# Navi-*plus*: Managing Ambiguous GUI Navigation Tasks with Follow-up Questions

**Anonymous ACL submission**

## Abstract

Graphical user interfaces (GUI) automation agents are emerging as powerful tools, enabling humans to accomplish increasingly complex tasks on smart devices. However, users often inadvertently omit key information when conveying tasks, which hinders agent performance in the current agent paradigm that does not support immediate user intervention. To address this issue, we introduce a **Self-Correction GUI Navigation** task that incorporates interactive information completion capabilities within GUI agents. We developed the **Navi-*plus*** dataset with GUI follow-up question-answer pairs, alongside a **Dual-Stream Trajectory Evaluation** method to benchmark this new capability. Our results show that agents equipped with the ability to ask GUI follow-up questions can fully recover their performance when faced with ambiguous user tasks.

## 1 Introduction

Graphical User Interface (GUI) becomes the foundational approach of modern human-computer interaction with increasing numbers of screens filling people's lives. To augment human capabilities and mitigate mental burdens in operating digital devices, GUI automation agents have arisen in recent years. Following the advancements of (Multimodal) Large Language Models (LLMs or MLLMs), extensive efforts were invested in constructing these autonomous agents through large-scale continual pre-training (Cheng et al., 2024, Chai et al., 2024, Lin et al., 2024), grounding-augmented supervised fine-tuning (Li et al., 2024a, Sun et al., 2024), and utilization of Chain-of-Action-Thought (CoAT) in navigation(Zhang et al., 2024, Liu et al., 2025).

However, the previous paradigm of GUI navigation agents is limited to receiving full human instruction and performing actions serially, diminishing human control halfway through agent processing. Thus, a practical "elephant in the room" question arises: ***If some important information is missing from the human instruction (i.e., product specs or important dates), how can the agent continue the task that it is expected to finish?***

This issue has motivated us to review the formulation of the current GUI navigation task. In this paper, we propose a novel task called **Self-Correction GUI Navigation**, which endows GUI agents with a new ability to handle the ambiguity of human instruction. The core idea is to add an "ASK" action in the agents' action space, enabling it to engage in intermediate natural language interactions with human users by proposing follow-up questions (Figure 1).

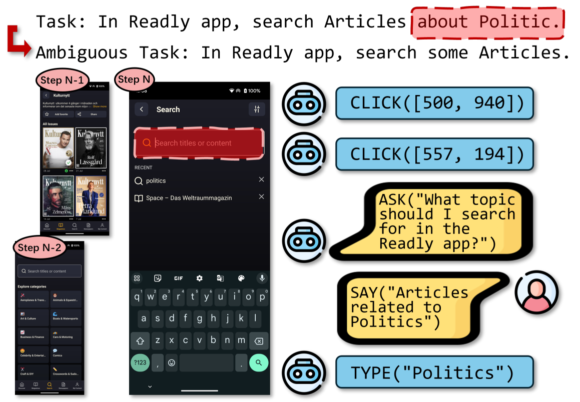

Task: In Readly app, search Articles about Politic.
Ambiguous Task: In Readly app, search some Articles.
Step N-1
Step N
Step N-2
CLICK([500, 940])
CLICK([557, 194])
ASK("What topic should I search for in the Readly app?")
SAY("Articles related to Politics")
TYPE("Politics")

Figure 1: Overview of our proposed Self-Correction GUI Navigation task. Agent proactively asks for missing information when the task is ambiguous.

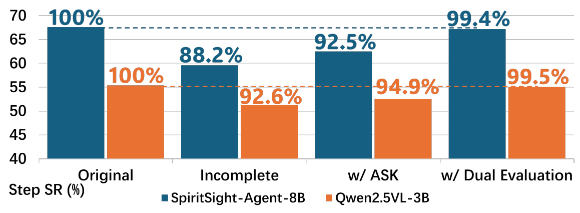


Figure 2: Comparison of agents' Step SR scores on original tasks and generated ambiguous tasks of AndroidControl-Navi*plus* data. Completing the task with ASK action can recover agents' performance.

We first design a data annotation pipeline to construct Navi*plus* dataset from existing trajectory datasets using open-source LLMs(MLLMs). GUI navigation trajectories with ambiguous task descriptions are intentionally and controllably generated, along with corresponding GUI follow-up question-answering (QA) pairs.

We then include fair and comprehensive evaluation metrics to benchmark GUI agents' capability to complete the GUI navigation task when the task description is ambiguous. A **Dual-Stream Trajectory Evaluation** method is proposed to separately compute metrics for the operational actions and for the additional ASK action, allowing direct model comparison.

Our experiments show that the missing information in task description can significantly harm GUI agents task success rate, but with the completion of ASK action, agents' can restore more than 99.4% of performance (Figure 2). Besides, we find that modern MLLM-based GUI agents can seamlessly learn the capability of proposing follow-up questions, and achieve satisfactory performance with timing accuracy above 0.932 and content accuracy exceeding 0.807.

## 2 Methods

### 2.1 Self-Correction GUI Navigation Task

When users describe tasks, it is possible that some key information be omitted. For example, when ordering oil paint online with the help of GUI agents, someone specified colors and sizes but forgot to mention the preferred delivery method due to unfamiliarity with the task flow, leading to ambiguity in the task description for the GUI automation agent.

To address the practical challenge of ambiguous task input, we propose a novel **Self-Correction GUI Navigation** task, aiming to benchmark and facilitate GUI agent feasibility when facing ambiguous task inputs. GUI agents' ability to correctly continue the task and to interact with human users to complete the missing information are the two main indications we consider. (See 3.1)

Specifically, we add an **ASK** action to the model's action space, and during the interaction between the model and the user, we provide a **SAY** action for the user to fill in the missing information. The interaction between the agent and the user will be logged into the agent's context, providing information to continue task navigation.

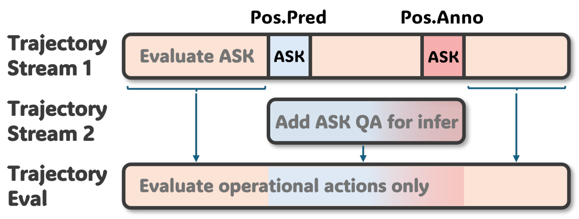


Figure 3: Visualize of our Dual-Stream Trajectory Evaluation method. Each line represents an episode's result.

### 2.2 Dataset Construction Pipeline

To intentionally generate ambiguous GUI navigation task descriptions and agent's follow-up questions for information completion, we develop this data construction process. We select AndroidControl (Li et al., 2024a) and Mind2Web (Deng et al., 2024a) as our data sources for they are collected by well-trained human annotators to ensure quality.

We start by generating low-level instructions for each step in the trajectories with InternVL2.5-26B (Chen et al., 2024c) to identify each action's operational intention. DeepSeek-V3 (Liu et al., 2024a) are then hired to decide whether a step is an informative step or a transactional step (for definitions, see Appendix C). Finally, DeepSeek-V3 is prompted to output an ambiguous task description that excludes selected informative steps, as "Task" and "Ambiguous Task" shown in Figure 1. A QA pair simulating the agent's follow-up question and the user's answer is also generated for each informative step, as the conversation between the agent and human shows in Figure 1.

More details of our data construction pipeline are described in Appendix D. For prompt templates see Appendix I. For dataset statistics see Appendix H.

### 2.3 Dual-Stream Trajectory Evaluation

Adding a new ASK action into agent action space makes the inference process and evaluation metrics inherently different from the original GUI navigation task. Since the evaluation is performed offline with pre-defined screenshots and trajectories, false-positive action predictions can occur if the ASK action appears before its annotated position. This over-strict criterion results in a diminish in observed agent performance.

So, we propose the **Dual-Stream Trajectory Evaluation** method with two key changes as depicted in Figure 3: (1) An ASK action is considered correct within a full task trajectory if it appears at or before the annotated step position. (2) If agents ASK in advance, execute a second inference by adding the ASK QA pairs into context to recover the operational action, like Figure 1.

See Algorithm 1 for pseudo code.

**Algorithm 1:** Dual-Stream Trajectory Eval

```
  Input: Episodes $E = \{E_1, E_2, \dots, E_M\}$
  Output: Metrics $\mu = [\mu_1, \mu_2, \dots, \mu_M]$
1 for each episode $E_m \in E$ do
2     1st Stream Infer: Predict action
        $A = \{a_1, a_2, \dots, a_i, \dots, a_j, \dots, a_n\}$,
        where predicted ASK at $a_i$, annotated
        ASK at $a_j$ $(i < j)$.
3     for $t = i$ to $j - 1$ do
4         Insert ASK QA pair into $E_m$ ;
5     2nd Stream Infer: Predict $a'_i, \dots, a'_j$.
6     for $k = i$ to $j$ do
7         Replace step $a_k$ into $a'_k$ in $A$.
8     Evaluate: Compute metrics $\mu_m$
```

## 3 Experimental Setup

### 3.1 Evaluation Metrics

**Operational Metrics** We follow the practice of Li et al. (2024a) and Deng et al. (2024a) to compute the Step Success Rate (Step SR or SSR) and whole task Success Rate (SR). A step is successful if the predicted action matches the annotated one in terms of the target element and text. A whole task is successful if all the steps it contains are successful.

**Follow-up Question Metrics** We propose to consider the timing and content relevance when evaluating the Self-Correction GUI Navigation task. The timing score focuses on the exact matches of ASK actions, and multiple scales including Precision, False Positive Rate (FPR), and F1 are calculated. The content relevance is measured by calculating the Cosine Similarity (CosSim) and METEOR score of the matched ASK actions.

For metrics formula see Appendix F.

### 3.2 Baseline Models and Implementation Details

We use the original data splits from AndroidControl (Li et al., 2024a) and Mind2Web (Deng et al., 2024a). We perform fine-tuning with the Qwen2.5-VL-3B (Qwen, 2025) and the SpiritSight-Agent-8B-base model adopting LoRA (Hu et al., 2021). For a more detailed explanation see Appendix E.

## 4 Results and Discussion

**Ambiguous task descriptions can significantly degrade agent performance.** Table 1 reports the performance of GUI agents on the AndroidControl-Navi*plus* dataset, comparing results for original and incomplete tasks. With the generated ambiguous task descriptions, the performance of baseline models drops significantly. SpiritSight-Agent's SR, for example, decreased by 31% compared to the full-task scenario. Extended results with more missing information and results on Mind2Web-Navi*plus*, are reported in Appendix G.

**The Dual-Stream Trajectory Evaluation is a more reasonable amendment to conventional navigation metrics.** The "SSR before/after" column in Table 1 reports the SSR before and after the annotated ASK steps. The SSR scores after ASK steps recover when ASK action is applied, indicating the missing information has been recollected. However, SSR scores before ASK steps decrease compared to the no ASK steps case rather than recovery, demonstrating that the original evaluation method failed to take early ASK into consideration. In contrast, by applying our proposed dual-stream evaluation method, the SSR before and after the ASK steps align more closely.

**Adding GUI follow-up questions recovers agent performances.** Table 1 also compares the performance of GUI agents using incomplete tasks, with the addition of ASK actions and with Dual Stream Evaluation. The first three columns of operation metrics show that incorporating GUI follow-up QA annotations for training leads to a 26.5% increase for SpiritSight-Agent and an 8% increase for Qwen2.5VL in SR. The performance of all baseline models recovered to over 99.4% of their original Step SR scores and achieved an average SR score of 100.8% with the dual-stream evaluation. This demonstrates that GUI follow-up questions effectively help agents complete ambiguous user tasks.

**The ability to propose GUI follow-up questions is perfectly acquired by MLLM-based GUI agents.** The ASK action's timing performance, as shown in Table 1, indicates that the agents can effectively ask follow-up questions when needed. Additionally, the cosine similarity score for the ASK content exceeds 0.807, while the Meteor score is above 0.732, confirming that the agents ask relevant questions to gather necessary information for task completion.

| Task Setting | Operations | | | ASK Timing | | | ASK Content | |
|---|---|---|---|---|---|---|---|---|
| | SSR | SR | SSR before / after | Precision | FPR | F1 | CosSim | Meteor |
| SpiritSight-Agent-8B | | | | | | | | |
| w/ Original Task | 67.6 | 24.2 | 65.7 / 70.2 | | - | | - | |
| w/ Incomplete Task | 59.6 | 16.7 | 60.8 / 60.2 | | - | | - | |
| | 88.2% | 69.0% | 92.5% / 85.8% | | | | | |
| + ASK | 62.5 | 18.7 | 51.3 / 70.3 | 0.463 | 0.091 | 0.454 | 0.807 | 0.732 |
| | 92.5% | 77.3% | 78.1% / 100.1% | | | | | |
| + Dual Eval | **67.2** | **23.1** | **64.2 / 68.9** | **0.935** | **0.005** | **0.603** | **0.807** | **0.732** |
| | **99.4%** | **95.5%** | **97.8% / 98.1%** | **+0.472** | **-0.086** | **+0.149** | | |
| Qwen2.5VL-3B | | | | | | | | |
| w/ Original Task | 55.4 | 10.0 | 56.0 / 55.9 | | - | | - | |
| w/ Incomplete Task | 51.3 | 8.7 | 51.6 / 52.3 | | - | | - | |
| | 92.6% | 87.0% | 92.1% / 93.6% | | | | | |
| + ASK | 52.6 | 9.5 | 47.9 / 56.2 | 0.574 | 0.055 | 0.487 | 0.832 | 0.750 |
| | 94.9% | 95.0% | 85.5% / 100.5% | | | | | |
| + Dual Eval | **55.1** | **10.6** | **54.9 / 56.2** | **0.947** | **0.004** | **0.585** | **0.832** | **0.750** |
| | **99.5%** | **106.0%** | **98.0% / 100.2%** | **+0.373** | **-0.051** | **+0.098** | | |

Table 1: Comparison of agents' performance on original and incomplete task descriptions, using ASK actions, and using dual-stream evaluation on AndroidControl-Navi*plus* dataset. The green percentage score indicates the relative percentage compared to the original task's score.

## 5 Related Work

### 5.1 Conversational AI Agents

Conversational AI agents emulate human conversations by understanding intentions and interacting with the provided environment to complete tasks or answer questions. The application of (multimodal) LLMs as conversational agents in various fields has garnered considerable attention, as they demonstrate remarkable performance in tasks such as decision-making (e.g. FILM (Min et al., 2021), ReAct (Yao et al., 2022)), tool usage (e.g. Toolformer (Schick et al., 2023), ToRA (Gou et al., 2023)), real-world interaction (e.g. DEPS (Wang et al., 2023), LABOR (Chu et al., 2024)), and multi-agent collaboration (e.g. CoMM (Chen et al., 2024a), L2MAC (Holt et al., 2024)).

We propose an interactive agent task to aid GUI automation, exploring the capability of conversational agents in GUI scenarios.

### 5.2 Conversational web navigation

Conversational web navigation is a novel task recently presented by WebLINX (Lù et al., 2024), wherein humans provide task descriptions to agents section by section, with each containing two or three actions. MT-Mind2Web (Deng et al., 2024b) proposed synthesizing conversational navigation data with existing GUI trajectory datasets by breaking down the full task description into lower-level instructions.

Some concurrent works, like AutoGLM (Liu et al., 2024b) and CogAgent-V2 (Hong et al., 2024), demonstrated an interesting ability to judge the sensibility of the next action and remind users to double-check the model-generated action. They also empowered the agents with the ability to inquire about supposedly missing information in task descriptions through immediate long-term navigation planning. These features are briefly mentioned in their blogs and are worth further research.

In contrast, we propose a straightforward yet effective method to enable GUI automation agents to actively interact with human users for missing task information. Our method obviates the need for long-range CoAT processing and reflection, while also remaining compatible with this format.

## 6 Conclusion

Through this work, we introduced a novel Self-Correction GUI Navigation task, enlightening the ability of GUI automation agents to natively interact and complete missing information when faced with ambiguous user tasks. Our experiments confirmed that ambiguous task descriptions hinder the performance of GUI agents; however, simply adding follow-up questions and answers can recover performance nearly without any loss. Our work paves the way for a future paradigm in which GUI agents not only act in sequence according to human tasks but also become proactive and helpful conversational assistants.

## Limitations

**Practiced only on offline GUI navigation datasets, Future works should evaluate on online benchmarks.** Our current Navi*plus* dataset only involves offline GUI navigation datasets as sources. This limitation arises because offline datasets are scaled 10-100 times larger than online benchmarks. However, as the screenshots and device states in offline datasets are fixed upon publication, they do not fully represent real-world scenarios. Future works should explore how the Self-Correction GUI Navigation task performs on online benchmarks, taking into account factors such as dataset scale and cross-dataset generalization.

**Practiced only on mobile and web platforms, Future works should involve more platforms.** Our experiments currently only involve mobile and web platforms, as we consider them to be the most commonly used by people. Nevertheless, future work should explore the Self-Correction GUI Navigation task on other platforms, such as desktop operating systems.

**Practiced only on ambiguous tasks with one step's information missed, Future works should explore more situations.** We proposed the Self-Correction GUI Navigation task and evaluated it based on the base case where only one key piece of information is omitted by the user in a single step. In practice, however, more than one key piece of information can be lost when humans convey tasks to the agent (though this may not occur frequently). These scenarios should be explored in future work.

**Practiced only with English datasets, Future works should involve broader language sources.** Our experimental datasets only contain English, which could potentially introduce language and cultural biases in the agent model. Currently, GUI navigation datasets in various languages are starting to emerge, and they could serve as complementary materials for future experiments.

**Future work should also discuss how the Self-Correction GUI Navigation task can be integrated with the planning capability.** We identified that our proposed Self-Correction GUI Navigation task can be seamlessly integrated with the navigation planning and reflection framework proposed by our concurrent work, CogAgent-V2. Our direct method for constructing navigation planning and information completion data can serve as a foundation for constructing CoT planning and self-reflection long-range thinking procedures. Although we have addressed its relevance in Section X, more discussion and experiments should be included in future work.

## References

Andrea Burns, Deniz Arsan, Sanjna Agrawal, Ranjitha Kumar, Kate Saenko, and Bryan A Plummer. 2022. A dataset for interactive vision-language navigation with unknown command feasibility. In *European Conference on Computer Vision*, pages 312–328. Springer.

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

Pei Chen, Boran Han, and Shuai Zhang. 2024a. Comm: Collaborative multi-agent, multi-reasoning-path prompting for complex problem solving. *arXiv preprint arXiv:2404.17729*.

Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, et al. 2024b. Guicourse: From general vision language models to versatile gui agents. *arXiv preprint arXiv:2406.11317*.

Xingyu Chen, Zihan Zhao, Lu Chen, Danyang Zhang, Jiabao Ji, Ao Luo, Yuxuan Xiong, and Kai Yu. 2021. Websrc: A dataset for web-based structural reading comprehension. *arXiv preprint arXiv:2101.09465*.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. 2024c. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*.

Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, et al. 2024d. How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites. *arXiv preprint arXiv:2404.16821*.

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

Kun Chu, Xufeng Zhao, Cornelius Weber, Mengdi Li, Wenhao Lu, and Stefan Wermter. 2024. Large language models for orchestrating bimanual robots. *arXiv preprint arXiv:2404.02018*.

Mostafa Dehghani, Basil Mustafa, Josip Djolonga, Jonathan Heek, Matthias Minderer, Mathilde Caron, Andreas Steiner, Joan Puigcerver, Robert Geirhos, Ibrahim M Alabdulmohsin, et al. 2023. Patch n'pack: Navit, a vision transformer for any aspect ratio and resolution. *Advances in Neural Information Processing Systems*, 36:2252–2274.

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

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

Yang Deng, Xuan Zhang, Wenxuan Zhang, Yifei Yuan, See-Kiong Ng, and Tat-Seng Chua. 2024b. On the multi-turn instruction following for conversational web agents. *arXiv preprint arXiv:2402.15057*.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Minlie Huang, Nan Duan, Weizhu Chen, et al. 2023. Tora: A tool-integrated reasoning agent for mathematical problem solving. *arXiv preprint arXiv:2309.17452*.

Samuel Holt, Max Ruiz Luyten, and Mihaela van der Schaar. 2024. L2mac: Large language model automatic computer for extensive code generation. In *The Twelfth International Conference on Learning Representations*.

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

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Zhiyuan Huang, Harry Ziming Cheng, Junting Pan, and Mingjie Zhan. 2024. Spiritsight agent: Advanced GUI agent with one look.

Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem Alshikh, and Ruslan Salakhutdinov. 2024. Omniact: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web. *arXiv preprint arXiv:2402.17553*.

Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. 2024a. On the effects of data scale on computer control agents. *arXiv preprint arXiv:2406.03679*.

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

Zhangheng Li, Keen You, Haotian Zhang, Di Feng, Harsh Agrawal, Xiujun Li, Mohana Prasad Sathya Moorthy, Jeff Nichols, Yinfei Yang, and Zhe Gan. 2024b. Ferret-ui 2: Mastering universal user interface understanding across platforms. *arXiv preprint arXiv:2410.18967*.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. 2024. Showui: One vision-language-action model for generalist gui agent. In *NeurIPS 2024 Workshop on Open-World Agents*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

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

Yuhang Liu, Pengxiang Li, Zishu Wei, Congkai Xie, Xueyu Hu, Xinchen Xu, Shengyu Zhang, Xiaotian Han, Hongxia Yang, and Fei Wu. 2025. Infiguiagent: A multimodal generalist gui agent with native reasoning and reflection. *arXiv preprint arXiv:2501.04575*.

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

Xing Han Lù, Zdeněk Kasner, and Siva Reddy. 2024. Weblinx: Real-world website navigation with multi-turn dialogue. *arXiv preprint arXiv:2402.05930*.

So Yeon Min, Devendra Singh Chaplot, Pradeep Ravikumar, Yonatan Bisk, and Ruslan Salakhutdinov. 2021. Film: Following instructions in language with modular methods. *arXiv preprint arXiv:2110.07342*.

Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, et al. 2024. Webcanvas: Benchmarking web agents in online environments. *arXiv preprint arXiv:2406.12373*.

Qwen. 2025. Qwen2.5-vl.

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

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

Qiushi Sun, Kanzhi Cheng, Zichen Ding, Chuanyang Jin, Yian Wang, Fangzhi Xu, Zhenyu Wu, Chengyou Jia, Liheng Chen, Zhoumianze Liu, et al. 2024. Os-genesis: Automating gui agent trajectory construction via reverse task synthesis. *arXiv preprint arXiv:2412.19723*.

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

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Shawn Ma, and Yitao Liang. 2023. Describe, explain, plan and select: interactive planning with llms enables open-world multi-task agents. *Advances in Neural Information Processing Systems*, 36:34153–34189.

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

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

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

Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. 2024. Ferret-ui: Grounded mobile ui understanding with multimodal llms. In *European Conference on Computer Vision*, pages 240–255. Springer.

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

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

## A Ethical Considerations

GUI automation agents have significant societal, security, and privacy implications. On one hand, they can free humans from repetitive operational tasks on digital devices and enhance work efficiency. On the other hand, if they fall into malicious hands, GUI agents could be misused to bypass anti-fraud systems or manipulate software to achieve harmful or unintended results. Additionally, GUI agents may make mistakes while performing tasks, leading to unacceptable results or unwanted side effects. There is also the risk of private information being leaked if proper data collection regulations are not in place. For these reasons, GUI automation agents must be fully regulated to ensure their broader use serves the social good.

## B Extanded Related Work

### B.1 GUI Navigation Agents and Datasets

Research on GUI navigation agents is gaining popularity, and the resulting works are becoming more diverse and powerful. The recent introduction of Vision Language Model (VLM) based methods, such as SeeClick (Cheng et al., 2024) and SeeAct (Zheng et al., 2024), has revolutionized the original text-only Large Language Models (LLMs) methods (Deng et al., 2024a) by providing more compact and informative screenshots as primary context sources.

Abundant data resources have been gathered to motivate the comprehensive development of VLM-powered GUI agents. Datasets like SeeClick(Cheng et al., 2024), GUICourse(Chen et al., 2024b), AMEX(Chai et al., 2024), SpiritSight(Huang et al., 2024), FerretUI-v1(You et al., 2024), FerretUI-v2(Li et al., 2024b), OS-ATLAS(Wu et al., 2024), ShowUI(Lin et al., 2024), AguVis(Xu et al., 2024) have collected large-scale screenshots annotated with element content and locations to enhance agents' GUI element grounding capability. Other works such as RICOSCA(Deka et al., 2017), Widget Captioning(Li et al., 2020), Screen2Words(Wang et al., 2021), ScreenQA(Hsiao et al., 2022), WebSRC(Chen et al., 2021), GUICourse(Chen et al., 2024b) have constructed screenshot-conditioned question-answering data to enhance the agents' domain knowledge in GUI environments. Furthermore, AMEX(Chai et al., 2024), GUICourse(Chen et al., 2024b), SpiritSight(Huang et al., 2024) have annotated the functionality of GUI elements to enable smoother generalization from element-level tasks to navigation tasks. Studies like Mind2Web(Deng et al., 2024a), OmniAct(Kapoor et al., 2024), GUICourse(Chen et al., 2024b), MoTIF(Burns et al., 2022), AITW(Rawles et al., 2024), GUI-Odyssey(Lu et al., 2024), AMEX(Chai et al., 2024), AndroidControl(Li et al., 2024a), OS-Genesis(Sun et al., 2024) have annotated real GUI navigation trajectories on various platforms for training and benchmarking practical GUI automation agents. AITZ(Zhang et al., 2024), AutoGLM(Liu et al., 2024b), CogAgent-V2(Hong et al., 2024), InfiGUIAgent(Liu et al., 2025) have augmented agents' task planning capabilities by using CoAT approaches and self-reflections to empower scaling at inference-time.

We continue to advance the development of GUI automation agents by addressing the unavoidable problem of ambiguous user task presentations, thereby enhancing the agents' capabilities and flexibility in real-world application scenarios.

## C Task Formulation

Following WebCanvas(Pan et al., 2024)'s symbol that a real-world GUI task episode $E$ is $(S, A, T, O)$, where:

- $S$ is the state space indicating the device status,
- $A$ is the action space,
- $T : S \times A \rightarrow S$ is the deterministic transition function,
- $O$ denotes the state observation space.

We first formulate a GUI navigation task as:

$$\text{Navi}(E, \text{Task}) \rightarrow \begin{cases} \{i_1, i_2, \ldots, i_n\}, \\ \{a_1, a_2, \ldots, a_n\}, \\ \{s_0, s_1, \ldots, s_n\} \end{cases}$$

where Task is the user's high-level instruction guiding the task, $a_i \in A$, $s_i \in S$, and $i_i \in I$ is the low-level instruction corresponding to the actions at each step.

In practical GUI navigation tasks, the execution process consists of both **informative steps**, which provide choices and guide decision-making in high-level instructions, and **transactional steps**, which represent inevitable actions necessary for task completion. Informative steps introduce decision branches (e.g., selecting options in a form), while transactional steps ensure the smooth flow of the task by performing essential operations (e.g., clicking confirmation buttons, and closing windows).

If some key information is omitted in the Task (becomes Task'), some informative steps may not be feasible, thus making the whole task less likely to be successful. To address this, we extend the action space $A$ by introducing an ASK action, which enables the agent to inquire about missing or ambiguous information. This modification results in the formulated Self-Correction GUI Navigation Task:

$$\text{Navi}(E, \text{Task'}) \rightarrow \begin{cases} \{i_1, i_2, \ldots, i_n\}, \\ \{a_1, a_2, \ldots, a_n\}, \\ \{s_0, s_1, \ldots, s_n\} \end{cases}$$

where the action space $A$ is adjusted to incorporate the ASK action.

## D Data Construction Details

**Low-level Instruction Completion** We start by generating low-level instructions for each step in the trajectories. This intention consists of an operational intention and an element description (e.g. the 'OK' button or the 'plus' button of the second product). The current action, along with a screenshot and the bounding box of the interacted element, is provided to InternVL2.5-26B(Chen et al., 2024c) to generate the low-level instructions for that action.

**Informative Step Decision** We then hired DeepSeek-V3(Liu et al., 2024a) to decide whether a step is an informative step or a transactional step. DeepSeek-V3 achieves a satisfactory accuracy rate when making the judgment, with 90% of the data passing human verification.

**Formation of Ambiguous Tasks** We present the full task description as a reference to DeepSeek-V3 and provide it with the informative steps to be removed. The model is prompted to output an ambiguous task description that excludes the selected informative steps while maintaining all other information and the original style. A QA pair simulating the agent's follow-up question and the user's answer is also generated for each informative step. In practice, the judgment of informative steps and the formation of ambiguous tasks are completed in one API call to minimize costs.

## E Experimental Setup In Detail

### E.1 Baseline Models

**Qwen2.5-VL (Qwen, 2025)** is a high-performance MLLM that natively incorporates computer use and phone use capabilities. It supports naive dynamic resolution (Dehghani et al., 2023) that can handle arbitrary image resolutions and map them into visual tokens linear to the number of image pixels.

**SpiritSight Agent (Huang et al., 2024)** is a pure-vision LLM-based GUI agent built upon InternVL2 (Chen et al., 2024d). It supports dynamic high-resolution max to 12 tiles of 448 × 448 images. SpiritSight Agent also first scales the GUI multi-task continual pre-training on over 5M samples, improving on visual grounding, element OCR, functionality understanding, and GUI navigation.

### E.2 Implementation Details

We use the original data splits from AndroidControl (Li et al., 2024a) and Mind2Web (Deng et al., 2024a). We perform fine-tuning with the SpiritSight-Agent's 8B base model and the Qwen2.5-VL's 3B model adopting LoRA (Hu et al., 2021). The LoRA rank is set to 64 for both SpiritSight-Agent-8B and Qwen2.5-VL-3B models. We fine-tune the models for one epoch, using a batch size of 64. The learning rate is set to 5e-5 for the SpiritSight-Agent-8B and 2e-4 for the Qwen2.5-VL-3B. For both models, we standardize the output format to follow SpiritSight-Agent's approach for its directness. All of our data and models are open-sourced under the Creative Commons Attribution 4.0 International License.

### E.3 Computational Budget

We train all models using the PyTorch library on an 8-GPU setup with NVIDIA A800-SXM4-80GB GPUs, leveraging the NVIDIA CUDA platform. The InternVL2.5 model is deployed on a single NVIDIA A800-SXM4-80GB GPU, and we use DeepSeek-Chat's official API for data generation. According to the DeepSeek platform, the API consumption is 130.1 million tokens.

## F Matrics Calculation Details

### F.1 SSR (Step Success Rate)

For each episode $e$, we compute the ratio of correct steps $C(e)$ to the total number of steps $T_e$. The Step Success Rate (SSR) is then computed as the average of this ratio over all episodes:

$$\text{SSR} = \frac{1}{N}\sum_{e=1}^{N}\frac{C(e)}{T_e}$$

Where: $N$ is the total number of episodes. $C(e)$ is the number of correct steps in episode $e$. $T_e$ is the total number of steps in episode $e$.

### F.2 SSR before/after

For each episode $e$, we compute the Step Success Rate (SSR) for the steps before and after a fixed ASK step at position $k$. The SSR before and after are then calculated as the averages across all episodes.

$$\text{SSR}_{\text{before}} = \frac{1}{N}\sum_{e=1}^{N}\frac{C_{\text{before}}(e)}{T_{\text{before}}(e)}$$

Where: $C_{\text{before}}(e)$ is the number of correct steps before the ASK step in episode $e$. $T_{\text{before}}(e)$ is the total number of steps before the ASK step in episode $e$.

Similarity, for the steps after the ASK step:

$$\text{SSR}_{\text{after}} = \frac{1}{N}\sum_{e=1}^{N}\frac{C_{\text{after}}(e)}{T_{\text{after}}(e)}$$

Where: $C_{\text{after}}(e)$ is the number of correct steps after the ASK step in episode $e$. $T_{\text{after}}(e)$ is the total number of steps after the ASK step in episode $e$.

### F.3 SR (Success Rate)

For each episode $e$, we check whether each step $k$ is correct. If all steps in an episode are correct, the episode is considered successful. The Success Rate (SR) is then computed as the ratio of successful episodes to the total number of episodes:

$$\text{SR} = \frac{1}{N}\sum_{e=1}^{N}\mathbb{I}_{\text{success}}(e)$$

Where: $N$ is the total number of episodes. $\mathbb{I}_{\text{success}}(e)$ is the indicator function, where $\mathbb{I}_{\text{success}}(e) = 1$ if episode $e$ is successful (all steps correct), and $\mathbb{I}_{\text{success}}(e) = 0$ otherwise.

### F.4 CosSim (Cosine Similarity)

Given two ASK sentences $s_1$ and $s_2$, we compute their embeddings $\mathbf{e}_1$ and $\mathbf{e}_2$ using a Sentence Transformer model:

$$\mathbf{e}_1 = \text{SentenceTransformer}(s_1) \quad (1)$$

$$\mathbf{e}_2 = \text{SentenceTransformer}(s_2) \quad (2)$$

| # Del | AndroidControl |  |  |  |  |
|---|---|---|---|---|---|
|  | SS-Agent-8B |  | Qwen2.5VL-3B |  | PaLM2S |
|  | SSR | SR | SSR | SR | SSR |
| 0 | 67.6 | 24.2 | 55.4 | 10.0 | 64.8 |
| 1 | 59.6 | 16.7 | 51.3 | 8.7 | - |
|  | -11.8% | -31.0% | -7.4% | -13.0% |  |
| 2 | 55.8 | 12.4 | 46.8 | 5.0 | - |
|  | -17.5% | -48.8% | -15.5% | -50.0% |  |

| # Del | Mind2Web |  |  |  |  |  |
|---|---|---|---|---|---|---|
|  | SS-Agent-8B |  | Qwen2.5VL-3B |  | FlanT5XL |  |
|  | SSR | SR | SSR | SR | SSR | SR |
| 0 | 45.3 | 8.8 | 46.3 | 8.3 | 43.5 | 4.4 |
| 1 | 26.3 | 2.1 | 25.7 | 1.7 | - | - |
|  | -41.9% | -76.1% | -44.5% | -79.5% |  |  |

Table 2: Comparison of agent performance when information from varying numbers of steps is excluded from original tasks. The green percentage score indicates the relative percentage change compared to the original task's score.

The cosine similarity between the two embeddings is then calculated as:

$$\text{CosSim}(\mathbf{e}_1, \mathbf{e}_2) = \frac{\mathbf{e}_1 \cdot \mathbf{e}_2}{\|\mathbf{e}_1\|\|\mathbf{e}_2\|}$$

Where: $\mathbf{e}_1 \cdot \mathbf{e}_2$ is the dot product of the two embeddings. $\|\mathbf{e}_1\|$ and $\|\mathbf{e}_2\|$ are the magnitudes of the embeddings.

## G Extended Experiments

### G.1 More excluded steps and dataset

Table 2 reports the performance of GUI agents on the original AndroidControl and Mind2Web datasets, as well as on our Navi*plus* datasets with information missing in the descriptions. Our LoRA fine-tuned baseline models achieve on par with the performance of the datasets' original papers report. Furthermore, with the generated ambiguous task descriptions in our Navi*plus* data, the baseline models' performance drops significantly as more steps are removed from the full task descriptions. On AndroidControl-Navi*plus*, the Step SR decreased by over 15%, and SR was nearly halved compared to the full tasks. A similar trend was observed in Mind2Web-Navi*plus*.

### G.2 Hyper-parameter analysis

We conducted a hyperparameter analysis on SpiritSight-Agent-8B with LoRA ranks set to 16 and 64, training for epochs ranging from 0 to 2, and recorded steps from 200 to 2600 at intervals of 200. For a fair comparison with the baseline models from the original papers, we report the results of training 1 epoch. The results are presented in Figure 4

## H Dataset Statistics

### H.1 Task Length Distribution

Figure 5 shows the distributions of step lengths for both the total number of tasks and the number of tasks annotated as informative (related) in the Navi*plus* dataset. On average, the number of related steps is fewer than the number of steps for full tasks.

### H.2 Related Steps Percentage Analyzation

Figure 6 shows the distributions of the task percentages for 'enough' and 'not enough' steps, which are used to generate ambiguous tasks.

## I Prompt Templates

### I.1 Low-level Instruction Completion

For prompt template for low-level instruction completion step, see Figure 7.

### I.2 Informative Step Decision & Formation of Ambiguous Tasks

For prompt template for informative step decision and formation of ambiguous tasks step, see Figure 8.

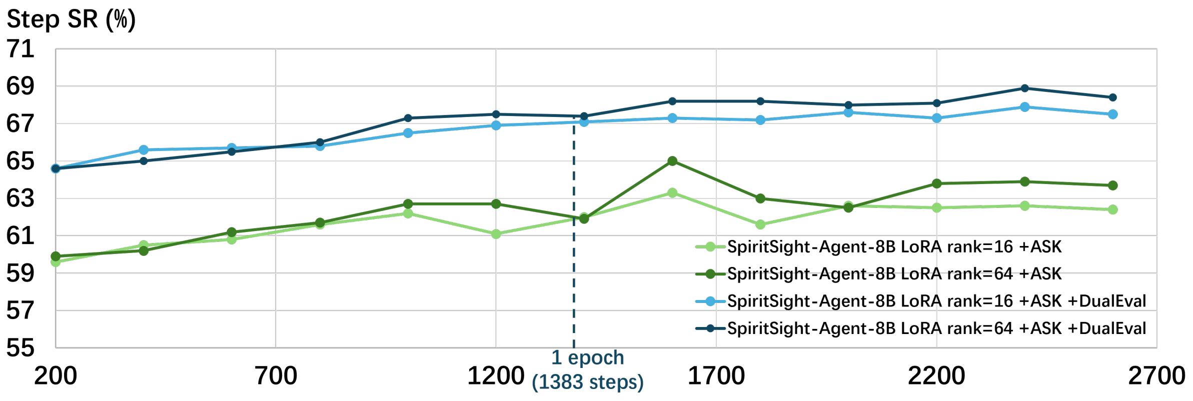
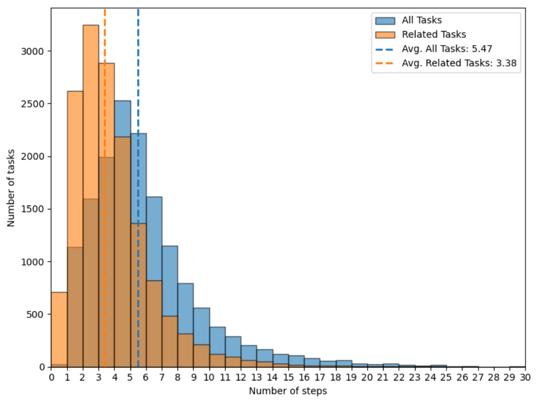
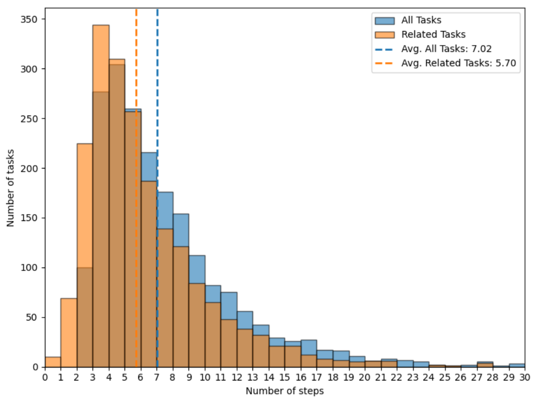


Figure 4: Hyper-parameter analysis on SpiritSight-Agent-8B with LoRA rank setting 16 and 64, and training epochs from 0-2 (steps from 200 to 2600).

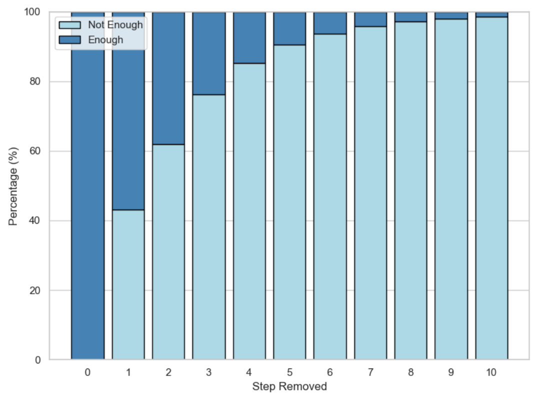


(a) AndroidControl-Navi*plus*.

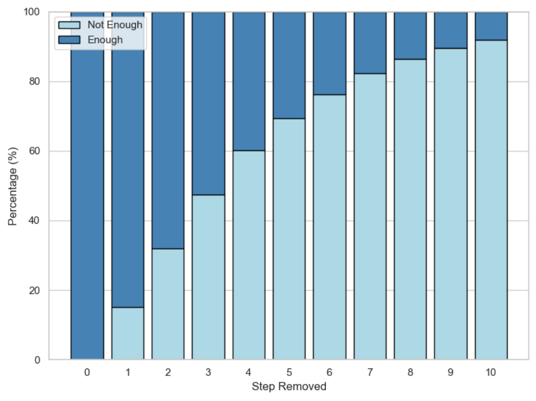


(b) Mind2Web-Navi*plus*.

Figure 5: The distribution of task lengths for the Full Task and the number of annotated Related Steps in Navi*plus* dataset.



(a) AndroidControl-Navi*plus*.



(b) Mind2Web-Navi*plus*.

Figure 6: The percentage distribution of tasks, both with and without enough steps to be removed, to generate ambiguous tasks in the Navi*plus* dataset.

Prompt Template for Low-level Instruction Completion

**Prompt:**
<image>
I will provide you with a full task description that is completed by performing a series of actions within web browser. These actions are performed sequentially as steps, and together they result in an operation trajectory for completing the task. Your mission is to generate a step instruction with the given a action code and the current screenshot content.

The action code indicates the clicked content or inputted content. The generated Step Instruction should be concise, directly related to the action code and its purpose. If the action code is CLICK(UnKnown), you need to identify the content in the red bbox to know what is exactly clicked.

Please output in JSON format, structured as follows:
{
"Full Task": "<Complete task description here>",
"Step Action Code": "<the provided action code>",
"Red Bbox Content": "<the content inside the red bbox>",
"Step Instruction": "<Generate a Step Instruction here>",
}
Do not output other explanations.

## Full Task: {task}
## Step Action Code: {action}

Figure 7: Prompt Template for Low-level Instruction Completion.

Prompt Template for Informative Step Decision & Formation of Ambiguous Tasks

**Prompt:**
You are a task simplification Specialist. You should maintain outputting accurate sentences. I will provide you with a full task description and one step instruction in this trajectory. The full task description is a GUI operation task completed by sequentially performing a series of steps within mobile phone apps.

Your mission is to create a **Simplified Task Description** by removing the information contained in a selected step instruction from the full task description.

### Please follow this step by step solution:
1. Repeat the full task description and the selected step instruction to make sure you understand the input information.
2. Find the overlapping information of the selected step instruction within the full task, base on named entity with specific information.
3. Form the simplified task description by removing the overlapping information from the full task description while making sure the remaining content unchanged. Only remove the related words, or replace specific entity with reference word.
4. Rephrase the generated simplified task description using the same imperative tone and style as the original task if necessary.
5. Generate a follow-up question to simulate as if the agents tries to clarify about the removed information.
6. Generate the human's clarifying answer based on the step instruction removed, do not straightly say the operation, but only say about the intention.

Please ensure that the output is in JSON format, structured as follows:

```
{
"Full Task": "<Complete task description here>",
"Selected Step to Exclude": "<Step to be removed here>",
"Overlapping Information": "<Details of the task description that overlap with the selected step. If no specific information is overlapped, say 'None'>",
"Incomplete Task Description": "<Generated task description without the selected step's information>",
"Rephrased Incomplete Task Description": "<If any rephrasing is required, show the final version here>",
"Follow Up Question": "<Generate a follow-up question that asks about the removed step>",
"Human Answer": "<Generate the human's clarifying answer, do not include operation>"
}
```

## Examples: {Few-shot Examples}

Figure 8: Prompt Template for Informative Step Decision & Formation of Ambiguous Tasks