FingerTip 20K: A Benchmark for Proactive and Personalized Mobile LLM Agents

Qinglong Yang **Haoming Li**

Haotian Zhao **Xiaokai Yan** **Jingtao Ding** **Fengli Xu** **Yong Li**

Abstract

Mobile GUI agents are becoming critical tools for enhancing human-device interaction efficiency, with multimodal large language models (MLLMs) emerging as dominant paradigms in this domain. Current agents, however, are limited to following explicit human instructions, resulting in insufficient capability for proactive intent anticipation. Additionally, these agents fail to leverage the contextual information associated with users during task execution, thereby neglecting potentially vast differences in user preferences. To address these challenges, we introduce the FingerTip benchmark. It contains two new tracks: proactive task suggestions by analyzing environment observation and users' previous intents, and personalized task execution by catering to users' action preferences. We collected unique human demonstrations of multi-step Android device interactions across a variety of everyday apps. These demonstrations are not isolated but are continuously acquired from the users' long-term usage in their real lives, and encompass essential user-related contextual information. Our experiments reveal challenges of the tasks we propose. The model fine-tuned with the data we collected effectively utilized user information and achieved good results, highlighting the potential of our approach in building more user-oriented mobile GUI agents. Our code is open-source at `https://anonymous.4open.science/r/FingerTip-57B8` for reproducibility.

1 Introduction

Recent studies have explored how to utilize multimodal large language models (MLLMs) to construct graphical user interface (GUI) control agents [1–5], with a significant direction being mobile phone UI control agents, as mobile phone UIs are among the most frequently interacted with by humans in daily life. These agents receive a natural language task instruction, such as "Set an alarm for 7:30 for me", and then perceive the device state by observing the device screen (via screenshots or textual UI trees), and generate actions (click, type, scroll, etc.) to interact with the device environment to fulfill human instructions.

Despite rapid progress, currently, most existing LLM-based mobile GUI agents are confined to a completely passive paradigm: they only perform tasks upon receiving a clear instruction. This paradigm restricts their ability to proactively offer task suggestions and assistance in the absence of direct human instructions. If users have to formulate detailed instructions for every intent when interacting with mobile GUI agents, it will significantly increase the cognitive burden of mobile phone usage. Moreover, humans sometimes may not clearly express some latent needs. Therefore, mobile GUI agents need to be more proactive to provide users with more comprehensive and seamless services. Furthermore, the existing agents utilize almost exclusively user instructions as textual information when performing tasks, without taking into account any additional user-related

Preprint. Under review.

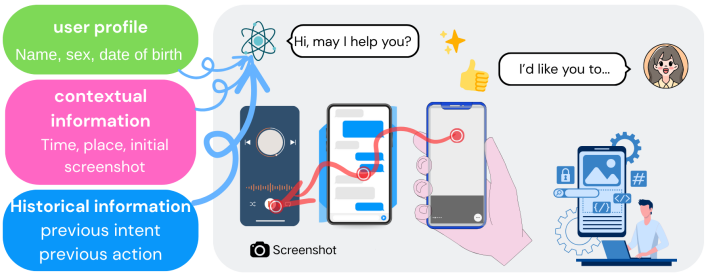


Figure 1: An overview task example in FingerTip.

information (e.g., time and location, user profile, user historical intents and actions), thus failing to provide personalized services to users.

To comprehensively evaluate the proactive and personalized capabilities of mobile GUI agents, we propose the FingerTip benchmark, which includes two new tracks: (i) proactive task suggestion, where the agent needs to integrate the user's past intents and the current environmental state to infer the user's potential current intent; (ii) personalized task execution, where the agent needs to refer to the user's past action preferences to execute current instructions. Since existing benchmarks do not provide such user-related contextual information, we spent two months collecting new diverse data from 91 users in their daily mobile phone usage, including 21,437 episodes covering 506 apps. We then conducted experiments on the FingerTip benchmark to evaluate the capabilities of generalist models and GUI-control agents built on specifically designed models and found that there is still much room for improvement in their proactive and personalized capabilities. We fine-tuned a small model using the collected data and achieved better results.

In summary, the main contributions of this work include:

- We propose the FingerTip benchmark, which includes two brand-new tracks, to evaluate the ability of mobile GUI agents to proactively predict user intents and offer suggestions, as well as their ability to personalize task execution in accordance with user preferences.
- We collect large-scale user-oriented mobile GUI-control data, derived from scenarios in users' daily lives, which includes user-related contextual information and users' long-term mobile phone usage patterns.
- We evaluate the capabilities of generalist models and GUI-control-specific models on the FingerTip benchmark, demonstrating the difficulty of the tracks we propose. The excellent performance of the model fine-tuned with our collected data highlights the potential of our approach in building more proactive and personalized mobile agents.

2 Related work

2.1 Mobile GUI-control datasets and benchmarks

Table 1 compares FingerTip to existing mobile GUI-control datasets and benchmarks. These datasets typically represent each data instance through two core components: a textual task instruction and its corresponding operational demonstration. The demonstration is encoded as a sequence of interface interactions (e.g., clicking, typing, scrolling) accompanied by relevant screenshots. What differentiates them is mainly whether they are single-step (grounding instructions to UI elements on the screen), and whether they have supplemental View Hierarchy (VH) data for each screenshot. These datasets share some common drawbacks. Firstly, their task instructions are either pre-defined by authors and annotators or generated by LLMs, and it is questionable to what extent they can reflect the real intents of people using mobile phones in their daily lives. Additionally, they collect task demonstrations mainly by having annotators operate simulators or connected phones on computers, which is not the real scenario of people using mobile phones. Finally, each data instance is isolated and does not contain any contextual information related to the user.

Table 1: Comparison of FingerTip to existing mobile GUI-control datasets and benchmarks.

Dataset & Benchmark	Platform	#Episode	#App or websites	#Avg steps	#Collection method	UI tree?	Screen?	Profile info?	Context info?	Task setting
AndroidLab	Android	10.5k	-	-	human annotate	yes	yes	no	no	execution
AMEX	Android	8k	110	12.8	human annotate	yes	yes	no	no	execution
AndroidControl	Android	15283	833	5.5	human annotate	no	yes	no	no	execution
AitW	Android	715142	357	6.5	human annotate	yes	yes	no	no	execution
AndroidLab benchmark	Android	138	9	8.6	select from AndroidLab	no	yes	no	no	execution
A3	Android	201	20	-	unknown	yes	yes	no	no	execution
SPHINX	Android	-	100	8.1	human annotate	yes	yes	no	no	execution
SPA-Bench	Android	340	58	-	human annotate	no	yes	no	no	execution
FingerTip	Android	21437	506	11.1	human annotate	yes	yes	yes	yes	proactive task suggestion & personalized task execution

For benchmarks, the success rate is the most commonly used metric, and some studies also consider efficiency and cost. A common approach to assessing the success of a task is to determine whether essential states have been reached [6–8]. Some studies also compare agents' actions to golden actions [9]. However, these golden actions do not take into account potentially vast differences in user preferences, that is, the action sequences of different users to complete similar tasks may be very different. In addition, current benchmarks have similar task forms, that is, given an existing instruction, how to perform actions to complete it. To the best of our knowledge, there is no mobile GUI agent benchmark that discusses how to proactively suggest tasks based on user-related information when instructions are unknown.

2.2 Mobile GUI-control agents

Mobile GUI agents are designed to understand the UI and automate tasks on mobile apps in a manner similar to that of humans. Current agents leverage the extensive world knowledge and powerful embodied capabilities of multimodal large language models (MLLMs) for complex task planning and reasoning in multi-step GUI-control tasks. One notable approach is to directly guide generalist models like GPT-4v to perform tasks through extensive prompt engineering [3, 10, 11, 1, 4, 12]. However, these methods require meticulously designed prompts to achieve the best results. Another research direction focuses on fine-tuning smaller models [13–17] on GUI-specific datasets to endow them with GUI grounding capabilities and the ability to break down high-level instructions, thereby enhancing their operational efficiency. Despite these advancements, most current agents are still confined to passively following explicit instructions and are unable to proactively predict user needs. Moreover, they do not take into account any user preferences when performing tasks. Some studies focus on proactively clarifying users' ambiguous instructions [18–20]; however, these studies still require users to provide initial instructions. Proactive Agent [21] predicts potential tasks by monitoring user activities and environmental states, but the input is text-only, and the task scenarios are mainly limited to computer or web environments rather than mobile ones.

3 Problem formulation

3.1 Proactive task suggestion

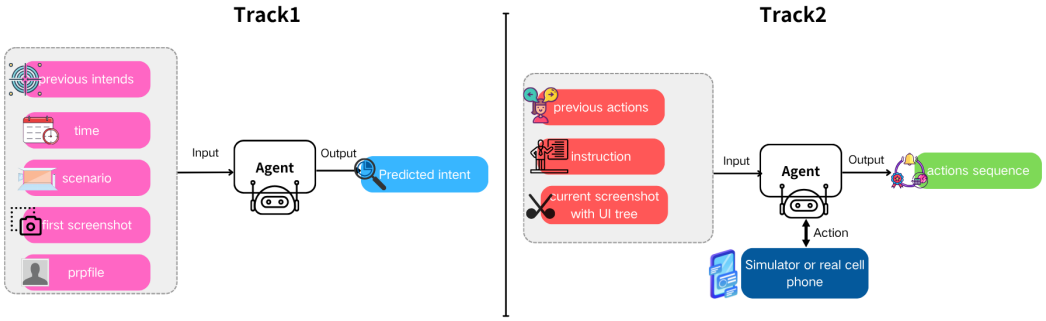


Figure 2: Demonstration of proactive task suggestion and personalized task execution.

In the FingerTip benchmark we propose, unlike the evaluation tasks of traditional mobile GUI agent benchmarks that rely entirely on explicit instructions, we introduce a new task where the agent proactively predicts the user's current intent and proposes tasks suggestion that the user might want to perform, as shown in Figure 2. The agent's task is to generate an intent prediction I based on the user profile U, the current time T, the current scenario S, the user's historical intents I_{history}, and the partial screenshots O observed at present. This can be formalized as:

$$I = f\ (U,\ T,\ S,\ I_{\text{history}},\ O) \tag{1}$$

where f represents the agent. I is a sentence that unambiguously predicts the intent of the user, including all necessary information. It should clearly state the name of the app that the user wants to use, and the final effect that the user wants to achieve. U includes common user attributes such as age, sex, occupation, etc. T represents the current timestamp, accurate to the second. S represents the current scenario. To avoid revealing real location information, we use some common location categories (such as residence, school, office, etc.) to represent it. I_{history} contains the user's historical intents in the recent period, up to 20 items, which may include the potential patterns and preferences of the user's mobile phone usage. O includes the first few screenshots of the user's current actions (e.g., opening the home page of a certain app), which should be able to reveal the initial part of the user's true intent (e.g., Use a certain app to...). It should be noted that we provide at most three such screenshots and at least zero. That is to say, the agent may need to make predictions in the complete absence of O. It is obvious that the fewer the number of screenshots, the more difficult it is to predict the user's true intent. We hope that the agent can utilize the above-mentioned user-related contextual information to infer the user's potential intents, and thereby proactively offer helpful task suggestions.

3.2 Personalized task execution

In addition to proactive task suggestion, we also aim to evaluate the agent's ability to carry out tasks under the condition of explicit instructions, that is, when the user's intent is known. The setting of this part is similar to the existing benchmarks. The difference lies in that we additionally assess the agent's ability to execute tasks in a personalized manner specifically catering to the action preferences of different users. Given user profile U, user instruction I_{true}, user's historical actions A_{history}, agent's action sequence A_{agent}, and the current screenshot O_t and the corresponding accessibility tree AT_t, the agent needs to perform the next action A_{t+1}, and then observe O_{t+1} and AT_{t+1}. This can be formalized as:

$$A_{t+1},\ O_{t+1},\ AT_{t+1} = f\ (U,\ I_{\text{true}},\ A_{\text{history}},\ A_{\text{agent}},\ O_t,\ AT_t) \tag{2}$$

where f represents the agent. I_{true} is equivalent to the user's true intent that needs to be predicted in proactive task suggestion, and here it serves as the instruction to be executed. A_{history} is the complete action sequence of the user when performing a similar task in the past, provided to the agent for in-context learning to imitate the user's action preferences. A_{agent}, on the other hand, is the action sequence $\{A_1, ..., A_t\}$ that the agent has already executed in the current task, helping the agent determine the progress of the task. The agent needs to constantly interact with the mobile phone environment until it believes that the user's instruction I_{true} has been fulfilled. We hope that the final sequence of agent actions A_{agent} can reflect the user's action preferences.

4 The FingerTip benchmark

4.1 Overview

The motivation for FingerTip data collection is to evaluate the dual tracks we have proposed, namely proactive task suggestion and personalized task execution. To this end, the most distinctive feature of the data should be user-oriented, containing sufficient user-related contextual information and being able to reflect the patterns and preferences of users in terms of intents and actions.

4.2 Data collection

The data collection process is shown in Figure 3. We first recruited 91 data collectors (hereinafter referred to as users) using Android phones through crowdsourcing. Each user filled out a questionnaire, which collected their user profiles. Users were required to download an APP developed by us on their

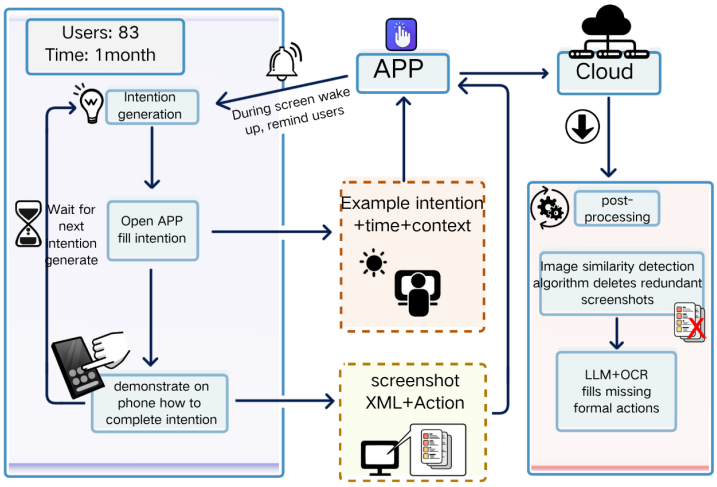


Figure 3: Data collection pipeline.

own daily used phones and use it to collect data. Specifically, whenever users had a real intent to use their phones in their daily lives, they could open the FingerTip APP, record their intent at that moment in one sentence, and select the location category they were in. Then, users needed to switch to the app involved in the intent they recorded and demonstrate the specific action sequence to complete this intent.

The FingerTip APP will automatically upload the intent they fill in (including time and location) and the demonstration process they provide (including screenshot sequences, corresponding accessibility tree XML file sequences, and UI action sequences) to the cloud server. This is regarded as the user collecting one piece of data. The APP may remind the user to collect data when they wake up the phone screen to prevent them from forgetting. Each user is required to use their phone to collect data for a minimum of one month and a maximum of two months, with a maximum of 12 pieces of data uploaded per day. In this way, users can fully customize the data they upload. Intents and demonstrations originate from the user's daily usage scenarios, rather than being pre-defined by researchers or generated by LLMs. Before the formal data collection began, each user underwent training to provide high-quality data. As all users were from mainland China, the data we collected mainly concerned third-party Chinese apps and covered a wide range of device types and Android versions.

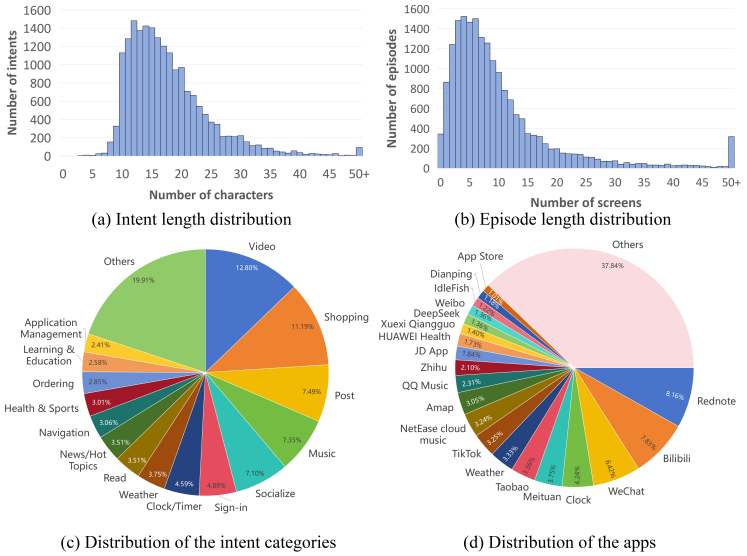
FingerTip APP is developed based on the accessibility features of the Android system. It can automatically record the type and coordinates, as well as optional text descriptions of each user action. The actions we collect are unified into an action space, as shown in Table 2. Among them, $terminate()$ is uniformly added to the last screenshot of all episodes. Due to the incomplete development of accessibility features in some Chinese apps, it may be impossible to automatically capture complete action information on certain interfaces of these apps, returning incorrect action formats that do not belong to the action space, or the returned actions lack coordinates. For these non-compliant actions, we use Qwen-VL-Max to observe the two screenshots before and after the action, and regenerate the action, then use PaddleOCR to check if the coordinates are correct.

4.3 Data statistics

The summary of data statistics is presented in Table 1. Additionally, Figure 4 reports the distribution of user intent length, episode length, intent categories, and app name in all data. The intent category of each intent is determined by DeepSeek-V3 [22].

4.4 Personalized action analysis

To verify the personalized differences in actions among different types of users, we first simply classified users into different categories based on age groups. Then, we randomly sampled one piece



(a) Intent length distribution (b) Episode length distribution

(c) Distribution of the intent categories (d) Distribution of the apps

Figure 4: Dataset statistics and distribution.

Table 2: Actions and Format.

Action	Format
click	coordinates=(x,y), content=”
long_click	coordinates=(x,y), content=”
type	text=”
scroll	coordinates=(x,y), direction=’down or up or right or left’
navigate_back	()
navigate_home	()
navigate_recent	()
wait	()
finish	()

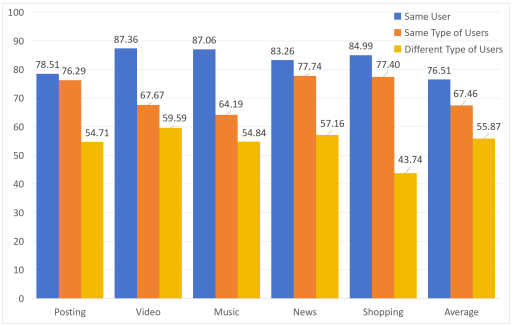


Figure 5: Personalized action analysis.

of data from each of the 40 intent categories. For the action sequence of such a piece of data, we calculated the Levenshtein similarity with the action sequence of the most intent-similar data from (i) the same user, (ii) the same type of users, and (iii) different type of users. All similarities were normalized to [0, 100] and plotted in Figure 5. It can be seen that even when performing similar intents, the similarity of action sequences with different types of users is significantly lower than that of the same user or the same type of users.

4.5 Data splits

Table 3: Details on FingerTip train, validation and test splits.

Split	# Episodes	# Screens	# Apps	# Categories
Train	16000	177674	460	40
Vali	4411	32859	41	0
Test-suggestion	1000	10412	155	38
Test-execution	200	2074	68	31

We created a training set, a validation set, and two test sets. The number of episodes and features in these sets are detailed in Table 3. Please note that the two test sets contain partially overlapping episodes. The test sets were formed by randomly sampling the last 20% of the data of each user, sorted by time, and then concatenated to ensure coverage of all users and that the proportion of data

from each user in the test sets is equal to their proportion in all data. These test sets were used in subsequent experiments.

5 Experiments

We conducted experiments on some generalist models and some GUI-control agents built on specifically designed models, evaluating their capabilities on the two tracks proposed in the FingerTip benchmark and assessing their performance under different task difficulties. Additionally, we fine-tuned a model using the collected data.

5.1 Experimental setup

Proactive task suggestion The LLMs we experiment with in this track include GPT-4.1, Qwen-VL-Max, DeepSeek-VL2 [23] and Qwen-2.5-VL-7B [24]. We set the temperature to zero for all models. For proactive task suggestion, the agent only needs one query to output the predicted intent. Since this is a brand new track proposed in our benchmark, there is no mature agent design available for direct use. We have designed a simple prompt to provide to all models evaluated in this track. This prompt contains all necessary inputs (see Section 3.1).

Personalized task execution In this track, in addition to the four generalist models mentioned above, we also experiment with three GUI-control agents built on specifically designed models, including Aguvis-7B [16], CogAgent-9B [15] and UI-TARS-1.5-7B [14]. For personalized task execution, the agent needs to interact with the environment in multiple steps to fulfill the user's instructions. We connect a physical phone to the computer via USB and use Android Debug Bridge (ADB) to provide this environment. Using an emulator would be a more convenient approach, but due to strict app control measures, most Chinese apps can only run on physical phones rather than emulators. For the generalist models, we designed a simple prompt to guide their output of the next action, with the action space consistent with Table 2. This prompt contains all necessary inputs (see Section 3.2). For the three GUI-control agents, they have specific format requirements for input and output. To ensure normal output effects, their original prompts were used, and the input information in Section 3.2 was uniformly integrated into these original prompts. Their output was converted into a form consistent with the action space.

Metrics In proactive task suggestion track, the goal of the agent is to maximize the textual similarity between the output and the user's true intent. We use a pre-trained model, paraphrase-multilingual-MiniLM-L12-v2 [25], to convert the agent's output and the user's true intent into embedding vectors and calculate their cosine similarity S_1. And, we calculate the Levenshtein similarity S_2 of these two strings. Both similarities are normalized to the range of [0, 1]. Finally, we take $Sim_1 = (S_1 + S_2)/2$ to comprehensively represent the text similarity. In addition to this numerical metric, we also use DeepSeek-V3 [22] to directly determine whether the agent's output and the user's true intent can be regarded as the same intent and provide a binary value to evaluate whether the agent successfully predicted the user's intent, thereby calculating the success rate SR_1.

In personalized task execution track, the primary goal of the agent is to successfully execute the user's instructions. We calculate the final success rate SR_2 by manually checking whether the environment state when the agent outputs $finished()$ matches the user's instructions. In addition, when the agent steps exceed 2.5 times the golden steps, the task is automatically considered a failure. Note that the path to successfully execute the user's instructions is not unique. The agent should also make the action sequence reflect the user's action preferences as much as possible. We do not require the agent's action at each step to be exactly the same as the user's golden action. Instead, we calculate the Levenshtein similarity S_I of the agent's complete action sequence and the user's complete action sequence as two strings. Then, following the approach in Section 4.4, we take the data from the different type of users that is most similar to the current user's intent, and calculate the Levenshtein similarity S_{II} of the agent's complete action sequence and this data's complete action sequence. Finally, we take the value $Sim_2 = S_I/S_{\text{II}}$. It is obvious that the larger this value is, the more similar the agent's action sequence is to that of the current user, and the more different it is from that of different type of users. In addition, we measure execution efficiency by comparing the agent steps with the user's golden steps to calculate the step ratio when the agent successfully execute the user's

instructions. For the two tracks, we also tallied the average time and token count consumed per query to assess the model's cost.

5.2 Overall Performance

The overall performance of the models we evaluated in proactive task suggestion is shown in Table 4. Note that here we set the number of O (the first few screenshots of the user's current actions) to 0. This makes the task quite challenging. GPT-4.1 performed the best among the generalist models, achieving $SR_1 = 7.2$ and $Sim_1 = 0.35$, but it also took the longest time. From SR_1, it can be intuitively seen that the success rate of all models in predicting the user's intent is very low, less than 10%. We consider it normal. The models almost only rely on 20 previous intents of the user to predict the user's current intent, which is difficult to achieve without prior knowledge of the user's characteristics and intent patterns.

Table 4: Overall performance of proactive task suggestion. Bold indicates the best performance.

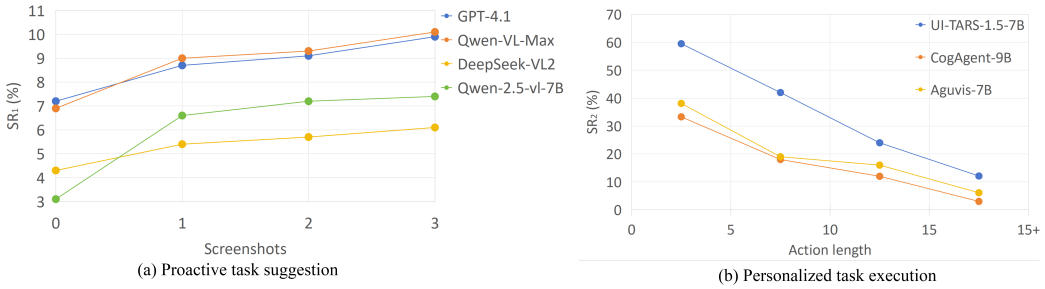
Model	SR_1 (%)	Sim_1	Time (sec)	Token
GPT-4.1	**7.2**	**0.35**	5.64	796
Qwen-VL-Max	6.9	0.33	1.98	950
Deepseek-VL2	4.3	0.25	**0.71**	**743**
Qwen-2.5-vl-7B	3.1	0.25	0.78	943

Table 5: Overall performance of personalized task execution. Bold indicate the best performance.

Model	SR_2 (%)	Sim_2	Step Ratio	Time (sec)	Token
GPT-4.1	**5.5**	0.98	**1.98**	8.02	2912
Qwen-VL-Max	4.5	**1.07**	2.06	4.17	2304
Deepseek-VL2	1.0	0.93	2.19	**3.46**	**2130**
Qwen-2.5-vl-7B	1.5	0.95	2.16	3.66	2213
Aguvis-7B	20.5	1.02	1.38	**6.86**	2494
CogAgent-7B	18.0	0.92	1.73	12.54	2808
UI-TARS-1.5-7B	**38.5**	**1.06**	**1.22**	10.15	**2440**

The overall performance of the models we evaluated in personalized task execution is shown in Table 5. GPT-4.1 and UI-TARS-1.5-7B achieved the best performance among the generalist models and GUI-control models respectively. The SR_2 of the generalist models were all very low, mainly due to their lack of precise GUI grounding ability, which led to incorrect UI coordinates being output even when they could correctly analyze the next action, thus failing to interact with the environment accurately. In contrast, the GUI-control models, having undergone targeted training, had stronger abilities to execute instructions and interact with the UI environment, resulting in higher SR_2, with UI-TARS-1.5-7B having the highest at 38.5. However, the Sim_2 of all models were approximately 1, indicating that the agent's action sequence did not favor either the current user or different types of users. This might suggest that the agent tends to complete tasks in a general way without catering to the specific action preferences of users, thus failing to complete tasks in a personalized manner.

5.3 Effect of task difficulty



(a) Proactive task suggestion

(b) Personalized task execution

Figure 6: Performance under different task difficulties.

We experiment with the models' performance under different task difficulty levels. For proactive task suggestion (see Figure 6.a), we varied the number of O (the first few screenshots of the user's current actions). The SR_1 of all models increased as the number of screenshots increased. This was expected. Clearly, if the agent knew the first screenshot of the user's current action, it could basically infer which app the user was using, thereby significantly narrowing the range of the user's intent. With the second and third screenshots, the agent could further narrow the user's intent range by analyzing the actions therein (e.g. clicking the search box).

For personalized task execution (see Figure 6.b), we calculated the SR_2 of GUI-control models on different subsets of action lengths in the test set (we did not calculate the SR_2 of generalist models because their SR_2 were all too low). It can be seen that as the action length increases, the SR_2 decreases. This is in line with expectations, as the greater the action length required to complete a certain instruction, the more complex the instruction is, and the more difficult it is to complete.

5.4 Effect of fine-tuning

Table 6: Performance of fine-tuned model.

Model	Proactive task suggestion		Personalized task execution		
	$SR_1(\%)$	Sim_1	$SR_2(\%)$	Sim_2	Step Ratio
Qwen-2.5-vl-7B	3.1	0.25	1.5	0.95	2.16
Qwen-2.5-vl-7B-FT	9.7 [+6.6]	0.49 [+0.24]	12.5 [+11.0]	1.21 [+0.26]	1.17 [-0.99]
GPT-4.1	7.2	0.35	5.5	0.98	1.98
UI-TARS-1.5-7B	-	-	38.5	1.06	1.22

To save resources, we only fine-tuned Qwen-2.5-VL-7B and adopted the parameter-efficient fine-tuning method of LoRA, with the LoRA rank set to 4. Following the method of sampling the test set, we randomly sampled 1,000 data episodes from the training set for fine-tuning. These data covered all users, and the proportion of data for each user was the same as their proportion in the training set. These 1,000 data episodes were reorganized according to the input and output formats of the two tracks, respectively. The prompts used in fine-tuning are the same as those we designed for generalist models. Finally, we trained separately on two tracks and obtained two fine-tuned models, each suitable for one of the two tracks.

The performance of the fine-tuned model on the two tracks is shown in Table 6. Despite using a smaller model and less training data, the fine-tuned model achieved significant performance improvements in all main metrics. In proactive task suggestion, compared with the best-performing generalist model GPT-4.1, our fine-tuned model achieved better performance in both SR_1 and Sim_1. In proactive task suggestion, compared with the best-performing UI-TARS-1.5-7B, our fine-tuned model had a lower success rate SR_2. We consider this acceptable because UI-TARS is a model specifically designed and extensively trained for GUI grounding and GUI control, and thus has a more general instruction execution capability. However, our fine-tuned model had a significantly higher Sim_2, indicating that the action paths it selects may not be optimal but are closer to the user's action preferences. Overall, the model fine-tuned on our collected data demonstrated stronger proactivity and personalization capabilities, being able to utilize user-related contextual information to extract potential intent patterns and action preferences from the user's past intents and actions, which existing models have not or find it difficult to consider.

6 Conclusions

We present FingerTip 20K, a benchmark advancing mobile GUI agents toward proactive task suggestion and personalized task execution. Our data captures longitudinal user interactions, enriched with contextual information to model user-specific patterns. Experiments reveal significant gaps in existing models' ability to leverage such contextual information. Fine-tuning Qwen-2.5-VL-7B on our data improved suggestion success rate while better aligning actions with user preferences, demonstrating the value of user-oriented training. This work establishes critical infrastructure for developing mobile agents that anticipate user needs and adapt to user action preferences.

References

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

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

[3] An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu Zhong, Julian McAuley, Jianfeng Gao, et al. Gpt-4v in wonderland: Large multimodal models for zero-shot smartphone gui navigation. *arXiv preprint arXiv:2311.07562*, 2023.

[4] Geunwoo Kim, Pierre Baldi, and Stephen McAleer. Language models can solve computer tasks. *Advances in Neural Information Processing Systems*, 36:39648–39677, 2023.

[5] 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.

[6] 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.

[7] Li Zhang, Shihe Wang, Xianqing Jia, Zhihan Zheng, Yunhe Yan, Longxi Gao, Yuanchun Li, and Mengwei Xu. Llamatouch: A faithful and scalable testbed for mobile ui automation task evaluation. *arXiv e-prints*, pages arXiv–2404, 2024.

[8] Juyong Lee, Taywon Min, Minyong An, Dongyoon Hahm, Haeone Lee, Changyeon Kim, and Kimin Lee. Benchmarking mobile device control agents across diverse configurations. *arXiv preprint arXiv:2404.16660*, 2024.

[9] Mingzhe Xing, Rongkai Zhang, Hui Xue, Qi Chen, Fan Yang, and Zhen Xiao. Understanding the weakness of large language model agents within a complex android environment. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6061–6072, 2024.

[10] 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.

[11] Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models. *arXiv preprint arXiv:2401.13919*, 2024.

[12] Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. *arXiv preprint arXiv:2306.07863*, 2023.

[13] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback, 2022. *URL https://arxiv. org/abs/2112.09332*, 2022.

[14] 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.

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

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

[17] 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.

[18] Zhuohao Wu, Danwen Ji, Kaiwen Yu, Xianxu Zeng, Dingming Wu, and Mohammad Shidujaman. Ai creativity and the human-ai co-creation model. In *Human-computer interaction. theory, methods and tools: thematic area, HCI 2021, held as part of the 23rd HCI international conference, hCII 2021, virtual event, July 24–29, 2021, proceedings, part i 23*, pages 171–190. Springer, 2021.

[19] Weiwen Chen, Mohammad Shidujaman, Jiangbo Jin, and Salah Uddin Ahmed. A methodological approach to create interactive art in artificial intelligence. In *HCI International 2020–Late Breaking Papers: Cognition, Learning and Games: 22nd HCI International Conference, HCII 2020, Copenhagen, Denmark, July 19–24, 2020, Proceedings 22*, pages 13–31. Springer, 2020.

[20] Cheng Qian, Bingxiang He, Zhong Zhuang, Jia Deng, Yujia Qin, Xin Cong, Zhong Zhang, Jie Zhou, Yankai Lin, Zhiyuan Liu, et al. Tell me more! towards implicit user intention understanding of language model driven agents. *arXiv preprint arXiv:2402.09205*, 2024.

[21] Yaxi Lu, Shenzhi Yang, Cheng Qian, Guirong Chen, Qinyu Luo, Yesai Wu, Huadong Wang, Xin Cong, Zhong Zhang, Yankai Lin, et al. Proactive agent: Shifting llm agents from reactive responses to active assistance. *arXiv preprint arXiv:2410.12361*, 2024.

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

[23] Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, et al. Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding. *arXiv preprint arXiv:2412.10302*, 2024.

[24] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[25] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[26] 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.

[27] Jingxuan Chen, Derek Yuen, Bin Xie, Yuhao Yang, Gongwei Chen, Zhihao Wu, Li Yixing, Xurui Zhou, Weiwen Liu, Shuai Wang, et al. Spa-bench: A comprehensive benchmark for smartphone agent evaluation. In *NeurIPS 2024 Workshop on Open-World Agents*, 2024.

[28] Jiaqi Zhang, Chen Gao, Liyuan Zhang, Yong Li, and Hongzhi Yin. Smartagent: Chain-of-user-thought for embodied personalized agent in cyber world. *arXiv preprint arXiv:2412.07472*, 2024.

[29] 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. *arXiv preprint arXiv:2410.24024*, 2024.

[30] Yuxiang Chai, Hanhao Li, Jiayu Zhang, Liang Liu, Guangyi Liu, Guozhi Wang, Shuai Ren, Siyuan Huang, and Hongsheng Li. A3: Android agent arena for mobile gui agents. *arXiv preprint arXiv:2501.01149*, 2025.

[31] Dezhi Ran, Mengzhou Wu, Hao Yu, Yuetong Li, Jun Ren, Yuan Cao, Xia Zeng, Haochuan Lu, Zexin Xu, Mengqian Xu, et al. Beyond pass or fail: A multi-dimensional benchmark for mobile ui navigation. *arXiv preprint arXiv:2501.02863*, 2025.

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

[33] Yu Shang, Peijie Liu, Yuwei Yan, Zijing Wu, Leheng Sheng, Yuanqing Yu, Chumeng Jiang, An Zhang, Fengli Xu, Yu Wang, et al. Agentrecbench: Benchmarking llm agent-based personalized recommender systems. *arXiv preprint arXiv:2505.19623*, 2025.

[34] Yuwei Yan, Yu Shang, Qingbin Zeng, Yu Li, Keyu Zhao, Zhiheng Zheng, Xuefei Ning, Tianji Wu, Shengen Yan, Yu Wang, et al. Agentsociety challenge: Designing llm agents for user modeling and recommendation on web platforms. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 2963–2967, 2025.

[35] Chenyang Shao, Xinyuan Hu, Yutang Lin, and Fengli Xu. Division-of-thoughts: Harnessing hybrid language model synergy for efficient on-device agents. In *Proceedings of the ACM on Web Conference 2025*, pages 1822–1833, 2025.

[36] Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, et al. Towards large reasoning models: A survey of reinforced reasoning with large language models. *arXiv preprint arXiv:2501.09686*, 2025.

[37] Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. Agentsquare: Automatic llm agent search in modular design space. *arXiv preprint arXiv:2410.06153*, 2024.

[38] Qingbin Zeng, Qinglong Yang, Shunan Dong, Heming Du, Liang Zheng, Fengli Xu, and Yong Li. Perceive, reflect, and plan: Designing llm agent for goal-directed city navigation without instructions. *arXiv preprint arXiv:2408.04168*, 2024.

[39] Lin Chen, Fengli Xu, Nian Li, Zhenyu Han, Meng Wang, Yong Li, and Pan Hui. Large language model-driven meta-structure discovery in heterogeneous information network. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 307–318, 2024.

A Appendix

A.1 Limitations

Our study has several limitations. Firstly, all 91 contributors live in mainland China, and mainly interact with Chinese third-party apps. The recorded linguistic habits, UI layouts and action patterns may differ markedly from other regions. Secondly, our LoRA fine-tuning uses only 1000 episodes and a single 7B model. Due to cost constraints, we did not conduct larger-scale fine-tuning experiments nor verify its generalization. Finally, we assume that screenshots can be stored and shared after anonymization. In practice, fine-grained UI traces can still contain unique visual features that allow re-identification. Techniques such as selective redaction or synthetic replay should be explored before large-scale deployment.

A.2 Broader impacts

FingerTip 20K aims to advance mobile agents that anticipate user needs and adapt to individual preferences. If developed responsibly, such agents could reduce the interaction barrier for elderly or motor-impaired users, reduce screen time by automating repetitive tasks, and serve as a test bed for privacy-preserving personalization research. At the same time, the technology entails risks. Continuous screen capture combined with explicit user profiles gives models an intimate view of personal life. An attacker compromising the agent, or a service provider lacking strong governance, could reconstruct sensitive behaviors, contacts or locations. We encourage future work on on-device processing, differential privacy and audit mechanisms.

A.3 Data collection

The data collection was carried out through crowdsourcing, and participants were paid in accordance with the living wage laws of their country. Participants were informed of the expected use of the collected data and signed a data usage agreement. They were asked not to upload any data related to private information. We provided participants with detailed guidance documents and video tutorials on how to operate the FingerTip APP for data collection. All participants went through a training phase during which they became familiar with the FingerTip APP and received targeted feedback from our manually checked data. They were encouraged to avoid using overly simplified or ambiguous language to collect clear and useful intent descriptions. It should be noted that the FingerTip APP only collects data when participants actively use it. It does not automatically collect data at other times.

A.4 Data format

Our data is publicly released at `https://www.kaggle.com/datasets/qinglongyang/fingertip-20k`. The data contains several folders named with numbers (i.e. user IDs), and each of these folders contains multiple folders named with timestamps (e.g., 20250309_133115), representing all the data episodes submitted by that user. For each data episode, the following information is included:

- *screenshots*: a list of screenshots for each observation encoded as JPGs.
- *accessibility trees*: a list of Android accessibility tree XML files for each observation.
- *actions*: a list of actions represented in the form of JSON dictionaries. Each screenshot corresponds to an action.
- *intent_description*: the user's true intent in this episode.
- *user_id*: the unique integer identifier of the user to whom this episode belongs. This information can be used to retrieve the corresponding user's user profile.
- *time*: the timestamp when this episode was collected.
- *scenario*: the category of location where the user was when this episode was collected.
- *app*: the name of the activity running when the episode was collected. This information is only used to launch the corresponding app in personalized task execution and is not provided to the LLM agent.

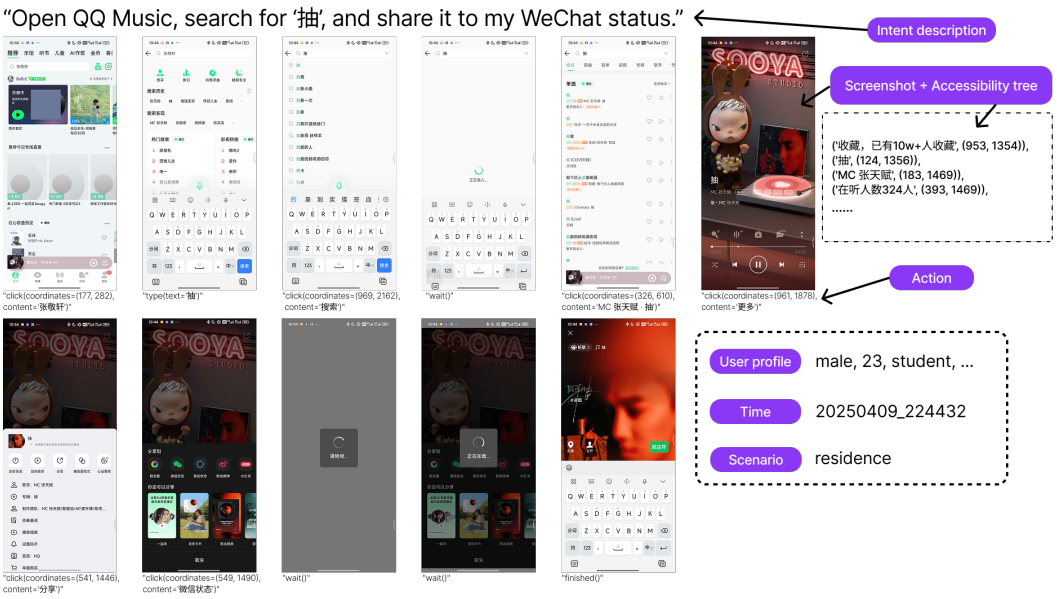


Figure 7: An example episode from FingerTip.

The example of an episode from FingerTip is shown in Figure 7.

Accessibility trees Note that when using accessibility trees, the LLM agent utilizes a list of all accessible UI elements and their coordinates corresponding to a certain screenshot, which is extracted from the metadata XML file through a Python function.

Table 7: User profile example.

Field	user_id	sex	age	occupation	address	marital_status	phone_brand
Example	55	male	20	student	Beijing	single	Huawei

User profile The types of information included in user profiles and an example can be seen in Table 7.

Scenario When users record their intents with the FingerTip APP, they need to select the category of the location they are in. Specifically, they can choose from the following 12 common categories: residence, office, school, dining place, shopping mall, medical institution, entertainment and leisure venue, sports venue, cultural venue, transportation, urban street, and natural outdoor spaces. If users think that none of these categories can describe the location they are in, they can fill in a new category on their own.

A.5 Prompts for the LLM agents

A.5.1 Prompt for proactive task suggestion

```
You are an Android GUI agent. You are given the first few screenshots of the user's
action (arranged in chronological order) and some supplementary information. You
need to infer the user's intent.

## Input
User_profile: {profile}
Time: {time}
Scenario: {scenario}
Previous_intents: {previous_intents}

## Note
- Express the user's intent unambiguously in one Chinese sentence, including all
necessary information.
- Clearly state the name of the app which the user is using, and the final effect
the user wants to achieve.
- Previous_intents contains the user's intents at certain times and in certain
scenarios in the past.
- Do not output anything other than the user's intent.

The user's intent:
```

A.5.2 Prompt for personalized task execution

```
You are an Android GUI agent. You are given an instruction and current screenshot
and some supplementary information. You need to perform the next action to complete
the instruction.

## Input
User_instruction: {instruction}
User_profile: {profile}
Screen_width_height: {size}
Screen_description: {screen_description}
Actions_reference: {actions_reference}
Previous_actions: {previous_actions}

## Action Space
click(coordinates=(x,y), content='')
long_click(coordinates=(x,y), content='')
type(text='')
scroll(coordinates=(x,y), direction='down or up or right or left')
press_back()
press_home()
press_recent()
wait()
finished()

## Note
- 'coordinates' should represent the coordinates of the click point. The origin is
the upper left corner of the screenshot, with x increasing to the right and y
increasing downward.
- 'content' should represent the original text at the click point or the description
 of the icon, usually in Chinese.
- 'text' should represent all the original text that the user intends to input. (
usually in Chinese, and usually included in User_instruction)
- 'press_back()', 'press_home()', 'press_recent()' means that go to previous screen,
 home screen, recent apps screen, respectively.
- 'wait()' means that wait until the next observation is received. This usually
occurs during loading or switching windows.
- 'finished()' means that the instruction is completed.
```

```
- Screen_description contains some correct 'content' and 'coordinates' of the UI,
which can be directly referenced.
- Actions_reference represents the complete sequence of actions that the user
performed when executing a similar instruction in the past, which can be used for
reference.
- Previous_actions contains the sequence of actions you have already performed under
 the current instruction.
- Only one action in Action Space can be taken. Do not output anything other than
the action to take.

The action to take:
```