Mobile-Bench-v2: A More Realistic and Comprehensive Benchmark for VLM-based Mobile Agents

Anonymous ACL submission

Abstract

VLM-based mobile agents are increasingly popular due to their capabilities to interact with smartphone GUIs and XML-structured texts and complete daily tasks. However, existing online benchmarks struggle with result replication due to dynamic environmental changes, while offline benchmarks, with their single-trajectory annotations, force the agents to follow the preferences of the annotator, limiting their reflections to complete tasks through multiple paths. Additionally, both types of benchmarks fail to assess whether agents can handle noise or engage in proactive interactions due to a lack of noise and overly full instructions. To address these limitations, we construct a more realistic and comprehensive multimodal offline benchmark named Mobile-Bench-v2, which includes a common task split with multi-path evaluations, a Noisy-APP split with pop-ups and ads, a contaminated split AITZ-Noise based on AITZ, and an ambiguous instruction split with preset Q&A interactions. We evaluate agent frameworks with VLMs on the common split using both single- and multi-path evaluation and assess the supervised fine-tuning agent on AITZ-Noise. Moreover, we explore whether incorporating noise into the original training data can overcome in-domain ad contamination. Data will be released in the future.

1 Introduction

LLM–based mobile agents (Wang et al., 2023; Ding, 2024) are increasingly popular due to their capability to interact directly with mobile Graphic User Interfaces (GUIs) and their potential to manage daily tasks autonomously. Unfortunately, LLM-based agents cannot fully comprehend the mobile GUI structure and widget functionality, relying solely on text such as Visual-Hierarchical, XML, HTML, or Accessible-Visited Trees. VLM-based agents (Ma et al., 2024; Zhang et al., 2024a) can provide a more comprehensive understanding of GUIs through visual perception and text assistance. This has led to recent work replacing foundational models with VLMs, resulting in some benchmarks for end-to-end mobile tasks based on GUI pages.

Existing VLM-based Agent benchmarks can be broadly categorized as online and offline: (1) Online evaluation involves the agent executing operations on a real device based on the user's high-level instructions until the task is completed. These benchmarks directly determine the success rate by checking the widget values in the final GUI. (2) Offline evaluation uses static datasets where the golden path is pre-executed on the device, with actions and screenshots saved offline. The agent generates the current action based on each step's screenshot and historical actions. Online benchmarks (Murthy et al., 2024; Deng et al., 2024a; Wang et al., 2024c) allow agents to complete tasks through various paths; however, due to the instability of the device environment, such as OS updates, APP updates, and user preference records, the evaluation results are fluctuating and unstable. Although offline benchmarks (Chai et al., 2024; Cheng et al., 2024; Rawles et al., 2024) are more convenient, static GUIs will gradually become obsolete. Considering the diversity of agent task solutions, the agent's good performance may only represent a good fit to the preferences encoded in the current benchmark annotations but does not necessarily indicate robustness or the ability to handle multi-path solutions. Benchmarks such as MobileAgentBench (Wang et al., 2024c) and AutoDroid (Wen et al., 2024) are constructed on real devices and evaluated within Google apps using the Android Accessibility Service; these apps feature clean pages without task-irrelevant ads, buttons, and pop-ups. At the same time, users may not be able to provide such precise and full instructions all at once (Wang et al., 2024f). Overall, existing benchmarks have several limitations, including a lack of multi-path evaluation, overly clean testing

Table 1: Comparison of Mobile-Bench-v2 to other benchmarks. **Scale**: # is the number of unique instructions on general third-party apps, average steps per instruction, and screenshots. * indicates step metrics are mis-annotated, and the tasks are not normal mobile GUI tasks.

Benchmarks	# Unique General Inst.	# Avg Steps	# Screen-shots	Task Category	Task Path	Realistic Environment	Ambiguous Noise
PIXELHELP	187	4.2	∼800	Navi.&QA	Single	✗	✗
MOTIF	480	4.5	∼21K	Navi.	Single	✗	✗
AITW	1,539	6.5	∼510K	Navi.	Single	✗	✗
AITZ	506	7.5	∼18K	Navi.	Single	✗	✗
AMEX	341	12.8	∼104K	Navi.	Single	✗	✗
SCREENSPOT	∼1,200	1	∼600	Grounding	Dot	✗	✗
MOBILEAIBENCH	*	*	*	QA	Dot	✗	*
MOBILEAGENTBENCH	100	20	∼2k	Navi.	Multiple	✓	✗
GUI ODEYSSEY	7,735	15.4	*	Navi.	Single	✗	✗
MOBILE-BENCH	832	*	14,144	Navi.	Multiple	✓	✗
Mobile-Bench-v2	12,856	7.28	∼48k	Navi.&QA	Both	✗	✓

environments, and overly explicit instructions.

To address the above limtations, we extend Mobile-Bench (Deng et al., 2024a) to form a new benchmark named Mobile-Bench-v2. Specifically, we make the following improvements: (1) **Offline Multi-path Tasks**: In contrast to the Mobile-Bench LLM online testing platform, we construct multi-modal offline data. To combine the advantages of both online and offline environments, we propose a multi-path testing approach. Based on the random walk graph-structured corpus of Mobile3M (Wu et al., 2024a), we use the GIAS (Generating Instructions From GUI Action Sequences) to construct 12k instructions. We verify the quality and stability of the generated instructions by performing multi-path sampling and human evaluation. Then, multiple closed-source frameworks and open-source agents are tested on this benchmark using both single-path and multi-path modes. (2) **Simulating realistic noisy environment**: We collect an additional sub-dataset named **Mobile-Bench-Noisy** with substantial ads and pop-ups to simulate a noisy environment and contaminate AITZ and AITW by inserting ads into original trajectories to build **AITZ-Noise**. (3) **Cross-lingual Evaluation**: Existing datasets and benchmarks set tasks in Google apps and English environments. However, these apps tend to be cleaner than their Chinese counterparts. We use native English ads from AITZ as in-domain noise, while ads collected from Chinese apps serve as out-domain ones. We also explore whether introducing English ads into the training data can enhance the agent's ability to handle in-domain noise contamination. (4) **Active Interactive Evaluation**: We construct a sub-dataset named **Mobile-Bench-ambiguous**, which allows agents to ask when necessary during task execution. Full and ambiguous instructions are built in the first round, and the questions raised by the agents during the warm-up process are assigned to each step, along with manually annotated answers in the second round.

Overall, our work makes three main contributions:

• We construct evaluation data based on Mobile3M's graph structure corpus and propose the GIAS for annotating instructions.

• We manually construct ambiguous instruction tasks and noise tasks, select apps with frequent ads as noise benchmarks and contaminate existing clean data by inserting ads.

• To the best of our knowledge, we are the first to create a comprehensive multimodal GUI evaluation benchmark for mobile GUI agents while also building multipath and noise evaluation.

2 Related work

2.1 Mobile Agents

Large language models (Achiam et al., 2023) emerge as autonomous agents (Li et al.; Wen et al., 2023) in the mobile domain and garner considerable attention. With the rapid development of vision-language models (VLMs), multimodal researchers build mobile GUI agents (Yang et al., 2023; Zheng et al., 2024) and multi-agent frameworks (Ding, 2024; Li et al., 2024; Wang et al., 2024b) based on closed-source VLMs. Meanwhile, some researchers focus on training agents with stronger element grounding (Cheng et al., 2024; Hong et al., 2024; Wu et al., 2024b), page navigation (Niu et al., 2024; Lu et al., 2024; Gou et al.,

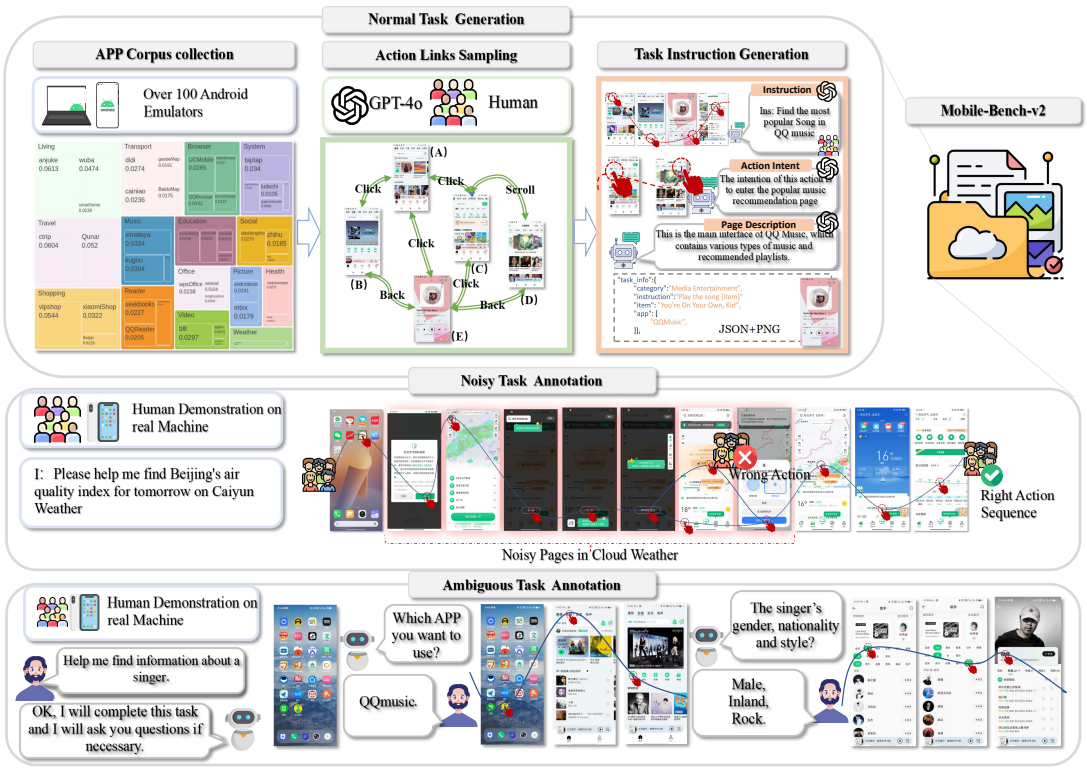


Figure 1: The **Mobile-Bench-v2** includes three types of tasks: **Common-split, Noisy-split, and Ambiguous-split**, and demonstrates the process of instruction generation and manual annotation for each task. In Noisy-split, the GUIs with red shading represent noise.

2024), GUI understanding (Chai et al., 2024; You et al., 2024; Baechler et al., 2024) and task planning capabilities (Zhang et al., 2024c; Nong et al., 2024; Xu et al., 2024) based on open-source VLMs. In addition, Bai et al. (2024) and Wang et al. (2024e) use joint online and offline reinforcement learning to enhance the generalization of mobile agents.

2.2 Mobile Agent Benchmarks

As shown in Table 1, AndroidEnv (Toyama et al., 2021) and MobilEnv (Zhang et al., 2023) are the first to create LLM agent evaluation environments based on reinforcement learning. MobileBench (Deng et al., 2024a) and AppBench (Wang et al., 2024a) introduce online benchmarks combining API and GUI, while MobileAgentBench (Wang et al., 2024c) establishes the first fully automated multimodal benchmark for VLM-based GUI agents. More offline benchmarks (Li et al., 2020a; Burns et al., 2021; Murthy et al., 2024) are released, which are primarily categorized into GUI understanding and task-oriented. (1) For task-oriented benchmarks, AITW (Rawles et al., 2024) and AITZ (Zhang et al., 2024b) create large-scale benchmarks based on Google apps, while AMEX (Chai et al., 2024) supplements these benchmarks by adding data for GUI understanding with similar app types. ScreenSpot (Cheng et al., 2024), Mobile3M (Wu et al., 2024a), and GUIOdyssey (Lu et al., 2024) focus on more granular element grounding and task planning. (2) Rico (Deka et al., 2017) is the first non-annotated GUI corpus, followed by ScreenQA (Hsiao et al., 2022), Widget Caption (Li et al., 2020b), and Screen2words (Wang et al., 2021), which is for Q&A, widget understanding, and page summarization. Subsequently, Mind2web (Deng et al., 2024b) incorporates additional GUI data of varying sizes, and Meta-GUI (Sun et al., 2022) provides tasks for multi-round dialogues.

3 Mobile-Bench-v2

3.1 Mobile Task Formulation

For the GUI agent, there are four essential capabilities: (i) **Planning** to determine the action step sequences. (ii) **Action Thought** to produce an action description at each step (*e.g.,* "open the flight detail page"), (iii) **Element Grounding** to identify a widget (*e.g.,* "[Click](x_1,y_1)") on the GUI, (iiii)

Action Reflection to evaluate whether the action result is correct. Given a mobile screenshot $\mathcal{S}$ (*e.g.,* a ctrip screenshot on Android) and a task T (*e.g., "Book a fight ticket from Chengdu to Beijing Sep.15 for me."*), a GUI agent should generate a sequence of executable actions. Specifically, at time step t, the agent should select an action $\boldsymbol{a}_t$ from the action space $\mathcal{A}$, which includes three types of actions: (1) Click. (2) Scroll. (3) Input.

$$\hat{a}_t = \begin{cases} [x_1, y_1, x_2, y_2], & \hat{a}_t \in Click \\ \mathcal{D} \in \{\uparrow, \downarrow, \leftarrow, \rightarrow\}, & \hat{a}_t \in Scroll \\ text, & \hat{a}_t \in Type \end{cases} \quad (1)$$

Based on the current environment observation $\mathcal{S}_t$, the action history $\mathcal{H}_{1:t-1}$=$\{\hat{a}_1, \hat{a}_2, ..., \hat{a}_{t-1}\}$, and the last step refection $\boldsymbol{f}_{t-1}$, the GUI agent will generate plan $\mathcal{P}_t$:

$$\mathcal{P}_t = \left\{ \hat{a}_t^{(1)} \cdots \hat{a}_t^{(n)} \mid (\hat{a}_1, \cdots, \hat{s}_{t-1}), f_{t-1}, \mathcal{S}_t \right\} \quad (2)$$

where $\mathcal{P}_t$ represents the planning of the next n actions starting from the current step. The environment observation $\mathcal{S}_t$ comprises an HTML document $text_t$ and a mobile screenshot $image_t$. If the current step is the first step of the entire task, the overall plan $\mathcal{P}_\mathcal{T}$ can be expressed as:

$$\mathcal{P}_\mathcal{T} = \left\{ \mathcal{A}_{1:n} \mid \boldsymbol{C}_a, \arg \max_{\{\mathcal{T}_{i_j}\}^k} \sum_{j=1}^{k} \text{SIM}(\mathcal{T}_i, \mathcal{T}_{i_j}) \right\} \quad (3)$$

where $\boldsymbol{C}_a$ is the corpus of each APP, which is collected by the GUI agent random walk. $\sum_{j=1}^{k} \text{SIM}(\mathcal{T}_i, \mathcal{T}_{i_j})$ is the top_k recall of the similarity comparison between the current task $\mathcal{T}_i$ and backup task $\mathcal{T}_{i_j}$.

3.2 Data Construction

The breadth-first search and random walk algorithms of Mobile3M are described in Appendix A and data distribution is shown in Figure 4.

Generating common instructions from action sequences. When we construct the Mobile3M graph corpus, the key challenge is how to annotate instructions for each trajectory that closely aligns with the intended actions. Building on Murty et al. (2024)'s fine-tuning of web agents to eliminate redundant actions from action sequences, two key points for pairing trajectories and instructions are the **intent understanding** and the **content changes** between different GUIs: (i) Using intents behind actions instead of themselves is more model-friendly to VLMs because coordinate-based actions without GUI pages cannot accurately reproduce the action scenes. Buttons with the same appearance but no textview may have entirely different meanings on the same GUI. In Figure 6, the same *'plus'* button represents adding *'Hazelnut Latte'* and *'Cookie Mocha'* respectively. Simply recording the action itself may lead to semantic confusion in action history. (ii) Providing content change descriptions to VLMs can reduce hallucinations when handling the sequential relationships between consecutive GUIs. Due to input length limitations, VLMs can typically process no more than eight 1K-resolution images in a single sequence, while Llama3.2-90B can only process a single image at a time.

Algorithm 1 GIAS Algorithm

Require: Start Page, P_0; End Page, P_t; Trajectory σ; Page Description des; GUI Pages, s; Action, a; Action Intent, T; Page Changes, C; Instruction, I; Task, $\mathcal{T}$
Ensure: Prompt, P; Few Shot Cases, F_S; Verified Flag, $\boldsymbol{Q}$;
Select $\sigma_i = \{s_{i_1}, s_{i_2}, \ldots, s_{i_{t-1}}\}$ from P_0 to P_t
for each $s_{ij} \in \sigma_i$ **do**
  for $j = 0$ to t **do**
    $des(s_{ij}) \leftarrow Qwen(s_{ij}, P_t)$
    $T_{i_{j:j+1}} \leftarrow des(s_{ij}), des(s_{i(j+1)}), a(s_{ij} \rightarrow s_{i(j+1)})$
    $C_{i_{j:j+1}} \leftarrow des(s_{ij}), des(s_{i(j+1)})$
    $I_{ij} \leftarrow \{C_{i0}, C_{i1}, \ldots, C_{it}\}, \{T_{i0}, T_{i1}, \ldots, T_{it}\}$
  end for
end for
for each path σ_i **do**
  $I_i \leftarrow Merge\ \{I_{i1}, I_{i2}, \ldots, I_{ik}\}$
end for
for each I_i **do**
  $I_i \leftarrow Simp(I_i, F_S, P_{sim})$
end for
$\mathcal{T}, Q \leftarrow Veri\ \{I_1, I_2, \ldots, I_i, P, F_S\}$ for all Trajectories σ_i, ensuring no redundant steps
return $\mathcal{T}$

To address the limitations above, we propose an automated instruction annotation method named **GIAS** (Generating Instructions From Mobile UI Action Sequences), which is shown in Figure 1. The whole process is as follows: (1) multi-path sampling based on fixed start and end pages; (2) GUI page content annotation; (3) action intent inference; (4) GUI Change Summary; (5) sub-instruction generation; (6) merging and simplification. The entire process is explained in detail in Algorithm 1. Specifically, we choose paths that start from nodes with the same name in Mobile3M and end at homogeneous nodes with different names (Homogeneous nodes refer to pages whose similarity or the number of identical UI elements exceeds the threshold (Lu et al., 2006)). Considering its diversity, we select trajectories that include at least two different types of actions and minimize the ratio of homogeneous pages. We deploy Qwen2-VL-

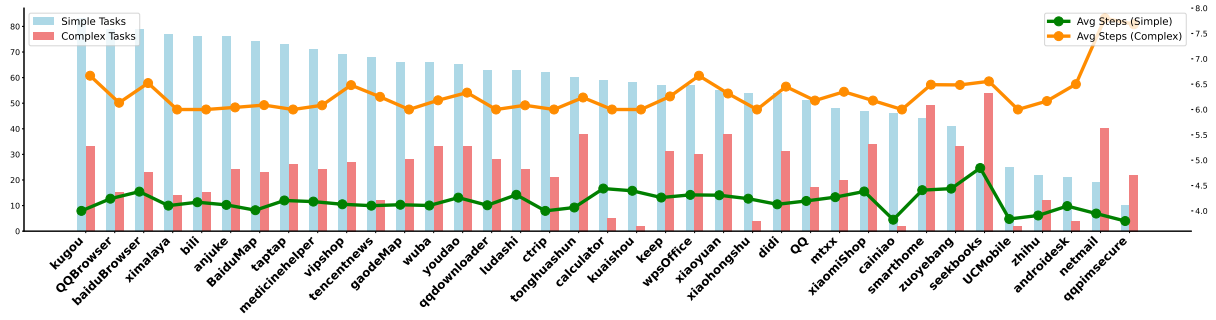


Figure 2: The **task distribution** chart is sorted by the number of simple tasks in descending order. The average steps for both simple and complex tasks in each app remain relatively balanced.

72B (Wang et al., 2024d) for page content, GPT-4 for change annotation, and then use GPT-4o for sub-task instruction generation, merging, and simplification. More details on GIAS can be seen in Appendix B.

Noisy app and ambiguous instruction data. Mobile-Bench-Noisy is primarily derived from manual annotation and contamination in existing data: (1) For manual annotation, we select apps from third-party markets; these apps contain unavoidable ads and pop-ups. When performing actions on these apps, we do not handle the following scenarios in advance: login, update, permission settings, ad pop-ups, and VIP subscriptions. In some instructions, to test the agent's response in unexpected situations, we deliberately click on ad pages incorrectly to see if it can recover from divergent paths. All noisy GUIs are additionally marked, and XML is dumped, which makes this benchmark adaptable for non-purely visual agents as well. (2) For data contamination, we randomly insert at least one ad into AITZ and AITW trajectory. For Mobile-Bench-Ambiguous, we first construct the full instructions annotate action trajectories, and remove key information from these instructions to simulate ambiguous instructions. Multiple sets of interactive Q&A are annotated at each step. For example, the full instruction is: '*I want a 16GB + 512GB MacBook Pro M4 in the Midnight version.*' The ambiguous instruction is: '*I want to buy a MacBook.*' In fact, it is only when the agent enters the selection page that he will know that the Apple M4 no longer offers the *Midnight version*. More details can be seen in appendix C.2.

Figure 3: The task distribution fan figure for five representative APPs: **on the left**, the distribution of simple tasks and their respective step counts; **on the right**, the distribution of complex tasks and their respective step counts.

3.3 Data Statistics

The apps and categories in the Mobile-Bench-v2 remain consistent with Mobile3M, comprising 15 categories and 49 apps, with each category containing at least three similar apps. It includes 12,854 test cases, which we divide into two categories based on action steps: simple tasks (4-6 steps) and complex tasks (7-11 steps). As shown in Figure 3, there are 9,620 simple tasks with an average of 4.62 steps and 3,234 complex tasks with an average of 7.21 steps. All instructions are generated using the GIAS algorithm along with state-of-the-art open and closed-source VLMs. Figure 2 shows the task distribution. We strive for balance, but some apps, like *SeekBooks* (a book-finding tool), experience imbalances. These occur because GIAS finds it hard to understand user intent, especially given the limited number of exploration steps in shopping apps. To analyze this phenomenon, we additionally compare the task proportions of five representative applications in Figure 3. As an observation, shopping apps (*DuApp&Zhuishushenqi*) have a higher proportion of complex tasks compared to simple tasks. In contrast, since *Baicizhan* features a clean page and straightforward functionality (*vocabulary learning*), constructing task instructions from random walk data becomes easier when the exploration depth is shallow. The noise and ambiguous

instruction test split each contains 100 instances, with each task in the ambiguous split including at least 5 additional manually constructed Q&A. The average trajectory lengths are 12.74 for the noise tasks and 7.53 for the ambiguous instruction tasks. Furthermore, we randomly insert one of 150+ ads at a step within one of the 2504 trajectories in AITZ and AITW, ensuring that it overlays the original target button while shifting a step.

3.4 Data Quality Verification

The common split is based on Mobile3M's filtering and annotation, while the noisy and ambiguous instructions are entirely manually annotated. Due to the random walk involved in the former, the instruction trajectories may include redundant actions, whereas the latter requires checking whether the noise is handled correctly and the quality of the Q&A. Therefore, we designed a data quality verification experiment shown in Table 2, extracting 100 data from the subsets to validate the quality of the instructions and annotated trajectories. Win Rate represents the proportion of instructions generated by GIAS that are equal to or exceed the quality of those manually validated and annotated. SE is almost equal to 1, indicating that there are almost no redundant steps in the annotation.

Metric	Simple	Complex	Noisy	Ambiguous
Annotation Step	4.62	7.21	12.74	7.53
Evaluation Step	4.57	7.07	12.36	7.12
Win Rate↑	86.0	72.5	100	86.0
SE↓	1.01	1.02	1.03	1.05

Table 2: Quality Verification experiment results. Crowdsourced annotations with quality verification performed by agent professionals.

4 Experiments

4.1 Experimental Setup

For Mobile-Bench-v2 common, noisy, and ambiguous splits, we experiment the agent frameworks such as AppAgent and MobileAgent-v2 with different fundamental VLMs: Qwen2-VL-72B (Wang et al., 2024d), Llama3.2-90B (Dubey et al., 2024), Qwen-VL-Max, GPT-4o (Achiam et al., 2023) and GPT-4v. We use OS-Atlas and Qwen2-VL-7B to evaluate AITZ-Noise and construct CoaT format data on AITZ trajectories to fine-tune them. To reduce close-source model costs, only zero-shot evaluations are done on a subset split of *Random-800 (600 simple and 200 complex)*, which has a similar sub-split distribution with the full split. Simple and complex splits set the maximum number of steps to 20 and 25. For ambiguous data, we set up an ablation study (1): Ambiguous Instruction: providing the agent with ambiguous instructions and step-by-step Q&A; (2) Full Instruction: providing the agent with a single comprehensive instruction covering all details upfront. We use LlamaFactory for fine-tuning and building the CoaT data as multi-turn dialogues. For multi-path, two GUIs are provided only during back actions, while other actions are restricted to the current GUI. Unlike Appagent, we annotate the widget types using a specific letter with a number (Figure 8).

4.2 Metrics

We establish all four metrics based on the methodologies proposed in Wang et al. (2024c) and Zhang and Zhang (2023),

Success Rate (SR): $N_{success}/M_{tasks}$, where $N_{success}$ is the number of completed tasks, judged by whether the agent reaches the final pages in multi-path evaluation or does completely correct actions in single-path evaluation.

Step Efficiency (SE): S_{actual}/S_{min}, where S_{actual} is the number of actual steps to complete a task, and S_{min} is the task's minimal annotated steps. This metric expresses if the agent performs unnecessary or redundant actions in multi-path evaluation.

Step Accuracy (Step.Acc): S_{tp}/S_{gt}, where S_{tp} is the number of predicted actions that match the golden actions, and S_{gt} is the number of golden actions. For click and scroll actions, the predicted action needs to be in the same area as the golden action, with the scroll action additionally requiring the same direction as the golden action. For input actions, the predicted text must achieve an F1 score of at least 0.5 compared to the golden action.

TYPE: S_{ttp}/S_{gt}, where S_{ttp} is the number of predicted actions that match the type of golden actions. For whole three actions, we use TYPE to check whether the action types are correct.

4.3 Main Result

Common data results. As shown in Table 3, the Qwen series models showcase superior performance in commom tasks, while Llama performs relatively poorly. Under AppAgent-v1, the best-performing Qwen2-VL-72B achieves 21.1% SR in the single-path evaluation of simple task and in the fewest SE(5.2) achieving highest 20.6% SR in the multi-path evaluation of the simple task. GPT-4o outperforms Qwen2-VL-72B in SR(25.6% >

Models	Cate.	Common-Simple			Common-Complex			Noisy Data			Ambiguous Data		
		Type/SE	Step. Acc	SR	Type/SE	Step. Acc	SR	Type/SE	Step. Acc	SR	Type/SE	Step. Acc	SR
AppAgent-v1 with Single-Agent Framework													
Qwen2-VL-72B	Single	95.2	60.3	21.1	93.5	53.8	5.0	78.0	24.4	0.0	91.2	43.5	1.0
	Multi	5.2	62.8	20.6	4.4	58.9	4.0	-	-	-	-	-	-
Qwen-VL-Max	Single	94.7	58.6	20.5	91.2	54.7	7.5	77.1	24.3	0.0	90.3	48.5	0.0
	Multi	5.9	67.6	12.6	4.3	63.1	6.6	-	-	-	-	-	-
Llama3.2-VL-90B	Single	86.4	22.4	2.6	87.0	24.3	1.0	69.7	11.2	0.0	85.4	15.5	0.0
	Multi	-	-	-	-	-	-	-	-	-	-	-	-
GPT-4v	Single	91.2	24.0	6.0	90.8	25.2	0.0	72.7	17.4	0.0	88.6	20.5	0.0
	Multi	6.1	29.7	3.0	4.5	29.4	0.0	-	-	-	-	-	-
GPT-4o	Single	80.4	57.6	18.5	69.2	40.6	1.5	52.3	18.2	0.0	64.7	33.9	0.0
	Multi	5.3	61.8	19.8	4.4	61.7	6.5	-	-	-	-	-	-
MobileAgent-v2 with Multi-Agents Framework													
Qwen2-VL-72B	Single	91.5	50.5	13.0	91.6	49.0	4.5	75.8	20.7	0.0	86.2	40.8	1.0
	Multi	5.4	54.9	15.1	4.4	58.6	4.0	-	-	-	-	-	-
Qwen-VL-Max	Single	74.2	17.0	3.0	68.8	12.3	2.0	66.5	4.2	0.0	66.6	7.0	0.0
	Multi	5.4	29.6	4.5	4.3	24.8	3.0	-	-	-	-	-	-
Llama3.2-VL-90B	Single	62.4	16.6	1.0	67.0	17.5	0.0	63.7	9.7	0.0	64.3	8.3	0.0
	Multi	-	-	-	-	-	-	-	-	-	-	-	-
GPT-4v	Single	90.8	22.9	3.8	90.6	28.3	0.5	62.5	12.6	0.0	91.0	15.6	0.0
	Multi	6.0	17.8	5.4	4.5	11.8	0.0	-	-	-	-	-	-
GPT-4o	Single	91.9	53.5	13.5	92.3	50.5	7.0	77.1	25.5	0.0	91.6	39.7	2.0
	Multi	4.9	57.6	25.6	4.2	56.3	7.5	-	-	-	-	-	-

Table 3: Results on MobileBench-v2 Common, Noisy and Ambiguous splits. Type is used in the single-path evaluation, while SE is used in the multi-path evaluation. The '-' indicates that the model does not support Reflection in the multi-path setting due to single-image support or window length limitations.

Agent	Benchmark	General		Google App		Install		Web Shopping		Total	
		Step. Acc	Noisy	Step. Acc	Noisy	Step. Acc	Noisy	Step. Acc	Noisy	Step. Acc	Noisy
Qwen2-VL	Normal	38.5	-	44.8	-	60.0	-	45.1	-	46.9	
	Noise	37.3	15.4	42.2	17.2	54.7	20.5	42.1	17.1	43.9	17.4
OS-Atlas	Normal	41.9	-	46.4	-	60.5	-	46.3	-	48.6	
	Noise	38.8	21.8	41.7	19.7	56.4	23.5	43.8	23.6	45.1	21.7

Table 4: Results on AITZ-Noise. Qwen2-VL and OS-Atlas are evaluated on AITZ and AITZ-Noise.

15.1%) on multi-path evaluation of the simple task, with fewer SE(4.9 < 5.4) under MobileAgent-v2 framework. Counterintuitively, GPT-4o achieved a higher Step.Acc and SR compared to GPT-4v and Llama3.2-VL-90B, but lower TYPE. This is because GPT-4o exhibits weaker instruction-following capabilities when page information and few-shot guidance are lacking. AppAgent-v1 outperforms in single-path evaluations, whereas MobileAgent-v2 excels in multi-path scenarios. This discrepancy can be attributed to the fact that, compared to the predefined correct action history in single-path testing, agents are more likely to get lost in the divergent paths of multi-paths. However, MobileAgent-v2's reflection and summarization mechanism effectively mitigates this issue.

Noisy data results. For noisy data, results in the third block of Table 3 demonstrate that none of the models are able to complete a task. Even superior models achieve very low Steps. Acc and TYPE on tasks involving noisy data. For instance, GPT-4o merely attains 25.5% Step.Acc and 77.1% TYPE under MobileAgent-v2. At the same time, under MobileAgent-v2, Qwen-VL-Max still could not be correctly output according to the prompt in the command format. The comparison between MobileAgent-v2 and AppAgent-v1 reveals that high-performing models, such as GPT-4o and Qwen2-VL-72B, achieve superior results in MobileAgent-v2, while low-performing models, such as GPT-4v and Llama3.2-VL-90B, perform relatively better in AppAgent-v1.

4.4 Ambiguous Instruction Ablation Study

As shown in the far-right column of Table 3, Qwen-VL-Max exhibits relatively strong results in TYPE (90.3%) and Step.Acc (48.5%) under AppAgent-v1, while Qwen2-VL-72B shows the best performance under MobileAgent-v2 with a SE of 86.2% and Step.Acc of 40.8%. This ablation study demonstrates that step-by-step Q&A can help the agent effectively ignore irrelevant content in task instructions for the current step. The results in Table 6 indicate that step-by-step Q&A contributes to

Agent	Training Data	General			Google App			Install			Web Shopping		
		Step. Acc	Noisy	SR	Step. Acc	Noisy	SR	Step. Acc	Noisy	SR	Step. Acc	Noisy	SR
Supervised Fine-tuning Setting(LoRA)													
CogAgent	AITW(CoaT)	40.4	-	11.5	38.1	-	11.3	45.2	-	17.3	39.1	-	13.4
Qwen2-VL-7B	AITZ(CoaT)	36.1	-	8.3	39.1	-	11.2	50.9	-	20.7	41.8	-	15.2
Qwen2-VL-7B	AITZ-Noise	39.8	98.0	11.7	42.3	99.0	16.6	60.9	100	30.4	41.5	99.0	13.3
OS-Atlas-7B	AITZ-Noise	46.2	99.0	18.7	50.2	99.5	21.3	62.4	100	33.0	44.8	99.0	17.3
Supervised Fine-tuning Setting(Full)													
Qwen2-VL-7B	AITZ-Noise	43.2	96.0	15.6	46.2	97.5	19.8	64.2	98.5	35.6	50.9	98.0	22.3
OS-Atlas-7B	AITZ-Noise	47.2	98.0	19.0	47.1	99.0	22.3	66.7	99.0	38.0	51.8	99.0	23.5

Table 5: Qwen2-VL, Cogagent, and OS-Atlas fine-tuned on AITZ-Noise, AITW, or AITZ and evaluation on AITZ-Noise. Metric "Noisy" means in-domain noisy step accuracy. More Experiments can be seen in Table 7.

Models	Full Instruction			Ambiguous Instruction		
	Type	Step. Acc	SR	Type	Step. Acc	SR
AppAgent-v1 with Single-Agent Framework						
Qwen2-VL-72B	82.9	41.5	1.0	91.2	43.5	1.0
Qwen-VL-Max	81.2	39.3	0.0	90.3	48.5	1.0
Llama3.2-VL-90B	67.9	7.6	0.0	85.4	15.5	0.0
GPT-4v	72.8	15.9	0.0	88.6	20.5	0.0
GPT-4o	59.7	31.9	1.0	64.7	33.9	0.0
MobileAgent-v2 with Multi-Agents Framework						
Qwen2-VL-72B	80.3	38.5	1.0	86.2	40.8	1.0
Qwen-VL-Max	57.6	3.3	0.0	66.6	7.0	0.0
Llama3.2-VL-90B	57.8	4.0	0.0	64.3	8.3	0.0
GPT-4v	84.6	13.9	0.0	91.0	15.6	0.0
GPT-4o	87.7	38.4	2.0	91.6	39.7	2.0

Table 6: Ablation study on Mobile-Bench-Ambiguous.

enhanced performance. Notably, under AppAgent-v1, the Step.Acc of Qwen-VL-Max and GPT-4v using step-by-step Q&A increased by 9.2% and 4.6% compared to full instruction. Full instructions may affect the model's ability to accurately identify tasks on the current page, whereas ambiguous instructions with step-by-step Q&A help the model better comprehend the page and execute more appropriate actions.

5 Discussion

In-domain noisy AITZ results. As shown in Table 4, the Step.Acc of Qwen2-VL and OS-Atlas decreased by an average of 3.0% and 3.5% from normal to in-domain noise. Given that the Noisy step accuracy is 17.4% and 21.7%, this indicates that smaller-scale agents fail to learn the features of advertisements during fine-tuning. As a result, they exhibit almost no generalization capability on transferred noisy data, even when only the background screenshot changes. More details can be found in Appendix B.2.

Mobile-Bench-Noisy out-domain data results. Unlike AITZ-Noise, the ads in Noisy-App are more dynamic and variable which is shown in Figure 11. Specifically, they exhibit the following three features: (1) After the pop-up ad countdown ends, the ad disappears automatically, and the agent's delayed instructions may cause accidental taps; (2) Some video ads cannot be closed during the early viewing stages; (3) The mis-taps caused by real ad noise may trigger app redirection. Due to these factors, in table 3, the Step.Acc of GPT-4o, GPT-4v, and Qwen-VL-Max decreased by 39.4%, 6.6%, and 30.4%, compared with common-complex task. The Step.Acc of models like GPT-4o on out-domain noise is, on average, 51% lower compared to in-domain agents.

Solving in-domain noise through fine-tuning. We are more focused on whether increasing the proportion of noisy training data can address the in-domain noisy problem. As shown in Table 5, Qwen2-VL, compared to the original AITZ training data, shows a Step.Acc improvement of 3.7%, 3.2%, 10.0%, -0.3% and SR improvement of 3.4%, 5.4%, 9.7%, -1.9% on the four sub-tasks. At the same time, full parameter fine-tuning outperforms LoRA in overall results but performs slightly worse than LoRA on noise step processing (an average of 1.5% lower). After training, the agent is able to correctly handle the vast majority of noisy steps (with an accuracy greater than 97%), demonstrating the effectiveness of training with noisy data.

6 Conclusion

In this paper, we propose Mobile-Bench-v2, an extension of the previous version that incorporates additional data modalities, multi-path evaluation, noisy data, and ambiguous instruction data. We also propose a novel instruction-free trajectory annotation method without human evaluation named GIAS. This benchmark provides a foundation for evaluating and optimizing GUI agents studies focused on planning decisions, noise robustness, and proactive interaction.

Limitations

Although multi-path validation similar to that on online machines was achieved on Mobile-Bench-v2, the diverse range of text inputs cannot be exhaustively covered, which differentiates it from online machines. Advanced agents such as AutoGLM (Liu et al., 2024) and others deployed by smartphone manufacturers could not be tested due to permission restrictions.

Ethics Statement

We have rigorously refined our dataset to remove any elements that could compromise personal privacy, thereby guaranteeing the highest level of protection for individual data. All data annotations were completed by crowdsourced volunteers, to whom we paid $0.5 per step as compensation and provided the necessary training. The human evaluation of our work was carried out through a meticulously randomized selection of IT professionals. This process ensured a gender-balanced and educationally diverse panel, reflecting a wide spectrum of perspectives and expertise.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Gilles Baechler, Srinivas Sunkara, Maria Wang, Fedir Zubach, Hassan Mansoor, Vincent Etter, Victor Cărbune, Jason Lin, Jindong Chen, and Abhanshu Sharma. 2024. Screenai: A vision-language model for ui and infographics understanding. *arXiv preprint arXiv:2402.04615*.

Hao Bai, Yifei Zhou, Mert Cemri, Jiayi Pan, Alane Suhr, Sergey Levine, and Aviral Kumar. 2024. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. *arXiv preprint arXiv:2406.11896*.

Andrea Burns, Deniz Arsan, Sanjna Agrawal, Ranjitha Kumar, Kate Saenko, and Bryan A Plummer. 2021. Mobile app tasks with iterative feedback (motif): Addressing task feasibility in interactive visual environments. *arXiv preprint arXiv:2104.08560*.

Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Dingyu Zhang, Peng Gao, Shuai Ren, and Hongsheng Li. 2024. Amex: Android multi-annotation expo dataset for mobile gui agents. *arXiv preprint arXiv:2407.17490*.

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

Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th annual ACM symposium on user interface software and technology*, pages 845–854.

Shihan Deng, Weikai Xu, Hongda Sun, Wei Liu, Tao Tan, Jianfeng Liu, Ang Li, Jian Luan, Bin Wang, Rui Yan, et al. 2024a. Mobile-bench: An evaluation benchmark for llm-based mobile agents. *arXiv preprint arXiv:2407.00993*.

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

Tinghe Ding. 2024. Mobileagent: enhancing mobile control via human-machine interaction and sop integration. *arXiv preprint arXiv:2401.04124*.

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

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. 2024. Navigating the digital world as humans do: Universal visual grounding for gui agents. *arXiv preprint arXiv:2410.05243*.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. 2024. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14281–14290.

Yu-Chung Hsiao, Fedir Zubach, Maria Wang, et al. 2022. Screenqa: Large-scale question-answer pairs over mobile app screenshots. *arXiv preprint arXiv:2209.08199*.

Wei Li, William E Bishop, Alice Li, Christopher Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on ui control agents. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Yanda Li, Chi Zhang, Wanqi Yang, Bin Fu, Pei Cheng, Xin Chen, Ling Chen, and Yunchao Wei. 2024. Appagent v2: Advanced agent for flexible mobile interactions. *arXiv preprint arXiv:2408.11824*.

Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. 2020a. Mapping natural language instructions to mobile ui action sequences. *arXiv preprint arXiv:2005.03776*.

Yang Li, Gang Li, Luheng He, Jingjie Zheng, Hong Li, and Zhiwei Guan. 2020b. Widget captioning: Generating natural language description for mobile user interface elements. *arXiv preprint arXiv:2010.04295*.

Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, et al. 2024. Autoglm: Autonomous foundation agents for guis. *arXiv preprint arXiv:2411.00820*.

Quanfeng Lu, Wenqi Shao, Zitao Liu, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, Yu Qiao, and Ping Luo. 2024. Gui odyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. *arXiv preprint arXiv:2406.08451*.

Wei Lu, Stephen Robertson, and Andrew Macfarlane. 2006. Field-weighted xml retrieval based on bm25. In *Advances in XML Information Retrieval and Evaluation: 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, November 28-30, 2005. Revised Selected Papers 4*, pages 161–171. Springer.

Xinbei Ma, Zhuosheng Zhang, and Hai Zhao. 2024. Coco-agent: A comprehensive cognitive mllm agent for smartphone gui automation. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 9097–9110.

Rithesh Murthy, Liangwei Yang, Juntao Tan, Tulika Manoj Awalgaonkar, Yilun Zhou, Shelby Heinecke, Sachin Desai, Jason Wu, Ran Xu, Sarah Tan, et al. 2024. Mobileaibench: Benchmarking llms and lmms for on-device use cases. *arXiv preprint arXiv:2406.10290*.

Shikhar Murty, Dzmitry Bahdanau, and Christopher D Manning. 2024. Nnetscape navigator: Complex demonstrations for web agents without a demonstrator. *arXiv preprint arXiv:2410.02907*.

Runliang Niu, Jindong Li, Shiqi Wang, Yali Fu, Xiyu Hu, Xueyuan Leng, He Kong, Yi Chang, and Qi Wang. 2024. Screenagent: A vision language model-driven computer control agent. *arXiv preprint arXiv:2402.07945*.

Songqin Nong, Jiali Zhu, Rui Wu, Jiongchao Jin, Shuo Shan, Xiutian Huang, and Wenhao Xu. 2024. Mobileflow: A multimodal llm for mobile gui agent. *arXiv preprint arXiv:2407.04346*.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. 2024. Androidinthewild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems*, 36.

Liangtai Sun, Xingyu Chen, Lu Chen, Tianle Dai, Zichen Zhu, and Kai Yu. 2022. Meta-gui: Towards multi-modal conversational agents on mobile gui. *arXiv preprint arXiv:2205.11029*.

Daniel Toyama, Philippe Hamel, Anita Gergely, Gheorghe Comanici, Amelia Glaese, Zafarali Ahmed, Tyler Jackson, Shibl Mourad, and Doina Precup. 2021. Androidenv: A reinforcement learning platform for android. *arXiv preprint arXiv:2105.13231*.

Bryan Wang, Gang Li, and Yang Li. 2023. Enabling conversational interaction with mobile ui using large language models. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–17.

Bryan Wang, Gang Li, Xin Zhou, Zhourong Chen, Tovi Grossman, and Yang Li. 2021. Screen2words: Automatic mobile ui summarization with multimodal learning. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pages 498–510.

Hongru Wang, Rui Wang, Boyang Xue, Heming Xia, Jingtao Cao, Zeming Liu, Jeff Z Pan, and Kam-Fai Wong. 2024a. Appbench: Planning of multiple apis from various apps for complex user instruction. *arXiv preprint arXiv:2410.19743*.

Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024b. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. *arXiv preprint arXiv:2406.01014*.

Luyuan Wang, Yongyu Deng, Yiwei Zha, Guodong Mao, Qinmin Wang, Tianchen Min, Wei Chen, and Shoufa Chen. 2024c. Mobileagentbench: An efficient and user-friendly benchmark for mobile llm agents. *arXiv preprint arXiv:2406.08184*.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. 2024d. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Taiyi Wang, Zhihao Wu, Jianheng Liu, Jianye Hao, Jun Wang, and Kun Shao. 2024e. Distrl: An asynchronous distributed reinforcement learning framework for on-device control agents. *arXiv preprint arXiv:2410.14803*.

Wenxuan Wang, Juluan Shi, Chaozheng Wang, Cheryl Lee, Youliang Yuan, Jen-tse Huang, and Michael R Lyu. 2024f. Learning to ask: When llms meet unclear instruction. *arXiv preprint arXiv:2409.00557*.

Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. 2023. Empowering llm to use smartphone for intelligent task automation. *arXiv preprint arXiv:2308.15272*.

Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. 2024. Autodroid: Llm-powered task automation in android. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, pages 543–557.

Qinzhuo Wu, Weikai Xu, Wei Liu, Tao Tan, Jianfeng Liu, Ang Li, Jian Luan, Bin Wang, and Shuo Shang. 2024a. Mobilevlm: A vision-language model for better intra-and inter-ui understanding. *arXiv preprint arXiv:2409.14818*.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, et al. 2024b. Os-atlas: A foundation action model for generalist gui agents. *arXiv preprint arXiv:2410.23218*.

Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. 2024. Aguvis: Unified pure vision agents for autonomous gui interaction. *arXiv preprint arXiv:2412.04454*.

Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2023. Appagent: Multimodal agents as smartphone users. *arXiv preprint arXiv:2312.13771*.

Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. 2024. Ferret-ui: Grounded mobile ui understanding with multimodal llms. *arXiv preprint arXiv:2404.05719*.

Danyang Zhang, Hongshen Xu, Zihan Zhao, Lu Chen, Ruisheng Cao, and Kai Yu. 2023. Mobile-env: an evaluation platform and benchmark for llm-gui interaction. *arXiv preprint arXiv:2305.08144*.

Jiayi Zhang, Chuang Zhao, Yihan Zhao, Zhaoyang Yu, Ming He, and Jianping Fan. 2024a. Mobileexperts: A dynamic tool-enabled agent team in mobile devices. *arXiv preprint arXiv:2407.03913*.

Jiwen Zhang, Jihao Wu, Yihua Teng, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. 2024b. Android in the zoo: Chain-of-action-thought for gui agents. *arXiv preprint arXiv:2403.02713*.

Li Zhang, Shihe Wang, Xianqing Jia, Zhihan Zheng, Yunhe Yan, Longxi Gao, Yuanchun Li, and Mengwei Xu. 2024c. Llamatouch: A faithful and scalable testbed for mobile ui task automation. *arXiv preprint arXiv:2404.16054*.

Zhuosheng Zhang and Aston Zhang. 2023. You only look at screens: Multimodal chain-of-action agents. *arXiv preprint arXiv:2309.11436*.

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

Category	APP	Unique Nodes	All Nodes	Action Steps	All Nodes(%)
Living	anjuke	57,286	190,102	1,334,428	6.13%
Living	wuba	38,667	147,009	903,586	4.74%
Living	smarthome	12,595	42,961	304,816	1.39%
Travel	ctrip	63,449	187,079	1,217,304	6.04%
Travel	Qunar	42,462	161,005	1,211,015	5.20%
Shopping	vipshop	72,468	168,531	1,036,086	5.44%
Shopping	xiaomiShop	21,666	99,770	755,718	3.22%
Shopping	duapp	18,925	38,926	223,379	1.26%
Transport	didi	12,786	84,865	637,400	2.74%
Transport	cainiao	20,593	73,132	480,223	2.36%
Transport	gaodeMap	13,674	59,142	319,377	1.91%
Transport	BaiduMap	13,552	54,322	280,498	1.75%
Browser	UCMobile	40,618	88,220	615,049	2.85%
Browser	baiduBrowser	36,016	70,282	401,348	2.27%
Browser	QQBrowser	18,500	44,006	218,828	1.42%
Browser	tencentnews	23,408	38,241	224,804	1.23%
System	taptap	24,759	105,461	624,941	3.40%
System	qqpimsecure	8,997	42,691	379,926	1.38%
System	ludashi	2,773	32,474	219,804	1.05%
System	qqdownloader	10,517	28,502	151,824	0.92%
System	calculator	4,265	15,819	97,005	0.51%
System	supercaculator	690	1,369	5,444	0.04%
Music	ximalaya	34,995	103,395	577,032	3.34%
Music	kugou	40,043	94,271	504,368	3.04%
Music	QQmusic	5,545	17,539	64,211	0.57%

Category	APP	Unique Nodes	All Nodes	Action Steps	All Nodes (%)
Reader	seekbooks	15,902	70,266	563,882	2.27%
Reader	QQReader	22,588	63,458	472,509	2.05%
Reader	zhuishushenqi	14,737	63,210	392,903	2.04%
Reader	pdfreader	495	1,507	5,211	0.05%
Social	xiaohongshu	45,324	85,362	525,519	2.75%
Social	zhihu	21,766	57,261	373,756	1.85%
Social	QQ	7,051	20,600	141,969	0.66%
Education	zuoyebang	19,884	70,661	507,146	2.28%
Education	Xiaoyuan	10,727	56,806	393,395	1.83%
Education	Youdao	8,756	35,121	206,035	1.13%
Education	Baicizhan	4,196	16,383	88,500	0.53%
Office	wpsOffice	11,156	73,739	486,661	2.38%
Office	Netmail	5,544	32,308	260,682	1.04%
Office	tonghuashun	6,410	30,722	163,297	0.99%
Office	QQmail	712	1,590	4,597	0.05%
Video	bili	46,080	91,891	471,940	2.97%
Video	qqlive	12,497	22,601	99,677	0.73%
Video	kuaishou	7,126	12,115	59,373	0.39%
Picture	androidesk	28,432	59,228	418,773	1.91%
Picture	mtxx	19,718	55,324	419,055	1.79%
Health	medicinehelper	15,046	83,832	547,880	2.71%
Health	keep	7,730	22,500	117,124	0.73%
Weather	pureweather	25,252	79,283	610,695	2.56%
Weather	cloudweather	1,956	3,904	19,339	0.13%
15	49	998,334	3,098,786	20,138,332	100%

Figure 4: The data distribution in Mobile3M.

A Mobile3M Dataset

Mobile3M is a large-scale dataset designed to systematically explore and analyze the functionality of mobile applications through UI-based interactions. It provides a comprehensive representation of user interface (UI) elements, interactions, and app navigation patterns. Mobile3M is characterized by the following key features:

(1) **Scale and Diversity**
Mobile3M includes over 20 million user interactions, covering 3 million screenshots and corresponding XML documents. These data are organized into directed graphs for 49 widely-used Chinese apps, where nodes represent UI pages, and edges capture user actions.

(2) **Detailed UI Representation**
Each UI page is described by both a screenshot and an XML document. The XML documents provide detailed structural information, including UI elements (e.g., buttons, text fields), their hierarchical relationships, and layout properties such as bounding boxes.

(3) **Action Space**
The dataset defines three fundamental user actions—*click*, *scroll*, and *input*—to simulate real-world app interactions. Each UI page contains an action space derived from its interactable elements, facilitating comprehensive modeling of user behaviors.

(4) **Graph-Based Organization**
Mobile3M employs a breadth-first search (BFS) algorithm to explore app functionality, representing the exploration results as graphs. This structure enables the identification of app workflows, the relationship between UI pages, and the possible transitions triggered by user actions.

(5) **Efficiency and Optimization**
To enhance exploration efficiency, Mobile3M incorporates a "unique page" mechanism that eliminates duplicates by comparing UI pages using a combination of element and pixel-based similarity thresholds. This reduces the exploration space, prevents redundant actions, and avoids cyclic sequences, ensuring more diverse and meaningful data coverage.

(6) **Balanced Action Distribution**
The dataset emphasizes balanced representation of user actions by prioritizing underrepresented interac-

tions, such as *input*. For example, random keywords are introduced for input actions, and scroll actions are executed in multiple directions to capture diverse app behaviors.

(7) **Task-Oriented Exploration**

Inspired by APPAgent, the dataset leverages a random walk algorithm to systematically interact with UI elements and record transitions between pages. The exploration process captures action traces, enabling task-driven navigation and detailed understanding of app functionalities.

B Data Analysis and Construction

B.1 GIAS Prompt

Input Format

You will be provided with a series of user interaction histories, each consisting of a caption describing the current page and an action performed by the user.

Your Task

Analyze each action and the corresponding page caption to determine what action was taken on that page. Summarize these actions into a task description, which should be a request. For example:

- *"I want to see what VIP privileges are available."*
- *"Help me find pants on sale."*
- *"Tell me what items are in my shopping cart."*

Important Notes:

1. *The task description and the sequence of actions should have a logical relationship.*
2. *The task description should be phrased as a request, reflecting the goal of the actions taken.*
3. *Actions and captions should be analyzed in sequence to deduce the user's objective.*

Output Format

"step-by-step description": *"Provide a series of interactions, where each entry corresponds to a screenshot caption of the current phone screen and the action performed on that page."*

"concise task": *"Summarize the user's overall goal based on the step-by-step description."*

Example

Caption 1:
This image shows a screenshot of a shopping application interface.
Action 1:
Click(Skincare Set)
Caption 2:
This image shows a screenshot of a shopping application interface. At the top, there is a search bar with the text "Skincare Set." Additionally, at the bottom of the page, there is a navigation bar with options like "All Products," "New Arrivals," "Moisturizing," "Dry Skin," "Niacinamide," and "Hyaluronic Acid." The current state is "All Products."
Action 2:
Click(New Arrivals)
Caption 3:
This image shows a screenshot of a shopping application interface. At the top, there is a search bar with the text "Skincare Set." Additionally, there is a navigation bar at the bottom with options like "All Products," "New Arrivals," "Moisturizing," "Dry Skin," "Niacinamide," and "Hyaluronic Acid." The current state is "New Arrivals." Below are multiple product recommendations.
Action 3:
Click(Ad)

Caption 4:
This image shows a product detail page. At the top, there is a pink banner that reads "Buy a set and get 13 items free," along with a product photo.
Output:
"step-by-step description":

1. *Click the "Skincare Set" product under the "Beauty" subcategory of "Recommended."*
2. *On the Skincare Set search results page, click the "New Arrivals" tab.*
3. *On the product details page, click the "Ad" tab.*

"Concise task":
Help me find the latest skincare set that is on promotion.

New Input and Task

Now, based on the following input, please generate the "step-by-step description" and "concise task":
`{trajectory_description}`

B.2 Data Contamination

The collected advertisements are shown in Figure 7. We embed them into the normal dataset and applied background whitening. We ensure that the elements that should have been clicked on the current page are no longer visible after the contamination. When splitting the training and test data, the position of the embedded advertisements is randomly assigned. However, the types of advertisements in the training data are largely consistent with those in the test data, and the same advertisements maintain consistent embedding positions.

Agent	Training Data	General		Google App		Install		Web Shopping		Total	
		Step. Acc	Noisy	Step. Acc	Noisy	Step. Acc	Noisy	Step. Acc	Noisy	Step. Acc	Noisy
Normal Data Supervised Fine-tuning											
Qwen2-VL-7B	AITZ	37.33	15.38	42.18	17.11	54.68	20.45	42.06	17.14	43.84	17.46
OS-Atlas	AITZ	38.81	21.79	41.61	19.73	56.37	23.57	43.71	23.57	45.16	21.62
In-domain Noise Supervised Fine-tuning											
Qwen2VL	AITZ + Noisy	43.07	77.56	47.63	73.68	60.64	75.76	44.02	75.00	48.20	75.79
OS-Atlas	AITZ + Noisy	44.92	82.05	49.64	76.32	63.06	79.55	48.01	78.57	50.99	79.56
Out-domain Noise Supervised Fine-tuning											
Qwen2VL	AITZ + Noisy	37.18	50.64	45.18	41.89	57.45	53.38	42.32	50.00	44.96	49.90
OS-Atlas	AITZ + Noisy	41.75	53.85	45.47	48.65	60.27	60.90	47.28	55.71	48.69	55.47

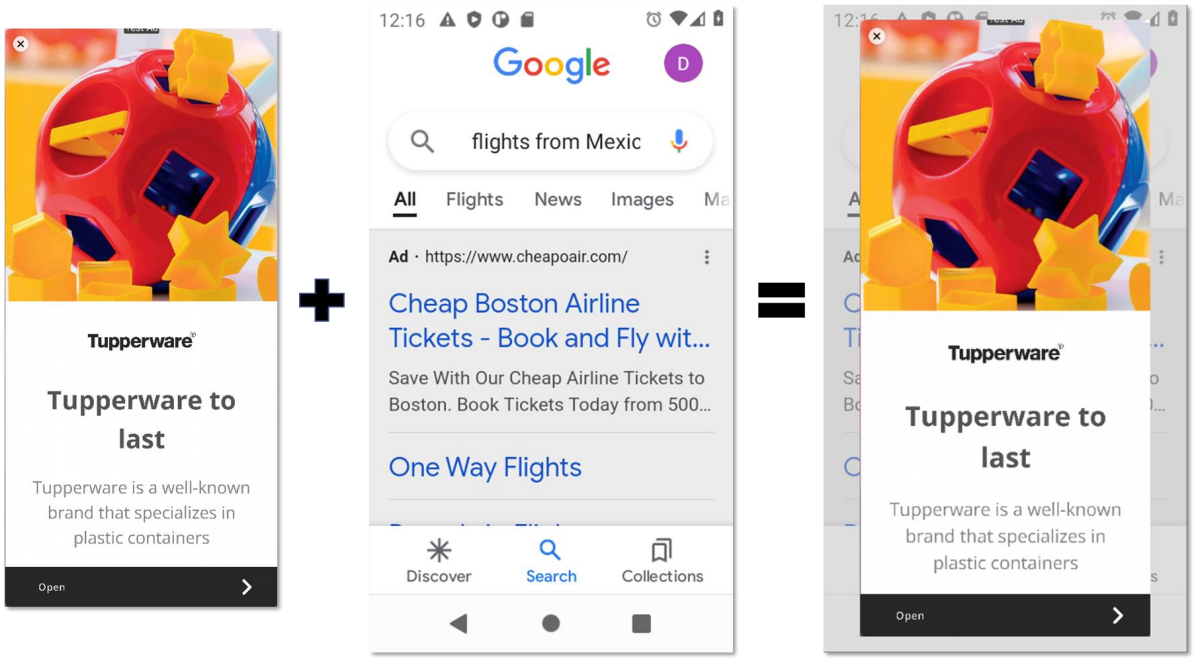
Table 7: Qwen2-VL and OS-Atlas fine-tuned on AITZ-Noise, AITW, or AITZ and evaluation on AITZ-Noise(Out-domain). Metric "Noisy" means out-domain noisy step accuracy.

C Experiment Details

C.1 Baseline Model Demonstration

AppAgent Below is the prompt we used. We did not re-adapt or adjust the prompt for different base models to ensure fairness.

```
I will give you the screenshot of a mobile app, the clickable UI element is labeled
with a letter 'c' and the number <ui_element> on the screen. The tag of each element is located at the center of
    the
element. Clicking on this UI element is a necessary part of proceeding with a larger task, which is to <
    task_description>.
In order to realize this larger task, you must first realize the current task <current_task_desc> in current
    screenshot.
Your task is to describe the functionality of the UI element concisely in one or two sentences. Notice that your
description of the UI element should focus on the general function. For example, if the UI element is used to
    navigate
```



AD top-up **GUI Screenshot** **Polluted GUI Screenshot**

Figure 5: Contaminated datasets are constructed by inserting advertisements and whitening the background of the original GUI screenshots.

Figure 6: The red boxed areas represent identical graphical controls and identical textual controls, which can create ambiguity in the action history.

```
to the chat window with John, your description should not include the name of the specific person. Just say:
"Clicking this area will navigate the user to the chat window". Never include the tag of the
UI element in your description. You can use pronouns such as "the UI element" to refer to the element.
```

Listing 1: Click Document Template

```
I will give you the screenshot of a mobile app, the clickable UI element is labeled
with a letter 'c' and the number <ui_element> on the screen. The tag of each element is located at the center of
    the
element. Clicking on this UI element is a necessary part of proceeding with a larger task, which is to <
    task_description>.
In order to realize this larger task, you must first realize the current task <current_task_desc> in current
    screenshot.
```

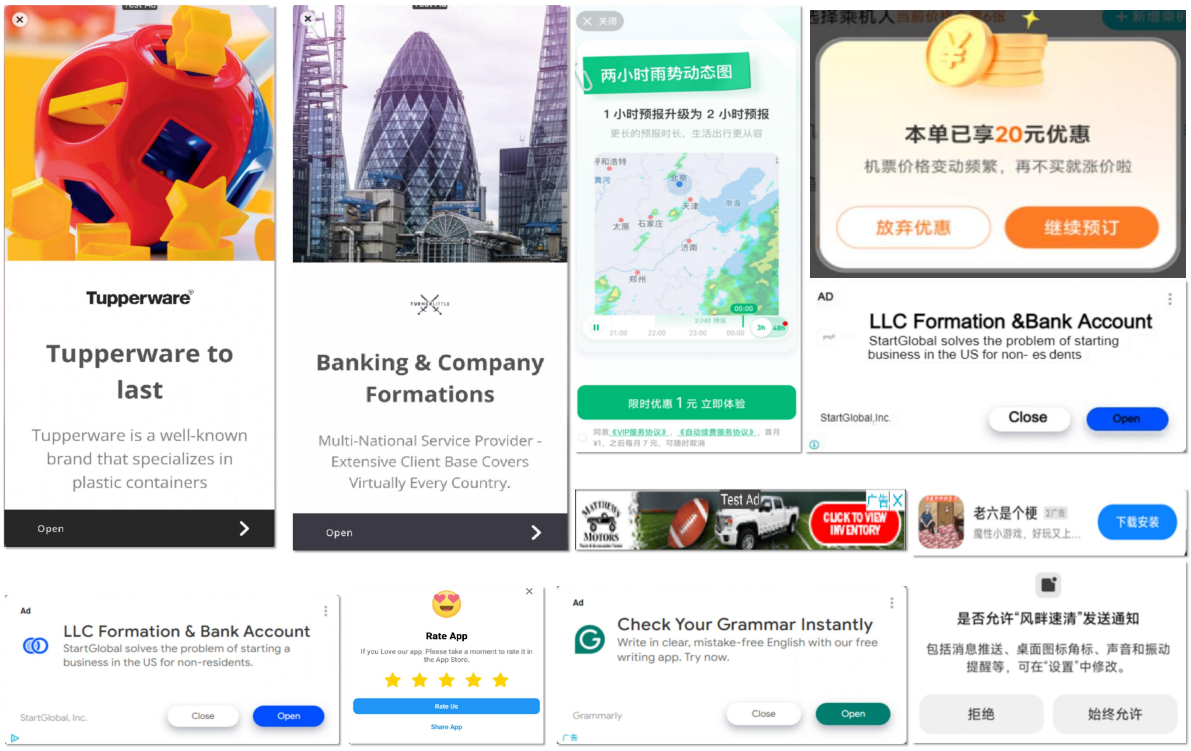


Figure 7: A collection of pop-up ads, collected from Google service official apps, third-party market apps, and mobile apps in mainland China.

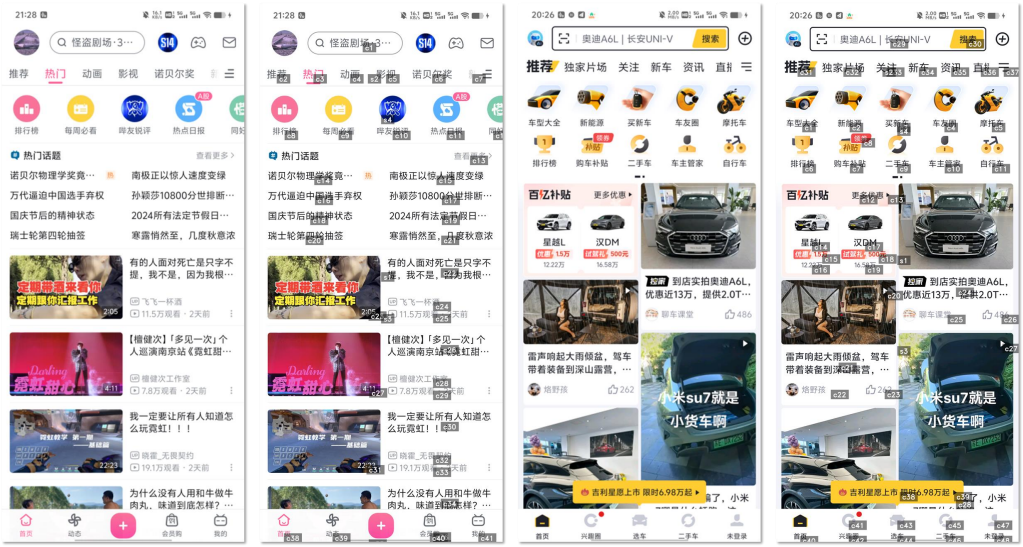


Figure 8: Appagent GUI labeled method.

```
Your task is to describe the functionality of the UI element concisely in one or two sentences. Notice that your
description of the UI element should focus on the general function. For example, if the UI element is used to navigate
to the chat window with John, your description should not include the name of the specific person. Just say:
"Clicking this area will navigate the user to the chat window". Never include the tag of the
UI element in your description. You can use pronouns such as "the UI element" to refer to the element.
```

Listing 2: Click Documentation Template

```
A documentation of this UI element generated from previous demos is shown below. Your
generated description should be based on this previous doc and optimize it. Notice that it is possible that your
understanding of the function of the UI element derived from the given screenshots conflicts with the previous doc,
because the function of a UI element can be flexible. In this case, your generated description should combine both.
Old documentation of this UI element: <old_doc>
```

Listing 3: Refine Documentation Suffix

```
You are an agent that is trained to perform some basic tasks on a smartphone. You will be given a
smartphone screenshot. The interactive clickable UI elements on the screenshot are labeled with tags starting from "c1".
The interactive scrollable UI elements on the screenshot are labeled with tags starting from "s1".The tag of each
interactive element is located in the center of the element. Every screenshot I've given you is a screenshot after
executing the correct action.

You can call the following functions to control the smartphone:

1. click(element: str)
This function is used to click an UI element shown on the smartphone screen.
"element" is a tag assigned to an UI element shown on the smartphone screen.
A simple use case can be click(c5), which taps the UI element labeled with "c5".

2. input(text_input: str)
This function is used to insert text input in an input field/box. text_input is the string you want to insert and must
be wrapped with double quotation marks. A simple use case can be text("Hello, world!"), which inserts the string
"Hello, world!" into the input area on the smartphone screen. This function is usually callable when you see a screenshot
about text inputing.

3. scroll(element: str, direction: str)
This function is used to scroll an UI element shown on the smartphone screen, usually a scroll view or a slide bar.
"element" is a tag assigned to an UI element shown on the smartphone screen. "direction" is a string that
represents one of the four directions: up, down, left, right. "direction" must be wrapped with double quotation
marks.
A simple use case can be swipe(s21, "up"), which scroll up the UI element labeled with "s21".

<ui_document>
The task you need to complete is to <task_description>, to complete this task you should perform current task
<current_task_desc>. Your past actions to proceed with this task are summarized as follows: <last_act>
Now, given the documentation and the following labeled screenshot, you need to think and call the function needed to
proceed with the task. Your output should include three parts in the given format:
Observation: <Describe what you observe in the image>
Thought: <To complete the given task, what is the next step I should do>
Action: <The function call with the correct parameters to proceed with the task.>
Summary: <Summarize your past actions along with your latest action in one or two sentences. Do not include the
tag in your summary>
You can only take one action at a time, so please directly call the function.
```

Instruction: Help me find the current popular audiobook content and browse different audiobook categories.

Step-by-step description:\n1. In the main interface of the music app, click the "Audiobook" option to enter the audiobook area.\n2. In the audiobook page, swipe up to browse different audiobook content.\n3. Continue swiping in the audiobook page to view more audiobook categories and content.\n

Figure 9: Common-Simple test case.

Listing 4: Task Template

C.2 Test Case Study

1. Common-Simple

2. Common-Complex

Figure 10 shows a Common-Complex case of Mobile-Bench-v2 and the GIAS results are as follows:

1. On the “My Gold” page of the mobile app, click the “Category” tab to enter the book category page.
2. On the category page, select the “Plot” category under the “Boys” tab.
3. In the plot category, select the “Return of the Strong” category to enter the list of books in this category.
4. In the “Return of the Strong” category, select the book “The First War God of the North.”
5. On the book details page of “The First War God of the North,” click the rating of 8.1 to view the ratings and reviews.
6. On the review page, click “Must-see masterpiece” to view specific book review details.
7. Enter the comment “Science Fiction” on the book review details page and submit it.

Task: Help me find and evaluate a book called *“The First War God of the North”*, view its ratings and related reviews, and add your own feedback under specific reviews.

3. Noisy Data

4. AITZ-Noisy

5. Ambiguous Data

Instruction: Help me find and evaluate a book called "The First War God of the North", view its ratings and related reviews, and add your own feedback under specific reviews.

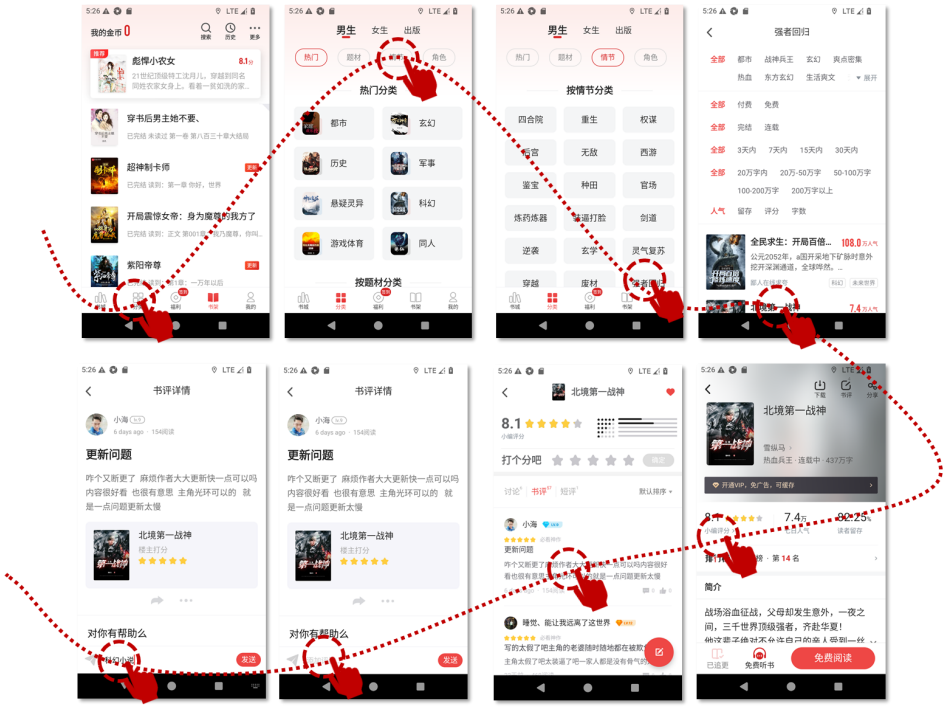


Figure 10: Common-Complex test case.

Instruction: Please help me find Beijing's air quality index for tomorrow on Caiyun Weather.

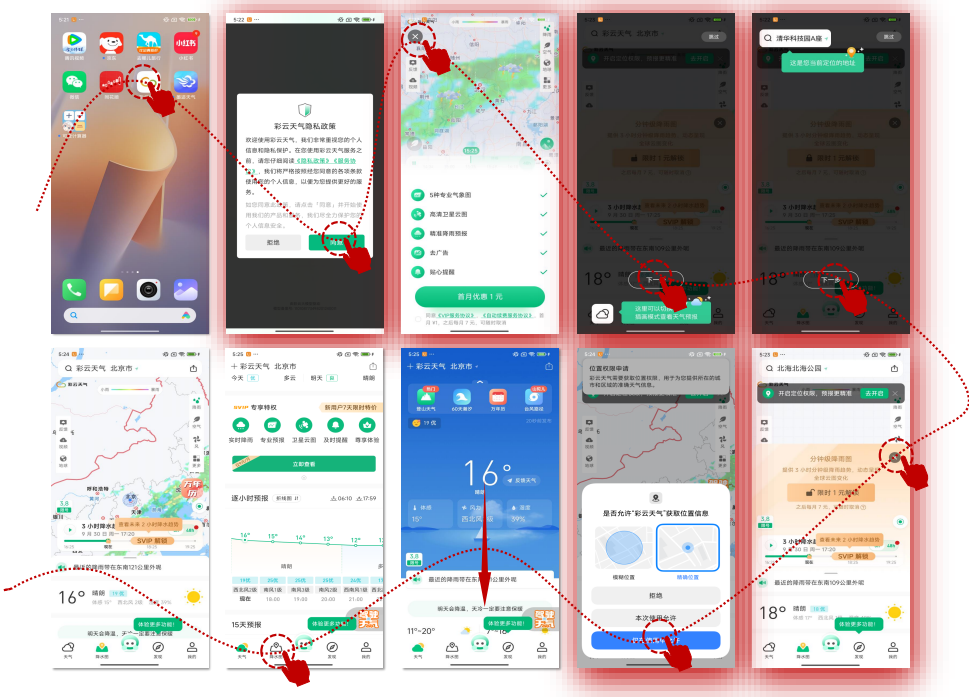


Figure 11: Noisy Data test case. The red shadow in the GUI screenshot are advertisements, pop-ups, or tutorial noise steps.

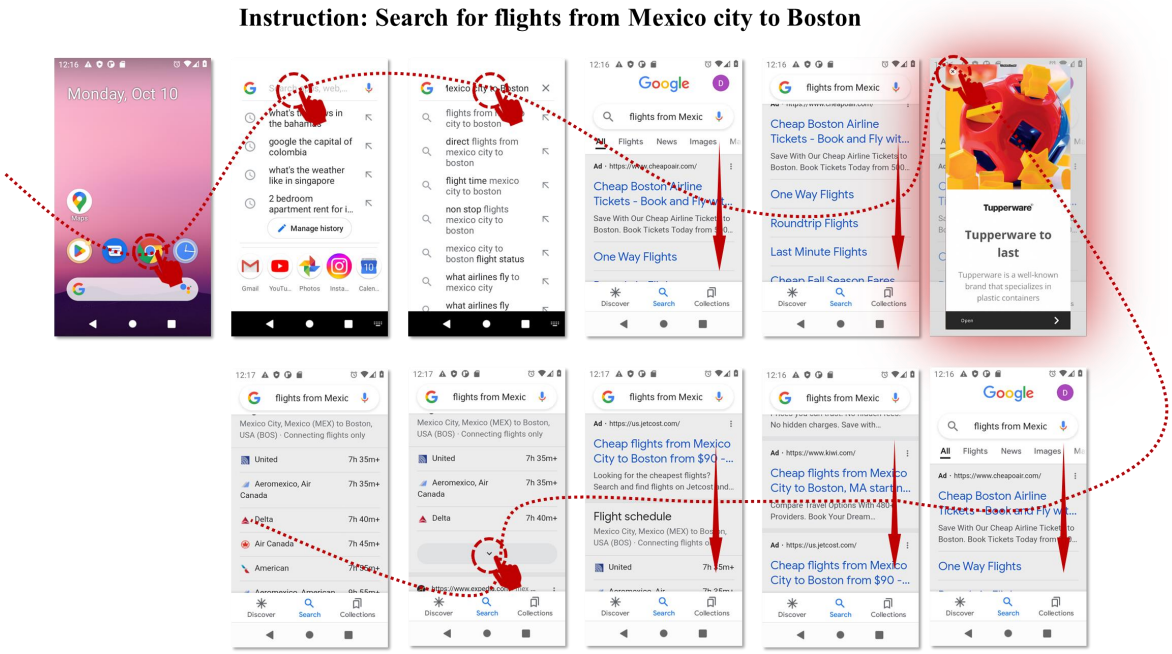


Figure 12: AITZ-Noisy test case. The red shadow GUI screenshot in the trajectory is artificially inserted noise.

Ambiguous Instruction	Find information about a movie
Q: Which app should be used?	A: Use Douban.
Q: Which app does this page belong to?	A: Douban.
Q: In which section should I search?	A: Movies.
Q: Do you want to browse a specific ranking?	A: Yes.
Q: For which time period?	A: Upcoming releases.
Q: How should it be ranked?	A: By popularity.
Q: Which movie do you want to check?	A: The most popular upcoming movie.
Q: What information do you need?	A: A complete summary.
Full Instruction	Find the most popular upcoming movie on Douban
Ambiguous Instruction	Find a midnight snack
Q: Which app should be used?	A: Use Ele.me.
Q: Would you like to filter by specific snack categories, speed,	A: Find new items from the nearest store that can
Q: Any other conditions?	A: deliver within 30 minutes.
Q: Do you have a specific price range?	A: No specific price range.
Q: Do you have a preferred cuisine or taste?	A: No preference, just quick delivery within 30 minutes.
Q: Which section should you search?	A: Food delivery.
Q: What are the speed requirements?	A: Within 30 minutes.
Q: What are the distance requirements?	A: Nearest store.
Full Instruction	Find a new delivery item from the nearest store on Ele.me that can deliver within 30 minutes.

Table 8: Test cases study on Mobile-Bench-Ambiguous.