OpenMobile: Building Open Mobile Agents with Task and Trajectory Synthesis

Kanzhi Cheng[1 2 3 *] **Zehao Li**[4] **Zheng Ma**[2†] **Nuo Chen**[1] **Jialin Cao**[1]
Qiushi Sun[5] **Zichen Ding**[4] **Fangzhi Xu**[3 6] **Hang Yan**[6] **Jiajun Chen**[1]
Luu Anh Tuan[3] **Jianbing Zhang**[1‡] **Lewei Lu**[2‡] **Dahua Lin**[2]

[1]Nanjing University [2]SenseTime [3]Nanyang Technological University
[4]Shanghai AI Laboratory [5]The University of Hong Kong [6]Xi'an Jiaotong University

Abstract

Mobile agents powered by vision-language models have demonstrated impressive capabilities in automating mobile tasks, with recent leading models achieving a marked performance leap, e.g., nearly 70% success on AndroidWorld. However, these systems keep their training data closed and remain opaque about their task and trajectory synthesis recipes. We present **OpenMobile**, an open-source framework that synthesizes high-quality task instructions and agent trajectories, with two key components: (1) The first is a scalable task synthesis pipeline that constructs a global environment memory from exploration, then leverages it to generate diverse and grounded instructions. and (2) a policy-switching strategy for trajectory rollout. By alternating between learner and expert models, it captures essential error-recovery data often missing in standard imitation learning. Agents trained on our data achieve competitive results across three dynamic mobile agent benchmarks: notably, our fine-tuned Qwen2.5-VL and Qwen3-VL reach 51.7% and **64.7% on AndroidWorld**, far surpassing existing open-data approaches. Furthermore, we conduct transparent analyses on the overlap between our synthetic instructions and benchmark test sets, and verify that performance gains stem from broad functionality coverage rather than benchmark overfitting. We release data and code at OpenMobile to bridge the data gap and facilitate broader mobile agent research.

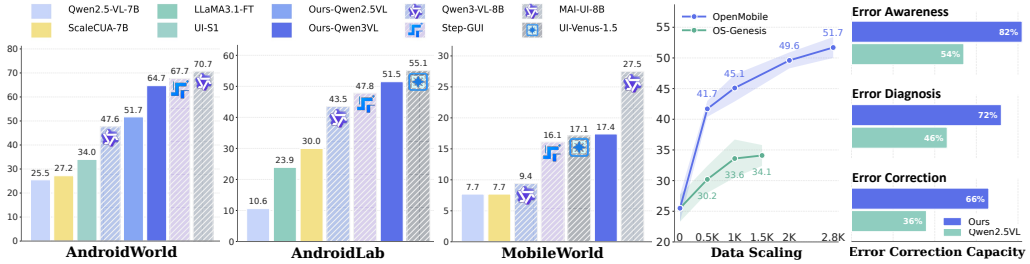

Figure 1: **Performance Comparison.** Task success rates across three dynamic mobile agent benchmarks. Our models significantly surpass open-data baselines and are competitive with leading closed-data systems. **Data Scaling.** AndroidWorld performance with increasing synthesized instructions. **Error Correction Capacity.** OpenMobile data substantially enhances the agent's error-recovery ability in live environments.

1 Introduction

Vision-language models (VLMs) have fueled rapid progress in mobile agents—autonomous systems that interact with smartphone interfaces to complete user tasks. Recent industry efforts such as Step-GUI (Yan et al., 2025), MAI-UI (Zhou et al., 2025), UI-Venus-1.5 (Gao

[*]Work done during internship at SenseTime. [†]Proj leadership. [‡]Corresponding author.

et al., 2026), and MobileAgent-v3.5 (Xu et al., 2026) have pushed the state of the art to striking levels, e.g., approaching 70% task success on the widely-adopted AndroidWorld benchmark (Rawles et al., 2024). These advances are innately driven by large-scale, high-quality agent training data, consisting of task instructions paired with execution trajectories.

However, these leading systems uniformly keep their trajectory data closed and remain opaque about how such data are produced. Meanwhile, the open-source community relies on public datasets like AndroidControl (Li et al., 2024) and AMEX (Chai et al., 2025), achieves only around 30% on the same benchmark (Lu et al., 2025b; Liu et al., 2025). Beyond this widening performance gap, the opacity prevents the community from understanding what data properties drive strong performance and generalization—leaving the open-source community unable to study, reproduce, or build upon these advances.

To bridge this gap, we present OpenMobile, an open data synthesis framework and dataset for mobile agent training. OpenMobile addresses two core challenges: **(1) Generating diverse, high-quality task instructions at scale in dynamic mobile environments.** Existing approaches typically *couple* exploration with generation by using a single trajectory as context for an LLM to task curation (Sun et al., 2025a; Murty et al., 2024). This dependency limits diversity to what a single local trajectory reveals. We instead *decouple* the two stages: first exploring the environment to build a global environment memory of the app's functionalities, then drawing on short-term memory from neighboring screens and long-term memory from semantically related functionalities within the app to compose complex, multi-step instructions. **(2) Collecting agent trajectories that yield effective training signal.** Expert trajectory distillation enables learners to imitate ideal behavior but often fails to address recovery from errors, which leads to a notable performance gap at test time. While self-evolution can mitigate this mismatch, it often suffers from slow convergence and is bounded by the learner's current performance caps. To this end, we further introduce a policy-switching strategy that alternates between the learner and expert models during rollout. We find that error-intervention switching, where a monitor detects deviations to trigger expert corrections, contributes to synthesizing error-recovery demonstrations while maintaining successful task completion.

Using OpenMobile, we synthesize 2.8K task instructions with corresponding 34K action steps across 20 Android apps. We conduct a comprehensive evaluation on three established online benchmarks, i.e., AndroidWorld, AndroidLab (Xu et al., 2025), and MobileWorld (Kong et al., 2025). Notably, our fine-tuned Qwen2.5-VL-7B and Qwen3-VL-8B achieve 51.7% and 64.7% task success on AndroidWorld and improve on the challenging MobileWorld from 9.4% to 17.4%, competitive with leading closed-data solutions and larger foundation models (Bai et al., 2025). Beyond raw performance, we address the community's growing concerns about potential data contamination through transparent experiments, confirming that our performance gains stem from broad functionality coverage and enhanced error-recovery capabilities rather than benchmark overfitting. These findings offer the open-source community a concrete foundation for building competitive mobile agents.

Our contributions are summarized as follows:

- We propose and open-source OpenMobile, a task and trajectory synthesis framework for mobile agents. It introduces decoupled task synthesis that builds global environment memory for instruction generation, and policy-switching trajectory rollout that captures corrective signals absent from expert distillation.
- We conduct comprehensive evaluation on three challenging dynamic benchmarks. Agents trained on our data achieve competitive performance with closed-data systems.
- We provide systematic analyses to examine data contamination risks and demonstrate that our performance gains are driven by broad functionality coverage and enhanced error-recovery capabilities, rather than benchmark overfitting.

2 Related Work

Building autonomous agents for digital automation is a long-standing goal in the AI and NLP community (Branavan et al., 2009; Shi et al., 2017; Shaw et al., 2023). Recent break-

throughs in LLMs have dramatically accelerated progress in this direction, enabling agents to plan and act across mobile, web, and desktop environments and tackle increasingly complex tasks (Rawles et al., 2024; Zhou et al., 2023; Xie et al., 2024; Sun et al., 2025c).

Digital Agents with Vision-Language Models. Early work relied on LLMs to interact with structured interface representations such as accessibility trees (Deng et al., 2023; Gur et al., 2023), or built agentic frameworks that operate computers via coding (Wu et al., 2024; Sun et al., 2024). The rapid progress of vision-language models has driven a shift toward end-to-end, vision-centric GUI agents (Cheng et al., 2024; Gou et al., 2024; Wu et al., 2025b;a). These agents take raw screenshots as input and complete tasks through human-like actions such as clicking and typing. Among these, proprietary systems such as Operator (OpenAI, 2025) and Anthropic's Computer-Use (Anthropic, 2024) stand out, achieving impressive performance by leveraging frontier foundation models (Yang et al., 2026a). Meanwhile, UI-TARS (Qin et al., 2025; Wang et al., 2025) set a milestone for open-weight agents through GUI pretraining, trajectory fine-tuning, and online reinforcement learning. More recently, industry efforts (Yan et al., 2025; Zhou et al., 2025; Gao et al., 2026; Xu et al., 2026) have further pushed mobile agent performance, achieving a 70% task success rate on AndroidWorld. At the core of these advances are large-scale synthesized task instructions and agent trajectories; however, both the data and the underlying synthesis recipes remain undisclosed.

On the other side, the open-source community has fallen increasingly behind. Human-annotated datasets such as AndroidControl (Li et al., 2024) and AMEX (Chai et al., 2025) have provided a foundation (Rawles et al., 2023; Lu et al., 2025a; Yang et al., 2026b; Sun et al., 2025b). However, these datasets contain significant annotation noise and lack rich thinking patterns. Models trained on them, such as ScaleCUA (Liu et al., 2025) and UI-S1 (Lu et al., 2025b), plateau at roughly 30% on AndroidWorld. There is a pressing need for scalable, open-source data synthesis recipes that can close this divide.

GUI Data Synthesis. Human annotation of GUI agent data is costly and labor-intensive, motivating growing interest in automated synthesis of task instructions and action trajectories. Early efforts adopt a *task-driven* paradigm, using strong language models to propose tasks from seed instructions and application descriptions (He et al., 2024; Lai et al., 2024). While straightforward, this approach lacks grounding in real-world context, often producing generic, underspecified, or infeasible instructions. This has motivated *interaction-driven* methods that first explore the target environment and then synthesize environment-grounded instructions from the observed context. A representative work is OS-Genesis (Sun et al., 2025a), which proposes reverse task synthesis, using random-walk trajectories to retrospectively infer meaningful task instructions. NNetNav (Murty et al., 2024) efficiently constructs complex web demonstrations through the synergy of an exploration strategy and a pruning labeler that filters low-quality trajectories. Subsequent efforts further advance this paradigm with more structured exploration strategies (Yang et al.; Gandhi & Neubig, 2025; Jiang et al., 2026) and more sophisticated instruction generation pipelines (Xie et al., 2025; Pahuja et al., 2025; Ramrakhya et al., 2025). Across these methods, exploration and instruction generation remain tightly *coupled*: each instruction is derived from a single exploration trajectory, which bounds diversity to local observations.

Once task instructions are available, the next step is to collect high-quality agent trajectories. A prevalent approach is expert distillation, where a strong agent model rolls out trajectories and a verifier model filters them for quality (Pan et al., 2024; Sun et al., 2025a; Lin et al., 2025). An alternative line explores self-evolution, where the agent iteratively executes tasks and retrains on its own successful trajectories to bootstrap performance (He et al., 2025; Qin et al., 2025). Recent efforts further co-generate verifiable evaluation scripts alongside instructions to facilitate RL training (Xue et al., 2026).

3 OpenMobile

The widening performance gap between proprietary and open-source mobile agents stems fundamentally from the lack of large-scale, high-quality open training data. To bridge this

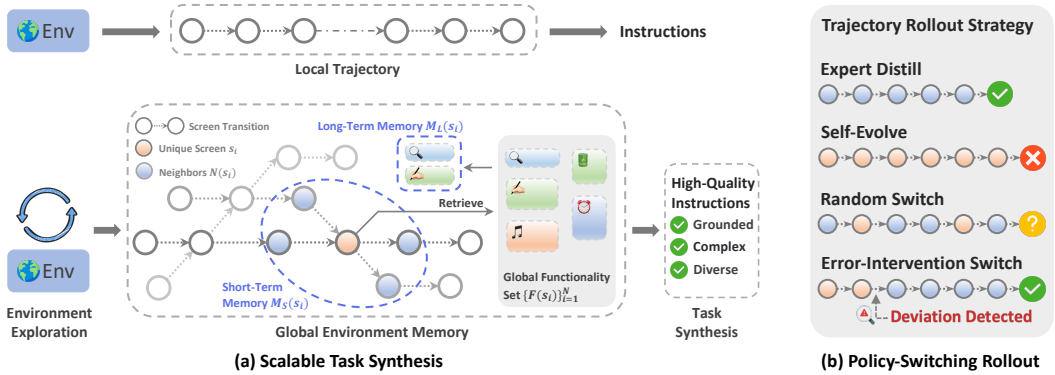

Figure 2: The overview of OpenMobile. (a) Scalable Task Synthesis. Instead of relying on a single local trajectory, we first explore the environment to build a global memory. By retrieving short- and long-term memories, we synthesize diverse, complex instructions that are contextually grounded and can be reasonably executed within the online environment. (b) Policy-Switching Rollout. Among different rollout strategies, error-intervention switching best captures error-recovery signals while ensuring task completion by detecting learner deviation and triggering expert correction.

divide, OpenMobile provides a data synthesis framework that produces two complementary assets: diverse, grounded task instructions over the broad functionalities of mobile environments (Section 3.1), and agent execution trajectories enriched with error-recovery signals that facilitate efficient agent training (Section 3.2). The overview of our methods is in Figure 2. Implementation details are provided in Appendix A.

3.1 Scalable Task Synthesis

Existing interaction-driven approaches tightly couple exploration with generation, deriving task instructions from individual exploration trajectories, which bounds instruction diversity to what one local trajectory reveals. We instead propose a decoupled paradigm inspired by how humans learn a new application: one first explores to build a structured, comprehensive understanding of the app's capabilities, then recalls and composes relevant functionalities when a complex need arises. Our pipeline mirrors this process in three stages: (i) exploring the environment to collect interaction experience, (ii) organizing the exploration data into a global environment memory $\mathcal{M}$, and (iii) drawing on short-term and long-term memory from $\mathcal{M}$ to synthesize compositional task instructions.

Environment Exploration The first stage aims to traverse target applications to collect information about the environment. Through successive interactions with the app, this stage yields a set of exploration trajectories—sequences of screen-action interactions that capture the transitions between different states. Our framework is agnostic to the specific exploration strategy and accommodates random walk (Sun et al., 2025a), structured coverage-based methods (Gandhi & Neubig, 2025; Ramrakhya et al., 2025; Shao et al., 2026), or human demonstrations. To demonstrate that our approach does not rely on sophisticated exploration heuristics, we adopt a simple random walk in this work. This choice is motivated by our observation that the critical factor lies not in exploration efficiency, but in how effectively the collected data is organized and used downstream.

Global Environment Memory Construction Multiple exploration sessions inevitably visit the same screen or environment state across different trajectories. We exploit these shared screens as natural anchoring points to weave all fragmented trajectories into a unified, interconnected structure. Specifically, we apply perceptual hashing to cluster visually similar screens, identifying a set of N unique screens $\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$, and aggregate transitions from all trajectories to form a neighborhood relation: for each screen s_i, its neighbors $\mathcal{N}(s_i) \subset \mathcal{S}$ are the screens directly reachable from or leading to s_i. We then enrich

each screen with its functionality set $\mathcal{F}(s_i) = \{f_1, f_2, \ldots, f_K\}$, extracted by a strong vision-language model, where each f_k is a natural-language description capturing the semantics of a UI element on the screen. To support cross-screen association, we compute semantic embeddings for all functionalities and build a per-app retrieval index. The resulting global environment memory

$$\mathcal{M} = \big(\, \mathcal{S},\ \mathcal{N},\ \{\mathcal{F}(s_i)\}_{i=1}^{N} \,\big)$$

captures each application's functionality landscape in a structured, queryable form.

Memory-Augmented Task Synthesis Given the global memory $\mathcal{M}$, task synthesis proceeds by associating and composing functionalities into coherent instructions. For each candidate screen s_i, we construct a context $\mathcal{C}(s_i)$ comprising three complementary views: (1) the recalled screen s_i itself, including its screenshot and annotated functionalities $\mathcal{F}(s_i)$, which serves as the focal point for generation; (2) short-term memory $\mathcal{M}_{\mathcal{S}}(s_i)$, the functionalities of neighboring screens $\mathcal{N}(s_i)$, reflecting what is locally reachable and naturally chainable with the current screen's capabilities; and (3) long-term memory $\mathcal{M}_{\mathcal{L}}(s_i)$, semantically related functionalities retrieved from distant screens within the same application, surfacing features that a user might associate through experience but that no single trajectory would reveal, encouraging cross-feature composition.

The full context $\mathcal{C}(s_i) = \big(s_i,\ \mathcal{M}_{\mathcal{S}}(s_i),\ \mathcal{M}_{\mathcal{L}}(s_i)\big)$, comprising both screenshots and textual descriptions, is presented to a VLM to generate task instructions based on this context. We also carefully design generation guidelines and in-context examples to steer the model toward producing high-quality instructions; the full prompt is provided in the Section A.5. The generated instructions undergo model-based quality filtering and embedding-based deduplication to yield the final instruction set.

3.2 Policy-Switching Rollout

With task instructions in hand, the next step is to collect agent trajectories for training. A straightforward approach is expert distillation: rolling out a strong model to produce demonstration trajectories for imitation learning. Although this yields high-quality trajectories, it restricts the learner to ideal behavior and fails to expose it to the mistakes it might make during inference, leaving the agent unable to recover from its own errors. An alternative is self-evolution, where the learner iteratively executes tasks and retrains on its own successful trajectories. This directly addresses the distribution mismatch but converges slowly, as the learner's improvement is bounded by its own capacity.

We propose policy-switching rollout, which combines the strengths of both paradigms. Given a task instruction I, an expert policy π_e, and a learner policy π_l, the rollout proceeds step by step: at each time step t, the agent observes the current screen o_t and selects an action a_t according to one of the two policies, controlled by a switching variable $z_t \in \{e, l\}$:

$$a_t \sim \pi_{z_t}(\cdot \mid I, o_t, h_t)$$

where h_t denotes the interaction history up to step t. By alternating between π_e and π_l, the resulting trajectories contain segments where the learner makes mistakes followed by expert corrections. This produces error-recovery experiences that are absent from pure distillation, while the expert's presence avoids the capacity ceiling inherent in self-evolution.

Switching Strategies. The design of the switching rule, i.e., how z_t is determined at each step, is critical. A natural choice is random switching, which lets the learner take over with a fixed probability p whenever the two policies disagree on the next action. However, mobile agent tasks are inherently multi-solution: multiple valid action sequences can lead to the same goal, so disagreement between π_e and π_l does not necessarily signal a learner mistake, making random switching a noisy proxy for identifying errors. Moreover, frequent switching between policies tends to disrupt coherent progress on complex tasks, resulting in fragmented trajectories that offer little usable training signal.

To address this, we design an error-intervention strategy. Instead of switching at every point of disagreement, we introduce a monitor $\mathcal{O}$ that tracks the learner's execution in real time. The rollout begins with the learner policy ($z_t = l$); only when $\mathcal{O}$ detects that the

learner has deviated from productive progress does it trigger a switch to the expert ($z_t = e$) to intervene and correct the trajectory back on track. The resulting trajectories are thus enriched with scarce error-recovery experiences, while expert intervention ensures sufficient task completion for effective training. We compare these switching strategies in Section 5.1.

3.3 The OpenMobile Dataset

We instantiate the above pipeline on the Android emulator provided by AndroidWorld (Rawles et al., 2024). Although we leverage its environment infrastructure, we do *not* incorporate any benchmark test instructions during synthesis to prevent data leakage; a detailed overlap analysis is provided in Section 5.2. For policy-switching rollout, an early-stage fine-tuned checkpoint serves as the learner π_l and `Gemini-3.1-Pro-Preview` as the expert π_e. We adopt the action space and response format of Qwen3-VL (Bai et al., 2025), and employ the expert model to rewrite each step's chain-of-thought reasoning for higher-quality supervision. The resulting dataset contains approximately 2,800 instructions and 34K corresponding action steps across 20 Android apps, with an average trajectory length of 12.2 steps and 129-word chain-of-thought reasoning per step.

4 Experiments

In this section, we train models on OpenMobile data and present main results on established dynamic benchmarks. We leave ablation studies and further analyses to the next section.

4.1 Experimental Settings

From the policy-switching rollout trajectories, we retain only expert steps for training while preserving the full interaction history, including learner errors, as context to expose the model to realistic error-recovery scenarios. We fine-tune two base models: Qwen2.5-VL-7B and Qwen3-VL-8B (Bai et al., 2025). The former has not undergone heavy GUI-specific post-training, offering a cleaner testbed for isolating data-driven gains; the latter serves as a stronger foundation to validate effectiveness on a more capable base and push the performance ceiling. All models are trained with LLaMA-Factory (Zheng et al., 2024) using standard supervised fine-tuning with a batch size of 32, a learning rate of 1e-5, for 3 epochs.

We also experiment with reinforcement learning (RL), including step-level RL (Lu et al., 2026) and trajectory-level agentic RL (Li et al., 2026). While the results prove that our synthesized trajectories remain effective, they do not yield significant improvements over SFT on dynamic benchmarks. We discuss these findings in Appendix C.

4.2 Evaluation Benchmarks

We evaluate on three established dynamic mobile agent benchmarks. We focus on dynamic benchmarks because static datasets such as AndroidControl fundamentally lack the ability to evaluate an agent's crucial error-recovery capability, and their annotation noise further undermines evaluation reliability, making them poor proxies for real-world agent performance (Lu et al., 2025b; Gao et al., 2026). Experimental details are provided in Section B.1.

AndroidWorld (Rawles et al., 2024) is the dominant mobile agent evaluation benchmark. It provides robust, reproducible environments and deterministic evaluation through Android emulators. It comprises 116 tasks across 20 real-world apps, with parameterized task templates that generate diverse variants via random seeds.

AndroidLab (Xu et al., 2025) is a systematic benchmark for mobile agents with reproducible evaluation. It covers 138 tasks across 9 apps on predefined Android virtual devices, and supports both language-only and multimodal agents.

MobileWorld (Kong et al., 2025) is a recent and more challenging dynamic benchmark. It contains 201 tasks across 20 apps, with a stronger focus on long-horizon and cross-app workflows, making it substantially harder than AndroidWorld for evaluating complex mobile agent capabilities. We evaluate on its GUI-only subset.

Method	Base Model	AndroidWorld		AndroidLab		MobileWorld	
		Pass@1↑	Pass@3↑	Pass@1↑	Pass@3↑	Pass@1↑	Pass@3↑
Commercial Models							
GPT-4o	–	30.6	–	31.2	–	–	–
Gemini-3-Pro	–	60.3	75.0	–	–	51.3	–
Open-Weight Models							
Qwen2.5-VL-7B	–	25.5 ± 2.6	34.9	10.6 ± 1.8	15.2	7.7 ± 0.9	10.3
Qwen3-VL-8B	–	47.6 ± 2.2	62.1	43.5	–	9.4	–
UI-Venus-7B	Qwen2.5-VL	49.1	–	41.3	–	8.5	–
Step-GUI-4B	Qwen3-VL	63.9	75.8	47.8	–	16.1	–
Step-GUI-8B	Qwen3-VL	67.7	80.2	–	–	–	–
MAI-UI-8B	Qwen3-VL	70.7	–	–	–	27.5	–
UI-Venus-1.5-8B	Qwen3-VL	73.7	–	55.1	–	17.1	–
MobileAgent-v3.5-8B	Qwen3-VL	71.6	–	–	–	33.3	–
Open-Data Models							
UI-S1-7B	Qwen2.5-VL	34.0	–	–	–	–	–
ScaleCUA-7B	Qwen2.5-VL	27.2 ± 2.2	36.2	30.0 ± 1.1	37.7	7.7 ± 0.4	8.6
Ours-7B	Qwen2.5-VL	51.7 ± 1.7	68.1	22.7 ± 0.4	37.0	14.8 ± 1.3	21.4
Ours-8B	Qwen3-VL	64.7 ± 3.2	78.0	51.5 ± 0.7	62.3	17.7 ± 2.2	24.8

Table 1: Main results on AndroidWorld, AndroidLab, and MobileWorld. We report Pass@1 and Pass@3, where higher values indicate better performance. OpenMobile outperforms open-data baselines by a large margin and rivals leading closed-data systems.

4.3 Main Results

OpenMobile data substantially improves mobile agent performance with strong generalization. As shown in Table 1, models fine-tuned on OpenMobile data significantly outperform corresponding baselines across all three benchmarks. The Qwen2.5-VL variant improves by over 25 absolute points on AndroidWorld, demonstrating the effectiveness of our synthesized trajectories in enhancing VLM agentic capabilities. Notably, although OpenMobile data is collected within the AndroidWorld environment, the resulting models generalize well to unseen settings, including novel apps in AndroidLab and long-horizon cross-app tasks in MobileWorld, e.g., achieving over 50% relative improvement on the latter. Overall, our models substantially surpass existing open-data approaches and are competitive with leading industrial efforts. These results highlight the potential of open data synthesis for building competitive mobile agents.

Base model capability remains critical. Despite training on the same data, the Qwen3-VL variant consistently outperforms Qwen2.5-VL variant by a clear margin, suggesting that inherent base model capability, e.g., GUI understanding and planning, play an indispensable role. While high-quality trajectory data can narrow the gap, improving the underlying foundation model remains equally important for pushing the performance ceiling. Additional experiments with larger models and comparisons with other methods are in Appendix D.

5 Analysis

In this section, we first ablate the key design choices behind OpenMobile (Section 5.1), and then investigate what drives its effectiveness (Section 5.2), including the examination of potential benchmark overfitting.

5.1 Ablation Study

OpenMobile produces diverse and high-quality instructions. We compare our decoupled task synthesis with OS-Genesis (Sun et al., 2025a) and a coupled baseline. The coupled baseline shares the same generation prompt as ours, but uses the screenshot sequence

OpenMobile vs.	Complexity	Soundness
OS-Genesis	0.68 / 0.22 / 0.10	0.44 / 0.48 / 0.08
Coupled Pipeline	0.26 / 0.62 / 0.12	0.06 / 0.90 / 0.04

(a) Human evaluation (win / tie / loss).

Method	Pass@1↑
OS-Genesis	34.1 ± 1.7
Coupled Pipeline	45.3 ± 2.2
OpenMobile	**48.3 ± 1.3**

(b) Task success rate.

Table 2: Ablation on task synthesis strategies. (a) Human evaluation of instruction quality over 50 pairwise comparisons. (b) AndroidWorld success rate with 1.5K trajectories.

from a single exploration trajectory as context instead of the global environment memory. We first sample 50 instructions from each method for pairwise human evaluation, where experienced annotators judge instruction quality in terms of complexity and soundness and select the better one or declare a tie. As shown in Table 2a, instructions synthesized by OpenMobile are notably more challenging than those from both baselines while maintaining comparable soundness. We further train models on the synthesized data and evaluate on AndroidWorld. Results in Table 2b show that OpenMobile achieves the best performance under a fixed budget of 1.5K trajectories.

Policy-switching rollout enriches error-recovery signals and boosts test-time performance. We compare four trajectory rollout strategies: (i) expert distillation, which collects trajectories using only the expert model; (ii) self-evolution, where the learner iteratively retrains on its own successful trajectories over 3 rounds; (iii) random switching, which alternates between the expert and learner randomly as described in Section 3.2; and (iv) our error-intervention switching. Detailed experimental settings are provided in Section B.2.

As shown in Table 3, error-intervention switching achieves the best downstream performance by introducing richer error-recovery signals during trajectory rollout. Furthermore, as shown in the right panel of Figure 1, we compare the error-recovery behaviors of our trained model and the base model during live execution, including error awareness, diagnosis, and correction. The results confirm that OpenMobile data substantially strengthens the agent's error-recovery capability, which in turn drives the downstream performance gains.

Rollout Strategy	Avg. ER	Pass@1↑
Expert Distillation	0.42	44.8 ± 1.7
Self-Evolution	0.10	33.8 ± 0.9
Random Switch	0.64	45.1 ± 0.9
Error-Intervention Switch	**1.56**	**48.3 ± 1.3**

Table 3: Ablation on rollout strategies. Avg. ER is the average number of error-recovery instances per trajectory, manually counted over 50 randomly sampled trajectories.

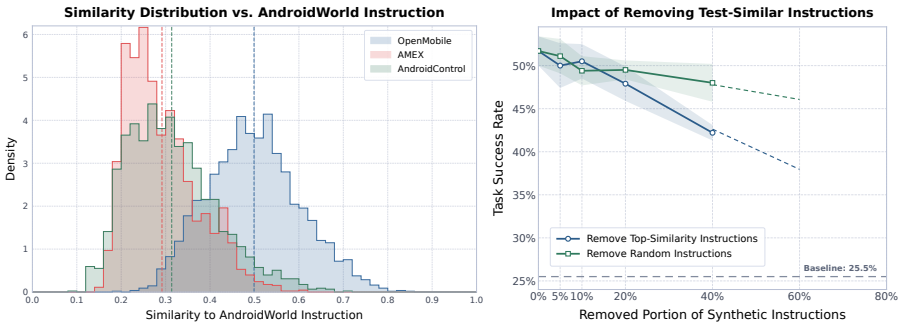

Figure 3: **Left:** Semantic similarity between synthetic and AndroidWorld instructions. Our synthesized instructions exhibit moderate functionality-level relevance, with only 3.5% exceeding a similarity of 0.7. **Right:** Impact of removing test-similar instructions from training. Removing a small fraction of the most similar instructions causes only a marginal performance drop, mitigating benchmark overfitting concerns.

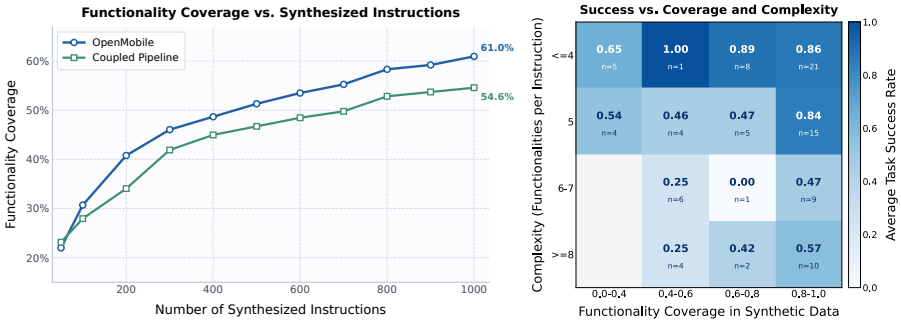

Figure 4: **Left:** Functionality coverage of AndroidWorld tasks as synthesized instructions scale. OpenMobile consistently achieves higher coverage than the coupled baseline. **Right:** Tasks with lower complexity (fewer required functionalities) and higher functionality coverage by synthetic data achieve higher success rates.

5.2 What Drives the Effectiveness of OpenMobile Data?

OpenMobile data is grounded in the benchmark environment, but does not overfit its test instructions. Since our data is synthesized within the AndroidWorld environment, a natural concern is whether the instructions simply replicate benchmark tests. To investigate, we compute semantic similarity between synthetic instructions and AndroidWorld test instructions using `openai/text-embedding-3-large`, and compare with AndroidControl and AMEX. As shown in Figure 3 (left), OpenMobile instructions are indeed more similar to the test set, which is expected given the shared environment and app suite. However, only 3.5% of our instructions exceed a similarity of 0.7, indicating moderate relevance rather than task-level duplication and alleviating concerns about data leakage, e.g., through rephrasing test instructions. The full list of most similar pairs is in Appendix E.

Furthermore, we experiment with removing the most test-similar synthetic instructions and compare with random removal to observe the downstream performance impact. As shown in Figure 3 (right), removing a small fraction, e.g., 10%, leads to only a marginal drop, indicating that our gains do not fragily depend on a few test-similar samples. However, as the removal ratio increases to 40%, performance degrades notably compared to random removal. This is because removing the most similar instructions inevitably strips away core app functionalities from the training data, preventing the model from acquiring essential skills. We analyze the role of functionality coverage in detail in the following section.

Broad functionality coverage drives agent performance. To understand what makes OpenMobile data effective, we conduct a functionality coverage analysis. We use an LLM to decompose each test task into required atomic functionalities. For instance, "Create a calendar event titled Meeting with Team for tomorrow at 10am" yields *create calendar event*, *set date*, *set title*, and *set start time*. We then measure what fraction of test-required functionalities are covered by synthesized instructions. As shown in Figure 4 (left), coverage steadily increases with instruction count, and OpenMobile consistently outperforms the coupled pipeline. This confirms the advantage of our decoupled design: the global memory provides a structured view of the environment's capabilities, while retrieving semantically related functionalities as long-term context encourages cross-feature composition, together driving broader and more diverse instruction synthesis.

We further examine how task complexity, i.e., the number of atomic functionalities per task, and functionality coverage jointly affect success rate. As shown in Figure 4 (right), tasks involving more functionalities are harder to complete (colors lighten from top to bottom), while tasks with higher coverage achieve higher success rates (colors deepen from left to right). This highlights the importance of functionality coverage for instruction synthesis: an effective synthesis method should maximize coverage of the environment's core functionalities, which is precisely the design principle behind OpenMobile.

6 Conclusion

We presented OpenMobile, an open data synthesis framework for building competitive mobile agents. Our framework addresses two key aspects of trajectory synthesis: (1) decoupling exploration from instruction generation to produce diverse, high-quality tasks, and (2) policy-switching rollout to enrich trajectories with error-recovery signals. Agents trained on OpenMobile data achieve strong performance and generalize well to unseen dynamic environments, substantially narrowing the gap with closed-data industrial systems. We also conduct transparent analyses on the overlap between synthetic and test instructions, confirming that these gains stem from broad functionality coverage and enhanced error-recovery capability rather than benchmark overfitting. We release all data and code with the hope that OpenMobile serves as a foundation for broader open mobile agent research.

References

Anthropic. Introducing computer use, 2024. URL https://www.anthropic.com/news/3-5-models-and-computer-use.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Satchuthananthavale RK Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 82–90, 2009.

Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Guozhi Wang, Dingyu Zhang, Shuai Ren, and Hongsheng Li. Amex: Android multi-annotation expo dataset for mobile gui agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 2138–2156, 2025.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9313–9332, 2024.

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

Apurva Gandhi and Graham Neubig. Go-browse: Training web agents with structured exploration. *arXiv preprint arXiv:2506.03533*, 2025.

Changlong Gao, Zhangxuan Gu, Yulin Liu, Xinyu Qiu, Shuheng Shen, Yue Wen, Tianyu Xia, Zhenyu Xu, Zhengwen Zeng, Beitong Zhou, et al. Ui-venus-1.5 technical report. *arXiv preprint arXiv:2602.09082*, 2026.

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

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

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. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6864–6890, 2024.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Hongming Zhang, Tianqing Fang, Zhenzhong Lan, and Dong Yu. Openwebvoyager: Building multimodal web agents via iterative real-world exploration, feedback and optimization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27545–27564, 2025.

Deyang Jiang, Jing Huang, Xuanle Zhao, Lei Chen, Liming Zheng, Fanfan Liu, Haibo Qiu, Peng Shi, and Zhixiong Zeng. Treecua: Efficiently scaling gui automation with tree-structured verifiable evolution. *arXiv preprint arXiv:2602.09662*, 2026.

Linjia Kang, Zhimin Wang, Yongkang Zhang, Duo Wu, Jinghe Wang, Ming Ma, Haopeng Yan, and Zhi Wang. Learning with challenges: Adaptive difficulty-aware data generation for mobile gui agent training. *arXiv preprint arXiv:2601.22781*, 2026.

Quyu Kong, Xu Zhang, Zhenyu Yang, Nolan Gao, Chen Liu, Panrong Tong, Chenglin Cai, Hanzhang Zhou, Jianan Zhang, Liangyu Chen, et al. Mobileworld: Benchmarking autonomous mobile agents in agent-user interactive and mcp-augmented environments. *arXiv preprint arXiv:2512.19432*, 2025.

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

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. *Advances in Neural Information Processing Systems*, 37:92130–92154, 2024.

Zehao Li, Zhenyu Wu, Yibo Zhao, Bowen Yang, Jingjing Xie, Zhaoyang Liu, Zhoumianze Liu, Kaiming Jin, Jianze Liang, Zonglin Li, et al. Os-themis: A scalable critic framework for generalist gui rewards. *arXiv preprint arXiv:2603.19191*, 2026.

Haojia Lin, Xiaoyu Tan, Yulei Qin, Zihan Xu, Yuchen Shi, Zongyi Li, Gang Li, Shaofei Cai, Siqi Cai, Chaoyou Fu, et al. Cuarewardbench: A benchmark for evaluating reward models on computer-using agent. *arXiv preprint arXiv:2510.18596*, 2025.

Zhaoyang Liu, JingJing Xie, Zichen Ding, Zehao Li, Bowen Yang, Zhenyu Wu, Xuehui Wang, Qiushi Sun, Shi Liu, Weiyun Wang, et al. Scalecua: Scaling open-source computer use agents with cross-platform data. *arXiv preprint arXiv:2509.15221*, 2025.

Quanfeng Lu, Wenqi Shao, Zitao Liu, Lingxiao Du, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, and Ping Luo. Guiodyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22404–22414, 2025a.

Zhengxi Lu, Jiabo Ye, Fei Tang, Yongliang Shen, Haiyang Xu, Ziwei Zheng, Weiming Lu, Ming Yan, Fei Huang, Jun Xiao, et al. Ui-s1: Advancing gui automation via semi-online reinforcement learning. *arXiv preprint arXiv:2509.11543*, 2025b.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Pengxiang Zhao, Guangyi Liu, et al. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 17608–17616, 2026.

Shikhar Murty, Hao Zhu, Dzmitry Bahdanau, and Christopher D Manning. Nnetnav: Unsupervised learning of browser agents through environment interaction in the wild. *arXiv preprint arXiv:2410.02907*, 2024.

OpenAI. Introducing operator, 2025. URL https://openai.com/index/introducing-operator/.

Vardaan Pahuja, Yadong Lu, Corby Rosset, Boyu Gou, Arindam Mitra, Spencer Whitehead, Yu Su, and Ahmed Hassan. Explorer: Scaling exploration-driven web trajectory synthesis for multimodal web agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 6300–6323, 2025.

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

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025.

Ram Ramrakhya, Andrew Szot, Omar Attia, Yuhao Yang, Anh Nguyen, Bogdan Mazoure, Zhe Gan, Harsh Agrawal, and Alexander Toshev. Scaling synthetic task generation for agents via exploration. *arXiv preprint arXiv:2509.25047*, 2025.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Androidinthewild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems*, 36:59708–59728, 2023.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. Androidworld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*, 2024.

Rui Shao, Ruize Gao, Bin Xie, Yixing Li, Kaiwen Zhou, Shuai Wang, Weili Guan, and Gongwei Chen. Hats: Hardness-aware trajectory synthesis for gui agents. *arXiv preprint arXiv:2603.12138*, 2026.

Peter Shaw, Mandar Joshi, James Cohan, Jonathan Berant, Panupong Pasupat, Hexiang Hu, Urvashi Khandelwal, Kenton Lee, and Kristina N Toutanova. From pixels to ui actions: Learning to follow instructions via graphical user interfaces. *Advances in Neural Information Processing Systems*, 36:34354–34370, 2023.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning*, pp. 3135–3144. PMLR, 2017.

Qiushi Sun, Zhirui Chen, Fangzhi Xu, Kanzhi Cheng, Chang Ma, Zhangyue Yin, Jianing Wang, Chengcheng Han, Renyu Zhu, Shuai Yuan, et al. A survey of neural code intelligence: Paradigms, advances and beyond. *arXiv preprint arXiv:2403.14734*, 2024.

Qiushi Sun, Kanzhi Cheng, Zichen Ding, Chuanyang Jin, Yian Wang, Fangzhi Xu, Zhenyu Wu, Chengyou Jia, Liheng Chen, Zhoumianze Liu, et al. Os-genesis: Automating gui agent trajectory construction via reverse task synthesis. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5555–5579, 2025a.

Qiushi Sun, Mukai Li, Zhoumianze Liu, Zhihui Xie, Fangzhi Xu, Zhangyue Yin, Kanzhi Cheng, Zehao Li, Zichen Ding, Qi Liu, et al. Os-sentinel: Towards safety-enhanced mobile gui agents via hybrid validation in realistic workflows. *arXiv preprint arXiv:2510.24411*, 2025b.

Qiushi Sun, Zhoumianze Liu, Chang Ma, Zichen Ding, Fangzhi Xu, Zhangyue Yin, Haiteng Zhao, Zhenyu Wu, Kanzhi Cheng, Zhaoyang Liu, et al. Scienceboard: Evaluating multimodal autonomous agents in realistic scientific workflows. *arXiv preprint arXiv:2505.19897*, 2025c.

Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, et al. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. *arXiv preprint arXiv:2509.02544*, 2025.

Qianhui Wu, Kanzhi Cheng, Rui Yang, Chaoyun Zhang, Jianwei Yang, Huiqiang Jiang, Jian Mu, Baolin Peng, Bo Qiao, Reuben Tan, et al. Gui-actor: Coordinate-free visual grounding for gui agents. *arXiv preprint arXiv:2506.03143*, 2025a.

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 preprint arXiv:2402.07456*, 2024.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and Yu Qiao. OS-ATLAS: Foundation action model for generalist GUI agents. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=n9PDaFNi8t.

Jingxu Xie, Dylan Xu, Xuandong Zhao, and Dawn Song. Agentsynth: Scalable task generation for generalist computer-use agents. *arXiv preprint arXiv:2506.14205*, 2025.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Haiyang Xu, Xi Zhang, Haowei Liu, Junyang Wang, Zhaozai Zhu, Shengjie Zhou, Xuhao Hu, Feiyu Gao, Junjie Cao, Zihua Wang, et al. Mobile-agent-v3. 5: Multi-platform fundamental gui agents. *arXiv preprint arXiv:2602.16855*, 2026.

Yifan Xu, Xiao Liu, Xueqiao Sun, Siyi Cheng, Hao Yu, Hanyu Lai, Shudan Zhang, Dan Zhang, Jie Tang, and Yuxiao Dong. Androidlab: Training and systematic benchmarking of android autonomous agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2144–2166, 2025.

Taofeng Xue, Chong Peng, Mianqiu Huang, Linsen Guo, Tiancheng Han, Haozhe Wang, Jianing Wang, Xiaocheng Zhang, Xin Yang, Dengchang Zhao, et al. Evocua: Evolving computer use agents via learning from scalable synthetic experience. *arXiv preprint arXiv:2601.15876*, 2026.

Haolong Yan, Jia Wang, Xin Huang, Yeqing Shen, Ziyang Meng, Zhimin Fan, Kaijun Tan, Jin Gao, Lieyu Shi, Mi Yang, et al. Step-gui technical report. *arXiv preprint arXiv:2512.15431*, 2025.

Bowen Yang, Kaiming Jin, Zhenyu Wu, Zhaoyang Liu, Qiushi Sun, Zehao Li, JingJing Xie, Zhoumianze Liu, Fangzhi Xu, Kanzhi Cheng, et al. Os-symphony: A holistic framework for robust and generalist computer-using agent. *arXiv preprint arXiv:2601.07779*, 2026a.

Qianlan Yang, Xiangjun Wang, Danielle Perszyk, and Yu-Xiong Wang. Self-guided hierarchical exploration for generalist foundation model web agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Rui Yang, Qianhui Wu, Zhaoyang Wang, Hanyang Chen, Ke Yang, Hao Cheng, Huaxiu Yao, Baoling Peng, Huan Zhang, Jianfeng Gao, et al. Gui-libra: Training native gui agents to reason and act with action-aware supervision and partially verifiable rl. *arXiv preprint arXiv:2602.22190*, 2026b.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 3: system demonstrations)*, pp. 400–410, 2024.

Hanzhang Zhou, Xu Zhang, Panrong Tong, Jianan Zhang, Liangyu Chen, Quyu Kong, Chenglin Cai, Chen Liu, Yue Wang, Jingren Zhou, et al. Mai-ui technical report: Real-world centric foundation gui agents. *arXiv preprint arXiv:2512.22047*, 2025.

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

A Implementation Details of OpenMobile Framework

A.1 Environment Exploration Algorithm

As described in Section 3.1, OpenMobile decouples exploration from instruction generation, making the framework agnostic to the specific exploration strategy. In this work, we follow OS-Genesis (Sun et al., 2025a) and adopt a simple random walk. Each session executes 10 steps, randomly selecting an interactable element from the current screen's accessibility tree to perform a click or type action. A blacklist of non-interactable elements is maintained to avoid redundant interactions. The primary objective of this phase is to maximize coverage of app states and functionalities for downstream task synthesis.

A.2 Global Environment Memory Construction

Given exploration trajectories consisting of screen-action transitions, we construct the global environment memory $\mathcal{M}$ through three stages: screen deduplication, functionality annotation, and semantic index construction. The full procedure is outlined in Algorithm 1.

Screen Deduplication. Exploration trajectories contain many visually identical or near-identical screens visited across different sessions. We compute a perceptual hash (pHash) for each screenshot and greedily cluster screens whose pHash similarity exceeds a threshold $\tau = 0.95$, selecting one representative per cluster. For each unique screen s_i, we aggregate all transitions from the original trajectories to identify its neighbor set $\mathcal{N}(s_i)$, i.e., screens directly reachable from or leading to s_i.

Functionality Annotation. For each unique screen, we use a strong LLM `Gemini-3.1-Pro-Preview` to extract a set of functionality descriptions $\mathcal{F}(s_i)$. Each functionality is a natural-language description capturing the semantics of a UI element (e.g., a button, toggle, or menu item). To improve annotation quality, we provide the model with the preceding screen and the action that led to the current screen as context. Elements are categorized as either *functionality* (app-provided features such as buttons and toggles) or *data* (user-generated content such as calendar events).

Semantic Index Construction. To enable cross-screen functionality retrieval, we compute semantic embeddings for all functionality descriptions within each app using a sentence embedding model `openai/text-embedding-3-large`. This produces a per-app retrieval index that supports efficient nearest-neighbor search during task synthesis. We apply greedy diversity filtering to ensure that retrieved functionalities are semantically distinct (pairwise cosine similarity below 0.8).

A.3 Memory-Augmented Task Synthesis

Given the global environment memory $\mathcal{M}$, we synthesize task instructions by presenting rich context to a strong vision-language model `Gemini-3.1-Pro-Preview` and prompting it to generate grounded, multi-step instructions.

Context Construction. For each candidate screen s_i, we construct a prompt that consists of three parts. First, we include the screenshot of s_i along with its annotated functionality descriptions, serving as the focal point for generation. Second, we retrieve screenshots and functionality descriptions from neighboring screens in the transition graph as short-term memory. Specifically, we include 1 predecessor screen (the screen that transitions into s_i) and up to 3 successor screens (screens reachable from s_i), providing the model with local navigation context. Third, we retrieve 30 semantically related functionalities from other screens within the same app as long-term memory. These are selected via embedding cosine similarity with a diversity constraint (pairwise similarity < 0.8) to surface distant but relevant features that encourage cross-functional composition. The assembled context, comprising both screenshots and textual descriptions, is fed to the model together with generation guidelines and in-context examples.

Algorithm 1 Global Environment Memory Construction

Input: Exploration trajectories $\mathcal{T} = \{(o_t, a_t, o_{t+1})\}$, similarity threshold τ
Output: Global environment memory $\mathcal{M} = (\mathcal{S}, \mathcal{N}, \{\mathcal{F}(s_i)\})$
// Stage 1: Screen Deduplication
Collect all screens $\mathcal{O} = \{o_t, o_{t+1} \mid (o_t, a_t, o_{t+1}) \in \mathcal{T}\}$
Compute perceptual hash $h(o)$ for each $o \in \mathcal{O}$
$\mathcal{S} \leftarrow \emptyset$
for each screen $o \in \mathcal{O}$ **do**
  if $\nexists s \in \mathcal{S}$ s.t. $\textsc{Sim}(h(o), h(s)) \geq \tau$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{o\}$ ▷ Add as new unique screen
  end if
end for
// Stage 2: Neighborhood & Functionality Extraction
for each unique screen $s_i \in \mathcal{S}$ **do**
  $\mathcal{N}(s_i) \leftarrow$ screens in $\mathcal{S}$ reachable from/to s_i via transitions in $\mathcal{T}$
  $\mathcal{F}(s_i) \leftarrow \text{VLM}(s_i, \text{context})$ ▷ Extract functionality descriptions
end for
// Stage 3: Semantic Index Construction
for each app A **do**
  Collect all functionalities: $\mathcal{F}_A = \bigcup_{s_i \in A} \mathcal{F}(s_i)$
  Compute embeddings $\mathbf{E}_A = \textsc{Embed}(\mathcal{F}_A)$
  Build retrieval index over $(\mathcal{F}_A, \mathbf{E}_A)$
end for
return $\mathcal{M} = (\mathcal{S}, \mathcal{N}, \{\mathcal{F}(s_i)\}_{i=1}^{|\mathcal{S}|})$

Quality Filtering. Generated instructions undergo three-stage filtering: (1) each instruction is scored by a strong LLM on complexity, clarity, and reasonableness (1–5 scale), and those with clarity < 4 or reasonableness < 4 are discarded; (2) remaining instructions are sorted by scores and greedily deduplicated using embedding cosine similarity with a threshold of 0.8, so the highest-rated instruction is retained within each semantic cluster; (3) per-app instruction counts are capped to ensure balanced app coverage during training.

A.4 Error-Intervention Policy Switching

Our error-intervention switching begins with the learner model executing the task. At each step, a monitor (`Gemini-3.1-Pro-Preview`) observes the recent action history and the last two screenshots to evaluate whether the previous action has caused the agent to deviate from the task objective. Once deviation is detected, the expert model is invoked to intervene. We find that providing the monitor's deviation analysis to the expert improves the quality of error-recovery signals, as the expert can better understand the current failure mode before correcting the trajectory. In our rollout, this error-intervention process is triggered at most twice. After each expert intervention, the expert executes at least 3 steps before control is returned to the learner.

A.5 Prompts Used in OpenMobile

We provide the complete prompts used in the OpenMobile pipeline below.

Prompt for Functionality Extraction

System Prompt

You are a GUI screenshot analysis expert. You will be provided with:
1. A screenshot of a UI screen (Screen Before) with the action area marked in red
2. The action type performed
3. The resulting screenshot after the action (Screen After)
4. The name of the Android app

Your task is to analyze the elements on the **second screenshot (Screen After)** ONLY. The first screenshot is provided only as context to help you understand the app's state.
Each element should be output as a dictionary:

```
{
    "type": "functionality" or "data",
    "label": "A short phrase describing its identifier on this screen",
    "description": "A few sentences describing this element's functionality"
}
```

The description should be **comprehensive and detailed**:

- Include the hierarchical location within the app (e.g., which menu, which settings page, which sub-section)
- Explain what this element does **at the phone/device level**, so that someone reading this description can fully understand the element's role and functionality without seeing the screenshot.

Here are examples showing bad descriptions and their improved versions:
Example 1:
– Bad: "A WiFi toggle that enables or disables WiFi connectivity."
– Reason: Too vague; does not specify location or device-level changes.
– Good: "This toggle under System Settings ¿ Network & Internet ¿ Wi-Fi enables or disables Wi-Fi on the device, allowing the phone to scan for available wireless networks and connect/disconnect from them."
Example 2:
– Bad: "A Reminder option enables users to set a reminder."
– Reason: Too vague; does not explain what scenario it is used for.
– Good: "In the calendar app's event creation/edit screen, this reminder option schedules a notification before the event starts (e.g., 10 minutes in advance), helping the user receive an alert at the chosen lead time."
Output a JSON list only. No markdown, no comments, no extra text. Start with `[` and end with `]`.

User Prompt

`[Image: Screen Before (with action area marked in red)]`
`[Image: Screen After]`
App: {app_name}
Action: {action_type}
The first image is Screen Before (with action area marked in red). The second image is Screen After. Please analyze the elements on the second image.

Prompt for Task Instruction Synthesis

System Prompt

You are a GUI explorer. Your goal is to explore a GUI environment and synthesize high-quality, high-difficulty, executable, high-level, multi-step GUI tasks/instructions.
You have already completed the exploration work. You have collected many screenshots from the current GUI environment, the transitions between them, and various functionalities within the corresponding app.
Now, you need to fully associate and imagine based on the following three sources of information to generate long-range, high-level tasks/instructions that are possible within the current app:

1. A recalled screenshot of a specific screen
2. Several screenshots in short-term memory that have transition relationships with this screenshot (screens that can be reached from the current screen)
3. Importantly, some functionalities retrieved from long-term memory that are associated with the current screen (semantically related functionalities from other screens in the same app)

Based on these three sources of information, you should fully associate, imagine, and generate long-range, high-level tasks/instructions that are possible within the current app.

Guidelines

1. The provided screenshots and functionalities are only a portion of your recalled memories serving as context. Your ONLY task is to synthesize clear multi-step GUI instructions. The instructions you synthesize do not need to have direct connections with the current screen or operations, but can be inferred from the context. However, to ensure the difficulty and complexity of generated tasks, you are encouraged to analyze, associate, and combine functionalities from your memories.
2. There are two types of tasks to generate:
   - **Action tasks**: Require performing a series of actions to accomplish a goal. For example: "Set an alarm for tomorrow at 8 AM that repeats every weekday."
   - **Question-answering tasks**: Require performing a series of actions and answering a question related to the environment's content. For example: "In my to-do list, how many tasks need to be completed this Wednesday? Answer the question with a single number."

   You should decide which type of task is appropriate to generate based on the context.
3. Synthesized tasks **must be clear and explicit**. Generated tasks should be specific with sufficient details, so that executors will not feel confused. For example, "Help me create a new event in the calendar" is too broad. It should include concrete configurations, e.g., date, time, title, description, duration, location, etc.
4. Synthesized tasks must be executable. **If you want to generate a task that involves operating on app data (for example, deleting an entry in the calendar), you MUST make sure the data you want to operate on is present in the given screenshots.**
5. Generated tasks should be diverse. Do not only focus on the app's main functions. Try to cover all functionalities of the app as much as possible, for example, elements or functions in corners of screens, or functionalities you associate from memories.
6. Generated tasks should be long-range. Do not generate single-step tasks such as clicking a button. You are encouraged to generate tasks that require executors to reason, plan, and complete in multiple steps. **You can also consider combining different sub-functions or sub-tasks into a long-range task, but ensure reasonableness.**
7. Generated tasks should be high-level. **Do not generate step-by-step instructions and detailed actions.** Instead, integrate multi-step instructions into a high-level intent to increase task difficulty. **They should be a single command that contains specific details, rather than step-by-step operations for completing a task.**
8. Generated tasks should start from the phone's home screen, not from the currently provided screen. Do not generate tasks that are bound to temporary states of the current interface (for example, a popup dialog that appears).
9. The operating environment is a virtual device with no network connection. Do not generate tasks that require internet connection or login. However, you can freely use data that is already saved in the existing app.

Example Tasks
Here are examples showing bad tasks and their improved versions:
Example 1:
– Bad: "Access and manage the list of all saved Bluetooth devices."
– Reason: Does not specify what "manage" means.
– Good: "View all existing Bluetooth devices, and if any exist, delete all of them."
Example 2:
– Bad: "Add a new recipe to the list using the plus button on the main recipe screen."
– Reason: Does not specify concrete content.
– Good: "In the Broccoli app, add a new recipe for 'Tomato and Egg Stir-fry', set the category to 'Stir-fry', and fill in the description as 'Mom's favorite dish'."
Example 3:
– Bad: "Check the battery usage statistics and enable Battery Saver mode if necessary."
– Reason: "If necessary" will confuse the executor.
– Good: "Write the top three items from battery usage statistics into the Markor app and save it as 'battery_usage_statistics', and enable Battery Saver mode."
Example 4:
– Bad: "Dismiss the voice search connection error by tapping the 'Keyboard' button, then manually type 'The Beatles' in the search bar."
– Reason: Includes a temporary state and assumes starting from the search interface.

– Good: "In {app name}, how many songs are included for The Beatles and Taylor Swift respectively? Answer with numbers separated by a comma."
Example 5:
– Bad: "In the Broccoli app, use the search function to find the recipe 'Salmon with Dill Sauce'. Open its details page and answer how many servings it yields."
– Reason: Contains too many specific operations; should be more high-level.
– Good: "In the Broccoli app, how many servings does 'Salmon with Dill Sauce' provide, and what is the total preparation time required?"
Example 6:
– Bad: "In Simple Calendar Pro, navigate to the 'Customize colors' menu, attempt to change the App icon color, and dismiss the warning popup."
– Reason: Contains unnecessary specific operations and temporary states.
– Good: "Set the app color of Simple Calendar Pro to blue."
Example 7:
– Bad: "In the Tasks app, what tasks do I have?"
– Reason: Too vague.
– Good: "In the Tasks app, which tasks due this week are not completed yet? Answer with titles only; if there are multiple, separate them with commas."
Example 8:
– Bad: "In the Audio Recorder app, configure the settings for high-fidelity recording. After entering the app, navigate to the setup menu and change the recording format to Wav, set the sample rate to 48kHz..."
– Reason: Contains too many step-by-step operations.
– Good: "Record an audio file in Wav format with 48kHz sample rate and Stereo channel using Audio Recorder, and save it as test_audio."

User Prompt

```
[Image: Current Screen]
[Image: Preceding Screen 1 (if available)]
[Image: Associated Screen 1--3 (if available)]
```

Current Screen
App: {app_name}
Elements on Current Screen ({N} items):
1. "type": functionality, "description": {description_1}
2. ...
These are screens that can transition into the current screen:
Preceding Screen 1
Elements ({N} items): ...
These are screens that can be reached from the current screen:
Associated Screen 1
Elements ({N} items): ...
Related Functionalities from Other Screens ({M} items)
These are semantically related functionalities from other screens in the same app:
1. {description_1}
2. ...
Your Task
Based on the above context, carefully analyze and think, then generate 1–3 high-quality GUI tasks. Each task should be a concise but high-level instruction in English. Output format (JSON array):

```
[
  {"reasoning": "...", "task": "task instruction 1"},
  {"reasoning": "...", "task": "task instruction 2"}
]
```

B Experiment Settings

B.1 Benchmark Evaluation Setup

We evaluate on three established dynamic mobile agent benchmarks: AndroidWorld, AndroidLab, and MobileWorld. All models are deployed with vLLM for inference. We observe that agent execution in dynamic environments exhibits inherent randomness, leading to

non-trivial variance in success rates across runs. To account for this, we run each benchmark three times and report the mean along with the half-range (i.e., (max − min) / 2) as a measure of variation. We additionally report Pass@3, the success rate when a task is considered solved if any of the three runs succeeds, to indicate the model's performance upper bound. Results of some baseline models are taken from prior work, e.g., UI-Venus-1.5 (Gao et al., 2026) and the MobileWorld leaderboard (Kong et al., 2025).

B.2 Policy-Switching Rollout Settings

We use Qwen2.5-VL-7B-Instruct as the base model (learner π_l) for all policy-switching ablations, with `Gemini-3.1-Pro-Preview` as the expert π_e. **Expert distillation.** The expert model executes all synthesized instructions. We retain trajectories where the expert signals task completion (i.e., outputs `complete` or `answer`) and convert them into step-level training samples. **Self-evolution.** The learner executes the synthesized instructions, and the expert serves as a judge to identify successful trajectories. Only successful trajectories are used to retrain the learner. This process is iterated for 3 rounds. **Random switching.** At each step, whenever the learner and expert predict inconsistent actions (e.g., different action types or different target elements), the learner's action is used in place of the expert's. However, the learner is not allowed to execute terminal actions (`complete` or `answer`) to ensure task completion by the expert. **Error-intervention switching.** The rollout begins with the learner policy. A monitor tracks the learner's execution and triggers a switch to the expert when deviation from productive progress is detected. The expert then intervenes to correct the trajectory back on track. Details of the monitor design are provided in Section A.4. All strategies are compared under a fixed budget of 1.5K trajectories. To quantitatively measure the error-recovery signals introduced by each strategy, we randomly sample 50 trajectories from each and manually inspect the average number of error-recovery signals per trajectory, defined as a step where the agent recognizes and attempts to correct a mistake from the previous step.

C Exploration with Reinforcement Learning

Beyond standard supervised fine-tuning (SFT), we explore the effectiveness of reinforcement learning (RL) on our synthesized data.

Step-Level RL. We first experiment with step-level RL, a commonly adopted paradigm for GUI agents. Following UI-R1 (Lu et al., 2025b), we define three reward signals for each agent step: a format reward, an action type reward, and a grounding reward, and train with standard GRPO. We then evaluate the resulting model on AndroidWorld. Results show that while step-level GRPO initially improves performance using synthesized trajectories, the gains quickly saturate and ultimately fail to surpass the SFT baseline. We attribute this to the inherent discrepancy between step-level optimization and multi-step execution in dynamic environments. Step-level RL tends to overfit to single-step outputs, failing to produce stable gains on long-horizon tasks that require sustained interaction with a changing environment. A similar observation has also been reported by UI-Venus-1.5 (Gao et al., 2026).

Trajectory-Level Agentic RL. We further explore trajectory-level agentic RL to enhance the agent's performance. We conduct these experiments using the OS-Themis (Li et al., 2026) framework, which provides an infrastructure of over a hundred Android emulator

Method	Base Model	AndroidWorld		AndroidLab		MobileWorld	
		Pass@1↑	Pass@3↑	Pass@1↑	Pass@3↑	Pass@1↑	Pass@3↑
Qwen3-VL-8B	–	47.6 ± 2.2	62.1	43.5	–	9.4	–
Ours-8B	Qwen3-VL	64.7 ± 3.2	78.0	51.5 ± 0.7	62.3	17.7 ± 2.2	24.8
Ours-8B-RL	Qwen3-VL	64.1 ± 0.5	77.6	53.9 ± 1.5	63.0	16.8 ± 0.9	20.5

Table 4: Results of trajectory-level RL.

instances for trajectory rollout, along with a multi-agent critic that leverages VLMs to judge task success or failure. We use an early checkpoint as the starting point for RL training, and filter the synthesized task instructions to retain only those that the checkpoint fails but the expert successfully completes, yielding 244 instructions in total. As shown in Table 4, while trajectory-level RL on our synthesized instructions does improve performance, it cannot consistently surpass its fully SFT-trained counterpart. We hypothesize that this is related to the limited diversity of our current environment setup and the stability of the RL framework itself. Addressing these limitations to further advance mobile agent capabilities remains an important direction for future work.

D Additional Experimental Results

Method	Pass@1↑	Pass@3↑
Qwen2.5-VL-7B	25.5 ± 2.6	34.9
Qwen2.5-VL-72B	27.6	–
UI-Venus-72B	65.9	–
Ours-7B	51.7 ± 1.7	68.1
Ours-72B	59.3 ± 0.9	72.8

(a) Scaling to larger models.

Method	Open	#Traj	Pass@1↑
OS-Genesis	✓	1.5K	17.4
HATS	✓	1K	24.4
AutoPlay	✗	20K	40.1
MobileGen	✗	0.5K	45.7
OpenMobile	✓	2.8K	**64.7**

(b) Comparison with data synthesis methods.

Table 5: (a) AndroidWorld results with larger model sizes. (b) Comparison with existing data synthesis methods on AndroidWorld.

To validate the effectiveness of OpenMobile data on larger models, we fine-tune `Qwen2.5-VL-72B-Instruct` using the same data. As shown in Table 5a, the larger model yields notably stronger performance, confirming both the scalability of our data and the importance of base model capability.

We also compare with existing mobile agent data synthesis methods (Sun et al., 2025a; Shao et al., 2026; Ramrakhya et al., 2025; Kang et al., 2026). As shown in Table 5b, OpenMobile achieves a substantially higher success rate with a moderate data scale. We note that direct comparison is imperfect, as these methods differ in base models and experimental settings, and some do not open-source full implementation details. Nevertheless, the results demonstrate the effectiveness of OpenMobile and position it as a strong starting point for future mobile agent data synthesis research.

E Similarity Between Synthetic and Test Instructions

To quantify the overlap between our synthesized instructions and the AndroidWorld test set, we compute pairwise cosine similarities using sentence embeddings from `openai/text-embedding-3-large`. As reported in Figure 3, our synthesized instructions exhibit moderate functionality-level relevance rather than task-level overlap with the benchmark, with only 3.5% exceeding a similarity of 0.7. Table 6 lists each AndroidWorld test instruction alongside its most similar synthetic counterpart from synthesized instructions.

Table 6: Complete list of nearest synthetic–test instruction pairs. Each row shows an AndroidWorld test instruction paired with the most similar OpenMobile synthetic instruction.

#	AndroidWorld Test Instruction	Most Similar Synthetic Instruction	Sim.
1	Record an audio clip using Audio Recorder app and save it.	In the Audio Recorder app, change the recording settings to use the Wav format and Stereo channel, then record a short audio clip.	0.715
2	Record an audio clip and save it with name "eVq3_review.m4a" using Audio Recorder app.	Record a high-quality audio clip using the Wav format and 48kHz sample rate in the Audio Recorder app.	0.635

(continued on next page)

(continued from previous page)

#	AndroidWorld Test Instruction	Most Similar Synthetic Instruction	Sim.
3	Open the file task.html in Downloads in the file manager; when prompted open it with Chrome. Then create a drawing using the three colors shown at the top and hit submit.	In the Files app, navigate to the Downloads folder and open the 'task.html' file using the Chrome browser.	0.743
4	Open the file task.html in Downloads in the file manager; when prompted open it with Chrome. Then navigate the X to the bottom-right cell, by using the direction buttons.	In the Files app, navigate to the Downloads folder and open the 'task.html' file using the Chrome browser.	0.803
5	Open the file task.html in Downloads in the file manager; when prompted open it with Chrome. Then click the button 5 times, remember the numbers displayed, and enter their product in the form.	In the Files app, navigate to the Downloads folder and open the 'task.html' file using the Chrome browser.	0.706
6	Take one photo.	Switch to Camera mode, enable the 3x3 grid lines overlay, and take a photo.	0.524
7	Take one video.	Switch the Camera app to Video mode and record a short video clip.	0.502
8	Pause the stopwatch.	Use the Stopwatch to record a lap time, then pause and reset the timer to zero.	0.626
9	Run the stopwatch.	Use the Stopwatch to record a lap time, then pause and reset the timer to zero.	0.610
10	Create a timer with 23 hours, 4 minutes, and 57 seconds. Do not start the timer.	In the Clock app, set a timer for 1 hour, 23 minutes, and 45 seconds and start the countdown.	0.574
11	Create a new contact for Ahmed dos Santos. Their number is +12432810546.	Create a new contact for 'Alice Smith' with the phone number '555-123-4567' in the Phone app.	0.527
12	Go to the new contact screen and enter the following details: First Name: Eva, Last Name: Smith, Phone: 119-168-9838, Phone Label: Work. Do NOT hit save.	In the Contacts app, create a new contact with the name "John Smith" and the phone number "555-1234".	0.614
13	Add the following expenses into the pro expense: name|amount_dollars|category_name|note Museum Tickets|$325.17|Entertainment|Urgent Social Club Dues|$425.35|Social|I may repeat this Museum Tickets|$485.01|Entertainment|I may repeat this	In Pro Expense, find the 'Club Membership' expense and update it by changing the category to 'Entertainment', setting the amount to 100, and changing the note to 'Monthly fee', then save the changes.	0.659
14	Add the expenses from expenses.jpg in Simple Gallery Pro to pro expense.	In Simple Gallery Pro, find the receipt from 'Innovate Solutions Ltd.' and answer what item was purchased and its price. Answer the question in the format: 'Item Name, Price'.	0.561
15	Go through the transactions in my_expenses.txt in Markor. Log the reimbursable transactions in the pro expense.	In the Pro Expense app, find the existing expense record for 'ProDev' and permanently delete it from the logs.	0.535
16	Add the following expenses into the pro expense: name|amount_dollars|category_name|note Club Membership|$56.67|Social|Urgent	In Pro Expense, find the 'Club Membership' expense and update it by changing the category to 'Entertainment', setting the amount to 100, and changing the note to 'Monthly fee', then save the changes.	0.706
17	Delete all but one of any expenses in pro expense that are exact duplicates, ensuring at least one instance of each unique expense remains.	In the Pro Expense app, find the existing expense record for 'ProDev' and permanently delete it from the logs.	0.544
18	Delete all but one of any expenses in pro expense that are exact duplicates, ensuring at least one instance of each unique expense remains.	In the Pro Expense app, find the existing expense record for 'ProDev' and permanently delete it from the logs.	0.544
19	Delete the following expenses from pro expense: Textbooks, Salary, Stationery.	In Pro Expense, permanently delete the 'School Supplies' transaction from the recent expenses list.	0.655
20	Delete the following expenses from pro expense: Night Out, Tailoring Services, Snacks.	In Pro Expense, permanently delete the 'School Supplies' transaction from the recent expenses list.	0.617
21	Delete the following expenses from pro expense: Taxi Fare.	Delete the 'Taxi Fare' transaction from the recent list in the Pro Expense app.	0.760
22	Delete the file smart_guitar_2023_07_08.mp3 from the Android filesystem located in the Podcasts folder within the sdk_gphone_x86_64 storage area.	In Retro Music, permanently delete the audio file for the song 'My Heart is Yours' from the device.	0.520
23	Move the file sci_fi_thriller.mp4 from Podcasts within the sdk_gphone_x86_64 storage area to the Movies within the same sdk_gphone_x86_64 storage area in the Android filesystem.	In the Files app, navigate to the internal storage 'sdk_gphone64_arm64', create a new folder named 'Project_Alpha', and then change the directory's view mode to the grid layout.	0.503
24	Update the Markor note backup_silly_tiger.txt by adding the following text, along with a new blank line before the existing content: "rMneE03kV42gdja3JKbM", and rename it to 0xeI_quick_house.md.	In the Markor app, open the 'silly_tree_copy' note, insert the text 'IMPORTANT:' in bold at the very beginning of the document, and save the file.	0.668

(continued on next page)

(continued from previous page)

#	AndroidWorld Test Instruction	Most Similar Synthetic Instruction	Sim.
25	Update the content of lively_wolf_2023_06_12.md to "n7g5D1zYaynRyGnucoSD" in Markor and change its name to ucbD_pretty_elephant.txt.	In Markor, update the file '2023_04_18_pr...' by replacing its content with a shopping checklist containing bolded 'Apples', italicized 'Bananas', and 'Oranges', then save the changes.	0.629
26	Create a new folder in Markor named folder_20260304_164543.	Create a new folder named 'Meeting_Notes' in the Markor app, and inside that folder, create a new file named '2023-10-27_Briefing.md'.	0.740
27	Create a new note in Markor named proud_banana_E4ao.txt with the following text: The pen is mightier than the sword.	In the Markor app, open the 'silly_tree_copy' note, insert the text 'IMPORTANT:' in bold at the very beginning of the document, and save the file.	0.605
28	Create a new note in Markor named wise_nurse_2023_06_26.txt with the following text: The early bird catches the worm.. Share the entire content of the note with the phone number +11843888619 via SMS using Simple SMS Messenger	In Simple SMS Messenger, save the phone number +16700234643 as a new contact named 'Daily Wisdom' and then initiate a call to this number.	0.547
29	Create a note in Markor named final_fancy_unicorn.md. Perform a paste operation in the note and save the note.	In Markor, create a new note containing the text 'Draft version', and then use the Search and Replace tool to replace the word 'Draft' with 'Final'.	0.599
30	Delete all my notes in Markor.	In the Markor app, delete all files in the Documents folder that contain the phrase 'april_workout_routine' in their names.	0.623
31	Delete the newest note in Markor.	In the QuickNote section of the Markor app, delete the existing line containing the date '2023-10-15' using the delete line toolbar option.	0.645
32	Delete the note in Markor named fancy_queen_2023_01_05.	In the QuickNote section of the Markor app, delete the existing line containing the date '2023-10-15' using the delete line toolbar option.	0.700
33	Edit note_5v02Y.md in Markor. Add to the bottom of the note The library book is due back on the 15th.	In the QuickNote section of the Markor app, delete the existing line containing the date '2023-10-15' using the delete line toolbar option.	0.551
34	Merge the contents of Markor notes best_lion_final.txt, alert_koala_2023_10_10.txt and edited_super_cat.md (in the same order) into a new Markor note named QpRyrbS1 and save it. Add a new line between the content of each note.	In Markor, open the file 'tough_frog_2023_08_05.txt', add a new line with the text 'Urgent' formatted in bold, and then save the file.	0.587
35	In Markor, move the note copy_helpful_umbrella.txt from MeetingMinutes to WorkProjects.	In the Markor app, add a new entry titled 'Organize documents' to the To-Do list, and then move the file 'final_meeting_notes_project_team.md' into a new folder named 'Work Archive'.	0.617
36	Create a file in Markor, called receipt.md with the transactions from the receipt.png. Use Simple Gallery to view the receipt. Please enter transactions in csv format including the header "Date, Item, Amount".	Create a new file named 'Groceries.md' in Markor, add a checklist item labeled 'Milk', and save the document.	0.581
37	Transcribe the contents of video copy_moment_10_.mp4 by watching it in VLC player (located in Download) and writing the sequence of strings shown on each frame to the text file copy_moment_10__transcription.txt in Markor as a comma separated list. For example, if the first frame shows the text "edna" and the second frame shows the text "pineapple", then the text file should contain only the following text: "edna, pineapple".	In the Markor app, what is the exact string of characters written on the first line of the file 'oGsN_note_X3...'? Answer the question with the text only.	0.467
38	Is the note titled 'Research Notes' in the Joplin app marked as a todo item? Respond with either 'True' if it is a todo or 'False' if not.	In the Joplin app, search for the item "Mortgage Payment Schedule" and determine if it is a to-do task (with a checkbox) or a standard note. Answer with "To-do" or "Note".	0.607
39	How many attendees were present in the meeting titled 'Marketing Campaign Planning' in the Joplin app? Express your answer as just a single number.	In the Joplin app, how many visible notes contain the word 'Plan' or 'Planning' in their title? Answer the question with a single number.	0.628
40	What quantity of matcha powder do I need for the recipe 'Lasagna' in the Joplin app? Express your answer in the format ¡amount¿ ¡unit¿ where both the amount and unit exactly match the format in the recipe.	In the Broccoli app, for the recipe containing the ingredient 'per individual taste', what is the default serving size displayed when you open the 'Adjust ingredients' dialog? Answer the question with a single number.	0.502
41	How many to-dos do I have in the 'Travel' folder in the Joplin app? Express your answer as just a single number.	In the Joplin app's 'All notes' list, how many to-do items start with the word 'Travel'? Answer the question with a single number.	0.845
42	Open the camera app. Clear any pop-ups that may appear by granting all permissions that are required.	Grant Android Auto the 'Device & app notifications' permission to allow it to read notifications, and then clear the app's cache.	0.528

(continued on next page)

(continued from previous page)

#	AndroidWorld Test Instruction	Most Similar Synthetic Instruction	Sim.
43	Add a favorite location marker for Malbun, Liechtenstein in the OsmAnd maps app.	In the OsmAnd app, configure the map settings to display both 'Favorites' locations and 'Transport' routes.	0.532
44	Add a location marker for 47.1303814, 9.5930117 in the OsmAnd maps app.	In the OsmAnd app, use the address search feature to locate the point at latitude 40.7306 and longitude -73.9352 using the coordinate search option, and show this location on the map.	0.579
45	Save a track with waypoints Schaan, Liechtenstein, Malbun, Liechtenstein, Planken, Liechtenstein, Rotenboden, Liechtenstein in the OsmAnd maps app in the same order as listed.	Using the 'Plan a route' feature in OsmAnd, manually create a custom route by placing four waypoints on the map and save this track to your 'My Places' collection with the name 'Sample Trip'.	0.606
46	Add the following recipes into the Broccoli app: title|description|servings|preparationTime|ingredients |directions Classic Margherita Pizza|An ideal recipe for experimenting with different flavors and ingredients.|1 serving|20 mins|to your liking|Spread pizza dough with tomato sauce, top with slices of mozzarella cheese and fresh basil leaves. Bake until crust is golden. Garnish with fresh herbs for a more vibrant taste. Garlic Butter Shrimp|A quick and easy meal, perfect for busy weekdays.|1 serving|2 hrs|see directions|Sauté shrimp in butter and minced garlic until pink. Sprinkle with parsley and serve with lemon wedges. Garnish with fresh herbs for a more vibrant taste. Mango Chicken Curry|A delicious and healthy choice for any time of the day.|3-4 servings|1 hrs|various amounts|Cook chicken pieces in a pan, add onions, garlic, and ginger. Stir in curry powder, coconut milk, and mango pieces. Simmer until chicken is cooked. Feel free to substitute with ingredients you have on hand.	In the Broccoli app, create a new recipe for 'Classic Margherita Pizza' under the 'Dinner' category, set the description to 'Simple and delicious', source it from 'Chef Mario', specify it serves 2 people, takes 45 minutes to prepare, and list 'Dough, Tomato Sauce, Mozzarella, Basil' as the ingredients.	0.760
47	Add the recipes from recipes.jpg in Simple Gallery Pro to the Broccoli recipe app.	In the Broccoli app, create a new recipe for 'Garden Salad', assign it to the 'Healthy' category, and use the device's camera to take and set a cover photo for the recipe before saving.	0.685
48	Add the recipes from recipes.txt in Markor to the Broccoli recipe app.	In the Broccoli app, find the 'Tomato Basil Bruschetta' recipe and add it to your Favorites.	0.595
49	Add the recipes from recipes.txt in Markor that take 4 hrs to prepare into the Broccoli recipe app.	In the Broccoli app, edit the 'Lentil Soup' recipe to change its preparation time to 45 minutes, and then mark the recipe as a favorite.	0.662
50	Add the following recipes into the Broccoli app: Recipe: Caprese Salad Skewers description: A quick and easy meal, perfect for busy weekdays. servings: 6 servings preparationTime: 4 hrs ingredients: various amounts directions: Thread cherry tomatoes, basil leaves, and mozzarella balls onto skewers. Drizzle with balsamic glaze. Garnish with fresh herbs for a more vibrant taste.	In the Broccoli app, the 'Caprese Salad Skewers' recipe is missing ingredient details. Edit the recipe to add 'Mozzarella balls' and 'Cherry tomatoes' to the ingredients list, and update the preparation time to '20 mins'.	0.780
51	Delete all but one of any recipes in the Broccoli app that are exact duplicates, ensuring at least one instance of each unique recipe remains	In the Broccoli app, clean up the recipe list by deleting the duplicate entries for 'Baked Cod with Lemon and Dill' so that only one such entry remains.	0.771
52	Delete all but one of any recipes in the Broccoli app that are exact duplicates, ensuring at least one instance of each unique recipe remains	In the Broccoli app, clean up the recipe list by deleting the duplicate entries for 'Baked Cod with Lemon and Dill' so that only one such entry remains.	0.771
53	Delete all but one of any recipes in the Broccoli app that are exact duplicates, ensuring at least one instance of each unique recipe remains	In the Broccoli app, clean up the recipe list by deleting the duplicate entries for 'Baked Cod with Lemon and Dill' so that only one such entry remains.	0.771
54	Delete the following recipes from Broccoli app: Eggplant Parmesan, Cauliflower Fried "Rice", Lemon Garlic Tilapia.	Delete the 'Eggplant Parmesan' recipe from your collection in the Broccoli app.	0.771
55	Delete the recipes from Broccoli app that use ghee in the directions.	Delete the 'Beef Stir Fry' recipe from the Broccoli app.	0.682
56	Delete the following recipes from Broccoli app: Turkey and Cheese Panini, Stuffed Bell Peppers, Thai Peanut Noodle Salad.	Delete the 'Thai Peanut Noodle Salad' recipe from the Broccoli app.	0.796
57	Delete the following recipes from Broccoli app: Chicken Caesar Salad Wrap.	Delete the 'Beef Stir Fry' recipe from the Broccoli app.	0.720
58	Delete the following recipes from Broccoli app: Mango Chicken Curry.	Delete the 'Beef Stir Fry' recipe from the Broccoli app.	0.752
59	Create a playlist in Retro Music titled "Acoustic Sessions 86" with the following songs, in order: City of Stars, Echoes of Silence	In Retro Music, create a new playlist named 'Acoustic Sessions' and add the songs 'City of Stars' and 'Distant Memories' to it.	0.829

(continued on next page)

(continued from previous page)

#	AndroidWorld Test Instruction	Most Similar Synthetic Instruction	Sim.
60	Add the following songs, in order, Shadows of Time, Eternal Flame, Golden Days to my playing queue in Retro music.	In Retro Music, find the song 'Distant Memories' in the 'Last added' list and add it to the playing queue.	0.648
61	Create a playlist in Retro Music titled "Electronic Chillout 553" with a duration between 45 and 50 minutes using the provided songs.	In Retro Music, create a new playlist named 'Chill Vibes' and add the song 'Hidden Paths' to it.	0.681
62	Create a playlist in Retro Music titled "Retro Pop Hits 458" with the following songs, in order: Bright Lights, Eternal Flame, Endless Summer. Then export the playlist to the Downloads directory on the device.	Create a new playlist named 'Night Drive' in the Retro Music app and add the songs 'Bright Lights' by Oliver and 'Eternal Flame' by Martina to it.	0.753
63	In Simple Gallery Pro, copy receipt_smart_vase_copy.jpg in DCIM and save a copy with the same name in Download	In Simple Gallery Pro, locate the receipt image for 'Innovate Solutions Ltd' within the DCIM folder, rotate the image 90 degrees clockwise, and mark it as a favorite.	0.647
64	In Simple Calendar Pro, create a calendar event on 2023-10-17 at 11h with the title 'Review session for Campaign' and the description 'We will review product launch. Snacks will be provided.'. The event should last for 45 mins.	In Simple Calendar Pro, create a new event on October 25 titled 'Budget Review' at 'Finance Dept' that starts at 10:00 and ends at 11:30, and add the note 'Prepare Q3 reports' in the description.	0.770
65	In Simple Calendar Pro, create a calendar event in two weeks from today at 20h with the title 'Workshop on Annual Report' and the description 'We will discuss upcoming project milestones.'. The event should last for 60 mins.	In Simple Calendar Pro, create a new event titled "Strategy Workshop" for October 20th starting at 14:00, set the location to "Main Hall", and add a description "Quarterly planning session".	0.767
66	In Simple Calendar Pro, create a calendar event for this Thursday at 5h with the title 'Call with HR' and the description 'We will discuss annual budget. Looking forward to productive discussions.'. The event should last for 45 mins.	In Simple Calendar Pro, create a new event on October 25 titled 'Budget Review' at 'Finance Dept' that starts at 10:00 and ends at 11:30, and add the note 'Prepare Q3 reports' in the description.	0.748
67	In Simple Calendar Pro, create a calendar event for tomorrow at 0h with the title 'Appointment for Campaign' and the description 'We will celebrate software updates.'. The event should last for 45 mins.	In Simple Calendar Pro, create a new event titled 'Project Meeting' for tomorrow at 2:00 PM, set it to repeat weekly, and add a reminder 10 minutes before the start.	0.748
68	In Simple Calendar Pro, create a recurring calendar event titled 'Review session for Project X' starting on 2023-10-24 at 18h. The event recurs weekly, forever, and lasts for 45 minutes each occurrence. The event description should be 'We will organize team roles. Let's be punctual.'.	In Simple Calendar Pro, create a new event titled 'Review session for Annual Report' for October 21st. Set the description to 'We will organize annual budget. Let's be punctual.' and configure the event to repeat yearly.	0.788
69	Do I have any events October 16 2023 in Simple Calendar Pro? Answer with the titles only. If there are multiples titles, format your answer in a comma separated list.	In Simple Calendar Pro, identify all events scheduled for October 17th and answer with their titles separated by a comma.	0.782
70	In Simple Calendar Pro, delete all the calendar events on 2023-10-25	In Simple Calendar Pro, find and delete the 'Workshop' event scheduled for October 24th.	0.718
71	In Simple Calendar Pro, delete all events scheduled for this Friday.	In Simple Calendar Pro, find and delete the 'Workshop' event scheduled for October 24th.	0.727
72	In Simple Calendar Pro, delete the calendar event on 2023-10-30 at 11h with the title 'Catch up on Annual Report'	In Simple Calendar Pro, find and delete the 'Workshop' event scheduled for October 24th.	0.751
73	What is on my schedule for October 19 2023 at 21:45 in Simple Calendar Pro? Answer with the titles only. If there are multiples titles, format your answer in a comma separated list.	In Simple Calendar Pro, switch the calendar view to 'Monthly and daily view', and then list the titles of all events scheduled for October 23rd. Answer with the titles separated by a comma.	0.757
74	What events do I have in the next week in Simple Calendar Pro? Assume the week starts from Monday. Answer with the titles only. If there are multiples titles, format your answer in a comma separated list.	In Simple Calendar Pro, identify all events scheduled for October 17th and answer with their titles separated by a comma.	0.750
75	Do I have any events between 4pm and 8pm October 27 2023 in Simple Calendar Pro? Answer with the titles only. If there are multiples titles, format your answer in a comma separated list.	In Simple Calendar Pro, identify all events scheduled for October 17th and answer with their titles separated by a comma.	0.744
76	What events do I have October 17 2023 in Simple Calendar Pro? Answer with the titles only. If there are multiple titles, format your answer as a comma separated list.	In Simple Calendar Pro, identify all events scheduled for October 17th and answer with their titles separated by a comma.	0.831
77	What is my first event after 11:00am October 21 2023 in Simple Calendar Pro? Answer with the titles only. If there are multiples titles, format your answer in a comma separated list.	In Simple Calendar Pro, switch the calendar view to 'Monthly and daily view', and then list the titles of all events scheduled for October 23rd. Answer with the titles separated by a comma.	0.773
78	What is the location of my Coding challenge event in Simple Calendar Pro? Answer with the location only.	In Simple Calendar Pro, what is the description for the 'Call with the Team' event, and is there a location currently set for it?.	0.597

(continued on next page)

(continued from previous page)

#	AndroidWorld Test Instruction	Most Similar Synthetic Instruction	Sim.
79	What is my next upcoming event in Simple Calendar Pro? Answer with the title only. If there are multiples titles, format your answer in a comma separated list.	In Simple Calendar Pro, identify all events scheduled for October 17th and answer with their titles separated by a comma.	0.753
80	When is my next meeting with Muhammad in Simple Calendar Pro? Express your answer in the format ¡month name¿ ¡day¿ ¡year¿ ¡hour in 24-hour format¿:¡minutes¿.	In Simple Calendar Pro, switch the view to the 'Simple event list', find the 'Meeting with Marketing' event, and answer with its start time in HH:MM format.	0.629
81	Create a new drawing in Simple Draw Pro. Name it amet_lively_eagle_final.jpg. Save it in the Pictures folder within the sdk_gphone_x86_64 storage area.	Using Simple Draw Pro, save a file named 'system_diagram' directly to the root directory of the internal storage (sdk_gphone64_arm64).	0.753
82	Reply to +13431223053 with message: Actions speak louder than words. in Simple SMS Messenger	In Simple SMS Messenger, find the conversation containing the message 'Actions speak louder than words.', and add the sender to your device's contacts with the first name 'Wisdom'.	0.669
83	Reply to the most recent text message using Simple SMS Messenger with message: When in Rome, do as the Romans do.	Find and delete the conversation thread containing the message "When in Rome, do as the Romans do." in the Simple SMS Messenger app.	0.621
84	Resend the message I just sent to Lily Pereira in Simple SMS Messenger	In Simple SMS Messenger, resume the draft conversation with +17401638798 by appending ' hope you are well' to the existing text and sending it.	0.572
85	Send a text message using Simple SMS Messenger to +15039078312 with message: Lorem Ipsum is simply dummy text.	Using Simple SMS Messenger, send a text message with the content 'padiNoBMVR' to the number +1 545-178-61614309.	0.631
86	Send a message to +17228051441 with the clipboard content in Simple SMS Messenger	In Simple SMS Messenger, use the dialer to enter the number +1 545-178-6161 and then start a new text message to this recipient.	0.571
87	Text the address of the event to David Wang that Emily Liu just sent me in Simple SMS Messenger	In Simple SMS Messenger, send a text message to +1 930-572-4145+2 with the content 'Please confirm if the meeting is still on for today'.	0.546
88	How many skate boarding activities did I do this week in the OpenTracks app? Assume the week starts from Monday. Express your answer as a single integer.	In the OpenTracks app, how many activities shown in the list were recorded on a Thursday? Answer with a single number.	0.718
89	What activities did I do October 6 2023 in the OpenTracks app? Answer with the activity type only. If there are multiple types, format your answer in a comma separated list.	In the OpenTracks app, identify all activities that have a recorded distance greater than 9 miles. Answer with the names of the activities separated by a comma.	0.693
90	How long was my climbing activity October 15 2023 in the OpenTracks app? Express your answer in minutes as a single integer.	In the OpenTracks app, what are the moving time and elevation gain recorded for the 'Morning Run' activity? Answer with the values separated by a comma.	0.611
91	What was the longest distance covered in a kayaking activity in the OpenTracks app this week? Assume the week starts from Monday. Express your answer as a single number in meters rounded to the nearest integer.	In the OpenTracks app list, how many recorded activities have a distance greater than 10 miles? Answer with a single number.	0.666
92	What was the total distance covered for swimming activities in the OpenTracks app from October 6 2023 to October 15 2023? Express your answer as a single number in meters rounded to the nearest integer.	In the OpenTracks app list, how many recorded activities have a distance greater than 10 miles? Answer with a single number.	0.659
93	What was the total duration of hiking activities in the OpenTracks app this week? Assume the week starts from Monday. Express your answer in minutes as a single integer.	In the OpenTracks app, how many activities shown in the list were recorded on a Thursday? Answer with a single number.	0.618
94	Turn bluetooth off.	Turn off the Nearby Share feature completely, then return to the Connection preferences menu and open the Bluetooth settings.	0.638
95	Turn bluetooth off.	Turn off the Nearby Share feature completely, then return to the Connection preferences menu and open the Bluetooth settings.	0.638
96	Turn bluetooth on.	Navigate to the Bluetooth pairing screen in Settings and find the phone's Bluetooth address. Answer the question with the full address string.	0.515
97	Turn bluetooth on.	Navigate to the Bluetooth pairing screen in Settings and find the phone's Bluetooth address. Answer the question with the full address string.	0.515
98	Turn brightness to the max value.	Configure the device display for better visibility by enabling 'Dark theme' and setting the 'Font size' to the largest available option.	0.403
99	Turn brightness to the max value.	Configure the device display for better visibility by enabling 'Dark theme' and setting the 'Font size' to the largest available option.	0.403

(continued on next page)

(continued from previous page)

#	AndroidWorld Test Instruction	Most Similar Synthetic Instruction	Sim.
100	Turn brightness to the min value.	Configure the device display for better visibility by enabling 'Dark theme' and setting the 'Font size' to the largest available option.	0.316
101	Turn brightness to the min value.	Configure the device display for better visibility by enabling 'Dark theme' and setting the 'Font size' to the largest available option.	0.316
102	Copy the following text to the clipboard: Reservation under: Mike	In the Markor app, append the sentence ' Dinner reserved at 7pm.' to the existing text in the file named '2023_08_11_good_vase.txt'.	0.375
103	Turn wifi off.	Turn off the "Adaptive connectivity" feature and set the Private DNS mode to "Off".	0.510
104	Turn wifi off.	Turn off the "Adaptive connectivity" feature and set the Private DNS mode to "Off".	0.510
105	Turn wifi on.	Set up a portable Wi-Fi hotspot named 'TravelRouter' with the password 'SecureNet99', and turn it on to share your cellular internet connection.	0.488
106	Turn wifi on.	Set up a portable Wi-Fi hotspot named 'TravelRouter' with the password 'SecureNet99', and turn it on to share your cellular internet connection.	0.488
107	Which tasks have I completed for October 18 2023 in Tasks app? Answer with the titles only. If there are multiples titles, format your answer in a comma separated list.	In the Tasks app, identify the titles of all tasks that are specifically due on 'Oct 8'. Answer by listing the titles separated by a comma.	0.782
108	How many tasks do I have due next week in Tasks app? Assume the week starts from Monday. Express your answer as a single integer.	In the Tasks app, count the number of visible tasks that are due on 'Tue'. Answer with the single number.	0.715
109	What tasks do I have due October 21 2023 in Tasks app? Answer with the titles only. If there are multiples titles, format your answer in a comma separated list.	In the Tasks app, identify the titles of all tasks that are specifically due on 'Oct 8'. Answer by listing the titles separated by a comma.	0.773
110	What are my high priority tasks in Tasks app? Answer with the titles only. If there are multiples titles, format your answer in a comma separated list.	In the Tasks app, how many tasks are currently active (not completed), and which of them has the highest priority? Answer the question with the number and task title, separated by a comma.	0.759
111	Which tasks with high priority are due October 16 2023 in the Tasks app? Answer with the title only. If there are multiples titles, format your answer in a comma separated list.	In the Tasks app, identify the titles of all tasks that are specifically due on 'Oct 8'. Answer by listing the titles separated by a comma.	0.779
112	What incomplete tasks do I have still have to do by October 21 2023 in Tasks app? Answer with the titles only. If there are multiples titles, format your answer in a comma separated list.	In the Tasks app, identify the titles of all tasks that are specifically due on 'Oct 8'. Answer by listing the titles separated by a comma.	0.740
113	Turn off WiFi, then enable bluetooth	Turn off the Nearby Share feature completely, then return to the Connection preferences menu and open the Bluetooth settings.	0.581
114	Turn on Wifi, then open the contacts app	Open the Contacts app and determine how many contacts are currently saved in the list. Answer with a single number.	0.516
115	Create a playlist titled "Documentary Insights Favorites" with the following files in VLC (located in Internal Memory/VLCVideos), in order: 2023_01_29_episode_46_HD.mp4, 2023_06_29_clip_38_export.mp4	In the VLC app, locate the 'Documents' folder within the internal memory storage and add it to the 'Favorites' list.	0.615
116	Create a playlist titled "Recipe Collection Ultimate Collection" with the following files in VLC (located in Internal Memory/VLCVideos), in order: moment_95__1KUB.mp4, scene_54_raw_gbYs.mp4, moment_52_HD_final.mp4, highlight_13_HD_2023_01_29.mp4. And then, create a playlist titled "Ultimate Fails Ultimate Collection" with the following files in VLC, in order: recording_41_HD_backup.mp4, recording_56__JRVN.mp4, 2023_08_10_scene_27_raw.mp4.	In the VLC app, navigate to the Browse tab and create a new playlist named "My Top Hits" using all the media files found in the "Music" folder.	0.619