AppAgent v2: Advanced Agent for Flexible Mobile Interactions

AppAgent v2: 用于灵活移动交互的高级代理

Yanda Li [1,2*] , Chi Zhang [2,4†] , Wenjia Jiang [4] , Wanqi Yang [1] , Bin Fu [2] , Pei Cheng [2] , Xin Chen [2] , Ling Chen [1] , Yunchao Wei [3]

李彦达 [1,2*]， 张驰 [2,4†]， 姜文佳 [4]， 杨万琦 [1]， 付斌 [2]， 程培 [2]， 陈鑫 [2]， 陈玲 [1]， 魏云超 [3]

[1] University of Technology Sydney [2] Tencent [3] Beijing Jiaotong University [4] Westlake University

[1] 悉尼科技大学 [2] 腾讯 [3] 北京交通大学 [4] 西湖大学

[1] Yanda.Li@student.uts.edu.au, ling.chen@uts.edu.au [3] wychao1987@gmail.com [4] chizhang@westlake.edu.cn

[1] Yanda.Li@student.uts.edu.au, ling.chen@uts.edu.au [3] wychao1987@gmail.com [4] chizhang@westlake.edu.cn

Abstract

摘要

With the advancement of Multimodal Large Language Models (MLLM), LLM-driven visual agents are increasingly impacting software interfaces, particularly those with graphical user interfaces. This work introduces a novel LLM-based multimodal agent framework for mobile devices. This framework, capable of navigating mobile devices, emulates humanlike interactions. Our agent constructs a flexible action space that enhances adaptability across various applications including parser, text and vision descriptions. The agent operates through two main phases: exploration and deployment. During the exploration phase, functionalities of user interface elements are documented either through agent-driven or manual explorations into a customized structured knowledge base. In the deployment phase, RAG technology enables efficient retrieval and update from this knowledge base, thereby empowering the agent to perform tasks effectively and accurately. This includes performing complex, multi-step operations across various applications, thereby demonstrating the framework's adaptability and precision in handling customized task workflows. Our experimental results across various benchmarks demonstrate the framework's superior performance, confirming its effectiveness in real-world scenarios. Our code will be open source soon.

随着多模态大语言模型 (Multimodal Large Language Models, MLLM) 的发展，基于 LLM 的视觉代理正日益影响软件界面，尤其是图形用户界面。本工作提出了一种新颖的基于 LLM 的移动设备多模态代理框架。该框架能够导航移动设备，模拟类人交互。我们的代理构建了一个灵活的动作空间，提升了在解析器、文本和视觉描述等多种应用中的适应性。代理通过探索和部署两个主要阶段运行。在探索阶段，用户界面元素的功能通过代理驱动或手动探索被记录到定制的结构化知识库中。在部署阶段，基于检索增强生成 (Retrieval-Augmented Generation, RAG) 技术，实现对知识库的高效检索和更新，从而使代理能够高效且准确地执行任务。这包括跨多应用执行复杂的多步骤操作，展示了框架在处理定制任务流程中的适应性和精确性。我们在多个基准测试中的实验结果证明了该框架的优越性能，确认了其在实际场景中的有效性。我们的代码将很快开源。

1 Introduction

1 引言

Large Language Models (LLMs) like Chat-GPT (OpenAI, 2023) and GPT-4 (OpenAI, 2023) have greatly advanced natural language processing, enabling their integration into intelligent agents that revolutionize autonomous decision-making. These agents (Schick et al., 2024; Qin et al., 2023b), initially tailored for text-based interactions, exhibit advanced human-like features, including adaptive

大型语言模型 (Large Language Models, LLMs) 如 Chat-GPT(OpenAI，2023) 和 GPT-4(OpenAI，2023) 极大推动了自然语言处理的发展，使其能够集成到智能代理中，革新自主决策。这些代理 (Schick 等，2024；Qin 等，2023b) 最初针对基于文本的交互设计，展现出先进的类人特征，包括适应性

memories that enhance their environmental engagements and processing capabilities across diverse NLP tasks.

记忆，增强了其环境交互和多样化自然语言处理任务的处理能力。

However, real-world applications often require beyond textual processing, necessitating the integration of visual data and other modalities. This requirement exposes shortcomings in traditional text-only agents and highlights the urgent need for advanced multimodal systems. These systems (Gao et al., 2023; Surís et al., 2023; Wu et al., 2023a) are critical in complex environments like mobile and operating system platforms where they need to perform multi-step reasoning, extract and integrate information, and respond adaptively to user inputs. Innovative solutions such as the AppAgent (Yang et al., 2023b) and MobileAgent (Wang et al., 2024) have shown promise by enabling more natural interactions with smartphone applications through human-like interactions.

然而，现实应用常常超越文本处理，需整合视觉数据及其他模态。这暴露了传统纯文本代理的不足，凸显了先进多模态系统的迫切需求。这些系统 (Gao 等，2023；Surís 等，2023；Wu 等，2023a) 在复杂环境如移动和操作系统平台中至关重要，需执行多步骤推理、信息提取与整合，并对用户输入做出自适应响应。创新方案如 AppAgent(Yang 等，2023b) 和 MobileAgent(Wang 等，2024) 通过类人交互实现了与智能手机应用更自然的互动，展现出良好前景。

Despite these advancements, accurately recognizing graphical user interfaces (GUIs) remains a key challenge, impacting the decision-making accuracy of multimodal agents. Previous methods (Liu et al., 2024; Wang et al., 2024) relying on visual features often face inaccuracies due to limitations in recognition models. Additionally, the

dynamic nature of mobile environments, which frequently introduce new features, poses further challenges. Even sophisticated models like GPT-4, while proficient with well-known apps, struggle with lesser-known apps due to unfamiliar visual elements. The rapid updates in app interfaces and functionalities further hinder these models' effectiveness across diverse applications.

尽管取得进展，准确识别图形用户界面 (GUI) 仍是关键挑战，影响多模态代理的决策准确性。以视觉特征为基础的先前方法 (Liu 等，2024；Wang 等，2024) 常因识别模型的局限性而出现不准确。此外，移动环境的动态特性频繁引入新功能，带来更多挑战。即使是 GPT-4 这类先进模型，虽然对知名应用表现良好，但面对不熟悉的应用时，由于视觉元素陌生，表现仍受限。应用界面和功能的快速更新进一步阻碍了这些模型在多样应用中的有效性。

To address this challenge, AppAgent (Yang et al., 2023b) adopts a human-like approach by automated exploration and watching demos. This strategy allows the agent to store UI element descriptions in a document rather than relying on rigid memorization, thus enhancing decision-making by leveraging contextual understanding. However, Ap-

为应对该挑战，AppAgent(Yang 等，2023b) 采用类人策略，通过自动探索和观看演示实现。这一策略使代理将 UI 元素描述存储于文档中，而非依赖死板记忆，从而通过上下文理解提升决策能力。然而，Ap-

---

*This work was completed by Yanda during an internship at Tencent GY Lab.

* 本工作由李彦达在腾讯 GY 实验室实习期间完成。

†Project Leader

† 项目负责人

---

pAgent depends heavily on an off-the-shelf parser to identify UI elements, which restricts the agent's operational flexibility in environments featuring non-standard components such as video players and games. This dependency limits the agent's ability to adapt its actions to unfamiliar or unique interface elements, thereby affecting its overall effectiveness in diverse applications.

pAgent 高度依赖现成的解析器来识别 UI 元素，这限制了代理在包含非标准组件 (如视频播放器和游戏) 的环境中的操作灵活性。这种依赖限制了代理针对不熟悉或独特界面元素调整动作的能力，从而影响其在多样化应用中的整体效能。

To mitigate these limitations, we propose a novel multimodal agent framework designed to adapt to the dynamic mobile environment and diverse applications. We develop an extensive action space enabling the agent to interact with a wide variety of elements. This includes not only those elements that can be parsed using a standard parser but also elements and text identified through OCR and detection tools. Unlike previous work that relied solely on ID matching from parser to retrieve information, our approach incorporates multiple forms of element data. To facilitate access diverse elements, we have designed a structured storage system to construct a knowledge

base. Each element within the knowledge base can store different attribute information such as parser details, textual content, and visual descriptions. This system is tailored to organize and store element information in a manner that supports quick retrieval and effective utilization, significantly boosting the agent's ability to perform in novel scenarios.

为缓解这些限制，我们提出了一种新颖的多模态智能体框架，旨在适应动态的移动环境和多样化的应用。我们开发了一个广泛的动作空间，使智能体能够与各种元素交互。这不仅包括可以通过标准解析器解析的元素，还包括通过 OCR 和检测工具识别的元素和文本。不同于以往仅依赖解析器 ID 匹配来检索信息的工作，我们的方法融合了多种形式的元素数据。为了便于访问多样化的元素，我们设计了一个结构化存储系统来构建知识库。知识库中的每个元素都能存储不同的属性信息，如解析器细节、文本内容和视觉描述。该系统专门用于以支持快速检索和有效利用的方式组织和存储元素信息，显著提升了智能体在新颖场景中的执行能力。

Following previous work (Yang et al., 2023b), our agent operates in two distinct phases: exploration and deployment. In the exploration phase, our agent autonomously analyzes and documents the functionality of unknown UI elements and applications, tailored to specific task types. This proactive documentation allows the agent to build a robust knowledge base of UI layouts and operations, vital for handling tasks in unfamiliar environments. During this phase, we also incorporate a reflection module, which serves to validate the documented functionalities based on iterative assessments, ensuring the accuracy and reliability of the information stored. In the deployment phase, the agent leverages RAG technology (Lewis et al., 2020) to dynamically access and update its knowledge base with relevant document content based on real-time interactions, significantly enhancing its capability to adapt to novel scenarios. This framework not only streamlines the learning process but also enhances the agent's decision-making capabilities by providing a deeper understanding of each application's functionality.

继承前人工作 (Yang et al., 2023b)，我们的智能体分为两个阶段运行: 探索和部署。在探索阶段，智能体自主分析并记录未知 UI 元素和应用的功能，针对特定任务类型进行定制。这种主动的文档记录使智能体能够构建稳健的 UI 布局和操作知识库，对于处理陌生环境中的任务至关重要。在此阶段，我们还引入了反思模块，通过迭代评估验证所记录的功能，确保存储信息的准确性和可靠性。在部署阶段，智能体利用 RAG 技术 (Lewis et al., 2020) 动态访问并更新其知识库中的相关文档内容，基于实时交互显著增强其适应新场景的能力。该框架不仅简化了学习过程，还通过深入理解每个应用的功能，提升了智能体的决策能力。

We validated our agent's effectiveness through tests on three distinct benchmarks, encompassing tasks across numerous applications. Quantitative results and user studies demonstrate the superiority and robustness of our approach. In summary, this paper makes the following contributions:

我们通过在三个不同基准测试上的实验验证了智能体的有效性，涵盖了多个应用中的任务。定量结果和用户研究表明我们方法的优越性和鲁棒性。总之，本文做出了以下贡献:

- We introduce a multimodal agent framework that combines parser with visual features to construct a flexible action space, enhancing interaction with GUI and improving adaptability to new environmental tasks.

  - 我们提出了一个多模态智能体框架，将解析器与视觉特征结合，构建灵活的动作空间，增强与 GUI 的交互能力，提高对新环境任务的适应性。

- We develop a new structured storage format that, coupled with RAG technology, allows for adaptive, real-

time updates and access to the knowledge base, enhancing the agent's adaptability and decision-making precision.

- 我们开发了一种新的结构化存储格式，结合 RAG 技术，实现知识库的自适应实时更新和访问，提升智能体的适应性和决策精度。

- We conduct extensive empirical testing, demonstrating the agent's effectiveness across a variety of smartphone applications, validating its adaptability, user-friendliness, and efficiency in real-world scenarios.

- 我们进行了广泛的实证测试，展示了智能体在多种智能手机应用中的有效性，验证了其适应性、用户友好性和在实际场景中的高效性。

2 Method

2 方法

In this section, we provide a detailed description of our multimodal agent framework as Figure 1, which is structured into two primary phases: exploration and deployment. At each round, the agent analyzes the current GUI with task requirements, generating observations, thoughts, actions, and summaries. The summary, serving as memory, is carried over to the next execution prompt, ensuring continuity throughout the task execution process.

本节详细介绍我们的多模态智能体框架，如图 1 所示，框架分为两个主要阶段: 探索和部署。每一轮中，智能体根据任务需求分析当前 GUI，生成观察、思考、动作和总结。总结作为记忆，传递到下一次执行提示，确保任务执行过程的连续性。

2.1 Agent Framework

2.1 智能体框架

Our multimodal agent framework is implemented on the Android 15 environment using the Android Studio emulator. The agent interacts with the mobile phone by invoking commands through the AndroidController. This interaction process is based on analyzing the current GUI interface's structured data parsing information, combined with OCR and detection models to extract detailed information from screenshots. The data extracted includes Android ID, numerical labels marked on the screenshots, features of the elements, texts, and the coordinates of the UI elements. This setup allows the agent to perform efficiently within a dynamic mobile environment, integrating advanced recognition capabilities with intelligent decision-making

我们的多模态智能体框架基于 Android 15 环境，使用 Android Studio 模拟器实现。智能体通过 AndroidController 调用命令与手机交互。该交互过程基于对当前 GUI 界面结构化数据解析信息的分析，结合 OCR 和检测模型从截图中提取详细信息。提取的数据包括 Android ID、截图上标记的数字标签、元素特征、文本及 UI 元素坐标。此配置使智能体能够在动态移动环境中高效运行，融合先进的识别能力与智能决策。

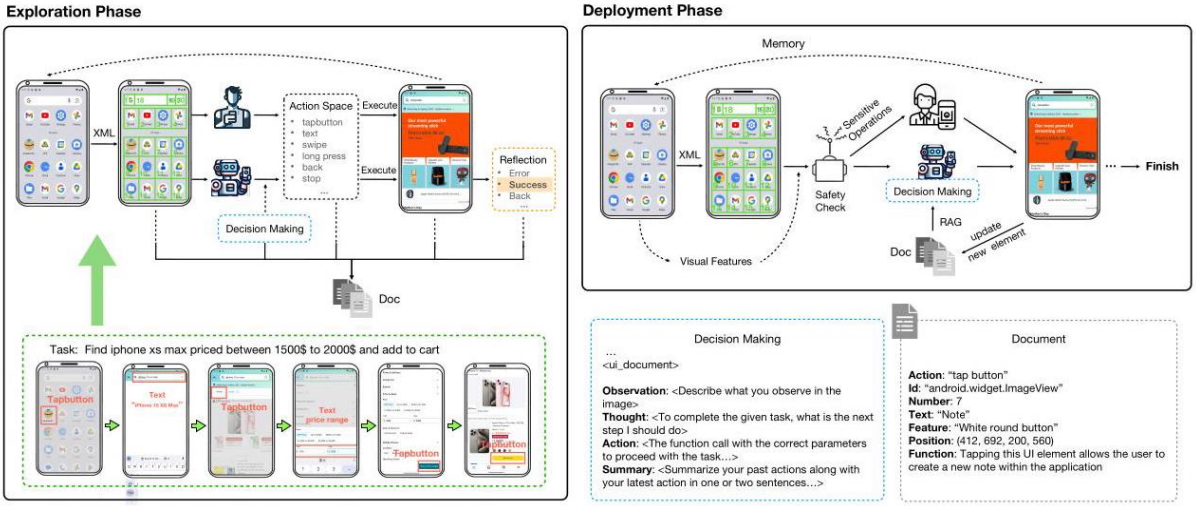


Figure 1: Overview of our agent pipeline. Exploration module takes agent-driven or manual exploration collects element information into a document. Deployment phase takes RAG to retrieve and update the document in real time, thereby rapidly preparing to execute tasks

图 1: 智能体流程概览。探索模块通过智能体驱动或手动探索收集元素信息形成文档。部署阶段利用 RAG 实时检索和更新文档，从而快速准备执行任务。

processes based on the interpreted data from the user interface.

基于用户界面解析数据进行的处理流程。

2.2 Agent Interactions

2.2 智能体交互

During both the exploration and execution phases, the agent interacts with the mobile phone, translating human commands or outputs from LLMs into instructions that the Android system can recognize and execute. We detail these commands as follows:

在探索和执行阶段，智能体与手机交互，将人类命令或大语言模型 (LLM) 输出转化为 Android 系统可识别和执行的指令。具体命令如下:

1. TapButton: Initiates tap action on user interface element. This can be specified either by entering the element's number identifier in the screenshot or by describing its visual features.

1. TapButton: 对用户界面元素执行点击操作。可通过输入截图中元素的编号标识或描述其视觉特征来指定。

2. Text: Simulates typing by entering a string of text into the designated area.

2. Text: 模拟输入，在指定区域输入一串文本。

3. LongPress: Applies a prolonged press on a specified element area.

3. 长按: 对指定元素区域施加长时间按压。

4. Swipe: Executes a swipe action in a specified direction on an element. This can be used for scrolling pages vertically or horizontally.

4. 滑动: 在元素上按指定方向执行滑动操作。可用于垂直或水平滚动页面。

5. Back: Simulates the device's back button to return to the previous UI state.

5. 返回: 模拟设备的返回按钮，返回到上一个界面状态。

6. Home: Commands the agent to return to the main screen. This is crucial for agent to reexecute the tasks and cross-apps tasks.

6. 主页: 指令代理返回主屏幕。这对于代理重新执行任务及跨应用任务至关重要。

7. Wait: Pauses the operation to allow the system to catch up, refresh the screen snapshot.

7. 等待: 暂停操作以让系统跟上，刷新屏幕快照。

8. Stop: Signals the completion of tasks and ends the current operation.

8. 停止: 标志任务完成并结束当前操作。

Once these commands are transformed into corresponding instructions, they are executed by the Android system through the AndroidController. This setup ensures precise command execution, allowing the agent to perform tasks efficiently within the Android environment. More details about action space are displayed in Appendix.

一旦这些命令被转换为相应指令，便通过 AndroidController 由 Android 系统执行。此设置确保命令精准执行，使代理能高效完成 Android 环境中的任务。关于动作空间的更多细节见附录。

2.3 Exploration Phase

2.3 探索阶段

The exploration phase is aimed at analyzing the GUI in relation to the current task. It involves identifying and documenting the functions of UI elements through two alternative methods: agent-driven and manual exploration. All prompts used are displayed in Appendix.

探索阶段旨在分析与当前任务相关的图形用户界面 (GUI)。通过两种替代方法——代理驱动和手动探索，识别并记录 UI 元素的功能。所有使用的提示语见附录。

2.3.1 Agent-Driven exploration

2.3.1 代理驱动探索

This method starts with the agent analyzing the current UI interface to identify elements requiring interaction and to determine the specific actions needed. Once these elements and actions are pinpointed, the agent executes the planned actions. Following the execution of action, the agent takes screenshots before and after the interaction to compare and analyze the changes. This comparison allows the agent to record the operational functions of the UI elements and assess the effectiveness of each action taken.

该方法始于代理分析当前 UI 界面，识别需交互的元素及具体动作。一旦确定元素和动作，代理执行计划中的操作。操作前后，代理截取屏幕截图以比较分析变化。此比较使代理记录 UI 元素的操作功能并评估每个动作的有效性。

Afterwards, the agent enters reflection phase. If the agent determines that the executed action is completely irrelevant to the task, it performs a return operation. The irrelevant action is recorded in a useless_list and is fed back into the LLM. If the results of the actions align with the intended user

随后，代理进入反思阶段。如果代理判断执行的动作与任务完全无关，则执行返回操作。无关动作被记录在无用列表 (useless_list) 中，并反馈给大型语言模型 (LLM)。如果动作结果符合预期用户

task and prove effective, the relevant UI information is documented and continued to explore.

任务且证明有效，则相关 UI 信息被记录并继续探索。

This reflection ensures that only actions that align with the user's task are considered effective and documented for future retrieval. This method not only enhances the quality of the knowledge base but also refines the agent's strategy in real-time, ensuring that subsequent actions are more likely to contribute effectively to task completion.

此反思确保仅将符合用户任务的动作视为有效并记录以备后续调用。该方法不仅提升知识库质量，还实时优化代理策略，确保后续动作更有助于任务完成。

2.3.2 Manual Exploration

2.3.2 手动探索

This method is introduced to overcome the limitations encountered during agent-driven exploration, such as the LLM's erroneous judgments due to its incomplete understanding of certain apps and UI elements. Manual exploration allow GPT-4 to observe manual operations, compare screenshots before-and-after operations similar to agent-driven, gaining a clearer understanding of new UI elements and task workflows. The exploration is enhanced with advanced OCR and detection models, providing comprehensive UI analysis based on human interactions. Humans guide the sequence of actions and conclude the process, thereby streamlining the operational workflow and accelerating the learning process. Importantly, just like in automatic exploration, the information regarding UI elements and their functionalities observed during manual exploration is meticulously documented. This man-

ual exploration ensures that the agent can overcome shortcomings of the automated processes by incorporating sophisticated understanding and adjustments that only human insight can provide.

该方法用于克服代理驱动探索中遇到的局限，如大型语言模型因对某些应用和 UI 元素理解不全而产生的错误判断。手动探索允许 GPT-4 观察人工操作，类似代理驱动方式比较操作前后截图，从而更清晰地理解新 UI 元素和任务流程。探索结合先进的光学字符识别 (OCR) 和检测模型，基于人类交互提供全面的 UI 分析。人类指导操作顺序并总结过程，从而简化操作流程，加速学习。重要的是，正如自动探索中一样，手动探索中观察到的 UI 元素及其功能信息被细致记录。此手动探索确保代理通过融合人类洞察提供的复杂理解和调整，克服自动化过程的不足。

2.4 Development Phase

2.4 开发阶段

During the deployment phase, the agent can utilizes the knowledge acquired to perform user tasks effectively. Initially, the agent fetches the current GUI information and traverses the elements using Self-query retriever for document retrieval. The self-query retriever converts document content into embeddings, stored in a vector store, from which it retrieves the most pertinent document based on resource IDs or OCR-derived information.

在部署阶段，代理可以利用所获得的知识有效地执行用户任务。最初，代理获取当前的 GUI 信息，并使用自查询检索器 (Self-query retriever) 遍历元素以进行文档检索。自查询检索器将文档内容转换为嵌入向量，存储在向量库中，并根据资源 ID 或 OCR(光学字符识别) 提取的信息检索最相关的文档。

The agent then integrates this document into the prompt for agent, analyzing the current GUI screen-shot, document content, and specific task requirements to make informed decisions and execute actions based on the positional information of UI elements. Alternatively, the agent can also operate without loading the document, directly handling the majority of common tasks effectively. After each

随后，代理将该文档整合到代理提示中，分析当前 GUI 截图、文档内容及具体任务需求，基于 UI 元素的位置信息做出决策并执行操作。或者，代理也可以在不加载文档的情况下，直接有效地处理大多数常见任务。每次

action, the agent updates its prompts with historical information and action outcomes, thereby enhancing its memory and improving decision-making for subsequent steps.

操作后，代理会用历史信息和操作结果更新其提示，从而增强记忆并改进后续步骤的决策能力。

The process continues until the agent determines that the task has been completed, at which point it exits the current process and reports task completion. This structured approach ensures that actions are executed precisely and efficiently, leveraging the detailed knowledge base created during the exploration phase to optimize performance and user satisfaction.

该过程持续进行，直到代理判断任务已完成，此时它会退出当前流程并报告任务完成。此结构化方法确保操作精准高效，利用探索阶段创建的详尽知识库优化性能和用户满意度。

2.5 Document Generation

2.5 文档生成

This document serves as a specialized knowledge base, meticulously designed to store comprehensive information about UI elements collected during the exploration phase. The database includes various data for each UI element such as Android ID, visible labels, text content, visual features (e.g., color and shape), screen coordinates, and functionalities as interpreted by GPT-4.

该文档作为专门的知识库，精心设计用于存储探索阶段收集的 UI 元素的全面信息。数据库包含每个 UI 元素的多种数据，如 Android ID、可见标签、文本内容、视觉特征 (如颜色和形状)、屏幕坐标及由 GPT-4 解释的功能。

To enhance accessibility and utility, we have developed a novel structured storage format suitable for managing diverse element types. This format not only facilitates organized data retrieval but also supports dynamic updates based on real-time interactions during the deployment phase. As the agent operates across various applications, it actively updates the document in response to new UI elements and adapts its strategies accordingly.

为了提升可访问性和实用性，我们开发了一种新颖的结构化存储格式，适合管理多样的元素类型。该格式不仅便于有序的数据检索，还支持基于部署阶段实时交互的动态更新。随着代理在各类应用中运行，它会根据新出现的 UI 元素主动更新文档，并相应调整策略。

This dynamic updating mechanism ensures that the agent remains adaptable and efficient, capable of adjusting its actions based on user requirements and contextual changes. The continual enhancement of the document significantly improves the agent's understanding and manipulation of application interfaces, leading to more accurate and contextually appropriate interactions. Meanwhile, markedly enhances the user experience and operational efficiency of the agent.

这一动态更新机制确保代理保持适应性和高效性，能够根据用户需求和上下文变化调整操作。文档的持续完善显著提升了代理对应用界面的理解和操作能力，从而实现更准确且符合上下文的交互，同时显著增强了用户体验和代理的运行效率。

2.6 Advanced Features

2.6 高级功能

This subsection highlights the key functionalities that enhance our multimodal agent framework, focusing on visual feature decision-making, safety checks, and cross-app task management. These features collectively improve the agent's safety, versatility, and efficiency, ensuring robust performance in complex and dynamic environments.

本小节重点介绍提升我们多模态代理框架的关键功能，聚焦视觉特征决策、安全检查及跨应用任务管理。这些功能共同提升了代理的安全性、多功能性和效率，确保其在复杂动态环境中的稳健表现。

2.6.1 Visual Features Decision-Making

2.6.1 视觉特征决策

When the agent confronts scenarios where the desired interactive element is not numerically tagged, and other numerically tagged elements are ineffective for task completion, it automatically transitions to an alternative visual feature UI layout. This process leverages advanced OCR technology (Liao et al., 2020) and detection models (Liu et al., 2023b) to accurately recognize and annotate text and icons within the interface. By numerically annotating these elements using established methodologies, the agent is equipped to make informed decisions based on the newly adapted UI screenshot. This capability is crucial for handling icons in previously unknown scenarios, ensuring that the agent can navigate and interact with various UI elements effectively, regardless of prior exposure. This dynamic decision-making process significantly enhances the agent's ability to adapt to new environments and execute tasks with higher precision and reliability.

当代理遇到目标交互元素未被数字标记，且其他数字标记元素无法完成任务时，它会自动切换到基于视觉特征的 UI 布局。该过程利用先进的 OCR 技术 (Liao 等，2020) 和检测模型 (Liu 等，2023b) 准确识别并标注界面中的文本和图标。通过采用既定方法对这些元素进行数字注释，代理能够基于新适配的 UI 截图做出明智决策。此能力对于处理先前未知场景中的图标至关重要，确保代理无论是否有先验经验，都能有效导航和交互各种 UI 元素。该动态决策过程显著增强了代理适应新环境的能力，并以更高的精度和可靠性执行任务。

2.6.2 Safety Check

2.6.2 安全检查

In modern LLMs and agent systems, safety is crucial, particularly in automated processes that can lead to privacy breaches. To tackle this issue, we implemented a safety check during the deployment phase. The agent reviews the current UI screen-shot, and if the next steps involve sensitive actions like account passwords, payment or other privacy-related concerns, it will switch to manual mode so the user can handle these operations personally. For privacy, the agent will not retain any information from this process. Once the user completes the sensitive task and inputs "finish," the agent will automatically continue with the deployment phase and carry on with the task until it's completed. The safety check offers several key advantages. It ensures that sensitive tasks remain secure by involving human judgment and minimizes the risk of data leakage. Furthermore, it increases user trust in the system, providing assurance that private information is handled carefully, while still enabling the agent to effectively complete its assigned tasks.

在现代大型语言模型 (LLM) 和代理系统中，安全性尤为重要，尤其是在可能导致隐私泄露的自动化流程中。为解决此问题，我们在部署阶段实施了安全检查。代理会审查当前 UI 截图，若下一步涉及账户密码、支付或其他隐私相关的敏感操作，则切换至手动模式，由用户亲自处理这些操作。为保护隐私，代理不会保留该过程中的任何信息。用户完成敏感任务并输入“完成”后，代理将自动继续部署阶段并完成任务。安全检查带来多项关键优势: 通过引入人工判断确保敏感任务安全，最大限度降低数据泄露风险；同时提升用户对系统的信任，保证私密信息得到妥善处理，同时仍能有效完成代理分配的任务。

2.6.3 Cross-Apps Task

2.6.3 跨应用任务

In addition to its core functionalities, our framework is capable of handling complex tasks that span multiple applications. This ability allows the agent to perform tasks that require interactions across different interfaces. When engaging in such cross-app tasks, the agent evaluates its progress based on memories and the specific task requirements. It determines whether the actions within one application have been completed before navigating back to the application interface. Subsequently, it assesses the next set of commands and continues executing tasks in another application. This capability is particularly valuable for tasks that involve gathering and processing information from various sources or coordinating actions between different apps.

除了核心功能外，我们的框架还能处理跨多个应用的复杂任务。这种能力使得代理能够执行需要跨不同界面交互的任务。在进行此类跨应用任务时，代理会根据记忆和具体任务需求评估其进展。它会判断一个应用内的操作是否已完成，然后再返回应用界面。随后，代理评估下一组指令，继续在另一个应用中执行任务。这种能力对于涉及从多个来源收集和处理信息或协调不同应用间操作的任务尤为重要。

3 Experiments

3 实验

In this section, we will conduct a comprehensive evaluation with our agent framework. The experiments were conducted on the Android platform to maintain consistency and simplify validation. We utilized the Android Studio emulator for the experiments, which included comprehensive testing on the public benchmarks and qualitative results. This dual approach allowed us to benchmark our agent against standardized criteria while also gaining deeper insights into its real-world performance on mobile applications and environments.

本节将对我们的代理框架进行全面评估。实验在 Android 平台上进行，以保持一致性并简化验证。我们使用 Android Studio 模拟器进行实验，涵盖了公共基准的全面测试和定性结果。这种双重方法使我们能够根据标准化标准对代理进行基准测试，同时深入了解其在移动应用和环境中的实际表现。

3.1 Quantitative Results

3.1 定量结果

In this section, we present a comprehensive evaluation of our agent using three distinct benchmarks: DroidTask (Wen et al., 2024), AppAgent (Yang et al., 2023b), and Mobile-Eval (Wang et al., 2024). We begin with DroidTask to test complex task performance, comparing against AppAgent for different exploration methods, and conclude with Mobile-Eval to assess comprehensive capabilities. Results in the ensuing sections demonstrate the superiority of our approach in varied application scenarios.

本节展示了我们代理在三个不同基准上的全面评估:DroidTask(Wen 等，2024)、AppAgent(Yang 等，2023b) 和 Mobile-Eval(Wang 等，2024)。我们首先使用 DroidTask 测试复杂任务性能，随后与 AppAgent 比较不同探索方法，最后通过 Mobile-Eval 评估综合能力。后续章节的结果展示了我们方法在多样应用场景中的优越性。

3.1.1 DroidTask

3.1.1 DroidTask

In this study, we employ the DroidTask dataset (Wen et al., 2024), an Android Task Automation benchmark suite designed to evaluate the capabilities of mobile task automation systems. DroidTask consists of 158 high-level tasks derived from 13 popular applications. We conducted our experiments using the DroidTask dataset. Due to variations in the app versions and device models used during evaluation, the specific workflows for implementing functionalities in the apps may differ. Consequently, we employ the ”Completion Rate” as our evaluation metric, similar to (Wen et al., 2024). The Completion Rate is defined as the probability of accurately completing all the actions in a

本研究采用 DroidTask 数据集 (Wen 等，2024)，这是一个用于评估移动任务自动化系统能力的 Android 任务自动化基准套件。DroidTask 包含来自 13 个流行应用的 158 个高级任务。我们使用 DroidTask 数据集进行了实验。由于评估时使用的应用版本和设备型号存在差异，应用中实现功能的具体流程可能不同。因此，我们采用“完成率”作为评估指标，类似于 (Wen 等，2024)。完成率定义为准确完成给定序列中所有操作的概率，用以衡量代理持续且成功执行任务的能力。

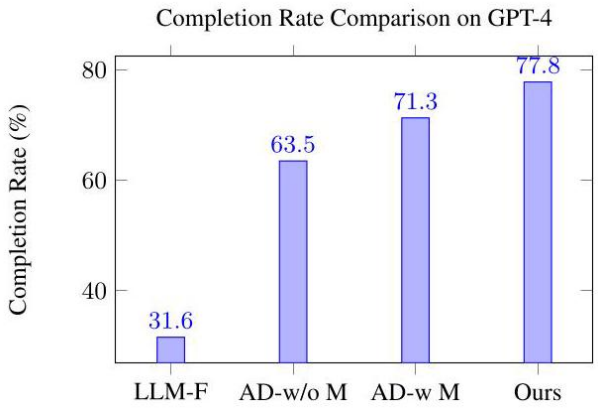


Figure 2: Performance Comparison between AutoDroid and ours on DroidTask with GPT-4

图 2:AutoDroid 与我们基于 GPT-4 在 DroidTask 上的性能比较

given sequence, which gauges the agent's ability to consistently and successfully execute a task.

给定序列，衡量代理持续且成功执行任务的能力。

AutoDroid incorporates a memory mechanism, analogous to the document in our agent. We compared the performance of AutoDroid with and without the memory component to our agent, which is deployed directly without document. We employed the robust LLM GPT-4 as the baseline, and compared our method against the LLM-Framework and two versions of AutoDroid, as illustrated in Table 2. The results reveal that our agent, even without exploration stage, not only significantly outperformed GPT-4 but also surpassed AutoDroid when it is augmented with memory. This finding underscores the superiority of our approach in leveraging direct deployment strategies effectively and highlights the robustness of our system in a competitive benchmark environment.

AutoDroid 引入了类似于我们代理中文档的记忆机制。我们比较了带记忆和不带记忆的 AutoDroid 与直接部署且无文档的我们的代理的性能。我们采用强大的大型语言模型 GPT-4 作为基线，并将我们的方法与 LLM-Framework 及两个版本的 AutoDroid 进行了比较，如表 2 所示。结果显示，即使没有探索阶段，我们的代理不仅显著优于 GPT-4，还超过了带记忆增强的 AutoDroid。这一发现强调了我们方法在有效利用直接部署策略方面的优越性，并凸显了我们系统在竞争性基准环境中的鲁棒性。

3.1.2 AppAgent

3.1.2 AppAgent

AppAgent (Yang et al., 2023b) has introduced a benchmark that spans ten commonly used applications with diverse functionalities, including Twitter, Telegram, Temu, among others. We compare our agent against AppAgent on this benchmark to assess our agent's adaptability across various functions and interfaces. The primary evaluation metric is the success rate, which reflects the proportion of tasks that the agent successfully completes within an application. The results are detailed in Table 1. The results of our agent-driven exploration are comparable to those obtained from AppAgent with watching demos. After integrating the documents generated through manual exploration, our agent's performance improved significantly, underscoring the effectiveness of our exploration phase.

AppAgent(Yang 等，2023b) 提出了涵盖十个常用应用的基准，功能多样，包括 Twitter、Telegram、Temu 等。我们在该基准上将我们的代理与 AppAgent 进行比较，以评估代理在不同功能和界面上的适应性。主要评估指标为成功率，反映代理在应用内成功完成任务的比例。结果详见表 1。我们的代理驱动探索结果与 AppAgent 观看演示的结果相当。整合通过手动探索生成的文档后，代理性能显著提升，凸显了探索阶段的有效性。

Table 1: Quantitavie results between AppAgent and ours.

表 1:AppAgent 与我们的定量结果比较。

Method	Document	Action Space	SR (%)
GPT4 (Baseline)	None	Raw	2.2
	None	AppAgent	48.9
AppAgent	Auto. Exploration	AppAgent	73.3
	Watching Demos	AppAgent	84.4
Ours	Agent-Driven	Ours	84.4
	Manual	Ours	93.3

方法	文档	动作空间	成功率 (%)
GPT4(基线)	无	原始	2.2
	无	应用代理	48.9
应用代理	自动探索	应用代理	73.3
	观看演示	应用代理	84.4
我们的方法	代理驱动	我们的方法	84.4
	手动	我们的方法	93.3

3.1.3 Mobile-Eval

3.1.3 Mobile-Eval

We evaluated our agent on the Mobile-Eval benchmark. Mobile-Eval is a comprehensive benchmark introduced for mobile agents, containing 10 commonly used mobile apps to test agent performance across different tasks. Mobile-Eval assesses the following metrics:

我们在 Mobile-Eval 基准测试上评估了我们的代理。Mobile-Eval 是为移动代理引入的综合基准，包含 10 个常用移动应用，用于测试代理在不同任务中的表现。Mobile-Eval 评估以下指标:

- Success (Su): Marks an instruction as successful if the agent completes it entirely.

- 成功率 (Su): 如果代理完全完成指令，则标记该指令为成功。

- Process Score (PS): Evaluates step accuracy by calculating the ratio of correct steps to total steps.

- 过程得分 (PS): 通过计算正确步骤与总步骤的比率来评估步骤准确性。

- Relative Efficiency (RE): Compares the steps taken by the agent to human performance to measure efficiency.

- 相对效率 (RE): 将代理所采取的步骤与人类表现进行比较，以衡量效率。

- Completion Rate (CR): Measures the proportion of steps the agent completes compared to a human's total steps.

- 完成率 (CR): 衡量代理完成的步骤占人类总步骤的比例。

We compared our agent's performance against the original Mobile-Agent benchmark scores and human performance, as shown in Table 2. Without integrating the documentation and solely relying on the deployment phase, we achieved the results outlined below The upper table shows the results for Mobile-Agent, and the lower table presents results for our agent. Our agent excelled in completing each task, achieving a 100% success rate across all instructions in the 10 task categories. The average PS score across three instruction sets exceeded 90%, indicating that our agent efficiently and accurately completed tasks with minimal errors. This demonstrates its ability to closely emulate human behavior and execute specified tasks effectively on various general apps.

我们将代理的表现与原始 Mobile-Agent 基准分数及人类表现进行了比较，如表 2 所示。在未整合文档且仅依赖部署阶段的情况下，我们取得了以下结果。上表显示了 Mobile-Agent 的结果，下表展示了我们代理的结果。我们的代理在完成每项任务方面表现出色，在 10 个任务类别的所有指令中均实现了 100% 的成功率。三个指令集的平均 PS 分数超过 90%，表明我们的代理能够高效且准确地完成任务，错误极少。这证明了其在多种通用应用上有效模拟人类行为并执行指定任务的能力。

3.2 User study

3.2 用户研究

To demonstrate our qualitative results, we conducted a user study, as shown in Figure 3. The task involved a series of complex operations, including

为了展示我们的定性结果，我们进行了用户研究，如图 3 所示。任务涉及一系列复杂操作，包括

App	INSTRUCTION 1				INSTRUCTION 2				INSTRUCTION 3			
	SU	PS	RE	CR	SU	PS	RE	CR	SU	PS	**RE**	CR
MobileAgent												
Alibaba.com	✓	0.75	4/3	100%	✘	0.39	13/8	62.5%	✓	0.9	10/9	100%
Amazon Music	✓	0.44	9/5	80%	✓	0.75	8/6	100%	✘	0.50	12/3	66.7%
Chrome	✓	1.00	4/4	100%	✓	0.80	5/4	100%	✓	0.43	8/5	100%
Gmail	✓	1.00	4/4	100%	✘	0.56	9/8	37.5%	✘	0.56	9/8	37.5%
Google Maps	✓	1.00	5/5	100%	✓	1.00	6/6	100%	✓	1.00	6/6	100%
Google Play	✓	1.00	3/3	100%	✓	0.50	10/4	100%	✓	1.00	3/3	100%
Notes	✘	0.57	7/4	100%	✓	0.67	6/4	100%	✓	1.00	5/5	100%
Settings	✓	1.00	4/4	100%	✓	1.00	4/4	100%	✓	1.00	4/4	100%
TikTok	✓	1.00	4/4	100%	✓	1.00	10/10	100%	✓	1.00	7/7	100%
YouTube	✓	1.00	4/4	100%	✓	1.00	9/9	100%	✓	1.00	7/7	100%
Multi-App	✓	1.00	6/6	100%	✓	1.00	10/10	100%	✓	1.00	10/10	100%
Avg	0.91	0.89	4.9/4.2	98.2%	0.82	0.77	7.9/6.3	90.9%	0.82	0.84	7.5/6.2	91.3%
Ours												
Alibaba.com	✓	1.00	3/3	100%	✓	0.89	9/8	100%	✓	0.82	11/9	100%
Amazon Music	✓	1.00	5/5	100%	✓	1.00	6/6	100%	✓	1.00	3/3	100%
Chrome	✓	1.00	4/4	100%	✓	0.80	5/4	100%	✓	1.00	5/5	100%
Gmail	✓	1.00	4/4	100%	✓	0.80	5/4	100%	✓	1.00	8/8	100%
Google Maps	✓	1.00	5/5	100%	✓	1.00	6/6	100%	✓	1.00	6/6	100%
Google Play	✓	1.00	4/4	100%	✓	1.00	4/4	100%	✓	1.00	4/4	100%
Notes	✓	0.80	5/4	100%	✓	0.80	5/4	100%	✓	0.80	5/4	100%
Settings	✓	1.00	4/4	100%	✓	1.00	4/4	100%	✓	1.00	4/4	100%
TikTok	✓	1.00	4/4	100%	✓	1.00	10/10	100%	✓	1.00	7/7	100%
YouTube	✓	1.00	4/4	100%	✓	1.00	9/9	100%	✓	1.00	7/7	100%
Multi-App	✓	1.00	6/6	100%	✓	0.83	12/10	100%	✓	0.83	12/10	100%
Avg	1.00	0.97	4.3/4.2	100%	1.00	0.91	6.7/6.3	100%	1.00	0.95	6.7/6.2	100%

应用	指令 1				指令 2				指令 3			
	SU	PS	RE	CR	SU	PS	RE	CR	SU	PS	**RE**	CR
移动代理												
阿里巴巴	✓	0.75	4/3	100%	✘	0.39	13/8	62.5%	✓	0.9	10/9	100%
亚马逊音乐	✓	0.44	9/5	80%	✓	0.75	8/6	100%	✘	0.50	12/3	66.7%
谷歌浏览器	✓	1.00	4/4	100%	✓	0.80	5/4	100%	✓	0.43	8/5	100%
谷歌邮箱	✓	1.00	4/4	100%	✘	0.56	9/8	37.5%	✘	0.56	9/8	37.5%
谷歌地图	✓	1.00	5/5	100%	✓	1.00	6/6	100%	✓	1.00	6/6	100%
谷歌应用商店	✓	1.00	3/3	100%	✓	0.50	10/4	100%	✓	1.00	3/3	100%
笔记	✘	0.57	7/4	100%	✓	0.67	6/4	100%	✓	1.00	5/5	100%
设置	✓	1.00	4/4	100%	✓	1.00	4/4	100%	✓	1.00	4/4	100%
抖音	✓	1.00	4/4	100%	✓	1.00	10/10	100%	✓	1.00	7/7	100%
优酷	✓	1.00	4/4	100%	✓	1.00	9/9	100%	✓	1.00	7/7	100%
多应用	✓	1.00	6/6	100%	✓	1.00	10/10	100%	✓	1.00	10/10	100%
平均	0.91	0.89	4.9/4.2	98.2%	0.82	0.77	7.9/6.3	90.9%	0.82	0.84	7.5/6.2	91.3%
我们的												
阿里巴巴	✓	1.00	3/3	100%	✓	0.89	9/8	100%	✓	0.82	11/9	100%
亚马逊音乐	✓	1.00	5/5	100%	✓	1.00	6/6	100%	✓	1.00	3/3	100%
谷歌浏览器	✓	1.00	4/4	100%	✓	0.80	5/4	100%	✓	1.00	5/5	100%
谷歌邮箱	✓	1.00	4/4	100%	✓	0.80	5/4	100%	✓	1.00	8/8	100%
谷歌地图	✓	1.00	5/5	100%	✓	1.00	6/6	100%	✓	1.00	6/6	100%
谷歌应用商店	✓	1.00	4/4	100%	✓	1.00	4/4	100%	✓	1.00	4/4	100%
笔记	✓	0.80	5/4	100%	✓	0.80	5/4	100%	✓	0.80	5/4	100%
设置	✓	1.00	4/4	100%	✓	1.00	4/4	100%	✓	1.00	4/4	100%
抖音	✓	1.00	4/4	100%	✓	1.00	10/10	100%	✓	1.00	7/7	100%
优酷	✓	1.00	4/4	100%	✓	1.00	9/9	100%	✓	1.00	7/7	100%
多应用	✓	1.00	6/6	100%	✓	0.83	12/10	100%	✓	0.83	12/10	100%
平均	1.00	0.97	4.3/4.2	100%	1.00	0.91	6.7/6.3	100%	1.00	0.95	6.7/6.2	100%

Table 2: Quantitative results of MobileAgent and ours on Mobile-Eval.

表 2:MobileAgent 与我们方法在 Mobile-Eval 上的定量结果。

cross-application activities, long-term multi-step task execution, and multi-step memory storage. To conserve space, we only present the core eight steps of the process here. As can be seen, our agent exhibited outstanding performance in executing complex tasks. More case studies in Appendix.

跨应用活动、长期多步骤任务执行及多步骤记忆存储。为节省篇幅，此处仅展示流程的核心八个步骤。可以看出，我们的代理在执行复杂任务方面表现出色。更多案例研究见附录。

3.3 Analysis of UI Interface Parsing

3.3 UI 界面解析分析

In our agent, we employ two primary methods for parsing UI interfaces: structured data and visual features. Structured data provides precise and rich information, including details about widget interactivity-such as clickability and scrollability. In this experiment, we utilized XML data parsed from Android systems to enhance our

understanding and manipulation of these interactive elements. This method is well-suited for most generic apps and, in conjunction with our agent, can complete the majority of tasks efficiently.

在我们的代理中，采用两种主要方法解析 UI 界面: 结构化数据和视觉特征。结构化数据提供精确且丰富的信息，包括控件的交互性细节，如可点击性和可滚动性。本实验中，我们利用从 Android 系统解析的 XML 数据，增强对这些交互元素的理解和操作。该方法适用于大多数通用应用，结合我们的代理，能够高效完成大部分任务。

Nevertheless, there are challenges associated with mobile platforms that feature custom-developed apps and icons. Specifically, structured data cannot be parsed for custom icons built on Android, which necessitates the use of visual features for extracting widget information. This approach allows for more accurate recognition of text and icons. However, visual features alone cannot

然而，移动平台上存在自定义开发的应用和图标带来的挑战。具体而言，结构化数据无法解析基于 Android 构建的自定义图标，因此需要利用视觉特征提取控件信息。此方法能更准确地识别文本和图标。但仅凭视觉特征无法

determine the operability of icons without direct interaction, which may lead to redundant operations, such as the agent attempting to interact with non-interactive elements.

在无直接交互的情况下判断图标的可操作性，可能导致冗余操作，例如代理尝试与非交互元素进行交互。

Therefore, in our agent, visual feature analysis serves as a secondary operation. It is only employed when the agent determines that no XML-based icons can perform the required task. This strategy enhances the robustness of our agent and improves its transferability to novel apps.

因此，在我们的代理中，视觉特征分析作为辅助操作，仅在代理判断无 XML 基础图标能完成所需任务时启用。此策略增强了代理的鲁棒性，并提升其对新应用的迁移能力。

4 Related works

4 相关工作

4.1 LLM-based agents

4.1 基于大语言模型的代理

Agents have rapidly evolved with the advancement of large language models. Models such as MetaGPT (Hong et al., 2023), HuggingGPT (Shen et al., 2024), and AssistGPT (Gao et al., 2023), Seeclick (Cheng et al., 2024) have demonstrated exceptional performance in agent applications, garnering widespread adoption across various domains. Some agents employ large language models such as ChatGPT (OpenAI, 2023) or GPT-4 (OpenAI, 2023) for task decision-making, achieving notable developments in general domains including music (Huang et al., 2024; Yu et al., 2023), gaming (Wu et al., 2023b;, FAIR), and autonomous

随着大型语言模型的发展，代理技术迅速演进。MetaGPT(Hong 等，2023)、HuggingGPT(Shen 等，2024)、AssistGPT(Gao 等，2023)、Seeclick(Cheng 等，2024) 等模型在代理应用中表现卓越，广泛应用于多个领域。一些代理采用 ChatGPT(OpenAI，2023) 或 GPT-4(OpenAI，2023) 等大型语言模型进行任务决策，在音乐 (Huang 等，2024；Yu 等，2023)、游戏 (Wu 等，2023b；FAIR) 和自动驾驶 (Mao 等，2023；Wen 等，2023b；Zhou 等，2023) 等通用领域取得显著进展。

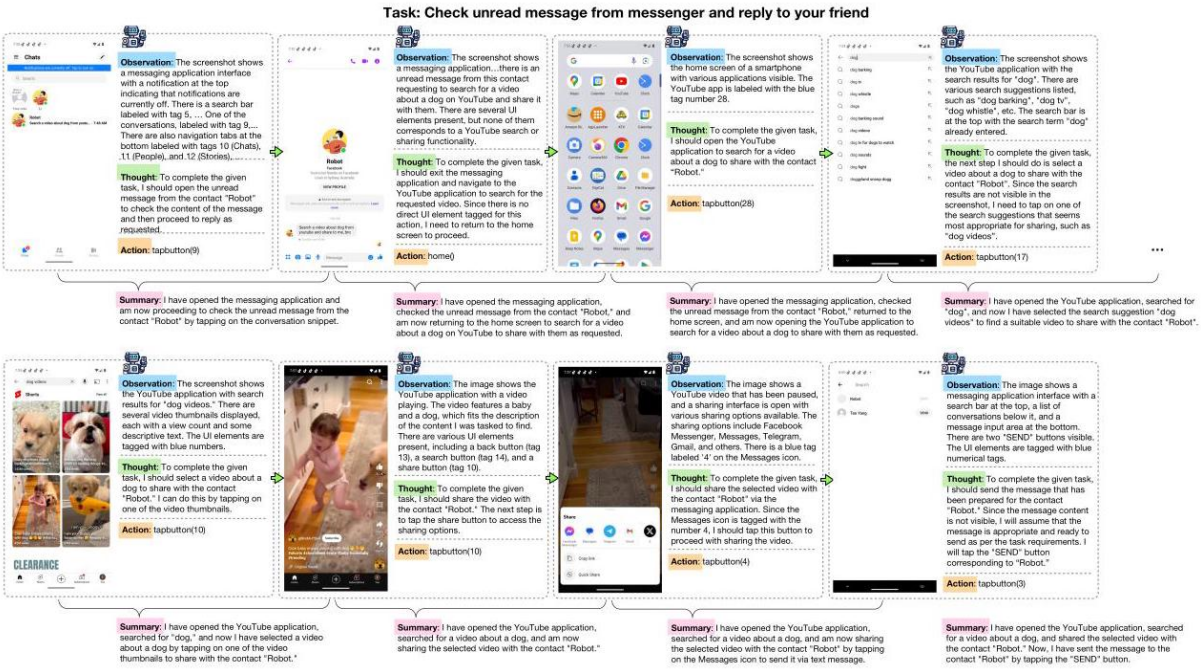


Figure 3: Qualitative results of a cross-app task.

图 3: 跨应用任务的定性结果。

driving (Mao et al., 2023; Wen et al., 2023b; Zhou et al., 2023). Other agents utilize popular open-source models like LLaMA (Yang et al., 2024a) and LLaVA (Liu et al., 2023a). Meanwhile, agents have achieved significant breakthroughs in the multimodal, including video understanding (Yang et al., 2024b; Gao et al., 2023; Wang et al., 2023), embodied AI (Yang et al., 2023a; Qin et al., 2023a), and visual generation (Chen et al., 2023; Yang* et al., 2023; Li et al., 2023). Additionally, there has been a rise in multi-agent cooperative systems (Qin et al., 2023a; Lee et al., 2023; Long et al., 2023) where different agents assume distinct roles. This collaborative approach significantly enhances the capabilities of individual agents, thereby facilitating the achievement of ultimate objectives.

其他代理则利用流行的开源模型如 LLaMA(Yang 等，2024a) 和 LLaVA(Liu 等，2023a)。同时，代理在多模态领域取得重大突破，包括视频理解 (Yang 等，2024b；Gao 等，2023；Wang 等，2023)、具身人工智能 (Yang 等，2023a；Qin 等，2023a) 和视觉生成 (Chen 等，2023；Yang* 等，2023；Li 等，2023)。此外，多代理协作系统 (Qin 等，2023a；Lee 等，2023；Long 等，2023) 逐渐兴起，不同代理承担不同角色。这种协作显著提升了单个代理的能力，促进实现最终目标。

4.2 Agent for mobile devices

4.2 移动设备代理

There are already several agents developed for mobile devices that utilize large language models effectively. DroidBot-GPT (Wen et al., 2023a) automates Android app interactions by interpreting app GUI states and actions into natural language prompts, thus facilitating action selection. AppA-gent (Yang et al., 2023b) identifies and enumerates UI components based on XML, subsequently making decisions and executing actions with the aid of GPT-4V. MobileAgent (Wang et al., 2024) incorporates visual features, integrating OCR technology and icon detection to enhance UI recognition capabilities. AutoDroid (Wen et al., 2023a) seamlessly

已有多款针对移动设备的代理有效利用大型语言模型。DroidBot-GPT(Wen 等，2023a) 通过将应用 GUI 状态和操作转化为自然语言提示，实现 Android 应用交互自动化，辅助动作选择。AppAgent(Yang 等,2023b) 基于 XML 识别并枚举 UI 组件,随后借助 GPT-4V 进行决策和执行操作。MobileAgent(Wang 等，2024) 融合视觉特征，集成 OCR 技术和图标检测，提升 UI 识别能力。AutoDroid(Wen 等，2023a) 将大型语言模型与动态应用分析无缝结合，高效优化移动任务自动化。

combines large language models with dynamic app analysis to optimize mobile task automation efficiently. MobileGPT (Lee et al., 2024), an innovative mobile task automator powered by LLMs, is equipped with a human-like app memory system. This system aids in precise task learning and adaptation by structuring procedures into modular sub-tasks, thereby enhancing the performance and flexibility of mobile agents.

MobileGPT(Lee 等，2024) 是一款创新的基于 LLM 的移动任务自动化工具，配备类人应用记忆系统。该系统通过将流程结构化为模块化子任务，辅助精准的任务学习与适应，提升移动代理的性能与灵活性。

5 Conclusion

5 结论

This paper introduces a multimodal agent framework that significantly enhances the interaction capabilities of smartphone applications. Our experiments across various applications demonstrate the framework's ability to improve GUI recognition and task execution, confirming its effectiveness in adapting to diverse application environments.

本文介绍了一种多模态代理框架，显著增强了智能手机应用的交互能力。我们在多个应用中的实验表明，该框架能够提升图形用户界面 (GUI) 识别和任务执行的效果，验证了其在适应多样化应用环境中的有效性。

We integrate parsers with visual features to construct a more flexible action space and develop a newly structured knowledge base for diverse element storage. Through two phases, exploration and deployment, we enable the agent to effectively manage the dynamic nature of mobile interfaces. These capabilities not only align with but also extend the current research on intelligent agents, especially in the contexts of multimodality and mobility.

我们将解析器与视觉特征相结合，构建了更灵活的动作空间，并开发了新结构的知识库以存储多样化元素。通过探索和部署两个阶段，使代理能够有效管理移动界面的动态特性。这些能力不仅符合当前智能代理的研究方向，尤其是在多模态和移动性背景下，还实现了扩展。

While building upon existing technologies, our approach contributes incremental advancements in the precision and adaptability of agents operating

在现有技术基础上，我们的方法在代理的精确性和适应性方面做出了渐进式改进，

within complex mobile environments. Future work will focus on enhancing cross-application functionalities and refining decision-making processes to further improve the efficiency and user experience.

以适应复杂的移动环境。未来工作将聚焦于增强跨应用功能和优化决策过程，进一步提升效率和用户体验。

6 Limitations

6 限制

Throughout the comprehensive testing process, we identified several limitations of our agent: Our method relies on the agent's ability to recognize numerical tags on the UI to determine specific UI elements. This approach can lead to confusion when the UI element itself contains numbers. Such errors can be mitigated through preliminary manual exploration and documentation to clarify the context.

在全面测试过程中，我们发现了代理的若干限制: 我们的方法依赖于代理识别界面上的数字标签以确定特定的 UI 元素。当 UI 元素本身包含数字时，这种方法可能导致混淆。通过预先的人工探索和文档记录以澄清上下文，可以减轻此类错误。

When attempting to interact with hidden UI elements, such as accelerating a video by clicking on the screen, the agent lacks the necessary prior knowledge and cannot detect the acceleration button within the current UI. This limitation hampers its ability to perform specific operations. Future work will focus on enhancing UI recognition and incorporating prior knowledge to address these issues effectively.

在尝试与隐藏的 UI 元素交互时，例如通过点击屏幕加速视频播放，代理缺乏必要的先验知识，无法在当前界面中检测到加速按钮。这一限制阻碍了其执行特定操作的能力。未来工作将致力于提升 UI 识别能力并引入先验知识，以有效解决这些问题。

7 Ethics Statement

7 伦理声明

Our research introduces a novel multimodal agent framework designed to interact seamlessly with smartphone applications, enhancing both user experience and decision-making capabilities. In developing and deploying this technology, we are committed to addressing several key ethical considerations:

我们的研究提出了一种新颖的多模态代理框架，旨在与智能手机应用无缝交互，提升用户体验和决策能力。在开发和部署该技术过程中，我们致力于解决若干关键伦理问题:

Privacy and Data Protection: We ensure strict adherence to global privacy standards, implementing robust data security measures to protect user information.

隐私与数据保护: 我们严格遵守全球隐私标准，实施强有力的数据安全措施以保护用户信息。

Reliability and Safety: We implement safety checks to ensure the reliability of our agent, particularly in dynamic environments.

可靠性与安全性: 我们实施安全检查，确保代理在动态环境中的可靠性。

Societal Impact: We consider the broader impacts of our technology, including potential effects on employment and environmental sustainability.

社会影响: 我们考虑技术的广泛影响，包括对就业和环境可持续性的潜在影响。

Continuous Monitoring: We commit to continuously monitoring and refining our technology to address emerging challenges and integrate user feedback.

持续监控: 我们承诺持续监控和完善技术，以应对新出现的挑战并整合用户反馈。

References

参考文献

Wei-Ge Chen, Irina Spiridonova, Jianwei Yang, Jian-feng Gao, and Chunyuan Li. 2023. Llava-interactive: An all-in-one demo for image chat, segmentation, generation and editing.

Wei-Ge Chen, Irina Spiridonova, Jianwei Yang, Jian-feng Gao, and Chunyuan Li. 2023. Llava-interactive: 一个集成图像聊天、分割、生成和编辑的全能演示。

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.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024. Seeclick: 利用 GUI 定位实现高级视觉 GUI 代理。arXiv 预印本 arXiv:2401.10935。

Meta Fundamental AI Research Diplomacy Team (FAIR)†, Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, et al. 2022. Human-level play in the game of diplomacy by combining language models with strategic reasoning. Science, 378(6624):1067-1074.

Meta 基础人工智能研究外交团队 (FAIR)†, Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty，Daniel Fried，Andrew Goff，Jonathan Gray，Hengyuan Hu 等。2022 年。通过结合语言模型与战略推理实现外交游戏中的人类水平对弈。《科学》，378(6624):1067-1074。

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.

高迪飞，季磊，周罗威，林庆鸿，陈乔雅，范子涵，寿正迈。2023 年。AssistGPT: 一个能够规划、执行、检查和学习的通用多模态助手。arXiv 预印本 arXiv:2306.08640。

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. 2023. Metagpt: Meta programming for multi-agent collaborative framework. arXiv preprint arXiv:2308.00352.

洪思睿，郑夏武，陈乔纳森，程宇恒，王金林，张策尧，王子立，邱家成，林子娟，周立阳等。2023 年。MetaGPT: 多智能体协作框架的元编程。arXiv 预印本 arXiv:2308.00352。

Rongjie Huang, Mingze Li, Dongchao Yang, Jia-tong Shi, Xuankai Chang, Zhenhui Ye, Yuning Wu, Zhiqing Hong, Jiawei Huang, Jinglin Liu, et al. 2024. Audiogpt: Understanding and generating speech, music, sound, and talking head. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pages 23802-23804.

黄荣杰，李明泽，杨东超，史嘉桐，常轩凯，叶振辉，吴宇宁，洪志清，黄嘉伟，刘景林等。2024 年。AudioGPT: 理解与生成语音、音乐、声音及口型动画。发表于 AAAI 人工智能会议论文集，第 38 卷，页 23802-23804。

Sunjae Lee, Junyoung Choi, Jungjae Lee, Hojun Choi, Steven Y Ko, Sangeun Oh, and Insik Shin. 2023. Explore, select, derive, and recall: Augmenting llm with human-like memory for mobile task automation. arXiv preprint arXiv:2312.03003.

李顺在，崔俊英，李正在，崔浩俊，Steven Y Ko，吴相恩，申仁锡。2023 年。探索、选择、推导与回忆: 为移动任务自动化增强具有人类记忆特征的大型语言模型。arXiv 预印本 arXiv:2312.03003。

Sunjae Lee, Junyoung Choi, Jungjae Lee, Munim Hasan Wasi, Hojun Choi, Steven Y. Ko, Sangeun Oh, and Insik Shin. 2024. Explore, select, derive, and recall: Augmenting llm with human-like memory for mobile task automation.

李顺在，崔俊英，李正在，Munim Hasan Wasi，崔浩俊，Steven Y. Ko，吴相恩，申仁锡。2024 年。探索、选择、推导与回忆: 为移动任务自动化增强具有人类记忆特征的大型语言模型。

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rock-täschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems, 33:9459-9474.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel 等。2020 年。面向知识密集型自然语言处理任务的检索增强生成。神经信息处理系统进展，33:9459-9474。

Yanda Li, Chi Zhang, Gang Yu, Zhibin Wang, Bin Fu, Guosheng Lin, Chunhua Shen, Ling Chen, and Yunchao Wei. 2023. Stablellava: Enhanced visual instruction tuning with synthesized image-dialogue data. arXiv preprint arXiv:2308.10253.

李彦达，张驰，余刚，王志斌，傅斌，林国胜，沈春华，陈玲，魏云超。2023 年。StableLLaVA: 通过合成图像对话数据增强视觉指令调优。arXiv 预印本 arXiv:2308.10253。

Minghui Liao, Zhaoyi Wan, Cong Yao, Kai Chen, and Xiang Bai. 2020. Real-time scene text detection with differentiable binarization. In Proc. AAAI.

廖明辉，万兆义，姚聪，陈凯，白翔。2020 年。基于可微分二值化的实时场景文本检测。发表于 AAAI 会议论文集。

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

刘世龙，程浩，刘昊天，张浩，李峰，任天河，邹雪岩，杨建伟，苏航，朱军等。2023a。LLaVA-Plus: 学习使用工具以创建多模态智能体。arXiv 预印本 arXiv:2311.05437。

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. 2023b. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. arXiv preprint arXiv:2303.05499.

刘世龙，曾昭阳，任天河，李峰，张浩，杨杰，李春元，杨建伟，苏航，朱军等。2023b。Grounding DINO: 将 DINO 与有根预训练结合用于开放集目标检测。arXiv 预印本 arXiv:2303.05499。

Yang Liu, Xinshuai Song, Kaixuan Jiang, Weixing Chen, Jingzhou Luo, Guanbin Li, and Liang Lin. 2024. Multimodal embodied interactive agent for cafe scene. arXiv preprint arXiv:2402.00290.

刘洋，宋欣帅，姜凯轩，陈伟星，罗景洲，李冠斌，林亮。2024 年。面向咖啡厅场景的多模态具身交互智能体。arXiv 预印本 arXiv:2402.00290。

Yuxing Long, Xiaoqi Li, Wenzhe Cai, and Hao Dong. 2023. Discuss before moving: Visual language navigation via multi-expert discussions. arXiv preprint arXiv:2309.11382.

龙宇星，李晓琦，蔡文哲，董浩。2023 年。先讨论再行动: 通过多专家讨论实现视觉语言导航。arXiv 预印本 arXiv:2309.11382。

Jiageng Mao, Yuxi Qian, Hang Zhao, and Yue Wang. 2023. Gpt-driver: Learning to drive with gpt. arXiv preprint arXiv:2310.01415.

毛家庚, 钱宇曦, 赵航, 王岳。2023 年。GPT-Driver: 用 GPT 学习驾驶。arXiv 预印本 arXiv:2310.01415。

OpenAI. 2023. Chatgpt. https://openai.com/blog/ chatgpt/. 1, 2.

OpenAI. 2023. Chatgpt. https://openai.com/blog/ chatgpt/. 1, 2.

OpenAI. 2023. Gpt-4 technical report. arXiv, pages 2303-08774.

OpenAI. 2023. Gpt-4 技术报告. arXiv, 页码 2303-08774.

Yiran Qin, Enshen Zhou, Qichang Liu, Zhenfei Yin, Lu Sheng, Ruimao Zhang, Yu Qiao, and Jing Shao. 2023a. Mp5: A multi-modal open-ended embodied system in minecraft via active perception. arXiv preprint arXiv:2312.07472.

秦一然, 周恩深, 刘启昌, 殷振飞, 盛璐, 张瑞茂, 乔宇, 邵靖. 2023a. Mp5: 通过主动感知实现的 Minecraft 多模态开放式具身系统. arXiv 预印本 arXiv:2312.07472.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023b. Toolllm: Facilitating large language models to master 16000+ real-world apis. arXiv preprint arXiv:2307.16789.

秦宇佳, 梁世豪, 叶一宁, 朱昆仑, 颜岚, 陆雅溪, 林彦凯, 丛鑫, 唐翔如, 钱比尔, 等. 2023b. Toolllm: 助力大型语言模型掌握 16000+ 真实世界 API. arXiv 预印本 arXiv:2307.16789.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2024. Toolformer: Language models can teach themselves to use tools. Advances in Neural Information Processing Systems, 36.

蒂莫·希克, 简·德维维迪-余, 罗伯托·德西, 罗伯塔·赖利亚努, 玛丽亚·洛梅利, 埃里克·汉布罗, 卢克·泽特尔-莫耶, 尼古拉·坎切达, 托马斯·西亚洛姆. 2024. Toolformer: 语言模型可以自学使用工具. 神经信息处理系统进展, 36.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2024. Hugging-gpt: Solving ai tasks with chatgpt and its friends in hugging face. Advances in Neural Information Processing Systems, 36.

沈永亮, 宋凯涛, 谭旭, 李东升, 卢伟明, 庄月婷. 2024. Hugging-gpt: 利用 ChatGPT 及其伙伴在 Hugging Face 上解决 AI 任务. 神经信息处理系统进展, 36.

Dídac Surís, Sachit Menon, and Carl Vondrick. 2023. Vipergpt: Visual inference via python execution for reasoning. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 11888- 11898.

迪达克·苏里斯, 萨奇特·梅农, 卡尔·冯德里克. 2023. Vipergpt: 通过 Python 执行实现视觉推理. 载于 IEEE/CVF 国际计算机视觉会议论文集, 页码 11888-11898.

Junke Wang, Dongdong Chen, Chong Luo, Xiyang Dai, Lu Yuan, Zuxuan Wu, and Yu-Gang Jiang. 2023. Chatvideo: A tracklet-centric multimodal and versatile video understanding system. arXiv preprint arXiv:2304.14407.

王俊科, 陈东东, 罗冲, 戴熙阳, 袁璐, 吴祖轩, 姜宇刚. 2023. Chatvideo: 以轨迹为中心的多模态多功能视频理解系统. arXiv 预印本 arXiv:2304.14407.

Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. arXiv preprint arXiv:2401.16158.

王俊阳, 许海洋, 叶嘉博, 闫明, 沈伟舟, 张骥, 黄飞, 桑吉涛. 2024. Mobile-agent: 具备视觉感知的自主多模态移动设备代理. arXiv 预印本 arXiv:2401.16158.

Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. 2024. Autodroid: Llm-powered task automation in android.

文浩, 李元春, 刘国宏, 赵善辉, 余涛, 李家俊, 蒋世奇, 刘云浩, 张雅琴, 刘云鑫. 2024. Autodroid: 基于大型语言模型的 Android 任务自动化.

Hao Wen, Hongming Wang, Jiaxuan Liu, and Yuanchun Li. 2023a. Droidbot-gpt: Gpt-powered ui automation for android. arXiv preprint arXiv:2304.07061.

文浩, 王洪明, 刘佳轩, 李元春. 2023a. Droidbot-gpt: 基于 GPT 的 Android 界面自动化. arXiv 预印本 arXiv:2304.07061.

Licheng Wen, Xuemeng Yang, Daocheng Fu, Xiaofeng Wang, Pinlong Cai, Xin Li, Tao Ma, Yingxuan Li, Linran Xu, Dengke Shang, et al. 2023b. On the road with gpt-4v (ision): Early explorations of visual-language model on autonomous driving. arXiv preprint arXiv:2311.05332.

文立成, 杨雪萌, 傅道成, 王晓峰, 蔡品龙, 李鑫, 马涛, 李颖萱, 徐林然, 尚登科, 等. 2023b. 与 GPT-4V(视觉) 同行: 视觉语言模型在自动驾驶领域的早期探索. arXiv 预印本 arXiv:2311.05332.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023a. Visual chatgpt: Talking, drawing and editing with visual foundation models. arXiv preprint arXiv:2303.04671.

吴晨飞, 尹胜明, 齐伟臻, 王晓东, 唐泽成, 段楠. 2023a. Visual ChatGPT: 与视觉基础模型对话、绘画与编辑. arXiv 预印本 arXiv:2303.04671.

Yue Wu, Xuan Tang, Tom M. Mitchell, and Yuanzhi Li. 2023b. Smartplay: A benchmark for llms as intelligent agents. arXiv preprint arXiv:2310.01557.

吴越, 唐轩, 汤姆·M·米切尔, 李元志. 2023b. Smartplay: 作为智能代理的大型语言模型基准测试. arXiv 预印本 arXiv:2310.01557.

Jingkang Yang, Yuhao Dong, Shuai Liu, Bo Li, Ziyue Wang, Chencheng Jiang, Haoran Tan, Ji-amu Kang, Yuanhan Zhang, Kaiyang Zhou, et al. 2023a. Octopus: Embodied vision-language programmer from environmental feedback. arXiv preprint arXiv:2310.08588.

杨景康, 董宇豪, 刘帅, 李博, 王子悦, 蒋晨成, 谭浩然, 康吉阿木, 张元涵, 周凯阳, 等. 2023a. Octopus: 基于环境反馈的具身视觉语言程序员. arXiv 预印本 arXiv:2310.08588.

Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. 2024a. Gpt4tools: Teaching large language model to use tools via self-instruction. Advances in Neural Information Processing Systems, 36.

杨锐, 宋琳, 李彦伟, 赵思杰, 葛一笑, 李秀, 单英. 2024a. Gpt4tools: 通过自我指导教大型语言模型使用工具. 神经信息处理系统进展, 36.

Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Ze-biao Huang, Bin Fu, and Gang Yu. 2023b. Appa-gent: Multimodal agents as smartphone users. arXiv preprint arXiv:2312.13771.

赵阳，刘家轩，韩宇成，陈鑫，黄泽彪，付斌，余刚。2023b。Appa-gent: 作为智能手机用户的多模态代理。arXiv 预印本 arXiv:2312.13771。

Zhengyuan Yang*, Linjie Li*, Jianfeng Wang*, Kevin Lin*, Ehsan Azarnasab*, Faisal Ahmed*, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. 2023. Mm-react: Prompting chatgpt for multimodal reasoning and action.

杨正远 *，李林杰 *，王建峰 *，林凯文 *，Ehsan Azarnasab*，Faisal Ahmed*，刘子成，刘策，Michael Zeng，王丽娟。2023。Mm-react: 为多模态推理与行动提示 ChatGPT。

Zongxin Yang, Guikun Chen, Xiaodi Li, Wenguan Wang, and Yi Yang. 2024b. Doraemongpt: Toward understanding dynamic scenes with large language models. arXiv preprint arXiv:2401.08392.

杨宗鑫，陈贵坤，李晓迪，王文冠，杨毅。2024b。Doraemongpt: 利用大型语言模型理解动态场景。arXiv 预印本 arXiv:2401.08392。

Dingyao Yu, Kaitao Song, Peiling Lu, Tianyu He, Xu Tan, Wei Ye, Shikun Zhang, and Jiang Bian. 2023. Musicagent: An ai agent for music understanding and generation with large language models. arXiv preprint arXiv:2310.11954.

于定尧，宋凯涛，陆佩玲，何天宇，谭旭，叶伟，张世坤，边江。2023。Musicagent: 基于大型语言模型的音乐理解与生成 AI 代理。arXiv 预印本 arXiv:2310.11954。

Xingcheng Zhou, Mingyu Liu, Bare Luka Zagar, Ekim Yurtsever, and Alois C Knoll. 2023. Vision language models in autonomous driving and intelligent transportation systems. arXiv preprint arXiv:2310.14414.

周兴成，刘明宇，Bare Luka Zagar，Ekim Yurtsever，Alois C Knoll。2023。自动驾驶与智能交通系统中的视觉语言模型。arXiv 预印本 arXiv:2310.14414。

A Prompt Structure Description

A 提示结构说明

In this section, we describe the main prompts used by our agent, highlighting their structure and purpose across different operational phases. The parts enclosed in bold black angle brackets are parameters that can be replaced during the coding phase, while the red text indicates areas to be filled in by the user, and the blue text represents annotations.

本节介绍我们代理使用的主要提示，重点说明其结构和在不同操作阶段的用途。用黑色粗体尖括号括起的部分为编码阶段可替换的参数，红色文本表示用户需填写的内容，蓝色文本为注释。

B Explanation of DroidTask Results

B DroidTask 结果说明

In figure 2, we present the performance of our agent and AutoDroid on the DroidTask benchmark. The differential in testing environments, AutoDroid's real device testing on specific Android phone compared to our emulator-based approach, alongside discrepancies in application versions between the two setups, precluded direct execution of some tasks. For a small subset of tasks that could not be completed, we identified alternative testing methods. For instance, whereas our application lacks a date-sorting option for document names, we considered sorting by the initial letter of the document names as an alternative. This adjustment maintains the same procedural flow and steps, albeit with a slightly different selection at the end. Additionally, there are tasks that our application does not support and for which no alternative exists; these cases were treated as error examples. Therefore, under identical conditions, the performance of our agent would be higher than currently observed.

图 2 展示了我们代理与 AutoDroid 在 DroidTask 基准测试上的表现。测试环境差异——AutoDroid 在特定安卓手机的真实设备上测试，而我们采用模拟器；以及两者应用版本不一致，导致部分任务无法直接执行。对于少数无法完成的任务，我们寻找了替代测试方法。例如，尽管我们的应用缺少按文档名日期排序的选项，但我们考虑以文档名首字母排序作为替代。此调整保持了相同的流程和步骤，仅末尾选择略有不同。此外，存在我们应用不支持且无替代方案的任务，这些视为错误案例。因此，在相同条件下，我们代理的表现将优于当前观察到的结果。

C Details of Action Space

C 动作空间详情

In this section, we provide a detailed description of the usage and parameters for each action space:

本节详细描述每个动作空间的用法及参数:

- TapButton(element: int/str): Initiates a tap action on a user interface element. For example, TapButton(5) taps the UI element labeled as '5'. TapButton ('hat') taps the UI element with text 'hat'.

- TapButton(element: int/str): 对用户界面元素执行点击操作。例如，TapButton(5) 点击标记为“5”的 UI 元素。TapButton('hat') 点击文本为“hat”的 UI 元素。

- Text(text: str): Simulates typing by entering a string of text into a designated input area. For instance, Text (”Hello, world!”) inputs the string ”Hello, world!” into the text field.

- Text(text: str): 模拟输入，在指定输入区域输入一段文本。例如，Text(”Hello, world!”) 在文本框中输入字符串“Hello, world!”。

- LongPress(element: int): Applies a prolonged press on a specified element. For example, LongPress(3) applies a long press to the element labeled ’ 3 ’.

- LongPress(element: int): 对指定元素执行长按操作。例如，LongPress(3) 对标记为“3”的元素进行长按。

- Swipe(element: int, direction: str, dist: str): Executes a swipe action in a specified direction on an element. For instance, Swipe(21, ”up”, ”medium”) swipes up on element ’21’ for a medium distance.

- Swipe(element: int, direction: str, dist: str): 在指定元素上执行某方向滑动操作。例如，Swipe(21, ”up”, ”medium”) 在元素“21”上向上滑动中等距离。

- Back(): Simulates the device’s back button to return to the previous UI state. Useful for navigating back without specific UI interactions.

- Back(): 模拟设备的返回按钮，返回到上一个界面状态。用于无需特定 UI 交互的返回操作。

- Home(): Commands the agent to return to the main screen, essential for resetting the environment or starting new tasks.

- Home(): 指令代理返回主屏幕，对于重置环境或开始新任务至关重要。

- Wait(): Pauses the operation for two seconds to allow system processes to complete.

- Wait(): 暂停操作两秒钟，以允许系统进程完成。

- Stop(): Ends the current operation, signaling the completion of tasks. Useful to terminate processes or to finalize script execution.

- Stop(): 结束当前操作，表示任务完成。用于终止进程或完成脚本执行。

D Case Study

D 案例研究

As illustrated in Figures 8, 9 and 10, we present several case studies showcasing the qualitative results obtained across diverse applications, tasks, and specialized functionalities. Figure 8 highlights a scenario where our agent triggers a safety check during sensitive operations. Figures 9 and 10 display the qualitative results of our agent

handling multi-step tasks. These examples demonstrate the robustness of our agent, emphasizing its capability to effectively manage a variety of complex scenarios.

如图 8、9 和 10 所示，我们展示了多个案例研究，展示了在不同应用、任务和专用功能中获得的定性结果。图 8 突出显示了我们的代理在敏感操作中触发安全检查的场景。图 9 和图 10 展示了代理处理多步骤任务的定性结果。这些示例展示了代理的鲁棒性，强调其有效管理各种复杂场景的能力。

Prompt for Auto-exploration

自动探索提示

You are an agent that is trained to complete certain tasks on a smartphone. You will be given a screenshot of a smartphone app. The interactive UI elements on the screenshot are labeled with numeric tags starting from 1.

你是一个被训练完成智能手机上特定任务的代理。你将获得一张智能手机应用的截图。截图上的交互式 UI 元素标有从 1 开始的数字标签。

Your job is to carefully analyze the difference between the two screenshots to determine if the action is in accord with the description above and at the same time effectively moved the task forward. Your output should be determined based on the following situations:

你的工作是仔细分析两张截图之间的差异，以判断操作是否符合上述描述，同时有效推进任务。你的输出应基于以下情况确定:

You can call the following functions to interact with those labeled elements to control the smartphone:

你可以调用以下函数与标记元素交互以控制智能手机:

[Detailed action space and examples, including tapbutton, text, etc.]

The task you need to complete is to <task_description>. Your past actions to proceed with this task are summarized as

你需要完成的任务是 <task_description>。你为推进该任务所做的过去操作总结如下

follows: <last_act>

如下:<last_act>

Now, given the following labeled screenshot, you need to think and call the function needed to proceed with the task.

现在，给定以下带标签的截图，你需要思考并调用完成任务所需的函数。

Your output should include three parts in the given format:

你的输出应包括以下三部分，格式如下:

Observation: <Describe what you observe in the image>

观察:< 描述你在图像中观察到的内容 >

Thought: <To complete the given task, what is the next step I should do>

思考:< 为完成给定任务，我下一步应做什么 >

Action: <The function call with the correct parameters to proceed with the task. If you believe the task is completed or

操作:< 使用正确参数的函数调用以继续执行任务。如果您认为任务已完成或

there is nothing to be done, you should output FINISH. You cannot output anything else except a function call or FINISH

无需执行任何操作，您应输出 FINISH。您不能输出除函数调用或 FINISH 之外的任何内容

in this field.>

在该领域。

Summary: <Summarize your past actions along with your latest action in one or two sentences. Do not include the numeric

总结:< 用一两句话总结你过去的操作及最新操作。请勿包含数字标签

tag in your summary>

标签在你的总结中 >

You can only take one action at a time, so please directly call the function.

你一次只能执行一个操作，请直接调用函数。

Figure 4: Prompt for agent-driven exploration phase.

图 4: 代理驱动探索阶段的提示。

Prompt for reflection

反思提示

I will give you screenshots of a mobile app before and after <action> the UI element labeled with the number '<ui_element>' on the first screenshot. The numeric tag of each element is located at the center of the element. The action of <action> this UI element was described as follows: <last_act> The action was also an attempt to proceed with a larger task, which is to <task_desc>.

我将给你展示一个移动应用的操作前后截图，<action> 第一个截图中标有数字‘<ui_element>’的 UI 元素。每个元素的数字标签位于元素中心。对该 UI 元素的 <action> 操作描述如下:<last_act> 该操作也是尝试推进一个更大任务，即 <task_desc>。

Your job is to carefully analyze the difference between the two screenshots to determine if the action is in accord with the description above and at the same time effectively moved the task forward. Your output should be determined based on the following situations:

你的任务是仔细分析两张截图的差异，以判断该操作是否符合上述描述，并且有效推动了任务进展。你的输出应基于以下情况判断:

[Detailed requirements for each situation, including back, ineffective, continue, success]

#Example for output format:

输出格式示例:

1. Decision: BACK

1. 决策: 返回

Thought: <explain why you think the last action is wrong and you should go back to the previous interface>

思考:< 解释为何认为上次操作错误，应返回上一界面 >

Documentation: <describe the function of the UI element>

说明:< 描述该 UI 元素的功能 >

2. Decision: INEFFECTIVE

2. 决策: 无效

Thought: <explain why you made this decision>

思考:< 解释为何做出此决策 >

3. Decision: CONTINUE

3. 决策: 继续

Thought: <explain why you think the action does not reflect the action description above and did not move the given

思考:< 解释为何你认为该操作未能反映上述操作描述，且未推动给定任务的进展 >

task forward>

任务进展 >

Documentation: <describe the function of the UI element>

文档说明:< 描述该 UI 元素的功能 >

4. Decision: SUCCESS

4. 决策: 成功

Thought: <explain why you think the action successfully moved the task forward>

思考:< 解释为何你认为该操作成功推动了任务的进展 >

Documentation: <describe the function of the UI element>

文档说明:< 描述该 UI 元素的功能 >

Figure 5: Prompt for reflection phase.

图 5: 反思阶段提示。

Prompt for action space

操作空间提示

#A prompt example for action space, take 'tap button' as an example.

操作空间提示示例，以“点击按钮”为例。

I will give you the screenshot of a mobile app before and after tapping the UI element labeled with the button <ui_element> on the screen. The numeric tag of each element is located at the center of the element.

我将给你展示一个移动应用在点击屏幕上标记为 <button> 的 UI 元素前后的截图。每个元素的数字标签位于元素中心。

Tapping this UI element is a necessary part of proceeding with a larger task, which is to <task_desc>.

点击该 UI 元素是执行更大任务的必要步骤，该任务是 <task_desc>。

Your task is to describe the functionality of the UI element concisely in one or two sentences. Notice that your description of the UI element should focus on the general function. For example, if the UI element is used to navigate to the chat window with John, your description should not include the name of the specific person. Just say: "Tapping this area will navigate the user to the chat window".

你的任务是用一到两句话简洁描述该 UI 元素的功能。请注意，描述应聚焦于一般功能。例如，如果该 UI 元素用于导航至与 John 的聊天窗口，描述中不应包含具体人名，只需说:“点击此区域将导航用户至聊天窗口”。

Never include the numeric tag of the UI element in your description. You can use pronouns such as "the UI element" to refer to the element.

切勿在描述中包含该 UI 元素的数字标签。你可以使用“该 UI 元素”等代词指代该元素。

Figure 6: Prompt for action space.

图 6: 操作空间提示。

Prompt for task_template

任务模板提示

You are an agent trained to perform basic tasks on a smartphone. When given a smartphone screenshot along with reference documents, your primary directive is to derive actionable insights from the documentation provided. These documents are essential for understanding the functionalities of UI elements that may not be immediately apparent from the screenshot alone. Your actions should be primarily informed by these documents, with the current UI interface analysis serving as a secondary reference.

你是一名经过训练的智能代理，负责在智能手机上执行基本任务。当收到智能手机截图及参考文档时，你的主要指令是从所提供的文档中提取可执行的见解。这些文档对于理解截图中不易直接察觉的 UI 元素功能至关重要。你的操作应主要基于这些文档，当前 UI 界面分析仅作为辅助参考。

Your decision-making process should prioritize actions as follows:

您的决策过程应优先考虑以下行动:

[Special requirements of decision-making.]

You can call the following functions to control the smartphone:

您可以调用以下函数来控制智能手机:

[Detailed action space and examples, including tapbutton, text, etc.]

<ui_document>

<ui_document>

The task you need to complete is to <task_description>. Your past actions to proceed with this task are summarized as

您需要完成的任务是 <task_description>。您为推进此任务所采取的过去行动总结如下

follows: <last_act>

如下:<last_act>

Now, given the documentation and the following labeled screenshot, you need to think and call the function needed to

现在，结合文档和以下标注的截图，您需要思考并调用所需函数以

proceed with the task. Your output should include three parts in the given format:

推进任务。您的输出应包括以下三部分，格式如下:

Observation: <Describe what you observe in the image>

观察:< 描述您在图像中观察到的内容 >

Thought: <To complete the given task, what is the next step I should do>

思考:< 为完成给定任务，我接下来的步骤应是 >

Action: <The function call with the correct parameters to proceed with the task. If you believe the task is completed or there is nothing to be done, you should output FINISH. You cannot output anything else except a function call or FINISH in this field.>

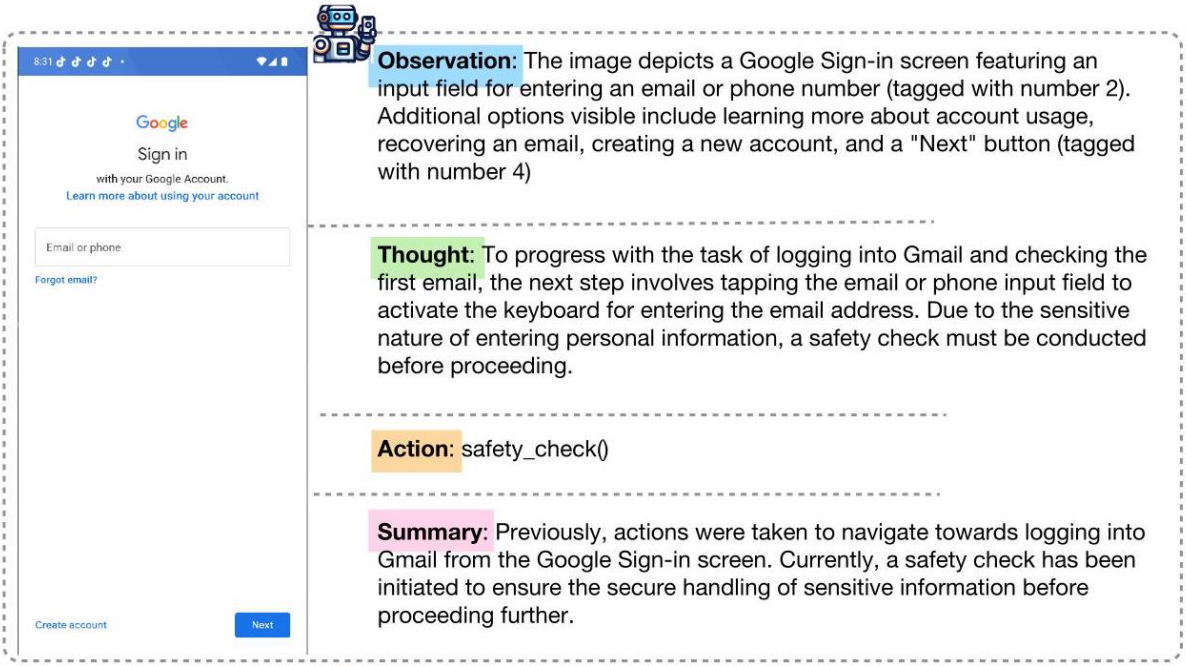
操作:< 调用带有正确参数的函数以推进任务。如果您认为任务已完成或无需操作，应输出 FINISH。此字段中只能输出函数调用或 FINISH，不能输出其他内容。>

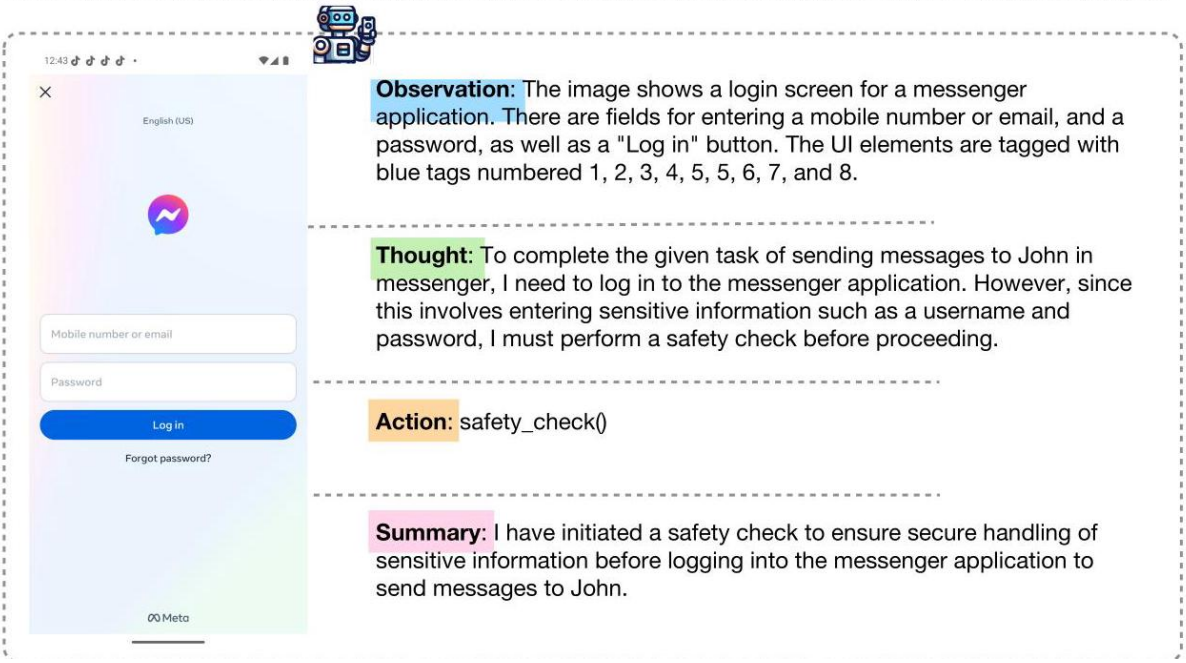
Summary: <Summarize your past actions along with your latest action in one or two sentences. Do not include the numeric tag in your summary> You can only take one action at a time, so please directly call the function.

总结:< 用一两句话总结您过去的行动及最新行动。总结中不包含数字标签 > 您一次只能执行一个操作，请直接调用函数。

Figure 7: Prompt for development phase.

图 7: 开发阶段提示。



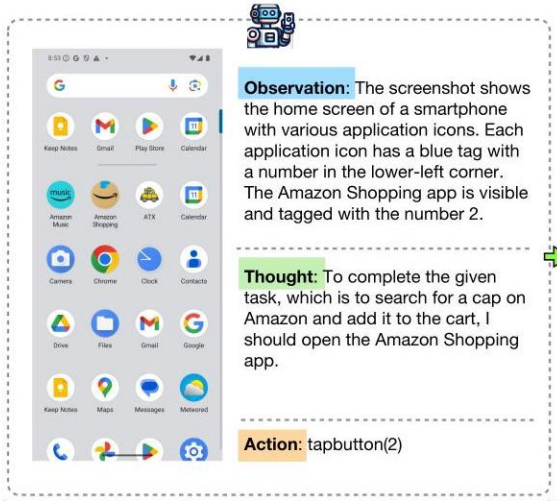


Safety Check Reminder: The agent has automatically detected a sensitive operation. The user will now proceed with the operation manually. Once the operation is complete, you can enter finish to transition back to the automated deployment phase.

安全检查提醒: 系统已自动检测到敏感操作。用户将手动继续该操作。操作完成后, 您可以输入 finish 以切换回自动部署阶段。

Figure 8: Safety check examples.

图 8: 安全检查示例。

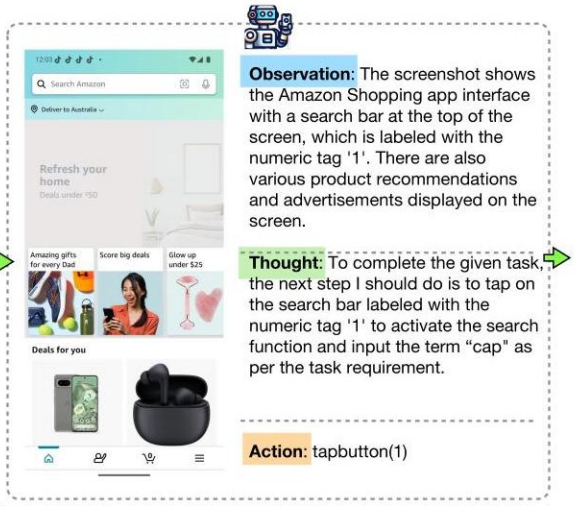


Observation: The screenshot shows the home screen of a smartphone with various application icons. Each application icon has a blue tag with a number in the lower-left corner. The Amazon Shopping app is visible and tagged with the number 2.

Thought: To complete the given task, which is to search for a cap on Amazon and add it to the cart, I should open the Amazon Shopping app.

Action: tapbutton(2)

Summary: I have opened the Amazon Shopping app to search for a hat and add it to the cart.

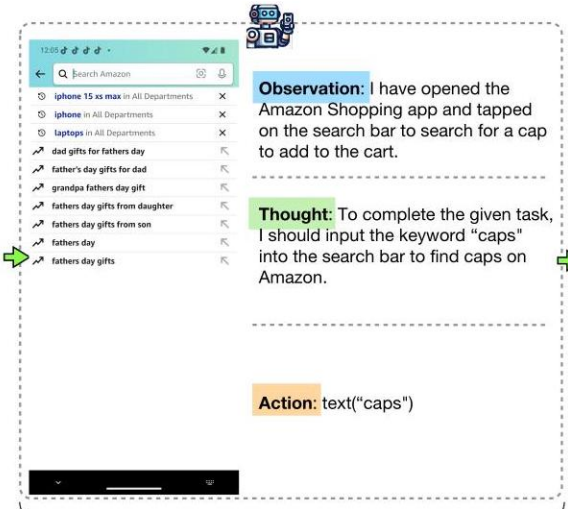


Observation: The screenshot shows the Amazon Shopping app interface with a search bar at the top of the screen, which is labeled with the numeric tag '1'. There are also various product recommendations and advertisements displayed on the screen.

Thought: To complete the given task, the next step I should do is to tap on the search bar labeled with the numeric tag '1' to activate the search function and input the term “cap" as per the task requirement.

Action: tapbutton(1)

Summary: I have opened the Amazon Shopping app and tapped on the search bar to search for a hat to add to the cart.

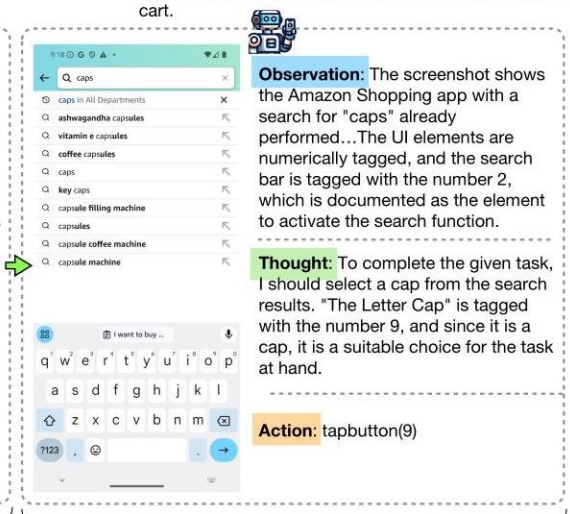


Observation: I have opened the Amazon Shopping app and tapped on the search bar to search for a cap to add to the cart.

Thought: To complete the given task, I should input the keyword “caps" into the search bar to find caps on Amazon.

Action: text(“caps")

Summary: I have opened the Amazon Shopping app, tapped on the search bar, and now I have entered the keyword "hat" to search for hats on Amazon.

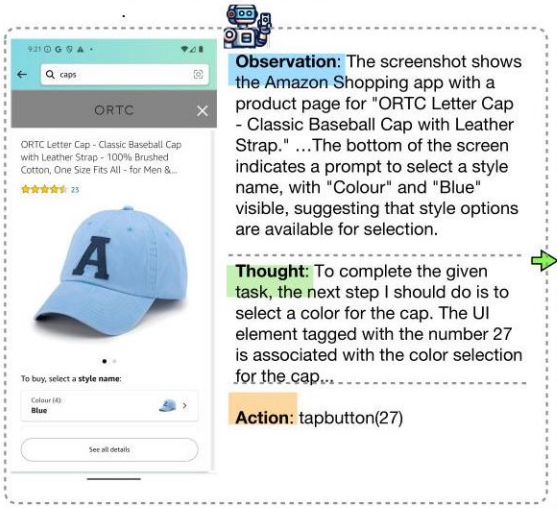


Observation: The screenshot shows the Amazon Shopping app with a search for "caps" already performed…The UI elements are numerically tagged, and the search bar is tagged with the number 2, which is documented as the element to activate the search function.

Thought: To complete the given task, I should select a cap from the search results. "The Letter Cap" is tagged with the number 9, and since it is a cap, it is a suitable choice for the task at hand.

Action: tapbutton(9)

Summary: I have opened the Amazon Shopping app, initiated a search for caps, and now I have selected "The Letter Cap" from the search results by tapping on it.

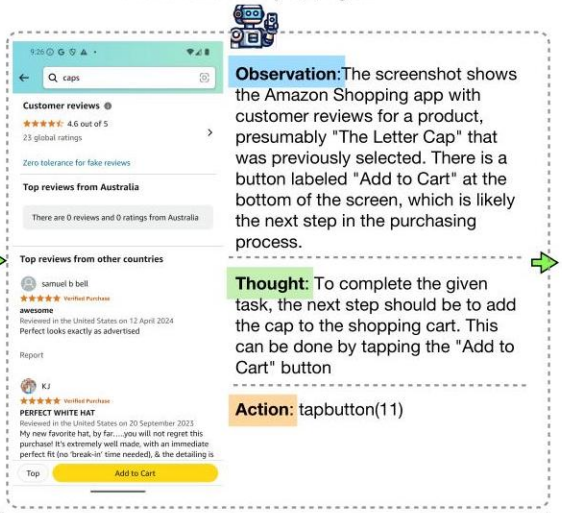


Observation: The screenshot shows the Amazon Shopping app with a product page for "ORTC Letter Cap - Classic Baseball Cap with Leather Strap." …The bottom of the screen indicates a prompt to select a style name, with "Colour" and "Blue" visible, suggesting that style options are available for selection.

Thought: To complete the given task, the next step I should do is to select a color for the cap. The UI element tagged with the number 27 is associated with the color selection for the cap…

Action: tapbutton(27)

Summary: I have opened the Amazon Shopping app, searched for caps, selected "The Letter Cap" from the search results, viewed the buying options, selected a style, and now I have proceeded to select a color for the cap by tapping on the corresponding UI element.



Observation:The screenshot shows the Amazon Shopping app with customer reviews for a product, presumably "The Letter Cap" that was previously selected. There is a button labeled "Add to Cart" at the bottom of the screen, which is likely the next step in the purchasing process.

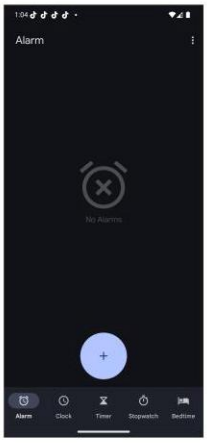
Thought: To complete the given task, the next step should be to add the cap to the shopping cart. This can be done by tapping the "Add to Cart" button

Action: tapbutton(11)

Summary: I have opened the Amazon Shopping app, searched for caps, selected "The Letter Cap" from the search results, viewed the buying options, selected a style, chosen a color, and now I have added the cap to the shopping cart by tapping the "Add to Cart" button.

Figure 9: Task: search cap in Amazon and add one to cart.

图 9: 任务: 在亚马逊搜索帽子并加入购物车。



Observation: The screenshot shows an alarm application with no alarms currently set. There is a UI element labeled with the numeric tag '2' which, according to the documentation, allows the user to create a new alarm when tapped.

Thought: To complete the given task, the next step I should do is tap the UI element labeled with the numeric tag '2' to create a new alarm.

Action: tapbutton(2)

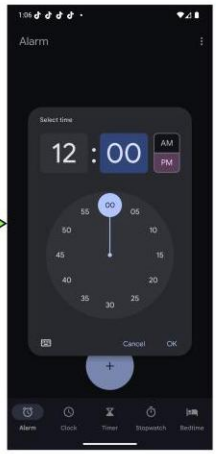
Observation: The screenshot shows an alarm clock application with the time currently set to 02:00. There are UI elements for hour and minute selection, AM/PM toggle, and buttons to cancel or confirm the alarm time. The hour section is high lighted, indicating that it is currently selected for adjustment.

Thought: To complete the given task, the next step I should do is switch from hour selection to minute selection by tapping the UI element labeled with the numeric tag '6'.

Action: tapbutton(6)

Summary: I have initiated the process of setting a new alarm by tapping the button to create a new alarm.

Summary: I have initiated the process of setting a new alarm and have now switched from hour selection to minute selection.



Observation: The screenshot shows an alarm clock application with the time selection interface open. The hour is set to 12, …and the 'OK' button, which is tagged with the number 19. .

Thought: To complete the given task, the next step I should do is to select the '30' minute option to set the alarm time to 12:30 pm.

Action: tapbutton(14)

Observation: The screenshot shows an alarm clock application with the time selection interface. The hour is set to '12', …There is also a button tagged with '19' labeled "OK", which is likely used to confirm the time selection.

Thought: To complete the given task, the next step I should do is confirm the selected time of 12:30 PM by tapping the "OK" button, which is tagged with '19'.

Action: tapbutton(19)

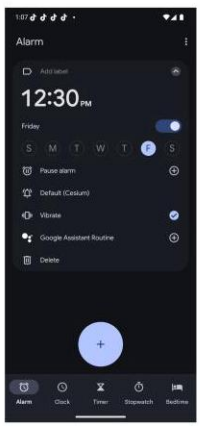
Summary:I have initiated the process of setting a new alarm, selected the hour, and now I have selected the '30' minute option to set the alarm for 12:30 pm.

Summary: I have set the alarm for 12:30 PM and confirmed the time selection by tapping the "OK" button.

Observation: The screenshot shows an alarm setting interface with various options. The time is set to 12:30 PM, …There are also options to delete the alarm and a button to create a new alarm. The days of the week are displayed, with Friday tagged as '12'.

Thought: To complete the given task, the next step I should do is to set the alarm to repeat every Friday. This can be done by selecting Friday from the days of the week options.

Action: tapbutton(12)



Observation:The screenshot shows an alarm setting interface with various options. The alarm time is set to 12:30 PM and Friday is highlighted, indicating that the alarm is set to repeat every Friday. There is a toggle labeled "Vibrate" wi th a numeric tag '17' next to it, which is currently enabled.

Thought: To complete the given task, the next step I should do is disable the vibration for the alarm as per the task requirements.

Action: tapbutton(17)

Summary: I have set the alarm for 12:30 PM, confirmed the time selection, and now I have selected Friday to set the alarm to repeat every Friday.

Summary: I have set an alarm for 12:30 PM to repeat every Friday and have now disabled the vibration for this alarm.

Figure 10: Task:set an alarm at 12:30 pm every Friday, disable the vibration.

图 10: 任务: 每周五下午 12:30 设置闹钟，关闭振动。