Monitor.'
- \* 'www.wikipedia.org': 'Look up the history of the Eiffel Tower on  
Wikipedia.'

## Here are some domains to skip:

- \* 'fbcdn.net': 'N/A'
- \* 'api.github.com': 'N/A'

Tasks should not require external knowledge, not modify the state  
of the web, and should not require logging in or creating an  
account. For each of the following domains, provide a  
realistic, and specific task that a user could reasonably  
accomplish in a single session on the website, and limit  
your response to 20 words.

**Figure 11: System prompt for the exploration phase of task generation.** We design the system prompt for task generation to detect and remove unsafe websites. This prompt ensures that tasks are passive, and do not modify content on a website. Refer to the next figures for the in-context examples used for the task proposer, and the system prompt used in the feedback step.

D AGENTS.TXT & STANDARDS FOR INTERNET AGENTS

Akin to `robots.txt` directives, website creators should have a standard format to specify how internet agents are allowed to interact with their websites, and what information on webpages agents are allowed to see. Desirable controls include rate limits for interactions, limits for maximum numbers of interactions, restrictions to allow agents to interact with certain pages and not others, and restrictions on the kind of data on webpages that agents are allowed to observe (achieved via tagging elements to hide their content from agents). In addition to restricting the data available to agents, website creators should have the ability to specify locations for “playgrounds” that replicate certain key functions of their site with virtual tasks and simulated data that are intended to teach agents how to operate their site while directing traffic from agents away from user-facing pages.

## E MORE DETAILS ON TASK GENERATION

We provide the system prompt used in the first phase of the task generation loop in Figure 11. This prompt was provided to Llama 3.1 70B, GPT-4o, and Gemini 1.5 Pro to generate tasks and filter

810 sites unsuitable for annotation in Section 4. We carefully designed this system prompt to enforce  
 811 that generated tasks are passive, and do not modify content on a website. In addition to this system  
 812 prompt, we employed a list of 100 hand-picked in-context examples of website URLs and appropriate  
 813 tasks, which are provided in the following JSON list. When querying an LLM, we randomly sample  
 814 16 in-context examples from the list, and provide only these examples to the LLM to generate a task  
 815 to guide exploration of the site. This improves diversity in the exploration phase.  
 816  
 817

```

818 [
819   {
820     "domain": "archive.org",
821     "task": "Identify the oldest book available in the public
822       ↪ domain on this site."
823   },
824   {
825     "domain": "arxiv.org",
826     "task": "Retrieve the latest preprint paper on machine
827       ↪ learning."
828   },
829   {
830     "domain": "wikibooks.org",
831     "task": "Find a freely available textbook on linear algebra
832       ↪ ."
833   },
834   {
835     "domain": "wiktionary.org",
836     "task": "Get the definition and etymology of the word 'serendipity'."
837   },
838   {
839     "domain": "openlibrary.org",
840     "task": "Locate an ebook about classic literature that is
841       ↪ available for borrowing."
842   },
843   {
844     "domain": "openculture.com",
845     "task": "Find a free online course on ancient history."
846   },
847   {
848     "domain": "theguardian.com",
849     "task": "Retrieve an article discussing recent trends in
850       ↪ renewable energy."
851   },
852   {
853     "domain": "medium.com",
854     "task": "Identify a highly rated blog post on productivity
855       ↪ hacks."
856   },
857   {
858     "domain": "goodreads.com",
859     "task": "Find the most popular book related to neuroscience
860       ↪ ."
861   },
862   {
863     "domain": "wired.com",
864     "task": "Retrieve an article about the latest advancements
865       ↪ in wearable technology."
866   },
867   {
  
```

```

864     "domain": "data.gov",
865     "task": "Identify the latest government dataset on climate
866         ↪ change."
867   },
868   {
869     "domain": "kaggle.com",
870     "task": "Find a well-documented data science competition on
871         ↪ image recognition."
872   },
873   {
874     "domain": "gov.uk",
875     "task": "Locate the latest UK government report on
876         ↪ healthcare."
877   },
878   {
879     "domain": "unsplash.com",
880     "task": "Find a high-resolution image of the Milky Way
881         ↪ Galaxy."
882   },
883   {
884     "domain": "pexels.com",
885     "task": "Retrieve a popular photo tagged with 'nature'."
886   },
887   {
888     "domain": "creativecommons.org",
889     "task": "Find an article explaining Creative Commons
890         ↪ licensing types."
891   },
892   {
893     "domain": "pypi.org",
894     "task": "Retrieve the most downloaded Python package for
895         ↪ data analysis."
896   },
897   {
898     "domain": "huggingface.co",
899     "task": "Identify a popular machine learning model on this
900         ↪ platform."
901   },
902   {
903     "domain": "sciencenews.org",
904     "task": "Find the most recent article on the health impacts
905         ↪ of air pollution."
906   },
907   {
908     "domain": "mit.edu",
909     "task": "Retrieve a publicly available research paper on
910         ↪ quantum computing."
911   },
912   {
913     "domain": "springer.com",
914     "task": "Identify the latest edition of a Springer book on
915         ↪ robotics."
916   },
917   {
918     "domain": "jstor.org",
919     "task": "Find a research paper discussing the history of the
920         ↪ Internet."
921   },
922   {

```

```

918     "domain": "biorxiv.org",
919     "task": "Retrieve the most recent bioRxiv preprint on CRISPR
920         ↪ technology."
921 },
922 {
923     "domain": "medrxiv.org",
924     "task": "Find a public health preprint related to COVID-19."
925 },
926 {
927     "domain": "commons.wikimedia.org",
928     "task": "Retrieve a high-resolution image of the Eiffel
929         ↪ Tower."
930 },
931 {
932     "domain": "scholar.google.com",
933     "task": "Find the most cited article by a specific
934         ↪ researcher."
935 },
936 {
937     "domain": "plos.org",
938     "task": "Locate the latest research paper on gene editing
939         ↪ published here."
940 },
941 {
942     "domain": "flickr.com",
943     "task": "Find a photo that has been released under a
944         ↪ Creative Commons license."
945 },
946 {
947     "domain": "datacite.org",
948     "task": "Retrieve metadata for a dataset related to
949         ↪ environmental studies."
950 },
951 {
952     "domain": "orcid.org",
953     "task": "Find the ORCID ID of a well-known researcher in AI
954         ↪ ."
955 },
956 {
957     "domain": "zotero.org",
958     "task": "Retrieve an article discussing citation management
959         ↪ tools."
960 },
961 {
962     "domain": "github.com",
963     "task": "Find the most starred repository on deep learning."
964 },
965 {
966     "domain": "figshare.com",
967     "task": "Retrieve an open dataset on climate patterns."
968 },
969 {
970     "domain": "zenodo.org",
971     "task": "Find the latest publication on open science
         ↪ practices."
972 },
973 {
974     "domain": "worldcat.org",
975     "task": "Locate a catalog entry for a rare book on botany."

```

```

972 },
973 {
974     "domain": "biodiversitylibrary.org",
975     "task": "Retrieve a scanned copy of an 18th-century
976         ↪ botanical illustration."
977 },
978 {
979     "domain": "genome.gov",
980     "task": "Find the latest update on the Human Genome Project
981         ↪ ."
982 },
983 {
984     "domain": "merriam-webster.com",
985     "task": "Retrieve the definition and usage of the word 'quantum'."
986 },
987 {
988     "domain": "stanford.edu",
989     "task": "Find the most recent online lecture on artificial
990         ↪ intelligence."
991 },
992 {
993     "domain": "edx.org",
994     "task": "Retrieve a TED Talk on leadership in technology."
995 },
996 {
997     "domain": "ted.com",
998     "task": "Find the latest ocean temperature data available."
999 },
1000 {
1001     "domain": "noaa.gov",
1002     "task": "Retrieve a dataset related to consumer behavior."
1003 },
1004 {
1005     "domain": "data.world",
1006     "task": "Find a course on data visualization."
1007 },
1008 {
1009     "domain": "curious.com",
1010     "task": "Retrieve a well-cited article on the psychological
1011         ↪ impact of social media."
1012 },
1013 {
1014     "domain": "theconversation.com",
1015     "task": "Identify a recent research paper on biodiversity
1016         ↪ conservation."
1017 },
1018 {
1019     "domain": "nature.com",
1020     "task": "Retrieve the latest article on genomics research."
1021 },
1022 {
1023     "domain": "pnas.org",
1024     "task": "Find a science news article on robotics
1025         ↪ advancements."
1026 },
1027 {
1028     "domain": "sciencedaily.com",
1029     "task": "Identify the top story on global health issues."

```

```

1026 },
1027 {
1028     "domain": "bbc.com",
1029     "task": "Retrieve a recent podcast episode about space
1030         ↪ exploration."
1031 },
1032 {
1033     "domain": "npr.org",
1034     "task": "Locate the most recent update on the global
1035         ↪ biodiversity status."
1036 }
1037 ]
1038

```

1039 We also provide the system prompt used in the second phase of the task generation loop, where  
1040 trajectories from agents are fed back to the task proposer, which generates a harder, grounded task.  
1041 This prompt instructs the task proposer to create a challenging task based on how an expert user  
1042 could be expected to use the shown website. The task proposer also predicts a list on intermediate  
1043 steps that can be used as a hint for agents, and a success criteria that can be used to improve the verifier.

1044

1045

```

1046 You are a helpful assistant designing tasks for a web automation
1047     ↪ script. I will show you previous runs of the script,
1048     ↪ including previous tasks, webpages, actions, and performance
1049     ↪ reviews, formatted in markdown. Help me design *challenging
1050     ↪ * new tasks.

```

```

1051 ## Formatting The Proposed Task
1052

```

1053 Format your task in the following JSON schema:

```

1054
1055     ````json
1056     {
1057         "proposed_task": str,
1058         "steps": List[str],
1059         "criteria": str
1060     }
1061     ````
```

1062 Here is what each key means:

- 1064 - `proposed\_task`: A specific, challenging task that an expert
 ↪ user might leverage this website to complete.
  - 1066 - Must not require making an account, logging in, submitting
 ↪ personal information, making a purchase, or placing an
 ↪ order.
- 1069 - `steps`: Steps an expert user would follow to complete the
 ↪ proposed task.
- 1071 - `criteria`: The required answer, and criteria to determine if
 ↪ the task was completed.

```

1072 ## Example Tasks For Inspiration
1073

```

1074 Suppose you want to design a task around the 'C-to-C Hose-Shut-Off
1075 ↪ Valve' on 'awg-fittings.com':

```

1076
1077     ````json
1078     {
```

```

1080 "proposed_task": "What is the C-to-C Hose-Shut-Off Valve length
1081     ↪ in mm?", 
1082 "steps": [
1083     "Navigate to 'awg-fittings.com'", 
1084     "Open the product catalog for fittings", 
1085     "Locate the product listing for the C-to-C Hose-Shut-Off
1086         ↪ Valve", 
1087     "Find the product length in mm, and respond with that length
1088         ↪ in the answer"
1089 ],
1090 "criteria": "The answer should include the specific length of
1091     ↪ '237 mm' for this product"
1092 }
1093 ````

1094 Suppose you want to design a task around the document 'The Angora
1095     ↪ cat; how to breed train and keep it' on 'biodiversitylibrary.
1096     ↪ org':
1097
1098 ````json
1099 {
1100     "proposed_task": "Open a scanned copy of 'The Angora cat; how
1101         ↪ to breed train and keep it'.", 
1102     "steps": [
1103         "Navigate to 'biodiversitylibrary.org'", 
1104         "Search for 'The Angora cat; how to breed train and keep it'
1105             ↪ in the search bar", 
1106         "Click on the title of the document in the search results", 
1107         "Confirm the correct document is displayed in an embedded
1108             ↪ PDF reader"
1109 ],
1110 "criteria": "The final webpage should display the correct
1111     ↪ document in an embedded PDF reader"
1112 }
1113 ````

1114 Suppose you want to design a task around the 'Generative
1115     ↪ Adversarial Networks' paper on 'scholar.google.com':
1116
1117 ````json
1118 {
1119     "proposed_task": "How many citations does the paper 'Generative
1120         ↪ Adversarial Networks' have?", 
1121     "steps": [
1122         "Navigate to 'scholar.google.com'", 
1123         "Search for 'Generative Adversarial Networks' in the search
1124             ↪ bar", 
1125         "Locate the correct paper in the search results", 
1126         "Find an up-to-date citation count, and respond with that
1127             ↪ count in the answer"
1128 ],
1129 "criteria": "The answer should include an up-to-date citation
1130     ↪ count, which is '80613' as of April 2025"
1131 }
1132 ````

1133 Suppose you want to design a task around the word 'serendipity' on
1134     ↪ 'wiktionary.org':

```

```

1134 ````json
1135 {
1136     "proposed_task": "What is the definition and etymology of the
1137         ↪ word 'serendipity'?", 
1138     "steps": [
1139         "Navigate to 'wiktionary.org'", 
1140         "Search for 'serendipity' in the search bar", 
1141         "Find the definition and etymology sections of the "
1142             ↪ 'serendipity' page", 
1143         "Summarize the contents of these sections in the answer"
1144     ],
1145     "criteria": "The answer should mention Serendip (or Serendib),
1146         ↪ coined by English writer and politician Horace Walpole in
1147             ↪ 1754"
1148 }
1149 ````
```

1150 Thanks for helping me design challenging new tasks, please follow  
1151 ↪ the instructions carefully. Start your response with an  
1152 ↪ analysis for how an expert user would leverage this website,  
1153 ↪ followed by a step-by-step breakdown of your proposed task,  
1154 ↪ and finally, enter your task in the JSON format. Respond in  
1155 ↪ 500 words.

## E.1 DETAILS FOR SAFETY EXPERIMENT

1156 Using these prompts for task generation, we remove unsafe websites. To evaluate the performance  
1157 of our filter, we employed a set of 100 curated websites, where 50 are manually verified as safe,  
1158 and 50 are manually verified as unsafe based on the filtering conditions. These sites were chosen to  
1159 span popular sites that typical users are likely familiar with, and less popular websites that may be  
1160 underrepresented in LLM training data.

```

1161 safe_sites_list = ['dhss.mo.gov', 'dizionario.corriere.it', '
1162     ↪ southgippsland.vic.gov.au', 'ds.iris.edu', 'lobbycontrol.de
1163     ↪ ', '4rsmokehouse.com', 'barnsleyfc.co.uk', 'wiwi.uni-
1164     ↪ wuerzburg.de', 'uplandca.gov', 'lsus.edu', 'wpcode.com', '
1165     ↪ webopedia.internet.com', 'tamko.com', 'premierchristian.news
1166     ↪ ', 'genome.jgi.doe.gov', 'burgerking.ca', 'thehugoawards.org
1167     ↪ ', 'radio.fm', 'thevinyldistrict.com', 'unilang.org', '
1168     ↪ raywhitegroup.com', 'grapevinetexas.gov', 'sanfrancisco.
1169     ↪ cbslocal.com', 'hyde-design.co.uk', 'breastcancerfoundation.
1170     ↪ org.nz', 'ludwigsburg.de', 'ignitionrobotics.org', '
1171     ↪ deliverit.com.au', 'kodokan.org', 'clickstay.com', '
1172     ↪ searchdatamanagement.techtarget.com', 'oceanario.pt', '
1173     ↪ wentworthpuzzles.com', 'catholicworldreport.com', 'quizlet.
1174     ↪ com', 'innovation.nhs.uk', 'synonyms.reverso.net', 'news.
1175     ↪ siemens.co.uk', 'readability-score.com', 'co.modoc.ca.us', '
1176     ↪ cityofmyrtlebeach.com', 'loire.gouv.fr', 'lawphil.net', '
1177     ↪ saem.org', 'parmigianoreggiano.it', 'engaging-data.com', '
1178     ↪ itf-tkd.org', 'aka.education.gov.uk', 'ub.uni-kl.de', '
1179     ↪ mottchildren.org']
```

1180 unsafe\_sites\_list = ['refer to the code release']

1181 We provide the list of unsafe sites in the code to keep this manuscript clean. Using these lists, we  
1182 conduct an experiment measuring the accuracy, precision, and recall of our safety filter for detecting  
1183 sites that are not suitable for training agents. In this experiment, we run the initial exploration phase

1188 of the task proposer with the same prompts used in the main experiments (shown in the last section),  
 1189 and we consider a website to be marked positive for unsafe content if the task proposer generates  
 1190 “N/A” rather than a task. We then calculate accuracy, precision, and recall for various LLMs.  
 1191

1192 **E.2 DETAILS FOR RELIABILITY EXPERIMENTS**  
 1193

1194 We evaluated the verifiable rate of tasks generated by the initial phase of task generation manually.  
 1195 For a set of 100 randomly sampled websites marked as safe by the task proposer in the initial phase,  
 1196 we attempted to complete the generated task, and checked that enough information is present on the  
 1197 website that a solution can be verified. In total, we annotated 300 tasks for Table 2 in 6 hours, and  
 1198 provide the 100 website URLs used in this experiment in the following code block.  
 1199

```
1200 reliability_sites_list = ['godaddy.com', 'chrome.google.com', '  

  1201   ↪ apple.com', 'support.cloudflare.com', 'support.apple.com', '  

  1202   ↪ edition.cnn.com', 'go.microsoft.com', 'google.de', 'w3.org',  

  1203   ↪ 'yandex.ru', 'bfdi.bund.de', 'microsoft.com', 'apps.apple.  

  1204   ↪ com', 'networksolutions.com', 'support.mozilla.org', 'yelp.  

  1205   ↪ com', 'cnn.com', 'ec.europa.eu', 'developer.mozilla.org', '  

  1206   ↪ icann.org', 'books.google.com', 'globeNewswire.com', '  

  1207   ↪ onlinelibrary.wiley.com', 'gnu.org', 'slideshare.net', '  

  1208   ↪ metacpan.org', 'porkbun.com', 'oag.ca.gov', 'spiegel.de', '  

  1209   ↪ linuxfoundation.org', 'help.opera.com', 'mayoclinic.org', '  

  1210   ↪ podcasts.apple.com', 'nhs.uk', 'addons.mozilla.org', 'google.  

  1211   ↪ fr', 'pewresearch.org', 'finance.yahoo.com', 'weforum.org',  

  1212   ↪ 'g2.com', 'savethechildren.org', 'news.com.au', 'biblia.com  

  1213   ↪ ', 'yr.no', 'engadget.com', 'microsoftstore.com', 'ema.  

  1214   ↪ europa.eu', 'theintercept.com', 'princeton.edu', '  

  1215   ↪ foodandwine.com', 'sfgate.com', 'voguebusiness.com', '  

  1216   ↪ ourworldindata.org', 'livingwage.org.uk', 'cms.law', '  

  1217   ↪ msdmanuals.com', 'websitestsetup.org', 'support.xbox.com', '  

  1218   ↪ treehugger.com', 'tripadvisor.com.pe', 'mondragon.edu', '  

  1219   ↪ greenparty.ca', 'aaojournal.org', 'restaurantpassion.com', '  

  1220   ↪ iwillteachyouoberich.com', 'moneyconvert.net', '  

  1221   ↪ gesundheitsinformation.de', 'ovc.uoguelph.ca', 'zdnet.be', '  

  1222   ↪ oxfordamerican.org', 'snackandbakery.com', 'journals.uic.edu  

  1223   ↪ ', 'confused.com', 'standards.globalspec.com', '  

  1224   ↪ onlyinyourstate.com', 'ahsgardening.org', 'wyze.com', '  

  1225   ↪ nornickel.ru', 'viessmann.fr', 'benetton.com', 'firecomm.gov.  

  1226   ↪ mb.ca', 'executedtoday.com', 'eukn.eu', 'fraeylemaborg.nl',  

  1227   ↪ 'verizon.com/about/news-center', 'orthodoxalbania.org', '  

  1228   ↪ cheapjoes.com', 'bake-eat-repeat.com', '  

  1229   ↪ plattformpatientensicherheit.at', 'hifinews.com', '  

  1230   ↪ cellsignal.com', 'thenotariessociety.org.uk', 'chosenfoods.  

  1231   ↪ com', 'westerndressageassociation.org', 'pridesource.com', '  

  1232   ↪ northtacomapediatricdental.com', 'strade-bianche.it', '  

  1233   ↪ pdairport.com', 'institute.sandiegozoo.org', 'raintaxi.com  

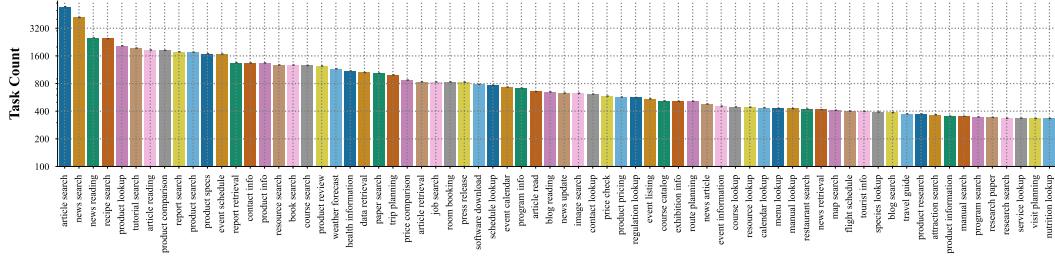
  1234   ↪ ']
```

1235 **E.3 AUTOMATIC TASK CATEGORIZATION**

1236 We employ *Llama 3.1 70B* to categorize tasks. We prompt *Llama 3.1 70B* with the system prompt in  
 1237 Figure 12 to assign a category in 3 words or less to encourage simple categories. Categories have  
 1238 16.9 tasks on average, and 953 categories have more than the mean, while 7741 have less than the  
 1239 mean. There is occasional overlap between categories, which can be observed in Figure 13, but for  
 1240 the purposes of understanding performance by category, overlap is acceptable provided categories  
 1241 have sufficiently large numbers of tasks, and performance per category can be accurately calculated.  
 We provide our task categorization script in the official code release.

1242 You are a helpful scientific assistant categorizing tasks on the  
 1243 ↪ web. You will observe a domain and web navigation task, and  
 1244 ↪ you should provide a concise categorization of the task in 3  
 1245 ↪ words or less. For example, if the domain is "google.com"  
 1246 ↪ and the task is "find a recipe for mashed potato", you may  
 1247 ↪ categorize the task as "recipe search".  
 1248  
 1249 **## Task Format**  
 1250  
 1251 Here is the format for the task:  
 1252  
 1253 **[domain]: [task]**  
 1254  
 1255 Here is what each part means:  
 1256  
 1257 **'[domain]'**: The domain of the website you are observing.  
 1258 **'[task]'**: The task a user is trying to accomplish on the website.  
 1259  
 1260 **## Response Format**  
 1261  
 1262 Respond with a category name for the task in 3 words or less, and  
 1263 ↪ provide only the category name, do not provide an  
 1264 ↪ explanation or justification for the categorization.  
 1265  
 1266 Here is the next task, please follow the instructions carefully.

1267 **Figure 12: System prompt for task categorization.** We employ *Llama 3.1 70B* to automatically  
 1268 label task categories for our dataset. We prompt the LLM to assign categories in 3 words or less, and  
 1269 set the sampling temperature to 0.5 to encourage predictions to use more consistent language. Using  
 1270 these categories, we seek to understand agent performance by category.



1281 **Figure 13: Largest categories for task generation.** We categorize 150k tasks generated by our  
 1282 pipeline in Section 4, and visualize the number of tasks in the largest 70 categories. Top categories  
 1283 include *article search*, *news search*, *recipe search*, and *product lookup*. The top 12 task categories  
 1284 have more than 1600 tasks assigned to each of them, the mean number of tasks per category is 16.9,  
 1285 and 89% of categories (7741 in total) have fewer than the mean number of tasks.

## F UNDERSTANDING AGENT CAPABILITIES & LIMITATIONS

1286  
 1287  
 1288 To complement the analyses presented in Section 5, we explore the categories of tasks that agents  
 1289 succeed at most frequently. Shown in Figure 14, we plot the average judge success probability  
 1290 prediction  $r_T$  versus task category for the top 70 most successful categories that have at least 100  
 1291 tasks assigned to them. Based on the figure, top categories include searching for *contact information*,  
 1292 finding *hours of operation*, looking up *biographical information*, obtaining current *weather forecasts*,  
 1293 and conducting *health research*. Based on these results, the top 22 categories are solved with an  
 1294 average success probability  $> 0.5$  using zero-shot agents based on *Llama 3.1 70B*. As stronger  
 1295

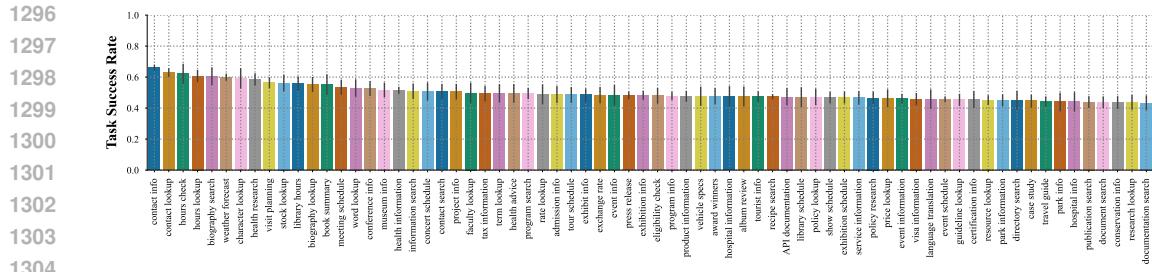


Figure 14: **Most solved categories for task generation.** We explore the completion rates for the top categories of tasks generated by our pipeline. We restrict our focus to categories where at least 100 tasks are assigned, and plots the success rates for the top 70 categories. Results show that 22 of these categories are solved with more than a 50% rate with zero-shot agents based on *Llama 3.1 70B*.

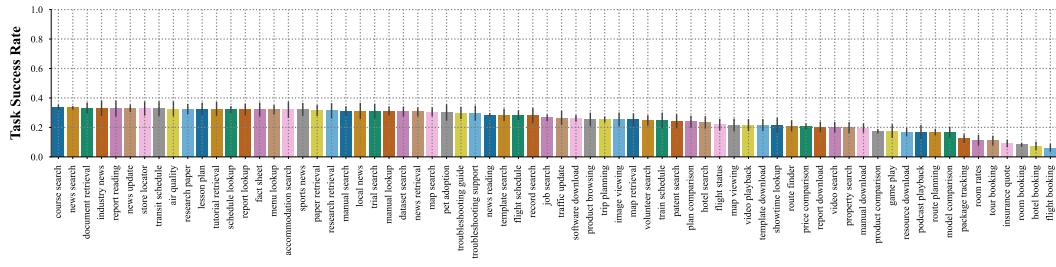


Figure 15: **Least successful categories for internet-scale task generation.** Similar to the previous figure, we explore the rates of task completion for the bottom 70 categories that have at least 100 tasks assigned to them. While the majority of the least successful categories have success rates greater than 20%, performance drops as low as 5%. Many of the categories shown in the plot above involve actions that are not feasible given the current limitations of the Playwright API, and may be possible in future work that extends agents to a fully-operable virtual computer environment. In addition, better LLM backbones are likely to improve performance.

models are developed, the success rates for agents running in our pipeline are likely to improve, and the quality of the data we generate will jointly improve.

In addition to studying the best-performing categories, we also explore the limitations of current agents via their least successful categories. Shown in Figure 15, we select the bottom 70 categories via their average judge success probability for categories with at least 100 tasks assigned. Many of these categories require agents to remember and reason about previous interactions, such as the *product comparison* category. For this category, an agent must review several products, and compare details from memory. In these cases, access to a note-taking tool may improve performance. Additionally, certain task categories involve requests that are not feasible given the limitations of the Playwright API, including categories for *downloading reports / manuals*, and *opening files*. While these tasks are not currently feasible, providing agents with a fully-operable virtual computer environment with applications pre-installed could unlock these abilities in future work.

## G AGENT, JUDGE & TASK PROPOSER SYSTEM PROMPTS

We provide the system prompt that powers the agent in this paper. This prompt is released in the code, alongside a fast HTML to Markdown processor we built. The agent prompt is carefully designed to elicit reasoning capabilities, and the experiment in Figure 10 shows the prompt is successful.

You are helping me complete tasks by operating a web browser. I  
 ↪ will share the current task, and a sequence of webpages and  
 ↪ actions from previous steps.

## Action Instructions

```

1350
1351 Based on the information we discovered so far, and the progress we
1352 ↪ made in previous steps, you are helping me determine the
1353 ↪ next action.
1354
1355 You will provide an action as JSON in a fenced code block:
1356
1357 ```json
1358 {
1359     "action_key": str,
1360     "action_kwargs": dict,
1361     "target_element_id": int | null
1362 }
1363 ```
1364 Actions have the following components:
1365
1366 - `action_key`: The name of the selected action.
1367 - `action_kwargs`: A dictionary of arguments for the action.
1368 - `target_element_id`: An optional id for the element to call the
1369 ↪ action on.
1370
1371 ## Action Definitions
1372 I've prepared an API documentation below that defines the actions
1373 ↪ we can use to complete the task.
1374
1375 ### Click Action Definition
1376
1377 - `click`: Click on an element specified by `target_element_id`.
1378
1379 ### Example Click Action
1380
1381 Suppose you want to click `[id: 5] Sales link`:
1382
1383 ```json
1384 {
1385     "action_key": "click",
1386     "action_kwargs": {},
1387     "target_element_id": 5
1388 }
1389 ```
1390 ### Hover Action Definition
1391
1392 - `hover`: Hover over an element specified by `target_element_id`.
1393
1394 ### Example Hover Action
1395
1396 Suppose you want to hover over `[id: 2] Company Logo image`:
1397
1398 ```json
1399 {
1400     "action_key": "hover",
1401     "action_kwargs": {},
1402     "target_element_id": 2
1403 }
1404 ```

```

```

1404     """ Scroll Action Definition
1405
1406     - 'scroll': Scroll the page by 'delta_x' pixels to the right and 'delta_y' pixels down.
1407     - 'delta_x': The number of pixels to scroll to the right.
1408     - 'delta_y': The number of pixels to scroll down.
1409
1410
1411     """ Example Scroll Action
1412
1413 Suppose you want to scroll down the page by 300 pixels:
1414
1415     ```json
1416     {
1417         "action_key": "scroll",
1418         "action_kwargs": {
1419             "delta_x": 0,
1420             "delta_y": 300
1421         },
1422         "target_element_id": null
1423     }
1424
1425
1426     """ Fill Action Definition
1427
1428     - 'fill': Fill an input element specified by 'target_element_id' with text.
1429     - 'value': The text value to fill into the element.
1430
1431     """ Example Fill Action (Text Input)
1432
1433 Suppose you want to fill '[id: 13] "Name..." (Enter your name text input)' with the text 'John Doe':
1434
1435     ```json
1436     {
1437         "action_key": "fill",
1438         "action_kwargs": {
1439             "value": "John Doe"
1440         },
1441         "target_element_id": 13
1442     }
1443
1444
1445     """ Example Fill Action (Range Slider)
1446
1447 Suppose you want to set '[id: 71] "$250 (5)" (range slider min: 0 max: 50 step: 1)' to the value of '$1000'. The slider has a range of 0 to 50 with a step of 1, and the value is currently set to '5'. You must translate the desired '$1000' to the correct underlying value of '20':
1448
1449
1450
1451     ```json
1452     {
1453         "action_key": "fill",
1454         "action_kwargs": {
1455             "value": "20"
1456         },
1457         "target_element_id": 71
1458     }

```



```

1512     ```
1513
1514     ### Goto Action Definition
1515
1516     - 'goto': Navigate to a new page ('target_element_id' must be null
1517     ↪).
1518     - 'url': The URL of the page to navigate to.
1519
1520     ### Example Goto Action
1521
1522     Suppose you want to open the DuckDuckGo search engine:
1523
1524     ```json
1525     {
1526         "action_key": "goto",
1527         "action_kwargs": {
1528             "url": "https://www.duckduckgo.com"
1529         },
1530         "target_element_id": null
1531     }
1532     ```

1533     ### Stop Action Definition
1534
1535     - 'stop': Stop when the task is complete, and report your progress
1536     ↪.
1537     - 'answer': Optional answer sent back to me.
1538
1539     ### Example Stop Action
1540
1541     Suppose the task is complete, and you want to stop and report your
1542     ↪ progress:
1543
1544     ```json
1545     {
1546         "action_key": "stop",
1547         "action_kwargs": {
1548             "answer": "The desired task is now complete."
1549         },
1550         "target_element_id": null
1551     }
1552     ```

1553     ## Formatting Your Response
1554
1555     Write a 200 word revised plan based on new information we
1556     ↪ discovered, and progress we made in previous steps. After
1557     ↪ your response, provide the next action as JSON in a fenced
1558     ↪ code block.

```

1558 We also provide the system prompt used by the judge. The system prompt instructs the judge to  
1559 predict JSON within a fenced code block that contains a “success” key, an “efficiency” key, and a  
1560 “self\_correction” key. The success key represents a score from 0 to 1 that estimates the probability  
1561 the task is successfully completed. The efficiency key represents a score from 0 to 1 that estimates  
1562 the probability the agent has taken the most efficient path to solve the task. The self correction key  
1563 represents a score from 0 to 1 that estimates the probability that the agent has demonstrated self  
1564 corrective behaviors during its completion of the task. These behaviors include when the agent  
1565 backtracks to a more promising state, re-plans when new information is discovered relevant to the  
task, and recognizes its own mistakes. These are generally behaviors we expect from successful

```

1566 agents, but for this paper we only filter by the success key to select training data for agents.
1567
1568
1569 You are helping me evaluate a browser automation script. I will
1570   ↪ share a task provided to the script, and a sequence of
1571   ↪ webpages and actions produced by the script.
1572
1573 ## The Action Format
1574
1575 The script produces actions as JSON in a fenced code block:
1576
1577 ````json
1578 {
1579   "action_key": str,
1580   "action_kwargs": dict,
1581   "target_element_id": int
1582 }
1583
1584 Actions have the following components:
1585
1586 - `action_key`: The name of the selected action.
1587 - `action_kwargs`: A dictionary of arguments for the action.
1588 - `target_element_id`: An optional id for the element to call the
1589   ↪ action on.
1590
1591 ## Action Definitions
1592
1593 I've prepared an API documentation below that defines the actions
1594   ↪ the script can use to complete the task.
1595
1596 ### Click Action Definition
1597
1598 - `click`: Click on an element specified by `target_element_id`.
1599
1600 ### Example Click Action
1601
1602 Here is an example where the script clicks '[id: 5] Sales link':
1603
1604 ````json
1605 {
1606   "action_key": "click",
1607   "action_kwargs": {},
1608   "target_element_id": 5
1609 }
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
2999

```

```

1620     "action_kwargs": {},
1621     "target_element_id": 2
1622 }
```
1624
1625 ### Scroll Action Definition
1626
1627 - 'scroll': Scroll the page by 'delta_x' pixels to the right and ` ↴ delta_y` pixels down.
1628   - 'delta_x': The number of pixels to scroll to the right.
1629   - 'delta_y': The number of pixels to scroll down.
1630
1631 ### Example Scroll Action
1632
1633 Here is an example where the script scrolls down the page by 300 ↴ pixels:
1634
1635
1636 ````json
1637 {
1638   "action_key": "scroll",
1639   "action_kwargs": {
1640     "delta_x": 0,
1641     "delta_y": 300
1642   },
1643   "target_element_id": null
1644 }
```
1645
1646 ### Fill Action Definition
1647
1648 - 'fill': Fill an input element specified by 'target_element_id' ↴ with text.
1649   - 'value': The text value to fill into the element.
1650
1651 ### Example Fill Action (Text Input)
1652
1653 Here is an example where the script fills '[id: 13] "Name..."' ( ↴ Enter your name text input) with the text 'John Doe':
1654
1655
1656 ````json
1657 {
1658   "action_key": "fill",
1659   "action_kwargs": {
1660     "value": "John Doe"
1661   },
1662   "target_element_id": 13
1663 }
```
1664
1665 ### Example Fill Action (Range Slider)
1666
1667 Here is an example where the script sets '[id: 71] "$250 (5)"' ( ↴ range slider min: 0 max: 50 step: 1) to the value of '$1000 ↴ '. This slider has a range of 0 to 50 with a step of 1, and ↴ the value is currently set to '5'. The script translates the ↴ desired '$1000' to the correct underlying value of '20':
1668
1669
1670
1671
1672
1673 ````json
```

```

```

1674     "action_key": "fill",
1675     "action_kwargs": {
1676         "value": "20"
1677     },
1678     "target_element_id": 71
1679 }
```
1680
1681
1682 ### Select Action Definition
1683
1684 - 'select': Select from a dropdown element specified by `→ target_element_id`.
1685   - 'label': The option name to select in the element.
1686
1687 ### Example Select Action
1688
1689 Here is an example where the script selects the option 'red' from
1690   → '[id: 67] "blue" (color select from: red, blue, green)':
1691
1692 ```.json
1693 {
1694     "action_key": "select_option",
1695     "action_kwargs": {
1696         "label": "red"
1697     },
1698     "target_element_id": 67
1699 }
```
1700
1701 ### Set Checked Action Definition
1702
1703 - 'set_checked': Check or uncheck a checkbox specified by `→ target_element_id`.
1704   - 'checked': Boolean value to check or uncheck the checkbox.
1705
1706 ### Example Set Checked Action
1707
1708 Here is an example where the script checks '[id: 21] "I agree to
1709   → the terms and conditions" (checkbox)':
1710
1711 ```.json
1712 {
1713     "action_key": "set_checked",
1714     "action_kwargs": {
1715         "checked": true
1716     },
1717     "target_element_id": 21
1718 }
```
1719
1720 ### Go Back Action Definition
1721
1722 - 'go_back': Go back to the previous page ('target_element_id' → must be null).
1723
1724 ### Example Go Back Action
1725
1726 Here is an example where the script goes back to the previous page
1727   → :

```

```

1728
1729     ```json
1730     {
1731         "action_key": "go_back",
1732         "action_kwargs": {},
1733         "target_element_id": null
1734     }
1735     ```
1736
1737     ### Goto Action Definition
1738
1739     - ``goto`': Navigate to a new page (`target_element_id` must be null
1740     ↪).
1741     - `url`': The URL of the page to navigate to.
1742
1743     ### Example Goto Action
1744
1745     Here is an example where the script opens DuckDuckGo search:
1746
1747     ```json
1748     {
1749         "action_key": "goto",
1750         "action_kwargs": {
1751             "url": "https://www.duckduckgo.com"
1752         },
1753         "target_element_id": null
1754     }
1755     ```
1756
1757     ### Stop Action Definition
1758
1759     - ``stop`': Stop when the task is complete, and report the progress.
1760     - `answer`': Optional answer from the script.
1761
1762     ### Example Stop Action
1763
1764
1765     Here is an example where the script stops and reports its progress
1766     ↪:
1767
1768     ```json
1769     {
1770         "action_key": "stop",
1771         "action_kwargs": {
1772             "answer": "The desired task is now complete."
1773         },
1774         "target_element_id": null
1775     }
1776     ```
1777
1778     ## Evaluation Instructions
1779
1780     Based on the progress of the script, you are helping me determine
1781     ↪ if the desired task has been completed successfully.
1782
1783     You will provide scores as JSON in a fenced code block:
1784
1785     ```json
1786     {
1787         "success": float,

```

```

1782     "efficiency": float,
1783     "self_correction": float
1784 }
1785 ```
1786
1787 ### Score Definitions
1788
1789 - 'success': Your confidence the desired task has been completed
1790   ↪ successfully.
1791   - range: 0.0 (not possible) to 1.0 (absolutely certain).
1792
1793 - 'efficiency': Your confidence the script has taken the most
1794   ↪ efficient path to complete the task.
1795   - range: 0.0 (not possible) to 1.0 (absolutely certain).
1796
1797 - 'self_correction': Your confidence the script has demonstrated
1798   ↪ self-corrective behaviors during its completion of the task.
1799   ↪ These behaviors include backtracking to a more promising
1800   ↪ state, replanning when new information is discovered, and
1801   ↪ recognizing its own mistakes.
1802   - range: 0.0 (not possible) to 1.0 (absolutely certain).
1803
1804 Write a 300 word analysis that establishes specific criteria to
1805   ↪ rigorously evaluate whether the task was completed, followed
1806   ↪ by which criteria the script has satisfied. After your
1807   ↪ response, provide your scores as JSON in a fenced code block.
1808   ↪

```

Finally, we provide the system prompt used in the task proposer to refine the task generated by the first iteration, and to raise the difficulty. The task proposer is instructed via the system prompt to produce JSON with a “proposed\_task” key that represents the task for the agent to complete, a “steps” key that represents the steps that an agent would follow to complete the task, and a “criteria” key that represents the criteria the judge will employ to determine if the task has been completed.

```

1813
1814 You are helping me instruct a language model agent that interacts
1815   ↪ with and navigates live webpages. We instructed the agent to
1816   ↪ complete an initial task, and I will share a sequence of
1817   ↪ webpages visited by the agent during its operation.

```

```

1818 ## Your Instructions
1819

```

```

1820 Help me refine the task, steps and criteria to raise the
1821   ↪ difficulty, while balancing the agent's capacity to
1822   ↪ successfully complete the task.

```

1823 You will provide a task as JSON in a fenced code block:

```

1824
1825   ````json
1826   {
1827     "proposed_task": str,
1828     "steps": List[str],
1829     "criteria": List[str]
1830   }
1831   ```

```

1832 Tasks have the following components:

```

1833
1834 - 'proposed_task': Instruct the agent to complete a task for you
1835   ↪ as if you are a real user that wants help on the website.

```

```

1836 - 'steps': Precise steps in an efficient trajectory that completes
1837   ↪ the task.
1838 - 'criteria': Ground truth answers and criteria to determine if
1839   ↪ the agent completes the task.
1840
1841 Tasks must adhere to the following guidelines:
1842
1843 - Must not require logging in, or making an account.
1844 - Must not require making a purchase, booking, or placing an order
1845   ↪ .
1846 - Must not require creating, deleting, or modifying any posts,
1847   ↪ articles, or webpages.
1848
1849 ## Example Tasks
1850
1851 I've prepared some examples to inspire your task design.
1852
1853 ### 'liveevents.iadb.org'
1854
1855 In this example, we explored 'liveevents.iadb.org' and saw an
1856   ↪ event page for the IDB Annual Meetings, which includes a
1857   ↪ list of the official hotels and instructions for official
1858   ↪ delegations.
1859
1860 ````json
1861 {
1862   "proposed_task": "I'm attending the IDB Annual Meetings and
1863     ↪ need to find accommodation. Please provide the address
1864     ↪ and phone number for the 'Pullman Santiago Vitacura' and
1865     ↪ 'Double Tree by Hilton' hotels. Additionally, what
1866     ↪ specific details do official delegations need to provide
1867     ↪ to access their special hotel block?",  

1868   "steps": [
1869     "Navigate to 'https://liveevents.iadb.org'.",
1870     "Click on the 'Hotels' link in the navigation menu.",
1871     "Locate 'Pullman Santiago Vitacura' in the 'OFFICIAL HOTELS
1872       ↪ FOR THE ANNUAL MEETINGS' list and extract its address
1873       ↪ and telephone number.",
1874     "Locate 'Double Tree by Hilton' in the same list and extract
1875       ↪ its address and telephone number.",
1876     "Read the instructions under 'HOTELS FOR OFFICIAL
1877       ↪ DELEGATIONS' to identify the required information for
1878       ↪ accessing the special hotel block.",
1879     "State the addresses and telephone numbers for both hotels
1880       ↪ and the required information for official delegations
1881       ↪ ."
1882   ],
1883   "criteria": [
1884     "The agent successfully navigates to the 'Hotels' page on '
1885       ↪ liveevents.iadb.org'.",
1886     "The address for Pullman Santiago Vitacura is stated as '
1887       ↪ Avenida Vitacura 3201 Vitacura, 7630578 Santiago,
1888       ↪ Chile'.",
1889     "The telephone number for Pullman Santiago Vitacura is
1890       ↪ stated as '+56 2 2944 7800'.",
1891     "The address for Double Tree by Hilton is stated as 'Avenida
1892       ↪ Vitacura 2727, Las Condes Santiago, Chile'.",
1893     "The telephone number for Double Tree by Hilton is stated as
1894       ↪ '+56 2 2587 7000'.",

```

```

1890
1891     "The agent states that official delegations need to include
1892         ↳ 'the name of your country' and 'the code (included in
1893         ↳ the invitation letters to the Governors)' to access
1894         ↳ the special hotel block."
1895     ]
1896 }
1897 ```
1898
1899     #### 'boldtcastle.com'
1900
1901     In this example, we explored 'boldtcastle.com' and saw a page with
1902         ↳ information about visiting Boldt Castle, including
1903         ↳ operating dates, admission prices, and how to get to Heart
1904         ↳ Island.
1905
1906     ````json
1907     {
1908         "proposed_task": "Help me plan a visit to Boldt Castle in 2025
1909             ↳ with one adult and one 6-year-old. Please provide the
1910             ↳ operating dates and hours for the 2025 season, the
1911             ↳ admission cost for just the castle for both of us, how to
1912             ↳ get to Heart Island, and the best phone number for
1913             ↳ general inquiries.",
1914         "steps": [
1915             "Navigate to 'boldtcastle.com'.",
1916             "Click on the 'Visiting' link.",
1917             "Click on the 'Plan Your Visit' link.",
1918             "Identify the 2025 season operating dates and hours for
1919                 ↳ Boldt Castle.",
1920             "Find the Boldt Castle-only admission price for an adult
1921                 ↳ (13+ years).",
1922             "Find the Boldt Castle-only admission price for a child aged
1923                 ↳ 6 (5-12 years).",
1924             "Locate information on how to get to Boldt Castle on Heart
1925                 ↳ Island.",
1926             "Find the general inquiry phone number for Boldt Castle.",
1927             "Synthesize all collected information into a concise answer
1928                 ↳ ."
1929         ],
1930         "criteria": [
1931             "State the 2025 season operating dates and hours for Boldt
1932                 ↳ Castle as May 10 - October 13, 10:30 AM - 6:30 PM.",
1933             "State the adult admission price for Boldt Castle only as
1934                 ↳ $13.50.",
1935             "State the admission price for a child aged 6 for Boldt
1936                 ↳ Castle only as $9.50.",
1937             "Provide the physical location of Boldt Castle (Heart Island
1938                 ↳ , Alexandria Bay, New York) and mention it's only
1939                 ↳ accessible by water.",
1940             "Provide the general inquiry phone number for Boldt Castle
1941                 ↳ as 315-482-9724."
1942         ]
1943     }
1944 ```
1945
1946     #### 'visitwestchesterny.com'
1947
1948     In this example, we explored 'visitwestchesterny.com' and saw a
1949         ↳ page that lists various coffee houses in Westchester County,

```

```

1944     ↳ including their names, addresses, and links to their
1945     ↳ descriptions.
1946
1947     ```json
1948     {
1949         "proposed_task": "Find a cozy coffee shop in Westchester County
1950         ↳ . Navigate to the 'Coffee Houses' section on the Visit
1951         ↳ Westchester NY website. Find a coffee shop described as '
1952         ↳ cozy' and provide its name, full address, and the exact
1953         ↳ sentence from its description that indicates it is cozy
1954         ↳ .",
1955         "steps": [
1956             "Navigate to 'visitwestchesterny.com'",
1957             "Click on 'Things to Do'",
1958             "Click on 'Food and Drink'",
1959             "Click on 'Coffee Houses'",
1960             "Scroll down to view the coffee shop listings.",
1961             "Identify 'Altamira Cafe Bar' (or any other coffee shop)
1962             ↳ described as 'cozy'.",
1963             "Extract the name and address of the identified coffee shop
1964             ↳ .",
1965             "Click the 'Details' link for the identified coffee shop.",
1966             "Identify and extract the exact sentence from the
1967             ↳ description on its dedicated page that indicates it is
1968             ↳ cozy."
1969         ],
1970         "criteria": [
1971             "Successfully navigate to the 'Coffee Houses' page.",
1972             "Identify a coffee shop described as 'cozy' (e.g., 'Altamira
1973             ↳ Cafe Bar').",
1974             "State the name of the identified coffee shop (e.g., '
1975             ↳ Altamira Cafe Bar').",
1976             "State the full address of the identified coffee shop (e.g.,
1977             ↳ '245 Main St., New Rochelle, NY 10801').",
1978             "Successfully navigate to the 'Details' page for the
1979             ↳ identified coffee shop.",
1980             "Correctly state the exact sentence from the description
1981             ↳ that indicates it is cozy (e.g., 'Relax in the cozy
1982             ↳ shop or take a treat to go with piping hot espresso, a
1983             ↳ cold coffee, delicious desserts and delightful
1984             ↳ sandwiches.'."
1985         ]
1986     }
1987     ```
1988
1989     ### `odetterestaurant.com`
1990
1991     In this example, we explored 'odetterestaurant.com' and saw a '
1992     ↳ Reservations' page, which lists policies for dietary
1993     ↳ accommodations, birthdays, a deposit requirement,
1994     ↳ cancellations, and rescheduling.
1995
1996     ```json
1997     {
1998         "proposed_task": "I want to make a dinner reservation for 4
1999         ↳ people at Odette, and one of my guests has a severe dairy
2000         ↳ allergy. I also want to request a birthday cake for the
2001         ↳ table. What are the key policies I need to be aware of

```

```

1998     ↳ regarding my guest's allergy, the cake request, and any
1999     ↳ deposit or cancellation rules for this reservation?", 
2000   "steps": [
2001     "Navigate to 'odetterestaurant.com'", 
2002     "Go to the 'Reservations' page", 
2003     "Identify the policy regarding dairy allergies and other
2004     ↳ dietary accommodations", 
2005     "Find the policy for requesting a birthday cake, including
2006     ↳ notice period and cost", 
2007     "Locate the deposit requirement per person for dinner
2008     ↳ reservations", 
2009     "Determine the cancellation or rescheduling policy and
2010     ↳ associated timeframe", 
2011     "Synthesize all relevant policies into a concise answer"
2012   ],
2013   "criteria": [
2014     "State that Odette is unable to accommodate guests with
2015     ↳ dairy allergies or intolerance.", 
2016     "State that cakes require a 72-hour notice and cost SGD78
2017     ↳ ++.", 
2018     "Confirm a deposit of SGD200 per person is required for
2019     ↳ dinner reservations.", 
2020     "State that all reservations are final and non-refundable,
2021     ↳ but changes can be made at least 72 hours prior to the
2022     ↳ reservation date."
2023   ]
2024   ```` 
2025   ### 'dottyabouticecream.co.uk'
2026 
2027 In this example, we explored 'dottyabouticecream.co.uk' and saw a
2028     ↳ form for hiring Dotty's ice cream van for corporate events,
2029     ↳ which includes fields for event details, guest count, and
2030     ↳ flavor inquiries.
2031 
2032 ````json
2033 {
2034   "proposed_task": "Inquire about hiring Dotty's ice cream van
2035     ↳ for a corporate event in Manchester, M1 1AE, on August 15
2036     ↳ th, 2024, from 2 PM to 4 PM, for 100 guests. Ask if
2037     ↳ vanilla, honeycomb crunch, and mango sorbet are available.
2038     ↳ Fill out the 'Get in Touch' form with your details (Jane
2039     ↳ Doe, jane.doe@example.com, 07123456789) and note you
2040     ↳ found them via a web search. Do not submit the form.", 
2041   "steps": [
2042     "Navigate to the 'Get in Touch' page on dottyabouticecream.
2043     ↳ co.uk.", 
2044     "Fill 'Jane Doe' into the 'Name' field.", 
2045     "Fill 'jane.doe@example.com' into the 'Email' field.", 
2046     "Fill '07123456789' into the 'Telephone Number' field.", 
2047     "Fill 'August 15th, 2024' into the 'Event Date' field.", 
2048     "Fill '2 PM - 4 PM' into the 'Time of Ice Cream Service'
2049     ↳ field.", 
2050     "Fill 'Manchester, M1 1AE' into the 'Venue Address (incl.
2051     ↳ Postcode)' field.", 
2052     "Fill '100' into the 'Number of Expected Guests' field.", 
2053     "Fill 'Web Search' into the 'Where Did You Hear About Dotty
2054     ↳ ?' field."
2055   ]

```

```

2052     "Fill the 'Message' field with an inquiry about the
2053         ↪ availability of 'Vanilla, Honeycomb Crunch, and Mango
2054         ↪ Sorbet' flavors for a corporate event.",
2055     "Confirm all specified fields are accurately filled, but do
2056         ↪ not click the 'Gimmie Ice Cream' submit button."
2057 ],
2058 "criteria": [
2059     "The agent successfully navigates to the 'Get in Touch' page
2060         ↪ .",
2061     "The 'Name' field is filled with 'Jane Doe'.",
2062     "The 'Email' field is filled with 'jane.doe@example.com'.",
2063     "The 'Telephone Number' field is filled with
2064         ↪ '07123456789'.",
2065     "The 'Event Date' field is filled with 'August 15th,
2066         ↪ 2024'.",
2067     "The 'Time of Ice Cream Service' field is filled with '2 PM
2068         ↪ - 4 PM'.",
2069     "The 'Venue Address (incl. Postcode)' field is filled with '
2070         ↪ Manchester, M1 1AE'.",
2071     "The 'Number of Expected Guests' field is filled with
2072         ↪ '100'.",
2073     "The 'Where Did You Hear About Dotty?' field is filled with
2074         ↪ 'Web Search'.",
2075     "The 'Message' field clearly inquires about the availability
2076         ↪ of 'Vanilla, Honeycomb Crunch, and Mango Sorbet'
2077         ↪ flavors for a corporate event.".
2078     "The agent does not submit the form by clicking the 'Gimmie
2079         ↪ Ice Cream' button."
2080 ]
2081 }
2082 ````

2083     ### `engineered.polestar.com`  

2084
2085     In this example, we explored 'engineered.polestar.com' and saw a
2086         ↪ page with information about Polestar Engineered Optimization
2087         ↪ for various Volvo models, including the 2023 Volvo XC60
2088         ↪ with a B5 Drive-E engine.
2089
2090     ````json
2091     {
2092         "proposed_task": "Is a Polestar Engineered Optimization
2093             ↪ available for a 2023 Volvo XC60 with a B5 Drive-E engine?
2094             ↪ If so, what are the primary performance benefits, how
2095             ↪ long does the installation take, and how would I find a
2096             ↪ dealer for this service?",  

2097         "steps": [
2098             "Navigate to engineered.polestar.com.",  

2099             "Under 'Can My Volvo Be Optimised?', select 'XC' then 'New
2100                 ↪ XC60' for the model.",  

2101             "Locate and select 'XC60 B5 Drive-E AWD Automatic 2023' or '
2102                 ↪ XC60 B5 Drive-E FWD Automatic 2023' to view its
2103                 ↪ optimization details.",  

2104             "Confirm if the vehicle is 'Approved for Polestar Engineered
2105                 ↪ Optimization'.",
2106             "Identify the primary performance benefits listed for the
2107                 ↪ optimization.",  

2108             "Determine the approximate installation time.",
2109     }

```

```

2106
2107     "Click on any 'Find a retailer' or 'Contact a dealer' links
2108         ↪ to see where they lead.",
2109     "Based on the website's information, describe how a user
2110         ↪ would find a dealer for installation.",
2111     "Synthesize all gathered information to answer the task."
2112 ],
2113 "criteria": [
2114     "Confirm that a 2023 Volvo XC60 with a B5 Drive-E engine is
2115         ↪ 'Approved for Polestar Engineered Optimization'.",
2116     "State the primary performance benefit as 'Power Mid-Range
2117         ↪ up to (hp) +3%' (from the specific product page) or '
2118         ↪ Up to +15% increased mid-range power' (from the
2119         ↪ general 'Get optimisation' page).",
2120     "State that the installation takes 'less than 60 minutes'.",
2121     "Clearly state that clicking the 'Find a retailer' or '
2122         ↪ Contact a dealer' links does not lead to a functional
2123         ↪ dealer search tool, and that users are advised to
2124         ↪ contact their local Volvo retailer directly for
2125         ↪ further questions."
2126 ]
2127
2128 `ajga.org`
```

In this example, we explored 'ajga.org' and saw a page with  
 ↪ information about Performance Based Entry (PBE) Stars for  
 ↪ junior golfers, including how they carry over to the next  
 ↪ season and tips for maximizing tournament opportunities.

```

2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048
4049
4050
4051
4052
4053
4054
4055
4056
4057
4058
4059
4060
4061
4062
4063
4064
4065
4066
4067
4068
4069
4070
4071
4072
4073
4074
4075
4076
4077
4078
4079
4080
4081
4082
4083
4084
4085
4086
4087
4088
4089
4090
4091
4092
4093
4094
4095
4096
4097
4098
4099
4100
4101
4102
4103
4104
4105
4106
4107
4108
4109
4110
4111
4112
4113
4114
4115
4116
4117
4118
4119
4120
4121
4122
4123
4124
4125
4126
4127
4128
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4150
4151
4152
4153
4154
4155
4156
4157
4158
4159
4160
4161
4162
4163
4164
4165
4166
4167
4168
4169
4170
4171
4172
4173
4174
4175
4176
4177
4178
4179
4180
4181
4182
4183
4184
4185
4186
4187
4188
4189
4190
4191
4192
4193
4194
4195
4196
4197
4198
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219
4220
4221
4222
4223
4224
4225
4226
4227
4228
4229
4230
4231
4232
4233
4234
4235
4236
4237
4238
4239
4240
4241
4242
4243
4244
4245
4246
4247
4248
4249
4250
4251
4252
4253
4254
4255
4256
4257
4258
4259
4260
4261
4262
4263
4264
4265
4266
4267
4268
4269
4270
4271
4272
4273
4274
4275
4276
4277
4278
4279
4280
4281
4282
4283
4284
4285
4286
4287
4288
4289
4290
4291
4292
4293
4294
4295
4296
4297
4298
4299
4300
4301
4302
4303
4304
4305
4306
4307
4308
4309
43
```

```

2160     "Identify and state that 'Plan your tournament schedule
2161         ↳ early to maximize playing opportunities and prevent
2162         ↳ missed deadlines.' is a key recommendation.",
2163     "Identify and state that 'Qualifiers are great opportunities
2164         ↳ for all players to earn Performance Stars and build
2165         ↳ their status.' is a second key recommendation."
2166   ]
2167 }
2168 ````

2169 ### `passports.gov.au`  

2170  

2171 In this example, we explored 'passports.gov.au' and saw a page
2172     ↳ with a section about the documents needed to prove
2173     ↳ Australian citizenship for individuals born in Australia on
2174     ↳ or after August 20, 1986.
2175  

2176 ````json
2177 {
2178     "proposed_task": "I was born in Australia on or after August
2179         ↳ 20, 1986, and am applying for my first Australian
2180         ↳ passport. What documents do I need to prove my Australian
2181         ↳ citizenship? Please list the primary document,
2182         ↳ acceptable alternatives, and any specific requirements
2183         ↳ for proving citizenship by birth based on my parents' or
2184         ↳ grandparents' status.",
2185     "steps": [
2186         "Navigate to 'passports.gov.au'.",
2187         "Navigate to the 'How it works' section.",
2188         "From 'How it works', navigate to 'Documents you need'.",
2189         "On the 'Documents you need' page, navigate to the '
2190             ↳ Citizenship' section.",
2191         "Within the 'Citizenship' section, locate the information
2192             ↳ for individuals 'Born in Australia on or after 20
2193             ↳ August 1986'.",
2194         "Identify the primary document required for proof of
2195             ↳ citizenship.",
2196         "Identify and list all acceptable alternative documents.",
2197         "Detail the specific scenarios for proving citizenship by
2198             ↳ birth, including those involving parents' or
2199             ↳ grandparents' documentation and the special case for
2200             ↳ permanent resident parents."
2201     ],
2202     "criteria": [
2203         "State that the primary document is the applicant's full
2204             ↳ Australian birth certificate.",
2205         "List an Australian citizenship certificate in the applicant
2206             ↳ 's name as an acceptable alternative.",
2207         "List an Australian passport issued in the applicant's name
2208             ↳ on or after 1 January 2000 that was valid for at least
2209             ↳ two years as an acceptable alternative.",
2210         "Detail the scenario where one parent was an Australian
2211             ↳ permanent resident or citizen, specifying the required
2212             ↳ parental documents (birth certificate, passport, or
2213             ↳ citizenship certificate).",
2214         "Explicitly mention that if both parents were Australian
2215             ↳ permanent residents when the applicant was born,
2216             ↳ evidence of citizenship must be obtained from the
2217             ↳ Department of Home Affairs.",
2218     ]
2219 }
2220 ````
```

```
2214     "Include the scenario involving grandparents' documents ( birth certificate, passport, or citizenship  
2215         certificate) if the parent was born in Australia on or  
2216         after 20 August 1986."  
2217     ]  
2218 }  
2219 ...  
2220  
2221 ## Formatting Your Response  
2222  
2223 Establish how the task can be refined in at most 300 words, and  
2224     synthesize relevant content and features on the website in  
2225     your response. After your response, provide a refined task  
2226     as JSON in a fenced code block.
```

## H DETAILS FOR TRAINING AGENTS

To understand the utility of data we obtained, we train agents and test on four relevant benchmarks: InSTA, WebVoyager (He et al., 2024), Mind2Web (Deng et al., 2023), WebLINX (Lù et al., 2024). In particular, our test set consists of a held-out set of 3,000 websites and tasks produced by the task generation feedback loop. Note these websites are not present in the training set. For WebVoyager (He et al., 2024), we transfer agents trained on our data zero-shot to 643 tasks on 15 websites WebVoyager (He et al., 2024). The websites in the WebVoyager benchmark are not present in the 20k trajectories we collected in Section 6.1 we used for training agents. For this experiment, we fine-tuned models based on *Qwen 3 1.7B* with a maximum sequence length of 16,384 tokens, and the most recent 5 observations, and actions in the context. We employed full fine-tuning on this model, with Adam, learning rate of  $5e-5$ , batch size of 32, `bfloat16`, and other parameters kept as the PyTorch defaults for Adam. Each model was trained using one epoch, a linear warm-up corresponding to the first 0.01 steps of training, and a linear decay to  $6e-5$  afterwards.

To filter data, we select trajectories that were scores as Judge (Success) = 1, which corresponded to 10.5k of 20k trajectories produced by the *Qwen 3 235B* data collection policy. Scores for filtering were produced by a *Qwen 3 235B* judge. We employed a simple filtering strategy that only considers the success score from the judge, and no other filtering conditions were used. Note the judge also predicts efficiency, and self correction scores, which could likely also help select the best data for training, but we did not explore filtering by these scores in this work.

For experiments on static benchmarks, we fine-tune `google/flan-t5-large` for Mind2Web, and `meta-llama/Llama-3.1-8B-Instruct` for WebLINX using official fine-tuning code released with corresponding benchmarks. We employ identical training hyperparameters to those used by Lù et al. (2024) for Llama in their official training code and Deng et al. (2023) for Flan to ensure that our results are comparable to previous work. Section 6.2 reports performance on the official `test_web` split of the WebLINX benchmark, and the official `test_website` split of the Mind2Web benchmark, where agents are tested on unobserved websites. The websites in these static benchmarks were not present in the dataset we generated for this experiment to ensure fairness.

## I HYPERPARAMETERS

We provide a list of the hyperparameters used in this work in Table 3. Hyperparameters for mixing our data with human data on static benchmarks are selected to mirror prior work in synthetic data (Trabucco et al., 2024), and to adhere to standard hyperparameters for WebLINX (Lù et al., 2024), and Mind2Web (Deng et al., 2023). We train using all available human data on these benchmarks, and add trajectories filtered using the previously discussed methodology, sampled at a 20% rate in the data-loader compared to an 80% rate for human data.

| Hyperparameter Name                               | Value                                                                                                                                                                                                                                       |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Models Used For Agents                            | Qwen/Qwen3-1.7B<br>Qwen/Qwen3-235B-A22B<br>meta-llama/Llama-4-Maverick-17B-128E-Instruct<br>meta-llama/Llama-3.1-70B-Instruct<br>meta-llama/Llama-3.3-70B-Instruct<br>google/gemini-2.5-flash                                               |
| Models Used For Judges                            | Qwen/Qwen3-235B-A22B<br>meta-llama/Llama-4-Maverick-17B-128E-Instruct<br>meta-llama/Llama-3.1-70B-Instruct<br>meta-llama/Llama-3.3-70B-Instruct<br>google/gemini-2.5-flash<br>google/gemini-1.5-pro<br>openai/gpt-4.1-mini<br>openai/gpt-4o |
| Common Crawl PageRank                             | cc-main-2024-apr-may-jun-host-ranks.txt.gz                                                                                                                                                                                                  |
| Number of sites before filtering                  | 1,000,000                                                                                                                                                                                                                                   |
| Number of tasks after filtering                   | 146,746                                                                                                                                                                                                                                     |
| Max Tokens Per Observation                        | 2,048                                                                                                                                                                                                                                       |
| Max Tokens Per Agent Trace                        | 1,024                                                                                                                                                                                                                                       |
| Max Tokens Per Judge Trace                        | 1,024                                                                                                                                                                                                                                       |
| Max Tokens Per Task Proposer Trace                | 1,024                                                                                                                                                                                                                                       |
| Last Steps Per Agent Context                      | 5                                                                                                                                                                                                                                           |
| Last Steps Per Judge Context                      | 5                                                                                                                                                                                                                                           |
| Last Steps Per Task Proposer Context              | 5                                                                                                                                                                                                                                           |
| Task Proposer Feedback Loops                      | 1                                                                                                                                                                                                                                           |
| LLM Sampling Temperature                          | 0.5                                                                                                                                                                                                                                         |
| LLM Sampling Top P                                | 1.0                                                                                                                                                                                                                                         |
| LLM Sampling Top K                                | default                                                                                                                                                                                                                                     |
| Fine-tuned LLM in Section 6.1                     | Qwen/Qwen3-1.7B                                                                                                                                                                                                                             |
| InSTA Training Epochs                             | 1                                                                                                                                                                                                                                           |
| InSTA Batch Size                                  | 32                                                                                                                                                                                                                                          |
| InSTA Learning Rate                               | 5e-5                                                                                                                                                                                                                                        |
| InSTA Optimizer                                   | Adam                                                                                                                                                                                                                                        |
| Mind2Web LLM                                      | google/flan-t5-large                                                                                                                                                                                                                        |
| Mind2Web Training Iterations                      | 11,505                                                                                                                                                                                                                                      |
| Mind2Web Batch Size                               | 32                                                                                                                                                                                                                                          |
| Mind2Web Learning Rate                            | 5e-5                                                                                                                                                                                                                                        |
| Mind2Web Optimizer                                | Adam                                                                                                                                                                                                                                        |
| WebLNX LLM                                        | meta-llama/Llama-3.1-8B-Instruct                                                                                                                                                                                                            |
| WebLNX Training Iterations                        | 10,000                                                                                                                                                                                                                                      |
| WebLNX Batch Size                                 | 16                                                                                                                                                                                                                                          |
| WebLNX Learning Rate                              | 5e-5                                                                                                                                                                                                                                        |
| WebLNX Optimizer                                  | Adam                                                                                                                                                                                                                                        |
| Data Filtering Condition                          | Judge (Success) = 1                                                                                                                                                                                                                         |
| Human Data Sampling Probability $p_{\text{real}}$ | 80%                                                                                                                                                                                                                                         |

Table 3: **Hyperparameters used in our paper.** We organize hyperparameters into seven sections, for the names of LLMs used as agents in the paper, the names of LLMs used as judges in the paper, the hyperparameters used for data collection, the sampling parameters for LLMs, the training parameters for static benchmarks, and the filtering and data mixing hyperparameters.

## J COST ANALYSIS FOR LLAMA 3.1 70B

To understand the significant reduction in cost that we obtain by running LLMs locally to generate data, we analyze the number of tokens processed by the LLM, and compute an expected cost if this were served using proprietary models. As the analysis shows, using *LLama 3.1 70B* is a feasible

option for running agents at this large scale, and results in the paper show that this choice of LLM backbone does not compromise performance. We have deep gratitude to the Llama team at Meta, and the Qwen team at Alibaba for working to make developments in language modeling available to the research community at no cost. We see up to a 95% reduction in cost with these models.

| Variable Name                                                                                             | Value            |
|-----------------------------------------------------------------------------------------------------------|------------------|
| Number of tasks                                                                                           | 146,746          |
| Average tokens per observation                                                                            | 1,024            |
| Max observations per agent context window                                                                 | 5                |
| Average agent / judge response size                                                                       | 512              |
| Max tokens per system prompt                                                                              | 1,024            |
| Average steps per task                                                                                    | 15               |
| Estimated tokens processed by the agent                                                                   | 14.65B tokens    |
| Tokens processed by the judge                                                                             | 1.35B tokens     |
| Total tokens processed                                                                                    | 16.00B tokens    |
| Expected API cost for <i>GPT-4.1</i>                                                                      | \$ 32,000.00     |
| Expected API cost for <i>Gemini 2.5 Pro</i>                                                               | \$ 20,000.00     |
| Expected AWS compute cost for serving <i>Llama 3.1 70B</i><br>(3,840 v100 GPU hours using spot instances) | \$ 1,575.70      |
| Percent saved using <i>Llama 3.1 70B</i>                                                                  | [95.08, 92.12] % |

Table 4: **Cost analysis for different LLM models in the fully-scaled pipeline.** This table provides statistics for the number of tokens that were processed by our pipeline, and why serving using a local LLM engine like vLLM is important for bringing down costs.