# Autonomous Evaluation and Refinement of Web Agents

Jiayi Pan[1]* Yichi Zhang[2] Nicholas Tomlin[1] Yifei Zhou[1] Sergey Levine[1] Alane Suhr[1]
[1]UC Berkeley [2]University of Michigan

## Abstract

*We show that domain-general automatic evaluators can significantly improve the performance of agents for web navigation and device control. We experiment with multiple evaluation models that trade off between inference cost, modularity of design, and accuracy. We validate the performance of these models in several popular benchmarks for digital agents, finding between 74.4 and 92.9% agreement with oracle evaluation metrics. Finally, we use these evaluators to improve the performance of existing agents via fine-tuning and inference-time guidance. Without any additional supervision, we improve state-of-the-art performance by 29% on the popular benchmark WebArena, and achieve a 75% relative improvement in a challenging domain transfer scenario. We release our code and data at* https://github.com/Berkeley-NLP/Agent-Eval-Refine.

## 1. Introduction

Given an instruction, e.g., *"Tell me the cost of my latest canceled order,"* an automated digital agent would be expected to first navigate to a user's profile page, then to a list of their previous orders, identify the most recent order that has been canceled, and return its total amount to the user. Such agents offer the long-term potential of making digital devices more accessible, while also simplifying tedious or mundane tasks. However, in the short term, even state-of-the-art agents still make mistakes on simple tasks. Evaluating such agents and characterizing their failure modes is not only important for understanding and improving the models, but also critical for safely deploying them in real world. In this paper, we demonstrate the opportunities and efficacy of using automated evaluation models to both characterize and improve agent performance, without requiring access to any extra supervision, such as expert demonstrations or evaluation functions.

We propose to automatically evaluate user instructions and arbitrary agent trajectories with domain-general neural models. We explore two main variants of this approach

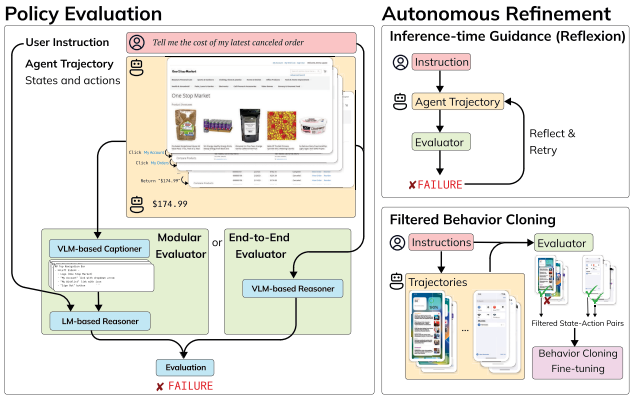


Figure 1. Method overview: A model-based evaluator provides evaluation of a digital agent's trajectory (left). It can be used as the reward function for Reflexion [29] or filtered behavior cloning to enhance model performance (right).

(Figure 1, left): first, a modular caption-then-reason approach where a vision-language model (VLM) first captions the screenshots, and a language model (LM) is used to reason about if an agent succeeds based on textual information; and second, an end-to-end approach where we prompt an advanced VLM like GPT-4V [2] to directly evaluate a trajectory. These two different approaches offer trade-offs in performance, cost, and transparency.

We first evaluate our proposed approach on its ability to match oracle evaluation metrics using WebArena [44] and Android-in-the-Wild [AitW; 27], achieving accuracies up to 82.1 and 92.9% respectively. We then show how these evaluation models can be used to refine existing agents through inference-time guidance or during training, without access to any hand-designed evaluation functions or additional demonstration data (Figure 1, right). When integrated as the reward function in Reflexion [29], the evaluator enhances the best-performing GPT-4 WebArena agent's success rate by up to 29% of relative improvement. Additionally, we evaluate in a domain transfer setting to device control in iOS, for which there is no existing benchmark environment or training data. When using our evaluation models to filter sampled trajectories to be used in behavior cloning, we see relative improvements of 75% accuracy in

*Email: jiayipan@berkeley.edu

this domain.

## 2. Related Work

Building automated digital agents that map from user instructions to executable actions has been a long-standing goal in the NLP and AI communities [3, 6, 7]. Recent advances in NLP and multimodal machine learning have supported the development of more capable agents, and many recent benchmarks and approaches cover instruction-conditioned tasks such as web navigation and device control.

**Digital Agents** Early modeling of language-conditioned autonomous agents focused on approaches that include semantic parsing [3, 24, 36], reinforcement learning [6, 7], and imitation learning [19]. The strength of pretrained language and language-and-vision modeling has renewed interest in building language-conditioned digital agents [11, 15, 17, 31, 43, 44]. For example, baseline approaches to WebArena [44] use few-shot prompting with language-only models, representing the environment state and action space with its document object model (DOM). More recent works in building these agents have moved from language-only modeling to vision-language modeling, representing the environment state space as its rendered pixel representation instead of relying on a DOM. Another line of work has applied inference-time techniques to improve model's performance, for example with inference-time exploration [41], intermediate plan revision [42] and error correction [32], and self-critique [35] on GPT-4 or GPT-4V. Concurrent to our work, OS-Copilot [35] proposes a self-critique component to autonomously refine Mac device control agents, implementing the critic as a LM that reasons about proposed tool implementations and error messages. In contrast to our work, this critic does not evaluate actual agent behavior in the execution environment or used in model training.

**Autonomous Refinement and Evaluation** Recently, there has been renewed interest in methods for improving policies at training [1, 5, 23, 26] or inference [29, 35, 39] time without human supervision. Approaches like Reflexion [29] assume access to an external evaluation function, leveraging it as the supervision signal to guide policy improvement at inference time. In contrast, we study applications of inference-time autonomous refinement without requiring access to the external evaluation function, and show that using our proposed domain-general evaluation models improves agent success rate by 29%. Meanwhile, methods which bootstrap policies with supervised training and refine them with reinforcement learning have been widely adopted; our proposed evaluators enable this paradigm in open, realistic digital agent scenarios, providing a relative improvement in performance of over 70%. Concurrent to our work, WebVoyager [16] also explores using GPT-4V as an automated proxy for human evaluation of web agents, though is not the primary focus of their work, and neither performs in-depth analysis of the quality of its judgments, nor explores its applicability to improving agents.

**Digital Agent Benchmarks** Recently-proposed benchmarks that study digital agents fall roughly into two categories: simulation-based benchmark and demonstration-based one. Simulation-based benchmarks include environment simulators that offer the ability to execute arbitrary agent trajectories. Early simulation environments such as WoB [25, 28], WebShop [38], and others [6] are limited in their domain coverage, realism, or generalizability of their evaluation functions. Recently proposed simulation environments like AndroidEnv [30], WebArena [44] and VisualWebArena [22], though far from perfect, have offered improvement across these dimensions. However, designing simulators, curating tasks, and handcrafting evaluation functions fundamentally limits their ability to mirror task and environment diversity of real environments.

In parallel, the community has focused on demonstration-based benchmarks that do not include an executable simulation environment, including PIXEL-HELP [24], MoTIF [8], Mind2Web [11], and AitW [27]. Notably, Mind2Web and AitW contain over 2K and 715K human trajectories respectively on a wide range of web navigation and device control tasks. Though primarily used for model training [11, 17, 27, 43], these datasets are also used for evaluating digital agents through reference-based metrics like action matching score. In this setting, an agent is given the prefix of a human demonstration and evaluated on its prediction of the next action to take. However, this metric requires human demonstrations and does not directly reflect agent's performance in real-world because it does not account for consequences of an agent's sequential decision process, alternative actions that diverge from the demonstration.

We propose a third approach in which arbitrary instructions and agent trajectories are directly evaluated by a model.

## 3. Domain-General Evaluators

We develop multiple domain-general automatic evaluators for digital agents. Given a user instruction $x$ and an initial environment state $s_0$, an agent generates and executes a sequence of actions $\overline{a} = \langle a_0, a_1, \ldots, a_n \rangle$, resulting in a sequence of state visits $\overline{s} = \langle s_0, s_1, s_2, \ldots, s_{n+1} \rangle$. In this work, we assume $a$ and $x$ are in text form, such as `<Type: ''Hello''>` and *"Check the weather"*, and each state $s$ is represented as a screenshot image. Given $x$, $\overline{a}$, and $\overline{s}$ as input,the model produces a scalar evaluation

$\bar{r} = \langle r_0, r_1, \ldots, r_n \rangle$ corresponding to each step of the trajectory:

$$\bar{r} = \text{evaluate}(x, \bar{a}, \bar{s}) \ .$$

The evaluator can provide either trajectory-level or per-step evaluations. For trajectory-level evaluation, $r_0 = \cdots = r_{n-1} = 0$, with $r_n = 1$ for successful trajectories and $r_n = 0$ otherwise. For per-step evaluation, we classify each step into three types, $r_i = 1$ indicates task success after action $a_i$, $r_i = p \geq 0$ indicates progress toward the goal, and $r_i = d < 0$ is assigned to actions that do not contribute to the objective. We query the model once for trajectory-level evaluation and $n$ times for per-step evaluation, reducing the model's task into a binary or ternary classification problem at each step.

We explore two methods for constructing the model:

1. An end-to-end approach that maps directly from instructions and screenshots to an evaluation via a pre-trained VLM.
2. A modular approach which first transcribes the observed screenshots into text descriptions using a VLM, and then uses a LM to map the descriptions, actions, and user instruction onto an evaluation.

Both methods have tradeoffs: in the first, we can apply advanced VLMs like GPT-4V. However, this approach is relatively expensive and relies on API calls to proprietary models. In the second, we can compose open-weight models to achieve slightly weaker performance, but with added benefits of explainability via modularity and low-cost local deployment.

### 3.1. End-to-End Approach

We directly provide an instruction-tuned VLM with $x$, $\bar{a}$, and $\bar{s}$. We prompt it to first produce a text-based reasoning process [34], then output its evaluation result. In our experiments, we use the proprietary vision-language model GPT-4V [2].[1]

### 3.2. Modular Caption-then-Reason Approach

Many existing approaches for joint reasoning about language and vision disentangle perception and reasoning. In these approaches, a VLM is first applied to visual input to generate a language-based description; then, a text-only model (e.g., a LM) takes as input this description and the user instruction to produce a response by reasoning only about linguistic inputs. Existing work applying this approach has mostly focused on joint reasoning about natural images and text, e.g., for visual question answering [14, 33, 40]. We take a similar approach here, where we first use a VLM to produce a description of the agent's observations given as $\bar{s}$ , then feed these descriptions, along with actions $\bar{a}$ and the user's instruction $x$ to an LM to produce a final evaluation.[2]

[1]Prompt templates and additional details are provided in Appendix A.1.

[2]Data collection process, hyper-parameters, and output examples are detailed in Appendix A.2.

**Captioner** One drawback to this modular approach is the potential for information loss, where the image description may not include all the details necessary for task success [33]. In our case, this could include missing or misrepresenting details about the screenshot, and indeed, we find that current open-weight VLMs struggle to produce detailed screenshot descriptions out of the box. In contrast, the most advanced, yet proprietary, VLMs can produce very detailed descriptions with adequate prompting.

To improve a captioner's ability to provide detailed, well-formatted descriptions, we collect a dataset of screenshots paired with descriptions, and use it to fine-tune an open-weight VLM. We first acquire screenshots from a variety web and device control domains, then use GPT-4V to provide an initial detailed description for each screenshot. We manually filter out or fix apparent errors in GPT-4V's output, resulting a total of 1,263 data points.[3] We use this data to fine-tune the QWen-VL [4] model. During both fine-tuning and at inference time, we provide text recognition results from EasyOCR[4] as an additional input to the VLM to reduce hallucination.

At inference time, we use our finetuned captioner model to acquire a description for each step in the agent trajectory. Critically, we do not provide this model access to the original user instruction, as we find this exacerbates model hallucinations; e.g., describing webpage attributes which would be relevant to the task, but are not actually present in the screenshot.

**Reasoner** Finally, we provide the actions, generated descriptions, and the original user instruction to a language-only instruction-tuned model. We experiment with prompting two LMs, Mixtral [20] and GPT-4, to produce a text-based thought and reasoning process as well as the final evaluation.

## 4. Experiments and Results

Our goal is to show how domain-general evaluation models can support the autonomous evaluation and refinement of digital agents, without requiring access to human demonstrations or oracle evaluation metrics. To this end, we first evaluate how these models perform **as autonomous evaluators** by comparing their judgments to benchmark-provided metrics and human judgements (Section 4.1). We then illustrate how these evaluation models, while imperfect, can serve **as discriminators in autonomous refinement settings** through both inference-time policy refinement [29] and filtered behavior cloning [filtered BC; 9, 10, 13] to support significant improvements in agent performance (Section 4.2).

Our rationale behind experiment design is to cover a broad range of domains and challenges. We use WebArena

[3]Table 3 in Appendix A.2 contains details of data sources and sizes.

[4]https://github.com/JaidedAI/EasyOCR

for both evaluation and inference-time refinement, as its built-in evaluation functions facilitate direct comparison. Android-in-the-Wild (AitW) is chosen for evaluation since it is widely used for training and evaluating Android agents, and is typically evaluated using a reference-based metric instead of task success. Lastly, we refine a model through filtered behavior cloning on iOS, where data scarcity poses a significant challenge to supervised methods.

**Environments** WebArena [44] is an offline web emulation environment and dataset that supports execution of arbitrary policies. WebArena comprises 812 human-written task instructions across various domains, including shopping, maps, and content management systems. Each instruction is paired with a handwritten test case that verifies agent success, e.g., by checking the status of a specific webpage element against a reference. We refer to this set of test cases as WebArena's oracle evaluator.

Android-in-the-Wild [AitW; 27] is a large-scale dataset for Android device control containing 715,142 human demonstrations of 30,378 unique instructions. In our experiments, we focus on a subset of 120 tasks randomly sampled from the AitW test set.[5] Unlike WebArena, AitW does not include an emulation environment for agent execution. Instead, the suggested evaluation metric is based on action matching: given a sequence of actions representing the prefix of a human demonstration, the agent is evaluated on its ability to predict the next action in the demonstration. While we compare against this reference-based metric in our experiments, we focus on end-to-end task-level success rate and implement an Android emulator to support execution of arbitrary trajectories.[6] We refer to human judgements on trajectory success as the oracle evaluation.

Despite significant interest in developing digital agents, progress in the domain of iOS device control has been modest, with the exception of [37], who collect a small unreleased dataset of human demonstrations in this domain. We curate a set of 132 tasks in the iOS domain, taking inspiration from tasks included in AitW. We experiment with using our proposed evaluation models to facilitate domain transfer, with the goal of applying the strongest model on AitW, CogAgent [17], to iOS. We develop a Python interface to the iOS emulator on macOS, and design its action space to align with the Android-in-the-Wild to facilitate domain transfer.[6]

**Evaluation Models** We evaluate three evaluation model variants:

- GPT-4V: End-to-end approach (Section 3.1) using GPT-4V.
- Captioner + Mixtral: Modular approach (Section 3.2) using a finetuned QWen-VL [4] to generate a trajectory description, and Mixtral [20] to provide the final evaluation.
- Captioner + GPT-4: Modular approach (Section 3.2) using a finetuned QWen-VL to generate a trajectory description, and GPT-4 to provide the final evaluation.

In most experiments, the evaluation model produces a trajectory-level evaluation, and takes as input only the last frame $s_{n+1}$ in the trajectory, along with the instruction $x$ and action sequence $\bar{a}$. Preliminary experiments suggested that model performance does not improve with information about previous states, likely due to limitations of existing models in processing long contexts. In the iOS experiments, the evaluation model takes as input the entire trajectory $\bar{s}$ and $\bar{a}$ and the instruction $x$, and produces a per-step evaluation.

**Agent Policies** We experiment with evaluating and refining the current state-of-the-art digital agents. In WebArena, this is a GPT-4-based agent described by [44]. For each task, GPT-4 is provided the user's instruction and the current DOM representation of the webpage derived from its HTML accessibility tree. GPT-4 is prompted to generate an action grounded in the DOM, e.g., clicking a button with a specific element ID. This agent achieves an end-to-end task success rate of 14.4% using the oracle evaluator.

The strongest agent on the AitW benchmark is CogAgent [17], followed by Auto-UI$_{\{\text{large, base}\}}$ [43]. These agents are implemented as neural vision-language models that map observations, represented as images, and instructions to executable actions. We also experiment with the human demonstrations provided in AitW.[7]

### 4.1. Automatic Evaluation

**WebArena** For each WebArena task and corresponding trajectory sampled from the GPT-4-based policy [44], we acquire task-completion judgments for each of the three evaluation systems described above. Table 1 shows the overall accuracy of the evaluator's predictions.[8] The end-to-end approach with GPT-4V achieves 80.6% accuracy, while Captioner + Mixtral, which uses only open-weight models, matches the oracle's evaluations for 74.4% of tasks, and replacing Mixtral with GPT-4 achieves the highest accuracy at 82.1%.

**Android-in-the-Wild** For the 120 sampled test tasks in AitW, we evaluate trajectories sampled from four policies:

[5] We subsample from the original test set of 1.4k tasks to facilitate acquiring human judgments of trajectories. See Appendix A.4 for details on a list of evaluated tasks and details on task sampling.

[6] Details on our emulators are available in Appendices A.4 and A.6.

[7] The human demonstrations use the original AitW emulator, which was not released by the authors; thus, these results are not directly comparable with the automated policies, which use the emulator we implement. However, the focus of our experiments is not to directly compare policies, but to compare evaluators across a variety of policies, tasks, and domains.

[8] Figure 4 in Appendix A.3 includes the confusion matrices of these predictions.

| | GPT-4V | Captioner + GPT-4 | Captioner + Mixtral |
|---|---|---|---|
| WebArena (%) | 80.6 | 82.1 | 74.4 |
| Android (%) | 90.6 | 89.8 | 92.9 |

Table 1. Comparison of evaluators accuracy against oracle evaluator or human judge in WebArena and Android.

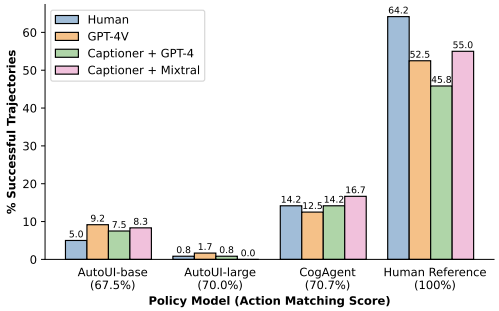


Figure 2. Evaluating models in Android-in-the-Wild with different evaluation methods. We use human judgments of trajectory success as oracle reference and compare it with judgments from our evaluation models and AitW's standard action matching score.

CogAgent [17], Auto-UI$_{\{\text{large, base}\}}$ [43], and human experts [27].[7] We acquire human judgments of trajectory success, as well as judgments from the three evaluation model variants.[9] Figure 2 shows the performance of all four agents as evaluated by humans and the three evaluator variants. Below each agent label we also include each policy's partial action match score [24], which is the standard reported metric for agents on AitW.[10]

Unsurprisingly, we find that the human reference trajectories achieve the highest performance as evaluated by all success metrics. However, our analysis reveals that about 36% of the human demonstrations we annotate are actually unsuccessful, with common errors including early stopping, completing the wrong task, and making mistakes with respect to the parameters of the task. The difficulty of collecting high-quality demonstration data at scale further demands automated evaluation methods that can either act as a quality filter or provide more direct evaluation than action matching score.

Among the three neural network policies, CogAgent achieves the highest success rates, followed by Auto-UI$_{\text{base}}$, while the performance of Auto-UI$_{\text{large}}$ is close to zero according to all evaluators. When comparing conclusions that can be drawn from the two styles of metrics – task success and action matching – there are three clear differences: first, that success rate lags far behind single-step action prediction; second, that relative performance of models changes depending on the metric used; and third, that using a reference-based metric on erroneous references could result in inflated impressions of model performance. In particular, while Auto-UI$_{\text{large}}$ appears to outperform Auto-UI$_{\text{base}}$ according to the action matching metric, it is clearly inferior in terms of overall task success rate. Quantitatively, all three evaluators achieve a Kendall correlation of 100% with the human judges, while the action matching score only obtains 66.7%. This highlights a fundamental drawback in a single-step metric like action matching: it does not reflect error propagation or distribution shift in the sequential prediction process of an arbitrary policy, which can be captured by whole-trajectory success metrics.

Measuring whole-trajectory success for the complex tasks that digital agents complete has typically required either human evaluation of individual trajectories, or manual creation of individual test cases, as in WebArena. We analyze the potential for automating this process using our three proposed evaluators. Table 1 shows the accuracy of each evaluator variant aggregated over trajectories from all four policies.[8] Overall, we find that our automated metrics correlate very strongly with human judgment: the Captioner + Mixtral variant shows the highest agreement with human judgment at 92.9% accuracy; replacing Mixtral with GPT-4 leads to a performance drop to 89.8%; and the end-to-end approach of GPT-4V achieves 90.6% accuracy.

### 4.2. Autonomous Refinement

**Reflexion on WebArena** We demonstrate how our proposed evaluation models can serve as a reward signal to guide an existing web agent at inference time, using the Reflexion technique [29] as an example. In Reflexion, an agent first attempts a task, and an external evaluator is used to judge whether its attempt was successful or not. If it is judged as unsuccessful, the agent will be prompted to reflect on the failure and retry. We experiment with improving the current state-of-the-art GPT-4-based WebArena agent.[11]

Figure 3 includes the agent's baseline performance, and performance using up to three rounds of Reflexion with the oracle evaluator (which serves as an upper bound) and our three evaluation systems as external supervision. We see the improvement our evaluators provide scales favorable with evaluator capability, with Captioner + Mixtral improves agent's relative success rate by 16% and GPT-4V based evaluator by 29%. All system variants, including the low-cost and locally-hosted variant Captioner + Mixtral, significantly enhance agent's performance while requiring no access to hand-designed evaluation functions.

Our preliminary study suggests that false negative evaluations have a more detrimental impact on agent's performance compared to false positives. If our evaluator predicts an execution is incorrect, but it was actually successful, this forces the agent to retry a successful execution, which

[9] Experimental setup details for AitW are provided in Appendix A.4

[10] Action matching scores are averaged across the subsets of AitW we sample from, as reported in [17] and [43].

[11] Reflexion prompts are detailed in Appendix A.5.

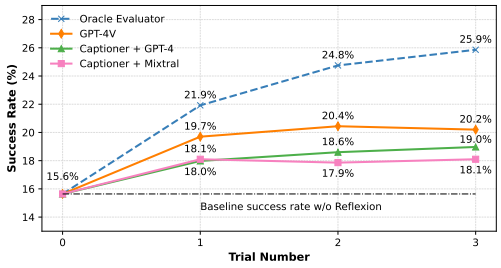


Figure 3. Results of applying Reflexion for up to 3 rounds using different evaluation systems on the WebArena benchmarks. Here, the oracle evaluator denotes performance using WebArena's built-in evaluation functions as the reward function; this provides an upper-bound of improvement using Reflexion.

nearly always leads a subsequent failure. In contrast, false positives only lose out on the opportunity to retry, which creates an upper bound of performance for the agent, but does not degrade its performance. Improving the robustness of inference-time algorithms under noisy supervision is an interesting future direction to explore.

**Filtered Behavior Cloning on iOS** We demonstrate how our evaluator can guide the refinement of a policy in low-resouce domain using filtered behavior cloning (filtered BC), without additional supervision. We use CogAgent [17] as the policy model for the experiment. CogAgent is an 18B-parameter VLM specialized in GUI understanding and navigation. It is primarily instruction-tuned with demonstrations from web navigation and Android device control, and incorporates a very limited, manually collected iOS dataset for training. Given a screenshot and an instruction, CogAgent first generates a high-level plan, followed by the low-level action. For data collection and testing purposes, we design 132 common tasks on iOS, with 80 tasks for training and 52 for testing. Given scaling limitations of emulation, including low speeds and restriction to emulation on macOS, we only experiment with iOS built-in apps and with the Captioner + Mixtral evaluator.

We first sample 737 trajectories from CogAgent, conditioned on the 80 training tasks. We use our evaluator to provide per-step evaluations to these trajectories, then apply filtered BC for fine-tuning using this data. Unlike standard fine-tuning, this method filters out data points with rewards below a specified threshold. We set this threshold at $\geq p$; i.e., we retain only state-action pairs that positively influence the success of a trajectory (Section 3). Additionally, we assess CogAgent's unmodified performance on iOS and explore a self-training approach by finetuning without data filtering as baselines for comparison.

Table 2 contains results for the 52 test tasks. iOS device control is a challenging task, with the baseline agent completing only 8 out of 52 tasks, yielding a 15% success rate. Self-training improves over the baseline by 3 tasks.

| Policy | # Successful Tasks |
|---|---|
| CogAgent Baseline | 8 |
| + Self-training | 11 |
| + Filtered BC (Ours) | 14 |

Table 2. Comparison of CogAgent and refined policies via self-training and filtered behavior cloning, including the number of successful tasks in our test set (out of 52 total tasks).

Filtered BC with our evaluator significantly improved the policy model's performance from 8 to 14 successes, marking a 75% relative improvement.

### 4.3. Error Analysis

We randomly sample 20 successful and 30 erroneous evaluations for each evaluation model in WebArena and manually annotate the sources of failure.[12] We categorize errors into three primary types, providing percentage estimates rounded to the nearest 5%.

1. Critical information lost from captions in the modular approach (10%); errors in screenshot understanding for the end-to-end GPT-4V approach (5%).
2. Errors in the reasoning process, observed in 50% of cases for GPT-4V/GPT-4-based methods and 70% for Mixtral-Captioner.
3. Ambiguities in task specification and success criteria, observed in 30% of cases for GPT-4V/GPT-4-based methods and 10% for Mixtral-Captioner.

We note that in our error categorization, a model must overcome errors in preceding categories to be assessed under the subsequent one. Consequently, Mixtral-Captioner's lower rate of Type 3 errors is mostly attributed to its higher frequency of Type 1 and 2 errors.

Additionally, we find the model provides the correct final evaluation, but incorrect reasoning, for about 10% of correct evaluations.

## 5. Conclusion

In this study, we design automatic methods to both evaluate and refine the performance of digital agents. We first describe a model that provides either trajectory-level or per-step evaluation of agent's performance. Subsequently, we propose two approaches to implement the model: an end-to-end approach using a pre-trained vision-language model, and a modular caption-then-reason approach using a VLM and a pre-trained language model together. These methods offer trade-offs between performance, cost, and modularity.

Using WebArena and Android-in-the-Wild as testbeds, we first validate the effectiveness of these evaluators against oracle evaluation metrics, and highlight their advantage

[12] Refer to Figures 5 through 11 in the Appendix for visual representations of these evaluations.

over standard reference-based metrics on AitW. We then show how the evaluators can be used to refine existing agents through both inference-time guidance and filtered BC. When integrated as the reward function in Reflexion, a method for inference-time refinement, our evaluators enhance the best-performing agent's success rate by up to 29%. Additionally, it boosts the performance of a strong device control policy in a domain transfer task by 75% via filtered behavior cloning, all without any extra supervision. Our findings show the potential of model-based automated evaluators for both evaluating and improving digital agents, which is especially critical in developing real-world agents where ground truth evaluation functions or human supervision are not always available.

## Limitations and Future Work

While our research demonstrates the potential of model-based evaluators in evaluating and improving digital agents, we also identify several areas for future exploration. First, current evaluators are still far from perfect, and any enhancement in their performance, e..g, from better representations of the action space or stronger base models, will likely directly translate to improved outcomes. Second, in this work, we focused on Reflexion and filtered behavior cloning. Future works can explore scaling up the experiments and developing better training and inference-time algorithms that are robust and efficient under noisy supervision. Finally, in this work we only make use of the evaluator's binary or ternary judgment, and discard the language-based explanation it generates. Future work can explore how to leverage this information, for example, to further enhance policies through language supervision or to provide scalable oversight of agent behavior.

## Ethics Statement

Most currently available digital agents are research artifacts. As the performance of these agents improve and they are increasingly deployed in the real world, they may pose security risks to their users. For example, a web agent with unconstrained access to a browser might be able to gain access a user's passwords, financial information, or social media messages. Better understanding the potential failure modes of these models in real-world use cases is critical to ensuring their safe deployment. We view our work as a first step in this direction: by developing domain-general evaluators, we hope to facilitate better understanding of models (and their risks) outside of simulated environments like WebArena. At the same time, human evaluation and oversight of these future systems will also be important for mitigating potential harms; although our work in this paper focuses on autonomous evaluation, we hope it will supplement, rather than supplant, human efforts.

**Acknowledgments**

We thank the Berkeley NLP group, especially Ruiqi Zhong, Andre He, Charlie Snell, Catherine Chen, Sanjay Subramanian, and Zineng Tang, as well as Allen Nie for feedback and discussions and Shuyan Zhou for assistance in setting up the WebArena experiments. This work was partially supported by an AI2 Young Investigator Grant. NT is supported by the DARPA SemaFor program.

## References


[1] Marwa Abdulhai, Isadora White, Charles Burton Snell, Charles Sun, Joey Hong, Yuexiang Zhai, Kelvin Xu, and Sergey Levine. LMRL Gym: Benchmarks for multi-turn reinforcement learning with language models. *ArXiv*, abs/2311.18232, 2023. 2

[2] OpenAI: Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Benjamin Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Raphael Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Lukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Ryan Kiros, Matthew Knight, Daniel Kokotajlo, Lukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Adeola Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel P. Mossing, Tong Mu, Mira Murati, Oleg Murk, David M'ely, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Ouyang Long, Cullen O'Keefe, Jakub W. Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alexandre Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Pondé de Oliveira Pinto, Michael Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario D. Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin D. Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas A. Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cer'on Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. GPT-4 technical report. 2023. 1, 3

[3] James Allen, Nathanael Chambers, George Ferguson, Lucian Galescu, Hyuckchul Jung, Mary Swift, and William Taysom. PLOW: a collaborative task learning agent. In *AAAI*, 2007. 2

[4] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-VL: A frontier large vision-language model with versatile abilities. *ArXiv*, abs/2308.12966, 2023. 3, 4

[5] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional AI: Harmlessness from AI feedback. *ArXiv*, abs/2308.12966, 2022. 2

[6] S.R.K. Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. Reinforcement learning for mapping instructions to actions. In *ACL-AFNLP*, 2009. 2

[7] S.R.K. Branavan, Luke Zettlemoyer, and Regina Barzilay. Reading between the lines: Learning to map high-level instructions to commands. In *ACL*, 2010. 2

[8] Andrea Burns, Deniz Arsan, Sanjna Agrawal, Ranjitha Ku-

mar, Kate Saenko, and Bryan A. Plummer. A dataset for interactive vision-language navigation with unknown command feasibility. In *ECCV*, 2022. 2

[9] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *NeurIPS*, 2021. 3

[10] Xinyue Chen, Zijian Zhou, Zheng Wang, Che Wang, Yanqiu Wu, and Keith Ross. BAIL: Best-action imitation learning for batch deep reinforcement learning. In *NeurIPS*, 2020. 3

[11] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. In *NeurIPS Datasets and Benchmarks Track*, 2023. 2, 13

[12] Brad Dwyer. Website screenshots dataset, 2020. 13

[13] Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. RvS: What is essential for offline RL via supervised learning? In *ICLR*, 2022. 3

[14] Jiaxian Guo, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Boyang Li, Dacheng Tao, and Steven Hoi. From images to textual prompts: Zero-shot visual question answering with frozen large language models. In *CVPR*, 2023. 3

[15] Izzeddin Gur, Hiroki Furuta, Austin V Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. In *ICLR*, 2024. 2

[16] Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. WebVoyager: Building an end-to-end web agent with large multimodal models. *ArXiv*, abs/2401.13919, 2024. 2

[17] Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and Jie Tang. CogAgent: A visual language model for GUI agents. *arXiv*, abs/2312.08914, 2023. 2, 4, 5, 6

[18] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022. 12

[19] Peter C. Humphreys, David Raposo, Tobias Pohlen, Gregory Thornton, Rachita Chhaparia, Alistair Muldal, Josh Abramson, Petko Georgiev, Alex Goldin, Adam Santoro, and Timothy P. Lillicrap. A data-driven approach for learning to control computers. In *ICML*, 2022. 2

[20] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts. *ArXiv*, abs/2401.04088, 2024. 3, 4

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 11

[22] Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. *ArXiv*, abs/2401.13649, 2024. 2

[23] Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. RLAIF: Scaling reinforcement learning from human feedback with ai feedback. *arXiv*, abs/2309.00267, 2023. 2

[24] Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. Mapping natural language instructions to mobile UI action sequences. In *ACL*, 2020. 2, 5

[25] Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration. In *ICLR*, 2018. 2

[26] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155, 2022. 2

[27] Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy P Lillicrap. AndroidInTheWild: A large-scale dataset for android device control. In *NeurIPS Datasets and Benchmarks Track*, 2023. 1, 2, 4, 5, 13

[28] Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of Bits: An open-domain platform for web-based agents. In *ICML*, 2017. 2

[29] Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *NeurIPS*, 2023. 1, 2, 3, 5, 11

[30] Daniel Toyama, Philippe Hamel, Anita Gergely, Gheorghe Comanici, Amelia Glaese, Zafarali Ahmed, Tyler Jackson, Shibl Mourad, and Doina Precup. AndroidEnv: A reinforcement learning platform for android. *ArXiv*, abs/2105.13231, 2021. 2

[31] Bryan Wang, Gang Li, and Yang Li. Enabling conversational interaction with mobile ui using large language models. In *CHI*, 2023. 2

[32] Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-Agent: Autonomous multi-modal mobile device agent with visual perception. *arXiv*, abs/2401.16158, 2024. 2

[33] Ziyue Wang, Chi Chen, Peng Li, and Yang Liu. Filling the image information gap for VQA: Prompting large language models to proactively ask questions. 2023. 3

[34] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS*, 2022. 3

[35] Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng

Kong. OS-Copilot: Towards generalist computer agents with self-improvement. *arXiv*, abs/2402.07456, 2024. 2

[36] Nancy Xu, Sam Masling, Michael Du, Giovanni Campagna, Larry Heck, James Landay, and Monica Lam. Grounding open-domain instructions to automate web support tasks. In *NAACL-HLT*, 2021. 2

[37] An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Qinghong Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu Zhong, Julian McAuley, Jianfeng Gao, Zicheng Liu, and Lijuan Wang. GPT-4V in Wonderland: Large multimodal models for zero-shot smartphone GUI navigation. *ArXiv*, abs/2311.07562, 2023. 4, 13

[38] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *NeurIPS*, 2022. 2

[39] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *NeurIPS*, 2023. 2

[40] Haoxuan You, Rui Sun, Zhecan Wang, Long Chen, Gengyu Wang, Hammad Ayyubi, Kai-Wei Chang, and Shih-Fu Chang. IdealGPT: Iteratively decomposing vision and language reasoning via large language models. In *Findings of the Association for Computational Linguistics: EMNLP*, 2023. 3

[41] Chi Zhang, Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. AppAgent: Multimodal agents as smartphone users. *arXiv*, abs/2312.13771, 2023. 2

[42] Chaoyun Zhang, Liqun Li, Shilin He, Xu Zhang, Bo Qiao, Si Qin, Minghua Ma, Yu Kang, Qingwei Lin, Saravan Rajmohan, et al. UFO: A UI-focused agent for Windows OS interaction. *arXiv*, abs/2402.07939, 2024. 2

[43] Zhuosheng Zhang and Aston Zhang. You only look at screens: Multimodal chain-of-action agents. *ArXiv*, abs/2309.11436, 2023. 2, 4, 5

[44] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *ICLR*, 2024. 1, 2, 4