GUICourse: From General Vision Language Model to Versatile GUI Agent

Anonymous ACL submission

Abstract

Utilizing Graphic User Interfaces (GUIs) for human-computer interaction is essential for accessing various digital tools. Recent advancements in Vision Language Models (VLMs) reveal significant potential for developing versatile agents that assist humans in navigating GUIs. However, current VLMs face challenges related to fundamental abilities, such as OCR and grounding, as well as a lack of knowledge about GUI elements functionalities and control methods. These limitations hinder their effectiveness as practical GUI agents. To address these challenges, we introduce **GUICourse**, a series of datasets for training visual-based GUI agents using general VLMs. First, we enhance the OCR and grounding capabilities of VLMs using the **GUIEnv** dataset. Next, we enrich the GUI knowledge of VLMs using the **GUIAct** and **GUIChat** datasets. Our experiments demonstrate that even a small-sized GUI agent (with 3.1 billion parameters) performs effectively on both single-step and multi-step GUI tasks. We further finetune our GUI agents on other GUI tasks with different action spaces (AITW and Mind2Web), and the results show that our agents are better than their baseline VLMs. Additionally, we analyze the impact of OCR and grounding capabilities through an ablation study, revealing a positive correlation with GUI navigation ability.

1 Introduction

Graphical User Interfaces (GUIs) are a pivotal medium for facilitating human-computer interactions, playing a crucial role across diverse applications. GUI agents are designed to complete complex tasks in these GUI systems, which can liberate humans from tedious and repetitive operations. Different GUI systems (e.g. websites and smartphones) usually use icons and images to convey specific information, making them suitable to be processed by visual-based agents. On the other hand, obtaining screenshots of GUI systems is straightforward, whereas accessing structured text is challenging (such as the code behind the iOS system). In this work, we focus on training visual-based GUI agents from general VLMs.

The performance of vision-based agents heavily relies on the fundamental capabilities and internal knowledge of their baseline VLMs. Although VLMs (Liu et al., 2023a; Peng et al., 2023; Zhu et al., 2023; Li et al., 2023a; Hu et al., 2023) demonstrate impressive capabilities in various tasks such as image captioning (Li et al., 2023b) and visual question answering (Liu et al., 2023b), these capabilities may not suffice for developing efficient visual-based GUI agents. For instance, we investigate the Qwen-VL-Chat, a powerful open-sourced VLM, on its ability to finalize GUI navigation instructions. As illustrated in Figure 1, it shows the two challenges of current VLMs: (1) Their OCR and grounding abilities are not supported to locate the website elements or designated text accurately; (2) They lack a comprehensive understanding of the functions and control mechanisms of website widgets. To solve these problems, we provide a pipeline for training GUI agents based on general VLMs. This involves enhancing their OCR and grounding capabilities, followed by training them to perform GUI navigation tasks and engage in related conversations.

Our approach mainly employs data-driven methods for training diverse agents. To cultivate versatile GUI agents, we require a substantial amount of high-quality GUI navigation data enriched with visual information (e.g., screenshots). While previous works have contributed various datasets, they often suffer from one or more shortcomings: (1) the GUI environments may be overly simplistic and disconnected from real-world scenarios (Liu et al., 2018; Humphreys et al., 2022); (2) the datasets tend to focus on narrow domains or scenarios (Yao et al., 2022; Sun et al., 2022a; Rawles et al., 2023);

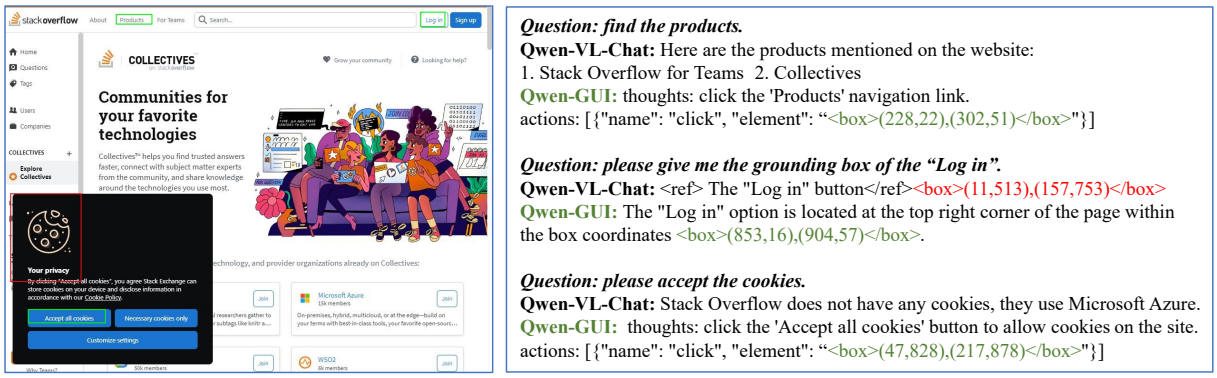


Figure 1: An example of several GUI tasks. Qwen-VL-Chat gives an incorrect bounding box for "Log in", which means its OCR and grounding capabilities are insufficient. Also, it can't finish GUI navigation tasks such as "find the products" and "accept the cookies" because it lacks the knowledge of GUI control actions. Trained on our datasets, our GUI agent Qwen-GUI performs these tasks well.

and (3) the overall size of the datasets may be insufficient for effective training of GUI agents (Deng et al., 2023). In this work, we primarily introduce the **GUIAct** dataset, which addresses these challenges by providing 67k human-verified single-step and 5,696 human-annotated multi-step GUI navigation instructions across various websites. Additionally, we present the **GUIEnv** dataset, comprising 10 million website page-annotation pairs, and the **GUIChat** dataset, featuring 44k single-turn QA pairs and 6k multi-turn dialogues alongside website screenshots, to support our training pipeline. In summary, our contributions are as follows:

(1) We establish a comprehensive pipeline and a series of datasets for building GUI agents based on VLMs. This pipeline consists of two learning stages aimed at enhancing fundamental abilities and acquiring knowledge specific to GUI systems.

(2) We train several visual-based GUI agents according to the pipeline and datasets. We find that even small-sized agentsperform effectively in GUI navigation tasks. Additionally, we demonstrate that integrating OCR and grounding data during the pre-training phase is helpful for these tasks.

2 Related Works

2.1 Vision Language Models

Currently, most Vision-Language Models (VLMs) use a bridge module to connect a vision encoder, such as CLIP-ViT (Dosovitskiy et al., 2020; Radford et al., 2021), and a language model, such as Vicuna (Touvron et al., 2023) and Qwen (Bai et al., 2023a). The bridge module maps the visual features from the vision encoder into the embedding space of the large language model with an MLP layer, exemplified by LLaVA (Liu et al., 2023b), or an attention-based resampler, such as BLIP-2 (Li et al., 2023b) and Qwen-VL (Bai et al., 2023b). Uniquely, Fuyu-8B (Bavishi et al., 2023) removes the external vision encoder and uses a decoder-only transformer architecture with a unified space for textual tokens and pixel information, enabling Fuyu-8B to process images of any resolution. Based on this, subsequent research efforts try to enhance VLMs in various domains, including visual detail recognition (Wu and Xie, 2023; Zhang et al., 2024), reliable responses (Sun et al., 2023b; Yu et al., 2023), object grounding (Peng et al., 2023), multilingual capabilities (Hu et al., 2023) and model efficiency (Zhu et al., 2024). However, many VLMs still have shortcomings when recognizing small-size text in high-resolution images, and providing the accurate location of the designated area at the pixel level. In this work, we provide the GUIEnv dataset to improve the VLMs' OCR and grounding abilities with high-resolution website screenshots.

2.2 GUI Agents

GUI agents are expected to help humans finish tasks on different GUI systems, such as simplified websites (Shi et al., 2017; Liu et al., 2018), simulated environments (Yao et al., 2022; Zhou et al., 2023), or real-world websites and smartphones (Deng et al., 2023; Rawles et al., 2023). Nowadays, the GUI agents are developing quickly due to the Large Models (LMs), Big Data, and data-driven methods. Some agents (Gao et al., 2023; Sun et al., 2023a; Ma et al., 2023; Kim et al., 2023; Xu et al., 2021; Zheng et al., 2023; Yang et al., 2023) using the train-free method and de-

Table 1: The overview of datasets we contributed for multi-stage GUI Agent training.

Name	Subset	Source	Num of Ins.
GUIEnv	global	C4	10M
	local	C4	0.7M
GUIAct	web-single	GPT-4V/Human	67k
	web-multi	Human	5,696
	smartphone	AITW-general	9,157
GUIChat	single-turn	GPT-4	44k
	multi-turn	GPT-4	6k

pending on closed-sourced LMs (e.g., GPT4 and GPT-4V (OpenAI, 2023)) to achieve GUI navigation with strategy design and prompt engineering. Also, some agents (Humphreys et al., 2022; Nakano et al., 2021; Qin et al., 2023) finish GUI tasks based on small-size open-sourced models, using data-driven methods (e.g. SFT).

Perception and control are the two important points for GUI agents. According to the perception modalities of environments, the GUI agents can split into text-only (Gur et al., 2023; Deng et al., 2023), image-text combined (Humphreys et al., 2022; Furuta et al., 2023), and vision-only (Shaw et al., 2023; Zhan and Zhang, 2023; Hong et al., 2023) methods. As for the control, GUI agents can split into position-free and position-needed methods. The position-free methods (Deng et al., 2023; Yao et al., 2022) generally provide indexes for every element in the GUI systems, and agents can finish actions by multiple choices. While the position-needed methods (Shaw et al., 2023; Zhan and Zhang, 2023; Hong et al., 2023) mean agents need to give the position information to finish some actions, such as "click" and "tap". In this work, we aim to train GUI agents with vision-only inputs and position-needed actions.

3 GUICourse

In this section, we introduce GUICourse, a series of datasets for improving VLMs' OCR and grounding abilities, enhancing VLMs' GUI knowledge, and helping GUI agents' interaction. As shown in Table 1, our GUICourse includes GUIEnv, GUIAct, and GUIChat three partitions. Examples from the **GUICourse** datasets are provided in Appendix C. We employ a combination of LLM-based auto-annotation and human annotation to construct our datasets. More details of the annotation process (e.g. the prompts and costs) are shown in Appendix A.2 and Appendix A.3.

3.1 GUIEnv: A Large Scale Dataset for OCR and Grounding Abilities

OCR and grounding are two fundamental abilities for visual-based GUI agents' perception and control. To augment these abilities, we create a large-scale dataset called GUIEnv with about 10M pre-training data GUIEnv-global and 0.7M SFT data GUIEnv-local. Each sample in GUIEnv-global is a long text with all the describable content on the full page, including text, grounding information, and layout sequence. Each sample in GUIEnv-local is a QA pair on a designed region as the "text2bbox" or "bbox2text" task.

Data Collection. We collected 4M URLs from the C4 (Cleaned Common Crawl Corpus) (Raffel et al., 2019) and employed Playwright[1] for rendering, ultimately producing 10M annotated screenshots(Lee et al., 2023). The 10M annotated screenshots correspond to the 10M samples of the ***GUIEnv-global*** data. As for the ***GUIEnv-local*** data, we select about 50k website screenshots with their elements from C4. These screenshots are usually huge because they include the websites' complete content. We process the data in three steps: (1) crop them to smaller partitions less than 1920×1080 resolution. (2) remove images with elements less than 10. (3) randomly simply 10 elements with texts and positions in each screenshot as the data of "text2bbox" and "bbox2text" tasks. Finally, we acquired 0.7M instructions for the GUIEnv-local data.

3.2 GUIAct: A GUI Navigation Dataset with Single-step and Multi-step Instructions

We argue that GUI agents can learn GUI knowledge such as the functions and the control methods of GUI elements from GUI navigation tasks. So, we provide the GUIAct dataset, which can be split into three partitions: "web-single", "web-multi", and "smartphone". Our main contributions are two subsets in website scenarios: a single-step website navigation subset with 67k instructions and a multi-step website navigation subset with 5,696 instructions. As shown in Table 2, although there are many similar navigation tasks in mobile application scenarios, the two subsets are the largest navigation datasets in real-world website scenarios with both single-step and multi-step instructions. An example of our multi-step instruction is shown

[1]https://github.com/microsoft/playwright

Table 2: Statistics of the GUIAct dataset compared with existing datasets. The columns indicate: the number of app/website domains, environments, and instructions (Dom., Env., and Inst.), the average number of turns per instance (Avg. Turns), instruction level, and scenarios. Our main contributions are the two largest subsets in website scenarios with both single-step and multi-step instructions.

Dataset	Dom.	Env.	Inst.	Avg. Turns	Inst. Level	Scenarios
MiniWoB++ (Liu et al., 2018)	-	100	100	3.6	Low-level	Simplified Mob./Webs
Rico (Deka et al., 2017)	27	9,772	10,811	-	-	(Traces) Mob. Apps
UIBert (Bai et al., 2021)	-	-	16,660	1.0	Low-level	Mob. Apps
UGIF (Venkatesh et al., 2022)	-	12	523	5.3	High & low	Mob. Apps
WebShop (Yao et al., 2022)	1	1	12k	11.3	High-level	Shopping Webs
RUSS (Xu et al., 2021)	-	22	80	-	High & low	Real-world Webs
PixelHelp (Li et al., 2020)	4	4	187	-	High & low	Mobile Apps
META-GUI (Sun et al., 2022b)	6	11	1,125	4.3	High-level	(Dialogues) Mob. Apps
MoTIF (Burns et al., 2022)	15	125	756	-	High & low	Mobile Apps
MIND2WEB (Deng et al., 2023)	5/31	137	2,350	7.3	High-level	Real-world Webs
AITW (Rawles et al., 2023)	-	357	715k	6.5	High & low	Android/Apps
WEBLINX (Lù et al., 2024)	-	155	2,337	43.0	High-level	(Dialogues) Webs
AndroidControl (Li et al., 2024)	-	833	15,283	4.8	High & low	Mobile Apps
GUI-World (Chen et al., 2024)	-	-	12,379	-	High-level	Web./Mob./Desk./XR
GUIAct (web-single)	50	13k	67k	1.0	Low-level	Real-world Webs
GUIAct (web-multi)	8/32	121	5,696	7.9	High-level	Real-world Webs

in Figure 2. The GUI agents only receive screenshots of GUI systems in the viewport, without any unseen elements, and output the action with position. We define a unified **action space** for different GUI systems, including 11 types of actions, and the details are shown in Appendix A.1. To enhance the scenario diversity of our dataset, we also convert a part of the samples in AITW depending on our action space and action format.

Data Collection. We contribute the ***GUIAct (web-single)*** dataset in four steps: *(1) Websites selection.* We use GPT-4 to gather 50 different scenarios (e.g., online shopping and E-learning) and hundreds of URLs. Then, we use these URLs as seeds to expand new websites by their hyperlinks, acquiring about 13k websites. *(2) Captures acquisition.* We use web snapshot tools to respond to website HTML, interactive elements, and screenshots based on their URLs. *(3) LLM-Auto Annotation.* We use GPT-4V to obtain single-step instruction-action pairs for each website. We give two images to GPT-4V in each request: an origin screenshot and a revised screenshot with interactive element identification. We get about 70k instruction-action pairs based on the 13k website screenshots. *(4) Data checking by human.* We hire annotators to check the automatic instruction-action pairs' quality. The annotators are requested to revise the inaccurate items or abandon them if they are hard to revise. After that, the accuracy of GUIAct (web-single) data improves from 55% to 92%, based on our sampled results. Finally, we get around 67k single-step action instructions in website scenarios, corresponding to 67k training samples.

We contribute the ***GUIAct (web-multi)*** dataset in three steps: *(1) Websites selection.* We concept 8 top-level common web scenarios: Shopping, Diet, Stay, Travel, Education, Health, Job, and Entertainment. Based on these scenarios, we collect 32 sub-scenarios and 121 well-known websites. *(2) Questions acquisition.* We use LLMs to generate instructions for each website. We collect 8k high-level instructions with the topic "searching special information" on the corresponding websites. *(3) Crowd-sourced Annotation.* We develop an annotation tool as a web browser plugin. Then, we hire annotators to execute operators to finish the instructions on the corresponding websites. If the instructions are inaccurate or have no answer, the annotators will revise or abandon them. Finally, We get 5,696 multi-step action instructions in website scenarios, corresponding to 44k training samples.

We contribute the ***GUIAct (smartphone)*** dataset using a subset of the AITW (Rawles et al., 2023) dataset. We select the data with the "General" tag and filter out the screenshots without the bottom navigation bar (because we will convert the "go back" and "go home" actions to "tap"). Then, we convert the origin actions to our action space. Finally, we get 9,157 multi-step action instructions in smartphone scenarios, corresponding to 67k training samples.

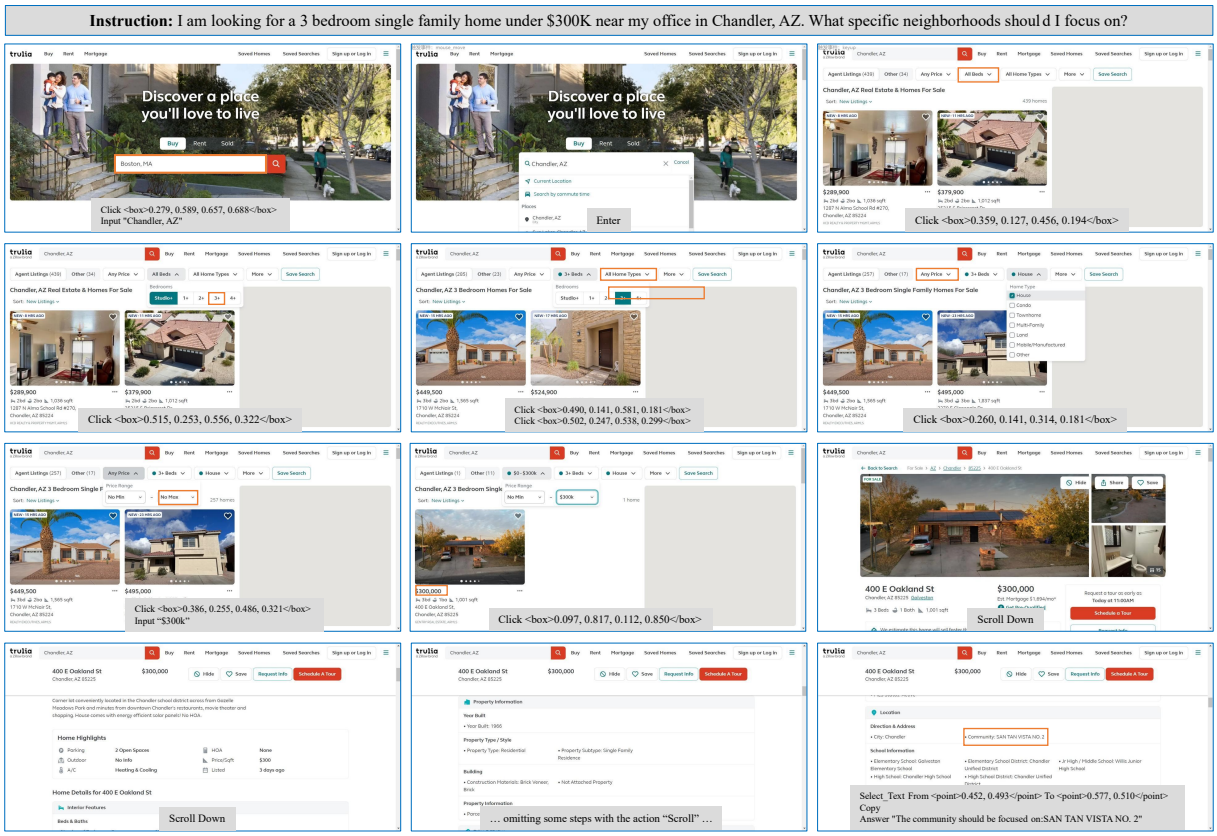


Figure 2: An example of our GUIAct dataset with 17 steps of action-screenshot pairs. Each step corresponds with one website screenshot and one or more actions. Some actions (e.g., "click") have their location information, represented by a bounding box.

3.3 GUIChat: A Text-Rich Multi-Modal QA Dataset for Interaction

Natural language interaction is important for better using GUI agents. It's necessary to mix conversation data during the training stage of GUI agents for better interaction. We introduce the GUIChat dataset, which includes lots of text-rich website screenshots and grounding boxes in dialogues. This dataset has about 44k single-turn QA pairs and 6k multi-turn dialogues in four aspects: visual information queries, human-centric issues, world knowledge inquiries, and complex reasoning tasks.

Data Collection. We contribute this dataset in three steps: *(1) Web Page Screenshot Acquisition.* Utilizing Playwright to render web pages and obtain screenshots, same as the collection process of GUIEnv. *(2) Textual Representation Extraction.* Extracting essential structured information and text with coordinates from a redundant DOM tree. *(3) Question-Answer Pair Generation.* Leveraging GPT-4 to construct question-answer pairs from the textual representations of the current website. The textual representations include the elements in the website with the position information. We use GPT-4 rather than GPT-4V to achieve this step because GPT-4 can get higher accuracy than GPT-4V when given accurate textual representations.

4 Experiments

We train several GUI agents depending on our training pipeline with our GUICourse (GUIEnv, GUIAct, and GUIChat datasets). First, we explore the performance of our GUI agents on our test datasets and common GUI tasks. Then, we do ablation studies of the whole training process based on MiniCPM-V, analyzing the influence of various factors. Finally, we use two examples to show the generalization of our GUI agents.

4.1 Training Settings

Depending on our training pipeline, we first improve the VLMs' fundamental abilities (OCR and grounding ability) with the GUIEnv dataset. Then, we improve the VLMs' GUI knowledge and conversation ability using the GUIAct and GUIChat datasets. Considering the data quantity and format

Table 3: The performance of our GUI agents on test datasets.

Agents	Web-Single			Web-Multi			Smartphone			Mean
	Type EM	Cli.Acc	StepSR	Type EM	Cli.Acc	StepSR	Type EM	Cli.Acc	StepSR	StepSR
MiniCPM-GUI										
+ GUIAct	89.1	52.4	55.2	62.3	42.0	48.6	64.5	29.6	43.3	49.0
+ High Resolution	91.8	72.9	69.2	68.2	42.4	47.1	72.6	47.5	52.0	56.1
+ GUIChat	**91.8**	**74.9**	**70.6**	67.0	45.4	**47.5**	71.7	44.7	53.3	57.1
Fuyu-GUI	90.1	67.1	63.5	68.8	51.2	47.1	72.1	29.1	40.4	50.4
Qwen-GUI	90.9	69.4	66.7	**68.9**	**52.5**	46.8	**73.0**	**55.7**	**58.1**	**57.2**

of GUICourse, we designate the GUIEnv-global data for pre-training, while the remaining datasets serve as supervised fine-tuning (SFT) data.

We train the MiniCPM-GUI agent based on MiniCPM-V (min, 2024) through the following steps: (1) We pre-train MiniCPM-V by integrating 10M samples from the GUIEnv-global dataset into the original pre-training data. (2) Subsequently, we continue the training in the SFT stage by merging approximately 70k samples from the GUIEnv-local dataset, along with all samples from the GUIAct and GUIChat datasets.

Additionally, we train other agents, Qwen-GUI and Fuyu-GUI, based on Qwen-VL (Bai et al., 2023b) and Fuyu-8B (Bavishi et al., 2023), respectively. However, unlike the training of MiniCPM-GUI, we only train them with the SFT data.

4.2 Performance on GUI Navigation Tasks

We evaluate our GUI agents through two methods: First, we test the performance of our agents on the test datasets split by ourselves. We split about 1.4k, 1k, and 2k samples from the "web-single", "web-multi", and "smartphone" partitions in GUIAct as test data. Then, we test the performance of our agents on other GUI navigation tasks with different action spaces, such as Mind2Web (Deng et al., 2023) and AITW (Rawles et al., 2023). We fine-tune our GUI agents and their baseline VLMs on the new GUI tasks. If our agents perform better, it demonstrates that they have learned more common GUI knowledge than baseline VLMs.

Results on our test datasets. We use the type exact match score (Type EM), click accuracy (Cli. Acc), and step success rate (StepSR) as metrics. Type EM is the accuracy of the actions' names. Cli.Acc is the average score of the success rate of the "click" and "tap" actions, which reflects the grounding ability. StepSR is the average score of all the actions' success rates. More details are shown in Appendix B.2.

Table 4: Results of our GUI agents on Mind2Web.

Agents	Cross-Task		Cross-Website		Cross-Domain	
	Ele.Acc	StepSR	Ele.Acc	StepSR	Ele.Acc	StepSR
SeeClick	28.3	25.5	21.4	16.4	23.2	20.8
Qwen-VL	23.2	20.3	16.8	14.0	14.1	12.3
Qwen-GUI	**27.9**	**24.4**	**19.3**	**15.6**	**20.5**	**17.5**
Fuyu-8B	8.3	6.6	4.8	4.0	3.6	3.0
Fuyu-GUI	**19.1**	**15.6**	**13.9**	**12.2**	**14.2**	**11.7**
MiniCPM-V	11.0	8.5	8.2	6.0	6.5	5.2
MiniCPM-GUI	**23.8**	**20.8**	**20.3**	**17.3**	**17.9**	**14.6**

Table 5: Results of our GUI agents on the AITW.

Models	Install	Goo.Apps	Single	WebShop.	Overall
SeeClick	66.4	54.9	63.5	57.6	60.6
Qwen-VL	**70.4**	57.8	70.1	64.7	65.8
Qwen-GUI	70.3	**61.2**	**71.6**	**66.1**	**67.3**
Fuyu-8B	45.9	40.0	**47.2**	40.8	43.5
Fuyu-GUI	**50.9**	**41.6**	45.7	**43.8**	**45.5**
MiniCPM-V	50.2	45.1	56.2	44.0	48.9
MiniCPM-GUI	**62.3**	**46.5**	**67.3**	**57.5**	**58.4**

As shown in Table 3, we can find that: (1) Small-size models can perform effectively on visual-based GUI navigation tasks. Qwen-GUI (9.6B) achieves the highest mean step success rate of 57.2 across the three datasets, while MiniCPM-GUI (3.1B) delivers comparable results with a mean step success rate of 57.1. (2) High resolution is important for GUI navigation tasks. The default resolution of MiniCPM-V is 448×448, and it can be improved to 1344×1344 by the flexible patching and slicing strategy (Ye et al., 2023). The mean step success rate improves to 56.1 from 49.0. (3) Our GUIChat dataset is helpful for tasks in website scenarios, although its main goal is to improve the interaction ability of GUI agents. The mean step success rate improved by 1 points.

Results on new GUI navigation datasets. Mind2Web and AITW are two commonly used

Table 6: The performance of MiniCPM-GUI with different amounts of the GUIEnv data.

Amount	Bbox2Text		Text2Bbox			Web-Single		
	EM Score	F1 Score	IoU@0.2	IoU@0.5	IoU@0.7	Type EM	Cli.Acc	StepSR
w/o GUIEnv	14.92	30.49	9.87	2.15	0.32	92.70	49.80	52.84
w/ 2.5M GUIEnv	35.87	57.38	51.39	25.32	9.76	91.84	66.40	64.82
w/ 5.0M GUIEnv	40.21	62.61	60.73	35.09	16.09	**92.98**	71.20	68.30
w/ 7.5M GUIEnv	43.60	63.46	65.34	40.67	20.06	91.56	71.50	68.58
w/ 10M GUIEnv	**44.12**	**64.78**	**68.02**	**47.96**	**23.28**	91.77	**74.90**	**70.57**

GUI tasks. Mind2Web has 2000 tasks with high-level instructions in website scenarios. AITW has 30k instructions and corresponding 715k operation trajectories in smartphone scenarios. To keep consistency with previous works, we choose the processed version of these two tasks by SeeClick (Cheng et al., 2024) and use the same evaluation methods as it does. We remove the "General" column of AITW in the evaluation because we convert and use a subset of the AITW-General dataset into our GUIAct dataset.

We mainly compared our GUI agents with their baseline VLMs. The results in Table 4 and Table 5 demonstrate that GUICourse helps GUI agents achieve better performance than their baseline VLMs. (1) In the Mind2Web task, Qwen-GUI exhibits a remarkable increase of 2-5 points in step success rate (StepSR) compared to its baseline Qwen-VL. Fuyu-8B and MiniCPM-V, which have weaker fundamental abilities, show more improvements of approximately 10 points. (2) In the AITW task, our agents still have better performance than their baseline VLMs. While, these improvements are relatively smaller compared to the Mind2Web task, likely because we use more data (e.g. GUIEnv and GUIChat) in the website scenario.

4.3 Ablation Study

In this section, we show the influence of VLMs' OCR and grounding abilities on simple GUI navigation tasks. We adjust the amount of the GUIEnv-global dataset in the pertaining data and analyze the final performances of MiniCPM-GUI.

Metrics of OCR and grounding abilities. We randomly sample 1.8k samples from the GUIEnv-local dataset as the test data of ***Bbox2Text and Text2Bbox*** tasks, testing the OCR and grounding ability of agents. We evaluate the Bbox2Text task by EM and F1 scores, which is the same as the SQuAD MRC (Rajpurkar et al., 2016) task. We evaluate the Text2Bbox task by IoU scores (IoU@0.2, 0.5, and 0.7).

Results. The performances of MiniCPM-GUI with different settings are shown in Table 6. We demonstrate that our GUIEnv dataset is useful to improve both the fundamental abilities of VLMs and the GUI navigation ability. As shown in Table 6, the scores of the "text2bbox" and "bbox2text" tasks, as well as the performance on GUI navigation tasks (web-single) is rising during the increase in the amount of GUIEnv data. We can see that the improvement is huge from "without GUIEnv" to "2.5M GUIEnv", which demonstrates that OCR and grounding abilities are necessary. The amount of GUIEnv data does not influence the "Type EM" because predicting action names is position-free.

4.4 Case Study

Multi-Turn Dialogue. As shown in Figure 3, we show a multi-turn dialogue example of our GUI agents. This example uses several unseen website screenshots and illustrates its proficiency across four key areas: (1) Analyzing the functions of GUI systems' elements; (2) Answering questions using its internal knowledge; (3) Creating actionable steps to accomplish specific tasks; (4) Extracting textual information from the screenshot along with its coordinates through grounding.

Simulated Environment. To evaluate the robustness of our GUI agents, we use a simulated smartphone environment by Android Studio[2] to do interactive testing. As shown in Figure 4, we let our GUI agent execute the same instruction with different initial environments. The execution processes are shown in the the left three and the right two screenshots respectively. When there is no "setting" icon on the initial environment, our GUI agent swipes first to find the "setting" icon. While the "setting" icon appears on the initial environment, our GUI agent taps directly. That means our agents can distinguish the detailed changes and know the icon of "settings", not only learn the fixed

[2]https://developer.android.com/

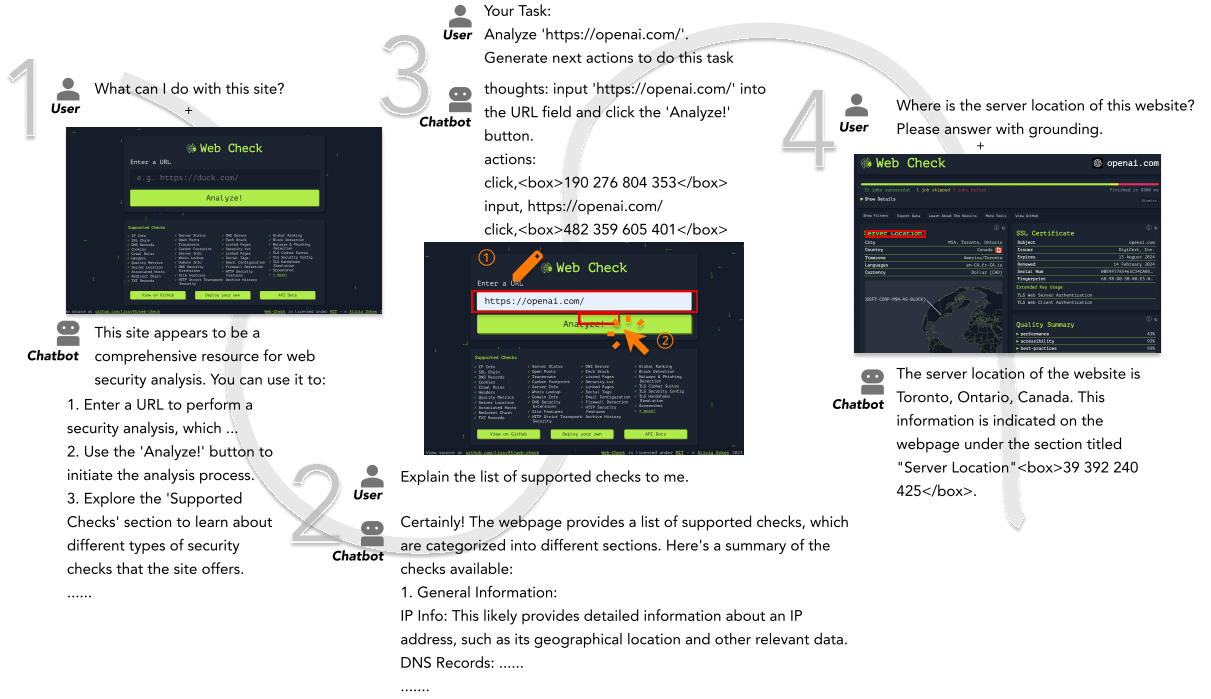


Figure 3: An example of a multi-turn conversation with our GUI agent, MiniCPM-GUI. We first ask the agent about possible operations on this website, then let it explain the terminologies in the screenshot. After that, we test the grounding ability of the agent by giving it a GUI navigation instruction and an information search question.

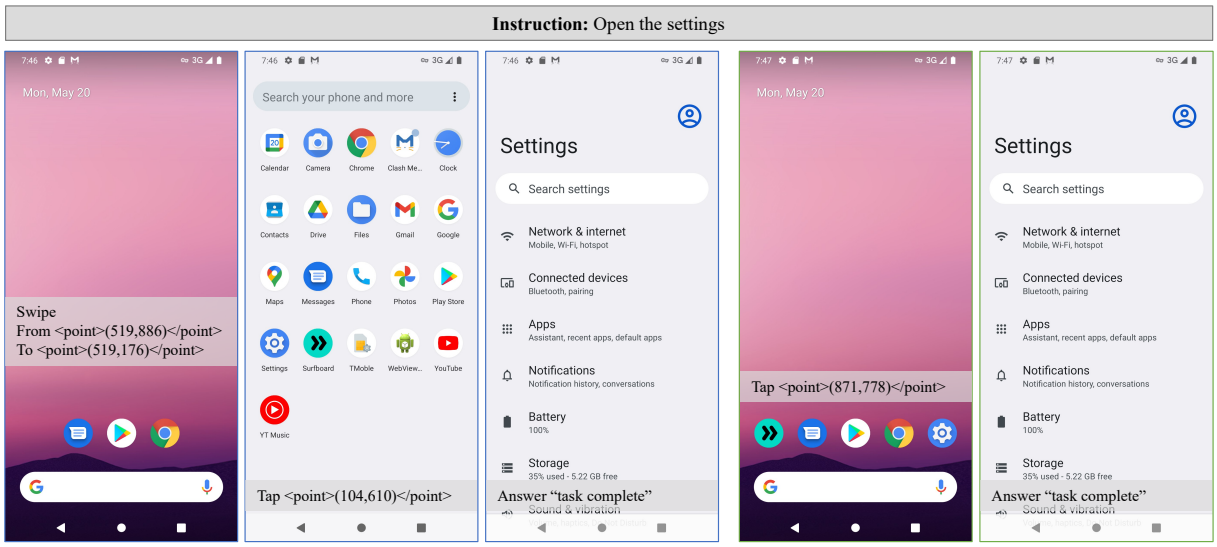


Figure 4: An example of our GUI agent executing the same instruction with different environments. The numbers in the "<point></point>" are the related positions from 0 to 1000, and the "(500,500)" corresponds to the center point of the screenshot.

action sequences in the training data. A long-chain example is shown in Appendix B.3.

5 Conclusion

In this work, we provide a comprehensive pipeline to train visual-based GUI agents from general VLMs. We construct GUICourse, a group of complete datasets (GUIEnv, GUIAct, and GUIChat) to help train GUI agents from general VLMs from three aspects respectively: improving the OCR and grounding abilities of VLMs, enhancing VLMs' GUI knowledge, and improving GUI agents' interaction ability. We train several GUI agents, evaluate them on GUI navigation tasks, and try them on simulated Android environments. Meanwhile, we conduct ablation studies to show the influence of OCR and grounding abilities.

Limitations

We use pretraining and supervised fine-tuning for training GUI agents using various VLMs. However, these might not be enough to achieve assistant-level GUI agents. We are considering the reinforcement learning methods such as RLHF in the future. We provide training data about website environments and fuse data about smartphone environments in our training. However, to build a general GUIAgent, only website and smartphone scenarios are not enough. We would like to build or gather more GUI instruction data about computer systems and professional software in the future.

References

2024. Minicpm-v.

Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, et al. 2021. Uibert: Learning generic multimodal representations for ui understanding. *arXiv preprint arXiv:2107.13731*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023a. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023b. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*.

Rohan Bavishi, Erich Elsen, Curtis Hawthorne, Maxwell Nye, Augustus Odena, Arushi Somani, and Sağnak Taşırlar. 2023. Introducing our multimodal models.

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.

Dongping Chen, Yue Huang, Siyuan Wu, Jingyu Tang, Liuyi Chen, Yilin Bai, Zhigang He, Chenlong Wang, Huichi Zhou, Yiqiang Li, et al. 2024. Gui-world: A dataset for gui-oriented multimodal llm-based agents. *arXiv preprint arXiv:2406.10819*.

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

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

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. *arXiv preprint arXiv:2306.06070*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Hiroki Furuta, Ofir Nachum, Kuang-Huei Lee, Yutaka Matsuo, Shixiang Shane Gu, and Izzeddin Gur. 2023. Multimodal web navigation with instruction-finetuned foundation models. *arXiv preprint arXiv:2305.11854*.

Difei Gao, Lei Ji, Luowei Zhou, Kevin Qinghong Lin, Joya Chen, Zihan Fan, and Mike Zheng Shou. 2023. Assistgpt: A general multi-modal assistant that can plan, execute, inspect, and learn. *arXiv preprint arXiv:2306.08640*.

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

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. 2023. Cogagent: A visual language model for gui agents. *arXiv preprint arXiv:2312.08914*.

Jinyi Hu, Yuan Yao, Chongyi Wang, Shan Wang, Yinxu Pan, Qianyu Chen, Tianyu Yu, Hanghao Wu, Yue Zhao, Haoye Zhang, et al. 2023. Large multilingual models pivot zero-shot multimodal learning across languages. *arXiv preprint arXiv:2308.12038*.

Peter C Humphreys, David Raposo, Tobias Pohlen, Gregory Thornton, Rachita Chhaparia, Alistair Muldal, Josh Abramson, Petko Georgiev, Adam Santoro, and Timothy Lillicrap. 2022. A data-driven approach for learning to control computers. In *International Conference on Machine Learning*, pages 9466–9482. PMLR.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2023. Language models can solve computer tasks. *arXiv preprint arXiv:2303.17491*.

Kenton Lee, Mandar Joshi, Iulia Turc, Hexiang Hu, Fangyu Liu, Julian Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. 2023. Pix2struct: Screenshot parsing as pretraining for visual language understanding. *Preprint*, arXiv:2210.03347.

Bo Li, Peiyuan Zhang, Jingkang Yang, Yuanhan Zhang, Fanyi Pu, and Ziwei Liu. 2023a. Otterhd: A high-resolution multi-modality model. *arXiv preprint arXiv:2311.04219*.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023b. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19730–19742. PMLR.

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

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

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. 2018. Reinforcement learning on web interfaces using workflow-guided exploration. *arXiv preprint arXiv:1802.08802*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023a. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*.

Shilong Liu, Hao Cheng, Haotian Liu, Hao Zhang, Feng Li, Tianhe Ren, Xueyan Zou, Jianwei Yang, Hang Su, Jun Zhu, et al. 2023b. Llava-plus: Learning to use tools for creating multimodal agents. *arXiv preprint arXiv:2311.05437*.

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

Kaixin Ma, Hongming Zhang, Hongwei Wang, Xiaoman Pan, and Dong Yu. 2023. Laser: Llm agent with state-space exploration for web navigation. *arXiv preprint arXiv:2309.08172*.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

OpenAI. 2023. Gpt-4 technical report.

Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, and Furu Wei. 2023. Kosmos-2: Grounding multimodal large language models to the world. *arXiv preprint arXiv:2306.14824*.

Yujia Qin, Zihan Cai, Dian Jin, Lan Yan, Shihao Liang, Kunlun Zhu, Yankai Lin, Xu Han, Ning Ding, Huadong Wang, et al. 2023. Webcpm: Interactive web search for chinese long-form question answering. *arXiv preprint arXiv:2305.06849*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. 2023. Android in the wild: A large-scale dataset for android device control. *arXiv preprint arXiv:2307.10088*.

Peter Shaw, Mandar Joshi, James Cohan, Jonathan Berant, Panupong Pasupat, Hexiang Hu, Urvashi Khandelwal, Kenton Lee, and Kristina Toutanova. 2023. From pixels to ui actions: Learning to follow instructions via graphical user interfaces. *arXiv preprint arXiv:2306.00245*.

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

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. 2023a. Adaplanner: Adaptive planning from feedback with language models. *arXiv preprint arXiv:2305.16653*.

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

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

Zhiqing Sun, Sheng Shen, Shengcao Cao, Haotian Liu, Chunyuan Li, Yikang Shen, Chuang Gan, Liang-Yan Gui, Yu-Xiong Wang, Yiming Yang, et al. 2023b. Aligning large multimodal models with factually augmented rlhf. *arXiv preprint arXiv:2309.14525*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Sagar Gubbi Venkatesh, Partha Talukdar, and Srini Narayanan. 2022. Ugif: Ui grounded instruction following. *arXiv preprint arXiv:2211.07615*.

Penghao Wu and Saining Xie. 2023. V*: Guided visual search as a core mechanism in multimodal llms. *arXiv preprint arXiv:2312.14135*.

Nancy Xu, Sam Masling, Michael Du, Giovanni Campagna, Larry Heck, James Landay, and Monica S Lam. 2021. Grounding open-domain instructions to automate web support tasks. *arXiv preprint arXiv:2103.16057*.

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

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

Jiabo Ye, Anwen Hu, Haiyang Xu, Qinghao Ye, Ming Yan, Guohai Xu, Chenliang Li, Junfeng Tian, Qi Qian, Ji Zhang, Qin Jin, Liang He, Xin Alex Lin, and Fei Huang. 2023. Ureader: Universal ocr-free visually-situated language understanding with multimodal large language model. *Preprint*, arXiv:2310.05126.

Tianyu Yu, Yuan Yao, Haoye Zhang, Taiwen He, Yifeng Han, Ganqu Cui, Jinyi Hu, Zhiyuan Liu, Hai-Tao Zheng, Maosong Sun, et al. 2023. Rlhf-v: Towards trustworthy mllms via behavior alignment from fine-grained correctional human feedback. *arXiv preprint arXiv:2312.00849*.

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

Jiarui Zhang, Jinyi Hu, Mahyar Khayatkhoei, Filip Ilievski, and Maosong Sun. 2024. Exploring perceptual limitation of multimodal large language models. *arXiv preprint arXiv:2402.07384*.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. 2023. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*.

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

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. 2023. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*.

Yichen Zhu, Minjie Zhu, Ning Liu, Zhicai Ou, Xiaofeng Mou, and Jian Tang. 2024. Llava-phi: Efficient multi-modal assistant with small language model. *arXiv preprint arXiv:2401.02330*.

A Datasets

A.1 Action Space

In previous works, various definitions of the GUI agents' action space have been proposed. However, these approaches often suffer from one or more of the following issues: (1) The partitioning of actions is disorganized and lacks coherence. (2) The action space fails to encompass all possible scenarios in GUI systems. To address these limitations, our work introduces a unified action space for GUI systems. Specifically, we have designed seven distinct groups of actions tailored for task-oriented GUI agents, as depicted in Figure 5:

(1) Pointing to something (e.g., buttons, icons): This action allows the agent to shift focus to specific elements within the GUI system, such as clicking on a search box or selecting empty spaces on websites. These actions include "click", "hover", and "tap".

(2) Inputting something: Agents can input relevant information into the GUI system, such as inputting search queries in a search box. These actions include "input", "type", and "paste".

(3) Browsing through pages: These actions enable the agent to navigate through pages within the GUI system, including page-up and page-down functionalities. These actions include "scroll", "drag", and "swipe".

(4) Logging information: Agents can log key information that aids in accomplishing the task, such as writing texts or copying essential paragraphs. These actions include "select text", "copy", and "write (by OCR)".

(5) Submitting (tables, search text): Agents can submit tables or the search context within the GUI system. We use the "enter" action to achieve this target.

(6) Selecting a value: This action allows the agent to choose a specific value from a drop-down menu. We use the "select" action to achieve this target.

(7) End of task: Agents can signal the completion of the task or respond (e.g., an answer, task completion status, or indication of task impossibility). We use the "answer" action to achieve this target.

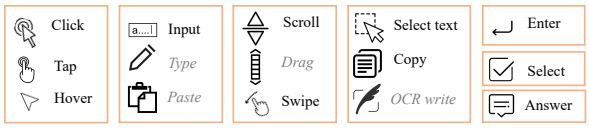


Figure 5: The actions in our action space. Our datasets utilize the actions displayed in black. The actions shown in gray are not used, as their functions can be effectively replaced by other actions within the same group.

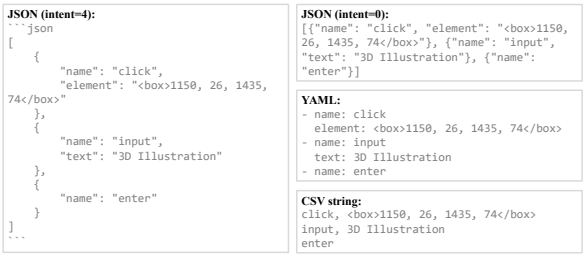


Figure 6: The four different action parse formats.

Actions Conversation from AITW dataset. During the contribution of our GUIAct-smartphone dataset, we use a subset of the AITW (Rawles et al., 2023) dataset, a large smartphone navigation data with General/Install/Google-Apps/Single/WebShopping tags. We use a subset of the partition with the "General" tag. To keep the consistency of our action space, we convert their actions: (1) dual-point gesture: we split the "dual-point gesture" actions into "tap" and "swipe" by the distance of the touch point and the left point. Then, we use the touch point as the "point" parameter of the "tap" action and use both points as the "dual-point" parameter of the "swipe" action.

(2) type: we rename the "type" action as the "input" action, and reserve the "text" parameter of "type" as the "text" parameter of "input".

(3) enter: we reserve the "enter" action.

(4) go back/home: we reflect on these two actions to "tap" actions. We find the position of the two buttons on the bottom navigation bar as the "point" parameter of the "tap“ action.

(5) task complete/impossible: we convert these two actions to "answer" actions, and the "task complete" and "task impossible" are the "text" parameters of the "answer" action.

The Format of Actions. Two types of formats need to be considered for actions: parse format and position format. We need a structured text format to extract actions from GUI agents' natural language responses, referred to as the parse format. Also, we need an absolute or related format to express the position information of the actions, referred to as the position format.

As for **parse format**, we try several formats such as JSON and YAML, as shown in Figure 6. In our experiments, we utilize JSON format as the action parse format for Qwen-VL and Fuyu-8B. For MiniCPM-V, we opt for the CSV format, which closely resembles natural language conventions.

As for **position format**, we use different position formats based on the models' preferences: absolute position format for Fuyu-8B and related position format for Qwen-VL. We also have a variant of related position formats, used for MiniCPM-V, which is scaling the related value to [0,1000) as integers (e.g., "<box>481 565 506 592</box>"). We show the position formats and the four parameters in Table 7.

A.2 LLM-Auto Annotation

We use LLMs to help our contribution processing for GUIAct (web-single), GUIAct (web-multi), and GUIChat datasets.

GUIAct (web-single): Website Selection. We use GPT-4 to gather website scenarios and website URLs, the prompts are shown in Figure 7.

GUIAct (web-single): LLM-Auto Annotation. We use GPT-4V to obtain instruction-action pairs for each web screenshot. To help GPT-4V comprehend both the content and the index of the elements, we provide two images for each request. As shown in Figure 8, one image is the origin screenshot without any mask box, and the other image is a revised screenshot with an element index and mask box on each element. Due to that, GPT-4V can get full information on the website viewport in the first image and get the element ID information in the second image.

GUIAct (web-multi): Questions Acquisition. We use LLMs (GPT-3.5 and Claude2) to generate English and Chinese questions for English and Chinese websites respectively. These questions all have special answers due to the information on the websites. The prompt template is shown in Figure 9.

GUIChat: Question-Answer Pair Generation. We use GPT-4 and employ a pipeline for generating both single-turn and multi-turn question-and-answer pairs. For single-turn question-and-answer generation, the process begins with generating 4 types of questions, followed by generating

Table 7: The position and distance formats. An element's position is composed of its left-top point and right-down point and can be expressed as <box>x1, y1, x2, y2</box>. A point's position is composed of its x and y coordinates, expressed as <point>x, y</point>. The position of dual points is composed of its start point and end point. The scroll parameter's distance information uses pixel values. There are "down" and "right" two parameters to signal y dim and x dim respectively. If the action is scrolling up, the value of the "down" parameter is negative.

Actions	Param.	Absolute	Related
Click, Hover, Select	**element**	<box>657, 434, 691, 455</box>	<box>0.481, 0.565, 0.506, 0.592</box>
Tap	**point**	<point>362, 1412</point>	<point>0.503, 0.981</point>
Swipe, Select_Text	**dual points**	from: <point>884, 236</point> to: <point>958, 233</point>	from: <point>0.730, 0.311</point> to: <point>0.791, 0.307</point>
Scroll	**scroll**	down 496 right 0	down 0.65 right 0

Prompt for website scenarios

Please provide 30 scenarios of using the internet and describe what conveniences this scenario can provide to people?
Please output in JSON format: [{"field": string, "description": string, "help": string}]

Response

```json
[
  {
    "scenario": "Online Shopping",
    "description": "Buying products or services via the internet.",
    "help": "Convenience of shopping from home, access to worldwide products, and easy price comparisons."
  },
  {
    "scenario": "Remote Work",
    "description": "Working from a non-office location using the internet.",
    "help": "Eliminates commute time, offers flexible schedules, and can lead to improved work-life balance."
  },
  {
    "scenario": "E-Learning",
    "description": "Attending educational courses online.",
    "help": "Enables remote learning, access to diverse courses, and self-paced education."
  },
  ...
]```
```

Prompt for website URLs

In the scenario of <E-Learning>, please provide 10 famous websites and their corresponding URLs.
Please output in JSON format: [{"field": string, "website name": string, "website URL": url_string}]

Response

```json
[
  {
    "scenario": "E-Learning",
    "website_name": "Coursera",
    "website_url": "https://www.coursera.org"
  },
  {
    "scenario": "E-Learning",
    "website_name": "edX",
    "website_url": "https://www.edx.org"
  },
  {
    "scenario": "E-Learning",
    "website_name": "Khan Academy",
    "website_url": "https://www.khanacademy.org"
  },

  ...

]```
```

Figure 7: The prompts for acquiring website scenarios and website URLs.

answers and answers with grounding. Multi-turn question-and-answer generation involves generating one round of questions and answers per turn based on the history of the conversation, with a maximum of five rounds. The prompts are shown in Figure 11.

A.3 Crowd-sourced Annotation

We hire humans to help us check or annotate the GUIAct (web-single) and GUIAct (web-multi) datasets.

Instructions Given to Participants. We have a data annotation handbook for the annotation task for GUIAct (web-multi) data. We show the main content in Figure 12.

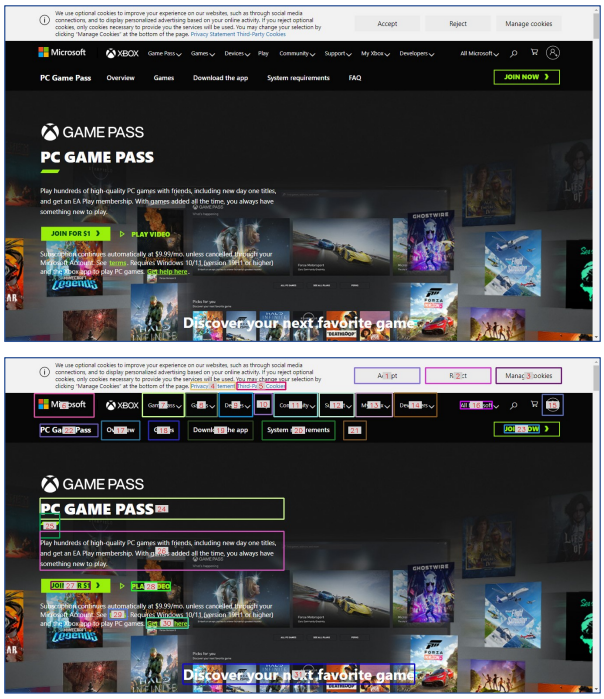
Cost. (1) We pay 0.6 RMB per item to check the LLM automatic annotation samples in the GUIAct (web-single) data. The checking efficiency of each item is about 40 seconds. So, the hourly salary we pay the annotator is 54 RMB. (2) We pay 5 RMB per item to annotate GUIAct (web-multi) data, and the annotation efficiency is approximately 5 minutes per item. So, the hourly salary we pay the annotator is 60 RMB. The prices we offer are higher than the minimum hourly wage in Beijing, China (13.91 RMB).

B Experiments

B.1 Training Details

Dataset Partitions. Our GUICourse includes three datasets: GUIEnv, GUIAct, and GUIChat.

(1) GUIEnv has 10M samples for pre-training (GUIEnv-global) and 0.7M samples for SFT (GUIEnv-local). Meanwhile, we split 1.8k samples in GUIEnv-local as test data for OCR and grounding abilities evaluation.



Prompt for website navigation data

1.Please describe the first image in detail.
2. Think about what you can accomplish by manipulating images (mouse or keyboard) ?
3. Based on the results of thinking given 6 tasks and possible operational steps, and combined with the second screenshot of the element sequence to explain.
4. Give specific mission goals, not vague or abstract descriptions, and avoid words like "Something".

candidate actions
type Click = {action: "click", element: number}
type Input = {action: "input", element: number, text: string}
type Enter = {action: "enter"}
type Scroll = {action: "scroll", "scroll_down": pixel}
type Answer = {action: "answer", text: string}

action explains
Scroll action need pixel number as input, and it can be a negative number.
Answer action is needed when you want to response to a question using the information in this image.

Please use the JSON format for output:
```
{
  "Description": string,
  "Task&Action": [
    {
      "Task": string,
      "Action_abstract": string,
      "Action_list": Click | Input | Enter | Scroll | Answer
    },
    ...
  ]
}
```

Response

```json
{
  "Description": "List of tasks with corresponding actions on the Xbox Game Pass webpage.",
  "Task&Action": [
    {
      "Task": "Accept cookie policy",
      "Action_abstract": "Click on accept button in the cookie policy banner.",
      "Action_list": { "action": "click", "element": 5 }
    },
    {
      "Task": "Join Game Pass for $1",
      "Action_abstract": "Click on the Join for $1 button.",
      "Action_list": { "action": "click", "element": 27 }
    },
    {
      "Task": "Scroll down to view more games",
      "Action_abstract": "Scroll down to reveal more of the game selection.",
      "Action_list": { "action": "scroll", "scroll_down": 1000 }
    },
    {
      "Task": "Navigate to the System Requirements page",
      "Action_abstract": "Click on the System Requirements link in the navigation menu.",
      "Action_list": { "action": "click", "element": 20 }
    }
  ]
}
```

Figure 8: The prompts and images for acquiring question-action pairs by GPT-4V.

(2) GUIAct is composed of three partitions: GUIAct (web-single), GUIAct (web-multi), and GUIAct (smartphone). GUIAct (web-single) is a dataset with single-step actions in website scenarios, which has about 67k instructions automatically generated by 13k screenshots. GUIAct (web-multi) is a dataset with multi-step actions and 5,696 complex instructions, which are annotated by humans. Also, we simply 9,157 instructions from AITW for smartphone scenarios, referred to as GUIAct (smartphone). There are a total of 67k, 44k, and 67k samples for the three partitions, and we split about 1.4k, 1k, and 2k samples as testing data. The residual samples are used for SFT training.

(3) GUIChat has 44k single-turn QA pairs and 6k multi-turn dialogues. We use all the samples for SFT.

Training Stages. Our training pipeline can be split into two stages due to the training targets. In the pre-training stage, we mainly improve the OCR and text-grounding abilities of VLMs, especially in complex website scenarios. In the SFT training stage, we mainly enhance the VLMs' knowledge about GUI systems, such as the function of elements and control actions, as well as the conversation abilities.

We only execute the whole training pipeline based on MiniCPM-V. For other used VLMs (such as Qwen-VL and Fuyu-8B), for which we can't acquire their pre-training data and details, we only execute the SFT training stage. (1) Pre-training stage: we mix GUIEnv-global data in the pre-training data and then execute the training. (2) SFT stage: we merge the data from three partitions as SFT data: 10% of the GUIEnv-local, all the data of GUIAct,

Prompt for website navigation questions
Please provide some instructions about <Information Inquiry> that are intended to find a specific piece of information or seek an answer to a question from a specific web page.
In this task, the instructions to be provided are for the <Shopping>-<Online Mall> scenario, where you need to consider what kind of information the user might need to want to know in that scenario and rely on that information to model the user's question (i.e., instruction).
Please give <35> such directives that are asking questions about information that may be contained in the website <eBay(https://www.ebay.com/)>.
Please keep the following issues in mind when designing directives:
1. short, with 1~2 sentence description and the serial number of the current directive before the directive.
2. rich in format, instructions should have as many different expressions as possible, and have a variety of linguistic expressions.
3. rich content, the instruction as far as possible to ask different content, with the diversity of content.
4. Specificity, the objects, people and events involved in the instruction should be concrete things that exist in reality, and should not be replaced by pronouns such as <a certain, these, this>.
5. The language of the instructions is English, and <###> is attached to the directive as a separator, followed by its Chinese translation.

Figure 9: The prompt template for acquiring questions about information inquiry on a designated website.

and all the data of GUIChat. The learning rate of MiniCPM-V is 5e-7, while the learning rate of Qwen-VL and Fuyu-8B is 1e-5. All the learning scheduler types are "cosine".

Resources Cost. For each agent, We use 1 $\times$ GPU A100 for inference, 8 $\times$ GPU A100 for SFT, and 128 $\times$ GPU A100 for pre-training. As for MiniCPM-V (3.1B), it takes about 2 days for the whole pre-training stage using the GUIEnv dataset and about 8 hours for the SFT stage using GUIAct and GUIChat datasets with 3 epochs. Moreover, it takes several hours for the evaluation. As for Qwen-VL (9.6B) and Fuyu-8B (9.4B), we only do SFT and inference processes. It takes about 1 day for training with the same SFT data.

B.2 The Evaluation of Actions.

Type Exact Match Score. We use Type Exact Match Score (Type EM) to calculate the accuracy for predicting the names of actions, and not care about the parameters of the actions. The Type EM score is also called action type match (Zhan and Zhang, 2023) or intend match (Lù et al., 2024) in other works.

Action Score and Success Rate. We calculate action scores and success rates for every action, as shown in Figure 8. The action scores are calculated similarity between the predicted actions and the golden actions. However, the action success rates judge if the predicted actions achieve the same function as the golden actions. We design different rules to calculate the action scores and success rates for various actions in our action space.

(1) "click" and "hover": For evaluating the two actions, we employ the Intersection over Union (IoU) score of their respective element regions (referred to as boxes) as their action scores. To determine the action success rate, we compare the

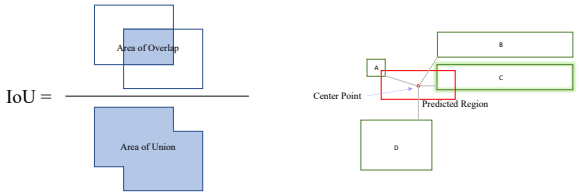


Figure 10: The IoU score and the element attach method. The left partition is the metrics of IoU scores and the right partition is the sketch of the element attach method. We first calculate the center point of the predicted element, and then calculate the distances (the gray dashed line) from this point to all the candidate elements. We select the element with the minimum distance as the final predicted element (the element "C" in this figure).

predicted elements' IDs with the golden elements' IDs using an exact match criterion. It is worth noting that we associate the predicted region with the element having the minimum distance, as illustrated in Figure 10. Generally, the success rates of "click" and "hover" actions (referred to as Cli.Acc) tend to be higher than their corresponding action scores.

(2) "tap": For the "tap" action, there are no candidate or golden regions available in the datasets. Instead, we determine the success of the action by calculating the distance between the predicted point and the golden point using the related position format. If the distance is less than 0.14, the action is considered successful, and the action score is calculated as $1 - \frac{distance}{0.14}$. Otherwise, the action score and success rate are set to 0.0.

(3) "input" and "answer": We utilize the F1 score as the action score. If the action score exceeds 0.5, we consider the action to be successful.

(4) "scroll" and "swipe": We convert the parameters of these actions into four directions: up, down, right, and left. If the predicted and golden results have consistent directions, the action is deemed

Table 8: The metrics of different actions.

Action	Param.	Examples	Score	Is Success
Click	element	<box>32, 20, 56, 44</box>	IoU	EM(el.)
Tap	point	<point>183, 505</point>	distance	distance < 0.14
Hover	element	<box>299, 118, 347, 166</box>	IoU	EM(el.)
Input	text	"French butter cake"	F1	F1 > 0.5
Scroll	pixel	down 500 right 0	direction	direction
Swipe	dual points	from <point1> to <point2>	direction	direction
Select Text	dual points	from <point1> to <point2>	IoU	IoU > 0.5
Copy	-	-	EM(type)	EM(type)
Enter	-	-	EM(type)	EM(type)
Select	element, text	the <element> value to "Female"	IoU + F1	EM(el.) + F1
Answer	text	"On April 16, 2021."	F1	F1 > 0.5

successful, and the action score is set to 1.0. Otherwise, the action score and success rate are set to 0.0.

(5) "copy" and "enter": There are no parameters in both actions. The action scores and the success rates are equal to the Type EM scores.

(6) “select_text”: This action involves selecting text within a swiped region. We evaluate the action score using the IoU metric. If the IoU score exceeds 0.5, we consider the action to be successful.

(7) “select”: there are two parameters (element and text) in this action. We use the IoU and F1 score for its element parameter and text parameter respectively. We calculate the average of the two scores as the action score. The action is considered successful only if both parameters meet the success criteria.

Step Success Rate. In our approach, we encounter scenarios where multiple actions need to be performed in a single step. For instance, based on a given screenshot, the required actions could be a combination of "click," "input," and "enter." Therefore, our datasets necessitate models that can predict one or multiple actions in a single step. To Evaluate the performance of a single step, *(1)* we first compare the top-n predicted actions with the golden actions, where n represents the number of golden actions. If the number of predicted actions is less than n, the scores of missing actions are set to 0.0. *(2)* Then, we calculate the average action score for each action that is being compared, providing an overall action score. *(3)* Finally, we calculate the step Success Rate (StepSR). We consider a step successful only if all the actions within that step are successfully executed.

B.3 Case Study

We show a long-chain example executing in the simulated smartphone environment in Figure 13.

C Data Examples

We show some examples of GUICourse in this section. An example of GUIEnv-global and an example of GUIEnv-local are shown in Figure 14. Some examples of GUIAct (web-single) are shown in Figure 15, and an example of GUIAct (web-multi) is shown in Figure 16. A single-turn QA example of GUIChat is shown in Figure 17 and a multi-turn QA example of GUIChat is shown in Figure 18.

D Ethical Considerations and Societal Impacts

Ethical Considerations. We provide fair wages for the annotators when constructing our datasets, and there is no personally identifiable information. When using existing assets, we cite the creators and use the open-sourced datasets or models.

Societal Impacts. Our data in GUICourse don’t need personal information or offensive content. Meanwhile, we don’t select websites that might include personal information or offensive content. However, some screenshots in our GUIEnv dataset are collected from the Cleaned Common Crawl Corpus, so we cannot guarantee that these website screenshots are absolutely free of personally identifying information or offensive content.

Prompt for single-turn QA (step1: generate questions)
I will provide you with a set of custom structured metadata representing a screenshot of a webpage. This metadata will enable you to fully comprehend the webpage. Subsequently, you will act as a curious human browsing this webpage image. Upon first viewing this image, you will ask two separate questions, randomly selected from the following four categories:

1. Observation-oriented question (Focusing on basic visual understanding of the layout and content of the webpage)
2. Individual-needs-centric question (Centered on the needs of a curious human)
3. World-knowledge-oriented question (Requiring additional world knowledge to answer)
4. Most challenging question (This should be the shortest question to ask but the longest to answer, requiring as much step-by-step reasoning as possible; it needs multiple pieces of information from the page, complex reasoning, world knowledge, and explanatory answers)

Here is the format of this metadata: The metadata structure and keys are organized into multiple levels, with each element enclosed in '<>' and containing comma-separated data items. These items include attributes such as bounding box, represented by (x1, y1, x2, y2) ranging from 0 to 1000, HTML tags, image source links, and text content or alt text.

{*Metadata*}
Please generating questions.

Prompt for single-turn QA (step2: generate answers)
I will provide you with a set of custom metadata representing a webpage screenshot. Through this metadata, you will fully understand the webpage and be able to identify locations through bounding boxes.

Your primary role is to assist a human user who is curious about and exploring the internet. You will receive questions from this user and are expected to provide precise answers, including step-by-step reasoning and detailed assistance. Finally, you should offer further inferences and insights based on the available information.

Here is the format of this metadata: The metadata structure and keys are organized into multiple levels, with each element enclosed in '<>' and containing comma-separated data items. These items include attributes such as bounding box, represented by (x1, y1, x2, y2) ranging from 0 to 1000, HTML tags, image source links, and text content or alt text.

{*Metadata*}
{*Question*}
Please generating answer.

Prompt for single-turn QA (step3: generate grounding boxes)
I will provide you with a set of custom metadata representing a webpage screenshot. Through this metadata, you will fully understand the webpage and be able to identify locations through bounding boxes, represented by (x1, y1, x2, y2) ranging from 0 to 1000.
Your task is to use the metadata you have to add appropriate bounding box annotations to an already written complete answer. These annotations will link the content of the answer to the original text's location on the webpage. The format is to follow each first mention of information related to a webpage element with a bracket containing the element's bounding box information, strictly adhering to the following format:
first mention of the webpage element [x1, y1, x2, y2]

Here is the format of this metadata: The metadata structure and keys are organized into multiple levels, with each element enclosed in '<>' and containing comma-separated data items. These items include attributes such as bounding box, represented by (x1, y1, x2, y2) ranging from 0 to 1000, HTML tags, image source links, and text content or alt text.

{*Metadata*}
{*Answer*}
Please generating answer.

Prompt for multi-turn QA
Your task is to generate multi-turn dialogues using custom metadata that represents a webpage screenshot. This metadata enables you to fully understand the webpage and identify elements through bounding boxes.

The dialogue structure is as follows:
User: Asks a question based on the given metadata.
Assistant: Answers the user's question with a detailed, informative response.

Here is the format of this metadata: The metadata structure and keys are organized into multiple levels, with each element enclosed in '<>' and containing comma-separated data items. These items include attributes such as bounding box, represented by (x1, y1, x2, y2) ranging from 0 to 1000, HTML tags, image source links, and text content or alt text.

{*Metadata*}
Please generating one round conversation.

Figure 11: The prompts for generating the GUIChat dataset.

标注目标 Annotation Target:

现实生活中，人们一般基于视觉信息，并依赖鼠标和键盘和浏览器进行交互，在网页上通过“鼠标点击”、“键盘输入”、“鼠标滚动”，“鼠标拖动”等操作<查找>并<浏览>网页，从而综合互联网中海量的信息来回答相应的问题
In real life, people generally rely on visual information, and rely on the mouse and keyboard to interact with the browser. On the web page, they can <find> and <browse> the web page through "mouse click", "keyboard input", "mouse scroll", "mouse drag" and other operations, so as to integrate the massive information in the Internet to answer the corresponding questions.

具体本次的标注任务，我们需要您首先仔细阅读给出的指令（一般是一个问题），然后通过Google浏览器在相应的网页上查找并浏览，得出该指令的回答，并同时通过Google浏览器插件对您的操作进行记录。在浏览过程中，如果遇到需要记录的信息，可以通过“鼠标框选“ 和“键盘Ctrl+C”随时记录信息，在浏览结束后，需要总结浏览到的信息，并回答指令中的问题，在作答框中填入结果。
For the specific annotation task of this time, we need you to first carefully read the given instructions (usually a question), then search and browse the corresponding webpage through Google Chrome to obtain the answer to the instructions, and at the same time record your actions through the Google Chrome plugin. During the browsing process, if you encounter information that needs to be recorded, you can use the "mouse box selection" and "keyboard Ctrl+C" to record the information at any time. After browsing, you need to summarize the information you have browsed, answer the questions in the instructions, and fill in the results in the answer box.

希望您能够仔细斟酌在浏览器上的每一次操作。界面中您的全部操作（除了点击空白位置外）、与操作相关的数据和最终的摘要以及答案都将被记录以用于指导模型，因此，您需要保证您在搜索相关信息、摘要相关信息和回答问题时进行的操作是精简和有效的。
We hope you can carefully consider every operation on the browser. All your actions in the interface (except for clicking on blank spaces), data related to the actions, and the final summary and answers will be recorded for guiding the models. Therefore, you need to ensure that the actions you perform when searching for relevant information, summarizing relevant information, and answering questions are concise and effective.

操作流程和注意事项 Operating procedures and precautions：

(1) 阅读：仔细阅读指令（英文），理解需要完成的任务（判断时候能够完成，如果不能则放弃该条指令并写明原因），并打开相应的网站（有些网站注册账号并提前登录）。
(1) Reading: Carefully read the instructions (in English), understand the tasks that need to be completed (determine when the y can be completed, if not, abandon the instruction and specify the reason), and open the corresponding website (some websites require registration account and login before your annotation).

(2) 开始记录：确认Google浏览器插件属于启用状态，输入指令编号（数字）、指令类型（选项）和指令本身（文本），并点击“开始记录”按钮。同时调整浏览器页面大小，使其小于 1920 * 1080像素（插件会有相应的提示）。
(2) Start recording: Confirm that the Google Chrome plugin is enabled, enter the command number (number), command type (option), and command itself (text), and click the "Start recording" button. Simultaneously adjust the browser page size to be less tha n 1920 * 1080 pixels (the plugin will provide corresponding prompts).

(3) 搜索：对所要回答的问题直接进行搜索、进行精简后搜索或提取出关键词进行搜索，使搜索引擎能够搜索到有助于回答问题的信息，如搜索结果不能令人满意则更换搜索内容重新搜索。
(3) Search: Directly search for the question to be answered, streamline the search, or extract keywords for search, so that s earch engines can find information that helps answer the question. If the search results are not satisfactory, replace the search c ontent and search again.

(4) 浏览：进入相关的详情页面中浏览信息，并将有助于回答该问题的信息进行记录（通过 copy操作， 插件自动记录）中，按需进行页面滚动，点击超链接，点击过滤按钮等等，直到找到足够信息可以回答指令中的问题。
(4) Browsing: Enter the relevant details page to browse information and record the information that helps answer the question (automatically recorded by the plugin through copy operation). Scroll the page as needed, click on hyperlinks, filter buttons , etc., until enough information is found to answer the questions in the instructions.

(5) 回答：在进行回答时，请注意，搜索和浏览已经结束，这将是最后一个操作。依赖 <Text: 之前的记录>、<Text: 之前复制的信息>、以及<Image: 当前的界面>来回答问题。点击“停止浏览并 作答”，写入问题的回答，若已尽最大努力仍无法获得足够信息以供回答，请在回答中填写<信息不足>或者<网页中不存在答案>并在其后写出无法回答的原因，如果记录合理仍可以视为一条标注好的数据。请注意：用英文回答，保证是流畅完整的句子。
(5) Answer: When answering, please note that search and browsing have ended, and this will be the last action. Relying on<Tex t: Previous Records>,<Text: Previously Copied Information>, and<Image: Current Interface>to answer questions. Click "Stop browsi ng and answer" to write the answer to the question. If you have tried your best but still cannot obtain enough information to an swer, please fill in "insufficient information" or "no answer on the webpage" in the answer and write the reason why you cannot ans wer. If the record is reasonable, it can still be considered as a labeled data. Please note: Answer in English to ensure smooth and c omplete sentences.

(6) 结束记录：点击“确认并提交答案”，将自动结束记录，等待插件下载记录数据。检查记录是否完整。
(6) End recording: Click "Confirm and Submit Answer" to automatically end the recording and wait for the plugin to download t he recording data. Check if the records are complete.

(7) 放弃记录：记录中出现任何问题，可以放弃本次记录，并重新开始。
(7) Abandon Record: If there are any issues with the record, you can abandon the current record and start over.

Figure 12: The main content of our data annotation handbook for GUIAct (web-multi).

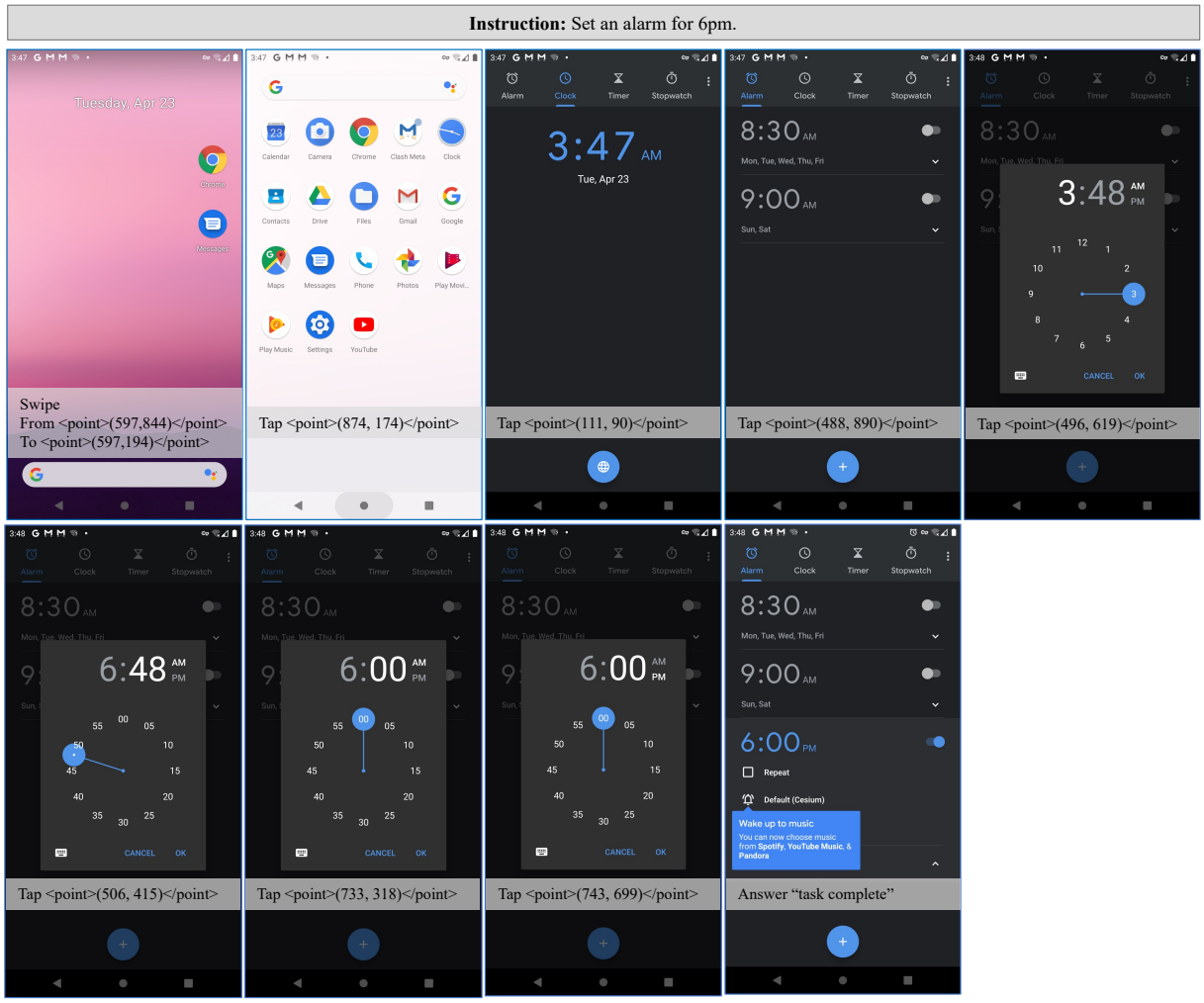


Figure 13: An example of our GUI agent executing the instruction with multiple steps in the simulated smartphone environment.

Figure 14: Some examples to show the data formats of our GUIEnv datasets. The GUIEnv-global task only provides a screenshot website, and the models need to predict all the texts and their positions according to the layout sequence. The GUIEnv-local task requires models to predict the region (a box represented by a left-top point and a right-down point) given the designated text or predict the text given the designated box.

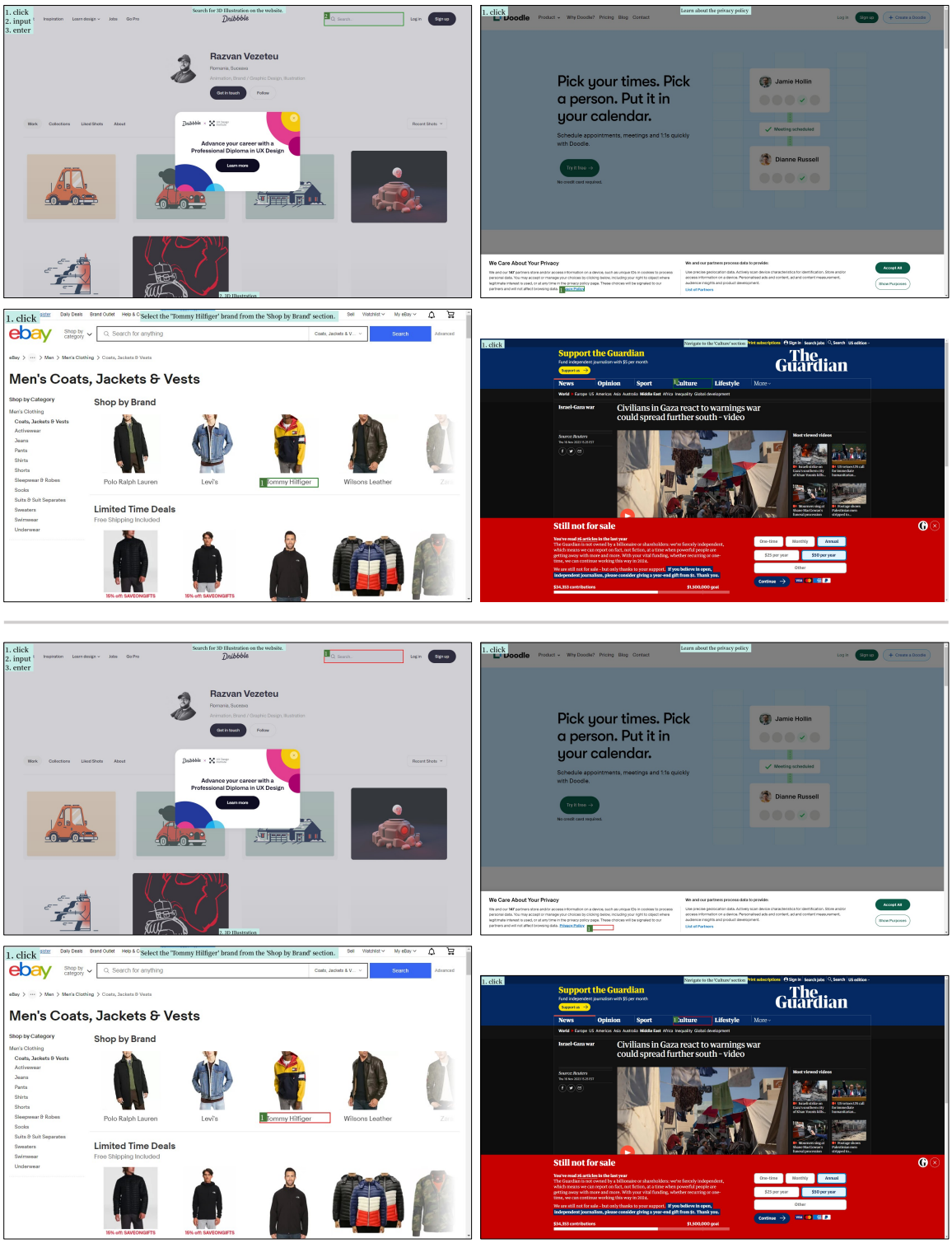


Figure 15: Four examples in the GUIAct-web-single datasets. The instructions are on the top of each screenshot, while the actions' names are on the left-top position. Moreover, we draw a box with an index to represent the "click" action. We use green color to visualize the golden actions and use red color to visualize the predicted actions.

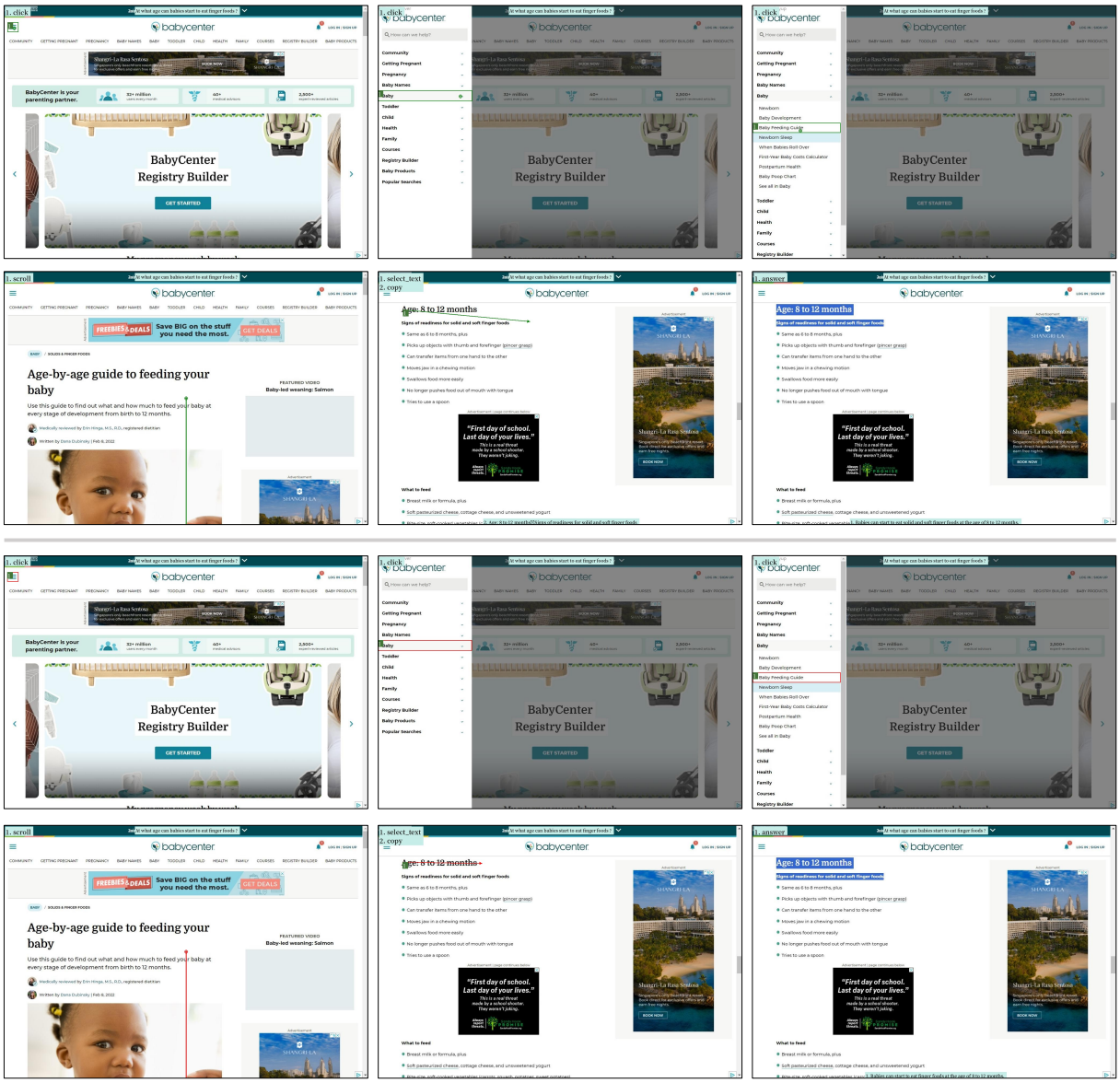


Figure 16: A 6-step navigation example of GUIAct-web-multi datasets. The instructions are on the top of each screenshot, while the actions' names in each step are shown in the left-top position. These graphs show the "click", "scroll", "select_text", "copy" and "answer" actions. We use green color to visualize the golden actions in the top-6 graphs and use red color to visualize the predicted actions in the bottom-6 graphs.

A Single-turn QA Example

Q: What role does the Bi-Parting door play in the operations at the Elmendorf AFB site (answer with grounding) ?

A: The Bi-Parting Door at Elmendorf Air Force Base in Anchorage, Alaska [10,276,390,305], plays a crucial role in the operations due to its design and inbuilt features.

Firstly, the door's configuration: It's a 6-Leaf Bi-Parting Door System [151,333,354,352]. It means the door is divided into six panels that split from the middle, allowing for efficient and swift operations.

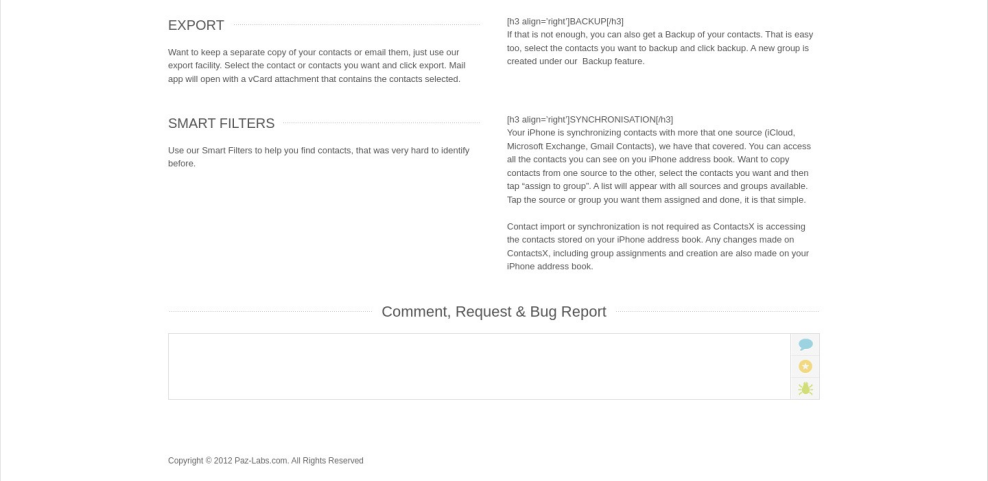
Secondly, the door's size is substantial: Two openings of 175' wide x 24' high each [76,370,238,389]. These dimensions indicate that the door can accommodate large military equipment, including aircraft such as the F-22 [10,407,431,602].

Thirdly, the door is equipped with several features to withstand the harsh Alaskan weather [10,625,77,644]. These features include 3" exterior insulated panel [40,660,223,679] and insulated translucent panels [40,690,237,709] to maintain thermal efficiency, unexposed bottom rails to prevent damage from environmental elements when the doors are closed [40,720,424,757], and dual weather seals for imperviousness to air infiltration and increased energy efficiency [40,768,409,805].

Finally, the doors are controllable from inside [40,816,427,853] and designed to stack inside the building when opened, suggesting usability and space-saving efficiency. Furthermore, the doors feature an electrical interlock switch that automatically disables the drive door if the personnel door is open, thereby enhancing the safety profile [40,864,425,901].

Therefore, the Bi-Parting Door at Elmendorf AFB, implemented by Norco [2,10,428,140], significantly contributes to the smooth functioning and safety of the operations by providing efficient access to large spaces, being weather-resistant, and incorporating the necessary safety mechanisms.

Figure 17: A case of GUIChat's single-turn QA.



A Multi-turn QA Example

Q: I'm interested in knowing how the export function works in this app. Is there a specific process or set of steps I need to follow?
A: Yes, there are specific steps to follow to use the export function of this app. Initially, you need to select the contacts that you wish to export. To do this, you probably need to navigate to the contacts section of the app and choose one or more specific contacts as per your requirements. Once you've selected the desired contacts, you have to click the 'export' option. The app will then generate a vCard attachment consisting of the chosen contacts. This operation specifically opens in the Mail app, where you can further process it—sending the contacts to another party via email, for instance, would be a breeze from here.

Q: You mentioned a "vCard attachment" that gets created upon exporting. Could you tell me more about what a vCard is and why it's useful?
A: Absolutely. A vCard is a digital file format standards for electronic business cards. vCards are typically attached in emails, but can be exchanged in other ways as well, such as through the web or instant messaging. They contain information such as names, phone numbers, addresses, emails, and even photos. In this context, when you export contacts, the application generates a vCard file that contains the selected contacts' details. This feature is useful as it provides a standard format for sharing contact details that can be easily read by a vast majority of email clients and contact management systems. Furthermore, it provides a secure way of transmitting contact information without data loss. It means that whoever receives this vCard can import it into their contact list and will have all the detailed information exactly as it was in your contact list.

Q: Since vCards seem quite useful, can I also create backups in the same format using this app?
A: Yes, this app indeed provides a facility to create backups of your contacts. To do this, you need to select the contacts you wish to backup, much like the export process. Upon clicking 'backup', a new group is created under the Backup feature within the app. However, it's not explicitly mentioned whether these backups are created in the vCard format or a different format, or if they're stored externally or within the app itself. But regardless of the method, this feature offers a layer of data safety, allowing you to restore your contacts if needed.

Figure 18: A case of GUIChat's multi-turn QA.