Article

Not peer-reviewed version

# OS Agents: A Survey on MLLM-Based Agents for General Computing Devices Use

Xueyu Hu , Tao Xiong , Biao Yi , Zishu Wei , Ruixuan Xiao , Yurun Chen , Jiasheng Ye , Meiling Tao , Xiangxin Zhou , Ziyu Zhao , Yuhuai Li , Shengze Xu , Shawn Wang , Xinchen Xu , Shuofei Qiao , Kun Kuang , Tieyong Zeng , Liang Wang , Jiwei Li , Yuchen Eleanor Jiang , Wangchunshu Zhou , Guoyin Wang , Keting Yin , Zhou Zhao , Hongxia Yang , Fan Wu , Shengyu Zhang * , Fei Wu

Posted Date: 26 December 2024

doi: 10.20944/preprints202412.2294.v1

Keywords: OS Agent; GUI; MLLM; LLM; computer use

 doi:10.20944/preprints202412.2294.v1



*Article*

# OS Agents: A Survey on MLLM-Based Agents for General Computing Devices Use

**Xueyu Hu [1,†], Tao Xiong [1,‡], Biao Yi [1,‡], Zishu Wei [1,‡], Ruixuan Xiao [1], Yurun Chen [1], Jiasheng Ye [2], Meiling Tao [3], Xiangxin Zhou [4,5], Ziyu Zhao [1], Yuhuai Li [1], Shengze Xu [6], Shawn Wang [7], Xinchen Xu [1], Shuofei Qiao [1], Kun Kuang [1], Tieyong Zeng [6], Liang Wang [4,5], Jiwei Li [1], Yuchen Eleanor Jiang [3], Wangchunshu Zhou [3], Guoyin Wang [8], Keting Yin [1], Zhou Zhao [1], Hongxia Yang [9], Fan Wu [10], Shengyu Zhang [1,*] and Fei Wu [1]**

[1] Zhejiang University; huxueyu@zju.edu.cn
[2] Fudan University
[3] OPPO AI Center
[4] University of Chinese Academy of Sciences
[5] Institute of Automation, Chinese Academy of Sciences
[6] The Chinese University of Hong Kong
[7] Tsinghua University
[8] 01.AI
[9] The Hong Kong Polytechnic University
[10] Shanghai Jiao Tong University
[*] Correspondence: sy_zhang@zju.edu.cn
[†] Project Lead.
[‡] Core Contributor.

**Abstract:** The dream to create AI assistants as capable and versatile as the fictional *J.A.R.V.I.S* from *Iron Man* has long captivated imaginations. With the evolution of (multimodal) large language models ((M)LLMs), this dream is closer to reality, as (M)LLM-based Agents using computing devices (e.g., computers and mobile phones) by operating within the environments and interfaces (e.g., Graphical User Interface (GUI)) provided by operating systems (OS) to automate tasks have significantly advanced. This paper presents a comprehensive survey of these advanced agents, designated as **OS Agents**. We begin by elucidating the fundamentals of OS Agents, exploring their key components including the environment, observation space, and action space, and outlining essential capabilities such as understanding, planning, and grounding. We then examine methodologies for constructing OS Agents, focusing on domain-specific foundation models and agent frameworks. A detailed review of evaluation protocols and benchmarks highlights how OS Agents are assessed across diverse tasks. Finally, we discuss current challenges and identify promising directions for future research, including safety and privacy, personalization and self-evolution. This survey aims to consolidate the state of OS Agents research, providing insights to guide both academic inquiry and industrial development. An open-source GitHub repository is maintained as a dynamic resource to foster further innovation in this field.

**Keywords:** OS Agent; GUI; MLLM; LLM; computer use

 doi:10.20944/preprints202412.2294.v1

# Contents

 doi:10.20944/preprints202412.2294.v1



## 1. Introduction

Building a superintelligent AI assistant akin to *J.A.R.V.I.S.*[1] from the Marvel movie *Iron Man*, which assists *Tony Stark* in controlling various systems and automating tasks, has long been a human aspiration. These entities are recognized as **Operating System Agents (OS Agents)**, as they use computing devices (e.g., computers and mobile phones) by operating within the environments and interfaces (e.g., Graphical User Interface (GUI)) provided by operating systems (OS). OS Agents can complete tasks autonomously and have the potential to significantly enhance the lives of billions of users worldwide. Imagine a world where tasks such as online shopping, travel arrangements booking, and other daily activities could be seamlessly performed by these agents, thereby substantially increasing efficiency and productivity. In the past, virtual assistants such as Siri [1], Cortana [2], Amazon Alexa [3] and Google Assistant[4] have already offered glimpses into this potential, but limitations in model capabilities such as contextual understanding [5], have prevented these products from achieving widespread adoption and full functionality.

Fortunately, recent advancements in (multimodal) large language models ((M)LLMs), such as Gemini [6], GPT [7], Grok [8], Yi [9] and Claude [10] series[2] have ushered in a new era of possibilities for OS Agents. These models boast remarkable abilities, enabling OS Agents to better understand complex tasks and use computing devices to execute. As illustrated in Figure 1, there has been a surge of OS Agents in both commercial products and academic research. Notable examples include the recently released Computer Use by Anthropic [12], Apple Intelligence by Apple [13], AutoGLM by Zhipu AI [14] and Project Mariner by Google Deepmind [15]. For instance, Computer Use leverages Claude [16] to interact directly with users' computers, aiming for seamless task automation. In the research community, a variety of works have been proposed to build (M)LLM-based OS Agents [17–28]. For instance, Wu et al. [22] proposes OS-Atlas, a foundational GUI action model that significantly improves GUI grounding and Out-Of-Distribution task performance by synthesizing GUI grounding data across various platforms. OS-Copilot [29] is an agent framework crafted to develop generalist agents that automate broad computer tasks, demonstrating robust generalization and self-improvement across diverse applications with minimal supervision. Given these advancements and the growing body of work, it has become increasingly important to provide a comprehensive survey that consolidates the current state of research in this area.

In this survey, we begin by discussing the fundamentals of OS Agents (§2), starting with a definition of what constitutes an OS Agent. As illustrated in Figure 2, we focus on three key components: the environment, the observation space, and the action space (§2.1). We then outline the essential capabilities OS Agents should possess, including understanding, planning, and grounding (§2.2). Next, we explore two critical aspects of constructing OS Agents (§3): (1) the development of domain-specific foundation models, covering areas such as architectural design, pre-training, supervised fine-tuning, and reinforcement learning (§3.1); and (2) the building of effective agent frameworks around these models, addressing core elements including perception, planning, memory, and action (§3.2). We also review the evaluation protocol (§4.1) and benchmarks (§4.2) commonly used to assess the performance of OS Agents. Finally, we discuss the challenges and future directions for OS Agents (§5), with a particular focus on issues related to safety and privacy (§5.1), as well as personalization and self-evolution (§5.2).

[1] J.A.R.V.I.S. stands for "Just A Rather Very Intelligent System", a fictional AI assistant character from the Marvel Cinematic Universe. It appears in Iron Man (2008), The Avengers (2012), and other films, serving as Tony Stark's (Iron Man's) personal assistant and interface for his technology.

[2] Rankings were determined using the Chatbot Arena LLM Leaderboard [11] as of December 22, 2024. For models originating from the same producer, rankings were assigned based on the performance of the highest-ranking model.

 doi:10.20944/preprints202412.2294.v1



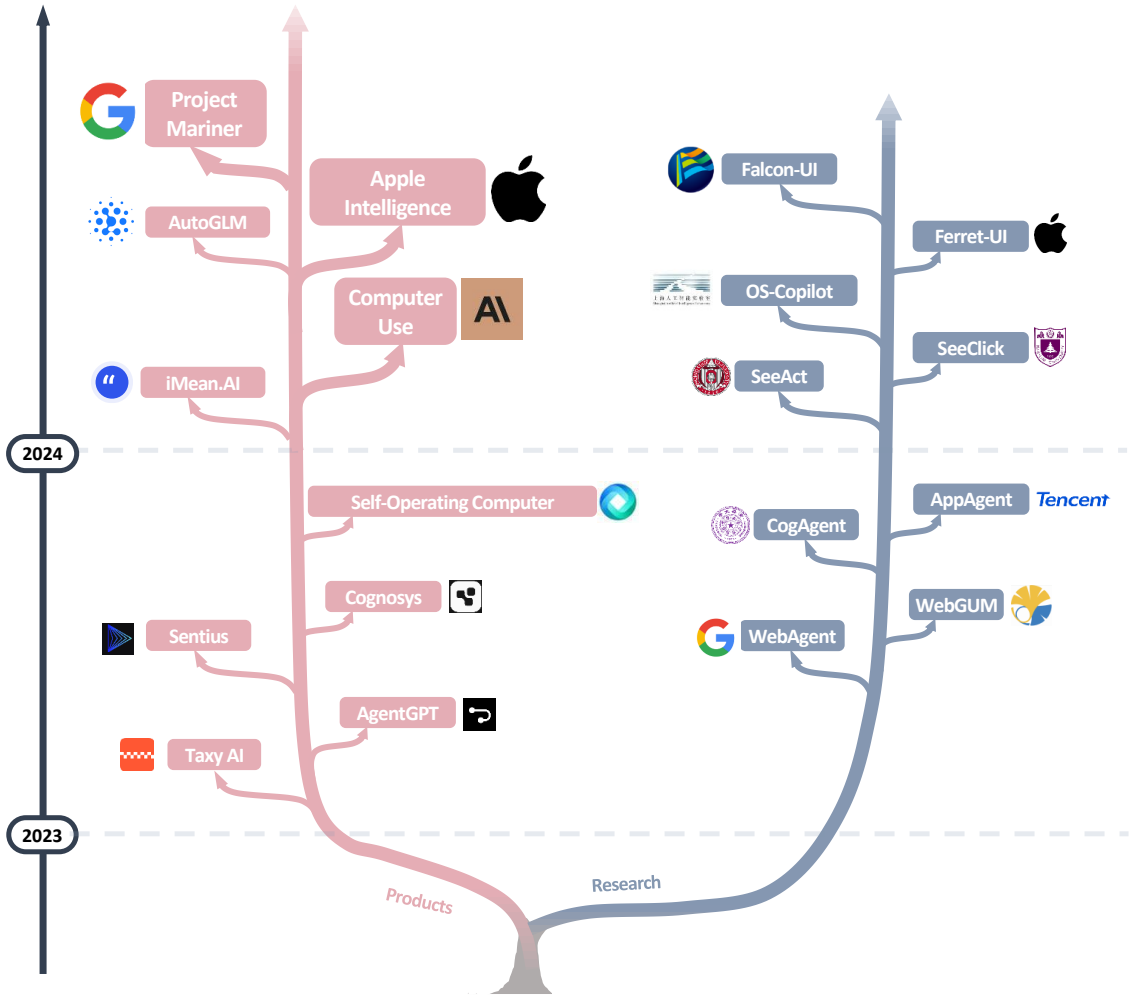


**Figure 1.** The representative commercial products and academic research related to OS Agents. Part of the materials used in this figure are adapted from this repo.

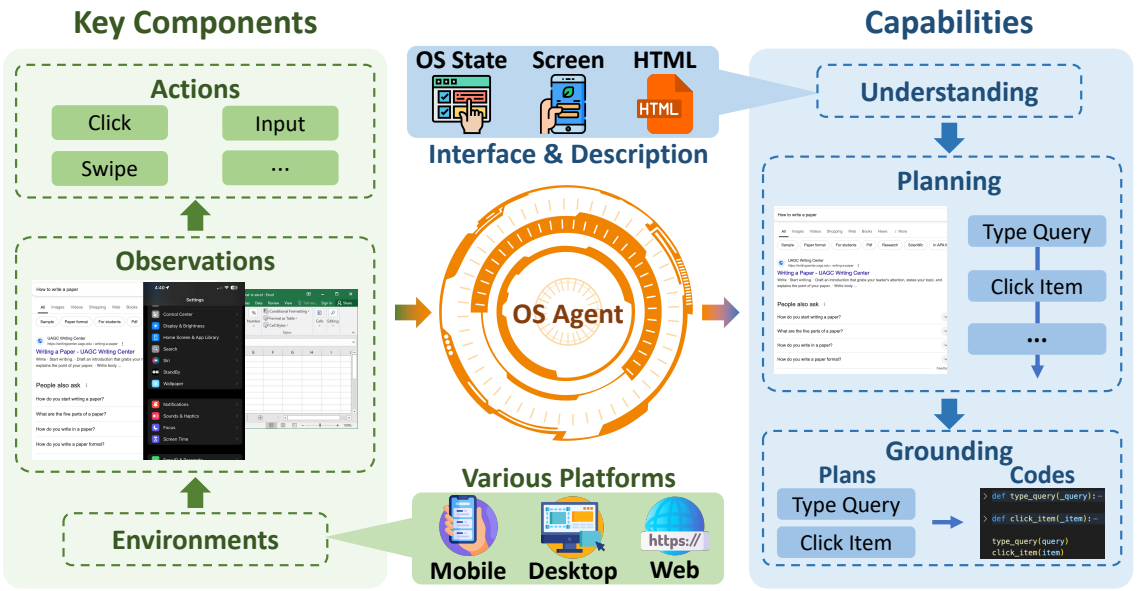


**Figure 2.** Fundamentals of OS Agents.

This survey aims to make contributions to the research and development of OS Agents by providing readers with a comprehensive understanding of their essential capabilities, offering insights into

 doi:10.20944/preprints202412.2294.v1



methodologies for building OS Agents based on (M)LLMs, and highlighting the latest research trends, challenges and future in this field. Recognizing that OS Agents are still in their early stages of development, we acknowledge the rapid advancements that continue to introduce novel methodologies and applications. To support ongoing developments, we maintain an open-source GitHub repository as a dynamic resource. Through this work, we aspire to inspire further innovation, driving progress in both academic research and industrial applications of OS Agents.

## 2. Fundamental of OS Agents

OS Agents are specialized AI agents that leverage the environment, input, and output interfaces provided by the operating system to generally using computing devices in response to user-defined goals. These agents are designed to automate tasks executed within the operating system, leveraging the exceptional understanding and generative capabilities of (M)LLMs to enhance user experience and operational efficiency. To achieve this, OS Agents are based on three key components: Environment, Observation Space, and Action Space, which together facilitate the agent's effective engagement with the operating system. Additionally, OS Agents necessitate three core capabilities: Understanding, Planning, and Grounding. These capabilities enable them to sequentially comprehend tasks, devise action strategies, and implement these actions effectively within the environment.

### *2.1. Key Component*

**Environment.** The environment for OS Agents refers to the system or platform in which they operate. This can include desktop [30–32], mobile [33–37] or web [38–43]. OS Agents interact with these diverse environments to perform tasks, gather feedback, and adapt to their unique characteristics. These environments encompass a diverse set of tasks, ranging from simple interactions such as information retrieval to complex multi-step operations, requiring agents to perform planning and reasoning across multiple interfaces, significantly increasing the complexity and posing challenges for OS Agents. We refer readers to §4.2 for detailed discussion.

**Observation Space.** The observation space encompasses the information OS Agents can access about the system's state and user activities. These observations guide the agents in comprehending the environment, making informed decisions, and determining the appropriate actions to achieve user-defined goals. Observation includes capturing outputs from the OS, such as screen images [24,26,44,45] with specific processing [23,27,46], or textual data, such as the description of the screen [29,30] and the HTML code [25,47] in web-based contexts. Multimodal input integrating these diverse data structure introduces significant challenges for agents to effectively understand and execute tasks. Further details are elaborated in §3.2.1.

**Action Space.** The action space defines the set of interactions through which OS Agents manipulate the environment using the input interfaces provided by the operating system. These actions can be broadly categorized into input operations [23,30,48], representing the primary methods of interacting with digital interfaces, navigation operations [24,49,50] which facilitate movement across the system's interface and extended operations, such as utilizing external tools or services [29,51]. These actions enable OS Agents to execute tasks, control applications, and automate workflows effectively. A comprehensive discussion can be found in §3.2.4.

### *2.2. Capability*

**Understanding.** A crucial capability of OS Agents is their ability to comprehend complex OS environments. These environments encompass a diverse array of data formats, including HTML code [17,52] and graphical user interfaces captured in screenshots [22,53]. The complexity escalates with length code with sparse information, high-resolution interfaces cluttered with minuscule icons, small text, and densely packed elements [18,27,54]. Such environments challenge the agents' perceptual abilities and demand advanced contextual comprehension. This comprehension is essential not only for tasks aimed at information retrieval [34] but also serves as a fundamental prerequisite for effectively executing a broad spectrum of additional tasks.

 doi:10.20944/preprints202412.2294.v1



**Planning.** Planning [55–57] is a fundamental capability of OS Agents, enabling them to decompose complex tasks into manageable sub-tasks and devise sequences of actions to achieve specific goals [29,30]. Planning within operating systems often requires agents to dynamically adjust plans based on environmental feedback and historical actions [28,44,58]. Reasoning strategies like ReAct [59] and CoAT [26] are also necessary to ensure effective task execution in dynamic and unpredictable scenarios.

**Grounding.** Action grounding is another essential capability of OS Agents, referring to the ability to translate textual instructions or plans into executable actions within the operating environment [22,47]. The agent must identify elements on the screen and provide the necessary parameters (e.g., coordinates, input values) to ensure successful execution. While OS environments often contain numerous selectable elements and possible actions, the resulting complexity makes grounding tasks particularly challenging.

## 3. Construction of OS Agents

In this section, we discuss effective strategies for constructing OS Agents. We begin by focusing on the development of foundation models tailored for OS Agents. Domain-specific foundation models [60–63] can significantly enhance the performance of OS Agents by incorporating specialized knowledge and capabilities essential for interacting with operating systems. This can be achieved through thoughtful model architecture design and targeted training strategies that align with specific tasks in this domain. In addition, we explore the construction of agent frameworks [64–67] that build upon these foundation models using non-tuning strategies. Techniques such as reasoning strategies and memory augmentation enable agents to accurately perceive their environment, generate effective plans, and execute precise actions without the need for fine-tuning. These approaches offer flexibility and efficiency, allowing OS Agents to generalize across diverse tasks and environments. By combining robust domain-specific foundation models with agent frameworks, we can further enhance the adaptability, reliability, and efficiency of OS Agents in automating complex tasks.

### *3.1. Foundation Model*

The construction of foundation models for OS Agents involves two key components: model architecture and training strategies. The architecture defines how models deal with input and output within OS environments, while training strategies enhance models with the ability of completing complex tasks. As illustrated in Figure 3, training strategies that are applied in construction of foundation models for OS Agents mainly include pre-training, supervised finetuning and reinforcement learning. Table 1 summarizes the architecture and training strategies used in the recent foundation models for OS Agents.

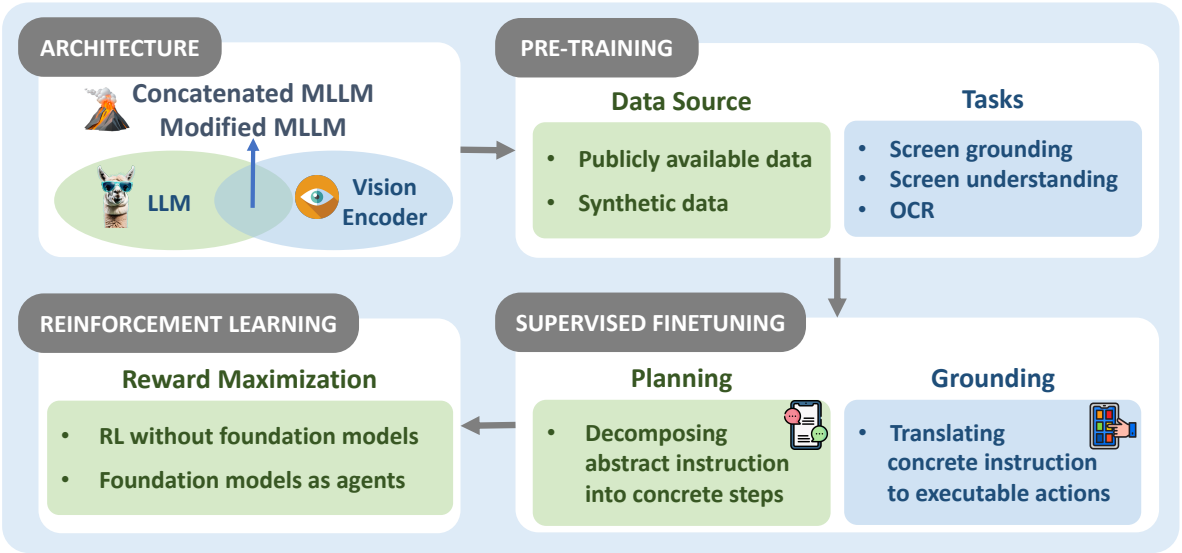


**Figure 3.** Summary of the content about foundation models for OS Agents in §3.1.

 doi:10.20944/preprints202412.2294.v1



**Table 1.** Recent foundation models for OS Agents. Arch.: Architecture, Exist.: Existing, Mod.: Modified, Concat.: Concatenated, PT: Pre-Train, SFT: Supervised Fine-Tune, RL: Reinforcement Learning.

| Model | Arch. | PT | SFT | RL | Date |
|---|---|---|---|---|---|
| OS-Atlas [22] | Exist. MLLMs | ✓ | ✓ | - | 10/2024 |
| AutoGLM [14] | Exist. LLMs | ✓ | ✓ | ✓ | 10/2024 |
| EDGE [21] | Exist. MLLMs | - | ✓ | - | 10/2024 |
| Ferret-UI 2 [68] | Exist. MLLMs | - | ✓ | - | 10/2024 |
| ShowUI [69] | Exist. MLLMs | ✓ | ✓ | - | 10/2024 |
| UIX [70] | Exist. MLLMs | - | ✓ | - | 10/2024 |
| TinyClick [71] | Exist. MLLMs | ✓ | - | - | 10/2024 |
| UGround [19] | Exist. MLLMs | - | ✓ | - | 10/2024 |
| NNetNav [72] | Exist. LLMs | - | ✓ | - | 10/2024 |
| Synatra [73] | Exist. LLMs | - | ✓ | - | 09/2024 |
| MobileVLM [74] | Exist. MLLMs | ✓ | ✓ | - | 09/2024 |
| UI-Hawk [75] | Mod. MLLMs | ✓ | ✓ | - | 08/2024 |
| GUI Action Narrator [76] | Exist. MLLMs | - | ✓ | - | 07/2024 |
| MobileFlow [53] | Mod. MLLMs | ✓ | ✓ | - | 07/2024 |
| VGA [20] | Exist. MLLMs | - | ✓ | - | 06/2024 |
| OdysseyAgent [77] | Exist. MLLMs | - | ✓ | - | 06/2024 |
| Textual Foresight [78] | Concat. MLLMs | ✓ | ✓ | - | 06/2024 |
| WebAI [79] | Concat. MLLMs | - | ✓ | ✓ | 05/2024 |
| GLAINTEL [80] | Exist. MLLMs | - | - | ✓ | 04/2024 |
| Ferret-UI [18] | Exist. MLLMs | - | ✓ | - | 04/2024 |
| AutoWebGLM [52] | Exist. LLMs | - | ✓ | ✓ | 04/2024 |
| Patel et al. [81] | Exist. LLMs | - | ✓ | - | 03/2024 |
| ScreenAI [82] | Exist. MLLMs | ✓ | ✓ | - | 02/2024 |
| Dual-VCR [83] | Concat. MLLMs | - | ✓ | - | 02/2024 |
| SeeClick [84] | Exist. MLLMs | ✓ | ✓ | - | 01/2024 |
| CogAgent [54] | Mod. MLLMs | ✓ | ✓ | - | 12/2023 |
| ILuvUI [85] | Mod. MLLMs | - | ✓ | - | 10/2023 |
| RUIG [86] | Concat. MLLMs | - | - | ✓ | 10/2023 |
| WebAgent [87] | Concat. LLMs | ✓ | ✓ | - | 07/2023 |
| WebGUM [88] | Concat. MLLMs | - | ✓ | - | 05/2023 |

#### 3.1.1. Architecture

A variety of architectures are employed to construct foundation models for OS Agents. It is common practice to build these models by leveraging existing open-source LLMs and MLLMs. Some architectures can be created by concatenating LLMs with vision encoders, enabling the models to process both textual and visual information. Additionally, MLLMs are frequently adapted by incorporating supplementary modules to address the specific requirements such as high-resolution image understanding.

 doi:10.20944/preprints202412.2294.v1



**Existing LLMs.** The architecture of existing LLMs can already process user instructions and read HTML code to perceive information contained in user interfaces. Therefore, several works [14,52, 81] directly chose open-source LLMs as backbone models without further optimizing architecture to develop foundation models for OS Agents, where T5 [80,89] and LLaMA [72,73] are popular architectures. WebAgent [17] combines Flan-U-PaLM with HTML-T5, a finetuned version of Long-T5-base. HTML-T5 reads user instructions together with HTML code of user interface and navigation history to produce a summary of the user interface and a plan for completing tasks specified in the user instruction, which would then be processed by the Flan-U-PaLM instance that generates executable Python code to execute user instructions.

**Existing MLLMs.** LLMs are capable of handling OS tasks, while an inescapable shortcoming of LLMs is that LLMs can only process textual input, while GUI are designed for human users that directly perceive vision information to operate the apps. For this, MLLMs, which additionally have the ability to process vision information while preserving the ability for complex natural language processing, are introduced. Various works [21,71,82] have shown that architectures of existing MLLMs such as LLaVA [19,20], Qwen-VL [76,77,84], InternVL [22,90], CogVLM [26,91], etc., can be effective for developing foundation models for OS Agents.

**Concatenated MLLMs.** Typical architecture of MLLMs consists of an LLM and a vision encoder connected by an adapter network or a cross-attention module. Several works [83,86] have shown that choosing LLMs and vision encoders that are suitable to process OS tasks and concatenating them in a way that is similar to that of existing MLLMs' could be a more suitable approach for constructing foundation models for OS Agents. For instance, Furuta et al. [88] and Thil et al. [79] chose T5 as the LLM in the structure, whose encoder-decoder architecture can better fit tree-architecture of HTML, enabling the model to better process GUI information by perceiving both text and image forms of the GUI.

**Modified MLLMs.** Further adjustments have been adopted to architectures of MLLMs to enhance understanding abilities of foundation models. For instance, most existing MLLMs can only process images of relatively low resolutions, typically 224×224, while common resolution of GUI screenshots is 720×1080. Resizing screenshots to fit the resolution vision encoders of MLLMs preserves features of general layout and most objects, but text and small icons cannot be well perceived, which sometimes would be vital for MLLMs to accomplish OS tasks. Some works have been proposed to enable MLLMs to perceive these features. CogAgent [54] introduced additional EVA-CLIP-L high-resolution vision encoder that accepts images of size 1120×1120, and added a cross-attention module to connect with the original MLLM. Ferret-UI [18] applied the idea of any-resolution, where screenshot images are both resized to fit the vision encoder and partitioned into sub-images, enabling the model to perceive and process visual features in all granularities. MobileFlow [53] chose Qwen-VL as the backbone with a GUI encoder (LayoutLMv3) added to the original architecture, which extracts embeddings of both images and OCR texts together with their positions. UI-Hawk [75] uses a vision encoder that applies a shape-adaptive cropping strategy to perceive details in the screenshot.

#### 3.1.2. Pre-Training

Pre-training [92–94] lays the foundation for model construction and is extensively employed to enhance the foundation models for OS Agents by expanding their understanding of GUI and facilitating the acquisition of the inherent correlations between visual and textual information. To achieve this, most existing pre-training approaches utilize continual pre-training from general pre-trained models with substantial textual or visual comprehension capabilities. This strategy leverages the established knowledge within these pre-trained models, thereby enhancing their performance on GUI-related tasks. One exception is Gur et al. [17], who trained their model from scratch, focusing specifically on parsing HTML text without incorporating the visual modality. To provide a comprehensive overview of their impact on the development of foundation models for OS Agents, data sources and tasks in pre-training will be discussed in the following.

 doi:10.20944/preprints202412.2294.v1



**Data source.** (1) *Publicly available data.* Some studies leverage publicly available datasets to quickly obtain large-scale data for pre-training. Specifically, Gur et al. [17] crawled and filtered web data to extract GUI-related information. Gur et al. [17] utilized CommonCrawl to acquire HTML documents, removing those with non-unicode or purely alphanumeric content, and extracted subtrees around '<label>' elements to train HTML-T5, a model capable of providing executable instructions. Similarly, Nong et al. [53] employed Flickr30K for modality alignment, enhancing the model's semantic understanding of images. However, relying solely on publicly available data for pre-training is insufficient to address the complex and diverse tasks required by OS Agents [19]. Consequently, (2) *Synthetic data.* Researchers incorporate synthetic data into the pre-training process, inspired by the real-world application scenarios of OS Agents. Cheng et al. [84] extracts visible text element positions and instructions to build grounding[3] and OCR task data based on HTML data obtained from the web, while Chen et al. [95] rendered entire websites after acquiring webpage links, segmented them into 1920×1080 resolution screenshots, and extracted features, thereby enriching the diversity of web data. Some studies [22] have noted that although similarities exist between different GUI platforms, pre-training solely based on web data struggles to generalize across platforms. To address this, they created multiple simulated environments and utilized accessibility (A11y) trees to simulate human-computer interaction, sampling cross-platform grounding data. Additionally, Wu et al. [74] proposed a data collection algorithm that simulates human interaction with smartphones by iteratively interacting with every element on each GUI page. This process represents the results as directed graphs and yielded a dataset containing over 3 million real GUI interaction samples.

**Task.** (1) *Screen grounding.* Many studies have demonstrated that pre-training enables models to extract 2D coordinates or bounding boxes of target elements from images based on textual descriptions [22,54,69,71,74,75,82,95]. In addition, Lin et al. [69], Cheng et al. [84] extended text-based grounding tasks by incorporating requirements for predicting text from center point coordinates and bounding boxes into the pre-training stage. (2) *Screen understanding.* Several studies posit that the foundation models for OS Agents should be capable of extracting semantic information from images, as well as analyzing and interpreting the entire content of the image. Wu et al. [22] emphasized that pre-training should equip MLLMs with the knowledge to understand GUI screenshots and identify elements on the screen. Furthermore, Zhang et al. [75], Baechler et al. [82] proposed screen question-answering as a task, where the former designed datasets targeting tasks involving counting, arithmetic operations, and interpreting complex data in charts. (3) *Optical Character Recognition (OCR).* OCR plays a crucial role in handling GUI elements that contain textual content. Hong et al. [54] constructed training data during the pre-training stage by using Paddle-OCR to extract text and bounding boxes from GUI screenshots, and validated the model's superior OCR capabilities on the TextVQA benchmark. Lin et al. [69] identified the capabilities of OCR as a critical evaluation criterion for constructing foundation models.

#### 3.1.3. Supervised Finetuning

Supervised Finetuning (SFT) has been widely adopted to enhance the planning and grounding capabilities of OS Agents. This requires efforts to collect domain-specific data to bridge the domain gap between tasks on natural images and GUIs [54], which is thus the key challenge herein.

For planning, researchers first collect multi-step trajectories and synthesize instructions for them. Gao et al. [90] traverse across the apps with fixed rules as well as LLMs, where the latter are applied to handle certain predefined scenarios and cases that fixed rules fail to cover. Ou et al. [73] uses online tutorial articles to build trajectories, where descriptions of steps are mapped into agent actions with LLMs. Chen et al. [96] builds directed graphs about navigation among webpages and finds the shortest path in the graph to obtain trajectories when generating data for certain tasks. These

[3] Given the varying interpretations of 'grounding' across different domains, in this subsubsection, the term 'grounding' specifically refers to visual grounding, which is the process of locating objects or regions in an image based on a natural language query. This definition differs from the one used in §2.2.

 doi:10.20944/preprints202412.2294.v1



trajectories are taken into advanced large language models, such as GPT4, to synthesize corresponding task instructions [18,54] as well as Chain-of-Thought reasoning process to decompose the tasks [52].

To synthesize data for grounding ability, researchers first connect the actions on the objects to GUI images and then synthesize instructions referring to them. Common strategies to draw the connections are rendering the source codes of GUIs. For example, Gou et al. [19], Chen et al. [21], Liu et al. [70], Kil et al. [83] render webpages with HTML and You et al. [18], Wu et al. [22], Baechler et al. [82], Gao et al. [90] leverage desktop or mobile simulators. A few attempts also leverage GUI detection models [18,26]. Compared to simply learning to operate on the source code, learning to operate with their visual form can show superior performance with the straightforward interaction between widgets [83]. Meanwhile, Meng et al. [20] shows learning with GUI images helps avoid hallucination and Liu et al. [70] demonstrates generalization to unseen GUIs. Then, to synthesize instruction referring to the widgets, Gou et al. [19] summarizes three typical expressions, namely referring to their salient visual features, locations, or functions. Notably, different GUIs may involve different action spaces, Wu et al. [22] find it necessary to adapt action sequences from different sources to a unified action space so as to avoid conflict among them during fine-tuning.

#### 3.1.4. Reinforcement Learning

Reinforcement learning (RL) [97] is a machine learning paradigm where agents learn optimal decision-making through interactions with an environment. By receiving feedback in the form of rewards, the agent iteratively refines its strategies to maximize cumulative rewards.

Early attempts [38,98–101] utilized RL to train agents to accomplish tasks on web and mobile Apps. We introduce several representative works as follows. Yao et al. [39] introduced WebShop, a simulated e-commerce website environment, based on which they trained and evaluated a diverse range of agents using reinforcement learning, imitation learning, and pre-trained multimodal models. The reward is determined by how closely the purchased product matches the specific attributes and options mentioned in the user instructions. Reinforcement learning is typically combined with behavior cloning or supervised fine-tuning to enhance performance. For example, Humphreys et al. [102] developed a scalable method using reinforcement learning and behavioral priors from human-computer interactions to control computers via keyboard and mouse, achieving human-level performance in the MiniWob++ benchmark. Zhang et al. [86] developed a multimodal model for automating GUI tasks by grounding natural language instructions to GUI screenshots, using a pre-trained visual encoder and language decoder, with RL to enhance spatial decoding by supervising token sequences with visually semantic metrics.

In the above RL-based works, large models generally function as feature extractors. More recently, research has progressed to the "LLMs as agents" paradigm, where LLMs serve as policy models and reinforcement learning is applied to align the large models with the final objectives. Thil et al. [79] improved web navigation in LLMs using the Miniwob++ benchmark by fine-tuning T5-based models with hierarchical planning and then integrating these with a multimodal neural network, utilizing both supervised and reinforcement learning. Fereidouni et al. [80] employs the Flan-T5 architecture and introduce training via Reinforcement Learning. They leveraged human demonstrations through behavior cloning and then further trained the agent with PPO. Liu et al. [14] followed the paradigm of LLMs as agents and proposed AutoGLM, foundation agents for autonomous control of computing devices through GUIs. They designed an intermediate interface that effectively disentangles planning and grounding behaviors, and developed a self-evolving online curriculum RL approach that enables robust error recovery and performance improvement. FengPeiyuan et al. [103] introduced a novel RL framework for LLM-based Agents, AGILE, integrating LLMs, memory, tools, and executor modules. RL enables LLMs to predict actions and the executor to manage them, enhancing decision-making and interactions. Reinforcement learning is also introduced to the agents based on vision-only models [104] and MLLMs [105,106].

 doi:10.20944/preprints202412.2294.v1



### 3.2. Agent Framework

OS Agent frameworks typically consist of four core components: Perception, Planning, Memory, and Action. The perception module collects and analyzes environmental information; the planning module handles task decomposition and action sequence generation; the memory module supports information storage and experience accumulation; and the action module executes specific operation instructions. As illustrated in Figure 4, these components work together to enable OS Agents to understand, plan, remember, and interact with operating systems. Table 2 summarizes the technical characteristics of recent OS Agent frameworks, including their specific implementations across these four core components.

**Table 2.** Recent agent frameworks for OS Agents. TD: Textual Description, GS: GUI Screenshots, VG: Visual Grounding, SG: Semantic Grounding, DG: Dual Grounding, GL: Global, IT: Iterative, AE: Automated Exploration, EA: Experience-Augmented, MA: Management, IO: Input Operations, NO: Navigation Operations, EO: Extended Operations.

| Agent | Perception | Planning | Memory | Action | Date |
|---|---|---|---|---|---|
| OpenWebVoyager [50] | GS, SG | - | - | IO, NO | 10/2024 |
| OSCAR [28] | GS, DG | IT | AE | EO | 10/2024 |
| PUMA [107] | TD | - | - | IO, NO, EO | 10/2024 |
| AgentOccam [108] | TD | IT | MA | IO, NO | 10/2024 |
| Agent S [109] | GS, SG | GL | EA, AE, MA | IO, NO | 10/2024 |
| ClickAgent [45] | GS | IT | AE | IO, NO | 10/2024 |
| LSFS [110] | GS, SG | - | - | EO | 09/2024 |
| NaviQAte [111] | GS, SG | - | - | IO | 09/2024 |
| PeriGuru [46] | GS, DG | IT | EA, AE | IO, NO | 09/2024 |
| OpenWebAgent [87] | GS, DG | - | - | IO | 08/2024 |
| LLMCI [112] | GS, SG | - | - | EO | 07/2024 |
| Agent-E [113] | TD | IT | AE, MA | IO, NO | 07/2024 |
| Cradle [114] | GS | IT | EA, AE, MA | EO | 03/2024 |
| CoAT [26] | GS | IT | - | IO, NO | 03/2024 |
| Self-MAP [115] | - | IT | EA | IO | 02/2024 |
| OS-Copilot [29] | TD | GL | EA, AE | IO, EO | 02/2024 |
| Mobile-Agent [116] | GS, SG | IT | AE | IO, NO | 01/2024 |
| WebVoyager [27] | GS, VG | IT | MA | IO, NO | 01/2024 |
| AIA [117] | GS, VG | GL | - | IO, NO | 01/2024 |
| SeeAct [47] | GS, SG | - | AE | IO | 01/2024 |
| AppAgent [23] | GS, DG | IT | AE | IO, NO | 12/2023 |
| ACE [30] | TD | GL | AE | IO, NO | 12/2023 |
| MobileGPT [118] | TD | GL | MA | IO, NO | 12/2023 |
| MM-Navigator [24] | GS, VG | - | MA | IO, NO | 11/2023 |
| WebWise [119] | TD | - | MA | IO, NO | 10/2023 |
| Li et al. [120] | TD | IT | AE | IO, NO | 10/2023 |
| Laser [25] | TD | IT | AE | IO, NO | 09/2023 |
| Synapse [121] | - | - | MA | IO | 06/2023 |
| SheetCopilot [122] | TD | IT | AE | EO | 05/2023 |
| RCI [58] | - | IT | AE | IO, NO | 03/2023 |
| Wang et al. [123] | TD | - | - | IO | 09/2022 |

 doi:10.20944/preprints202412.2294.v1



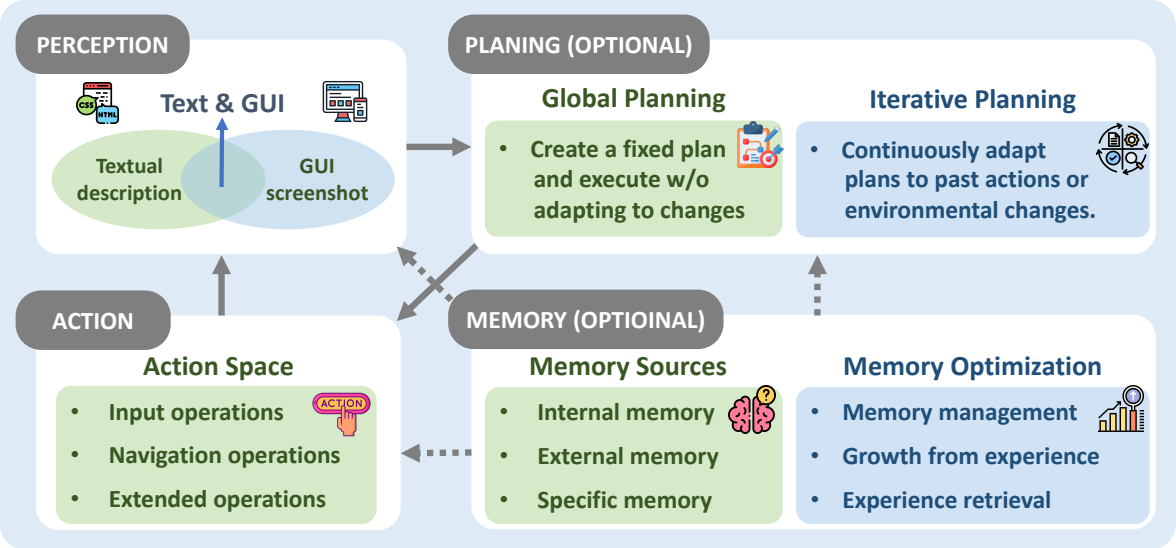


**Figure 4.** Summary of the content about agent frameworks for OS Agents in §3.2.

3.2.1. Perception

Perception is the process through which OS Agents collect and analyze information from their environment. In OS Agents, the perception component needs to observe the current environment and extract relevant information to assist with the agents' planning, action, and memory optimization. Perception can be broadly categorized into two types based on the input modality as follows:

**Textual Description of OS.** Early works [25,29,30,118,122–124] are limited by the fact that LLMs could only process textual input. Therefore, they mainly rely on using tools to convert OS states into text descriptions.

To facilitate LLMs' understanding, these text descriptions are often represented in a structured format, such as HTML, DOM, or accessibility tree. For instance, MobileGPT [118] converts mobile screens into a simplified HTML representation to help LLMs' comprehension. However, these approaches may generate irrelevant or redundant information, which can negatively impact the OS Agents' judgment of the environment and lead to incorrect actions. Therefore, some new approaches have been proposed to filter out invalid descriptions, ensuring that OS Agents only observe relevant information. For example, Agent-E [113] introduces a flexible DOM distillation approach that allows the agent to choose the most suitable DOM representation from three different implementations based on the specific task at hand. Li et al. [120] only expands the HTML representation when the agent takes action, compelling it to make rational decisions with limited information. WebWise [119] introduces a filtering function *filterDOM* to select relevant DOM elements based on predefined "tags" and "classes," filtering out unnecessary items.

**GUI Screenshot.** The emergence of MLLMs enables OS Agents to process visual inputs. Research is increasingly treating GUI screenshots as the perception input for OS Agents, which better aligns with human behavior. However, most existing vision encoders of OS Agents are pre-trained on general data, which makes OS Agents less sensitive to GUI elements.To enhance OS Agents' understanding and grounding ability without fine-tuning visual encoders, existing research focuses on using prompting techniques to describe GUI screenshots. These descriptionscan generally be categorized into three types: (1) *Visual description.* Most research [24,116] uses SoM prompting [125] to enhance OS Agents' visual grounding ability. They incorporate techniques like OCR and GUI element detection algorithms such as ICONNet [126] and Grounding DINO [127] to extract bounding boxes of interactive elements, which are then integrated into corresponding image regions to enhance agents' understanding of GUI screenshots. (2) *Semantic descriptiong.* Some studies improve OS Agents' semantic grounding ability by adding descriptions of these interactive elements. Specifically, SeeAct [47] enhances semantic grounding by using the HTML document of a website as the semantic reference for the GUI screenshot, thereby linking the visual elements with their corresponding semantic meaning in the HTML structure.

 doi:10.20944/preprints202412.2294.v1



(3) *Dual grounding.* Dual grounding combines both visual and semantic information to improve OS Agents' understanding of the visual environment. For instance, AppAgent [23] inputs a labeled screenshot along with an XML file that details the interactive elements to enhance agent understanding. OSCAR [28] introduces a dual-grounding observation approach, using a Windows API-generated A11Y tree for GUI component representation and adding descriptive labels for semantic grounding. PeriGuru [46] inputs a labeled screenshot and a detailed description generated through element and layout recognition. DUAL-VCR [83] employs a Dual-View Contextualized Representation approach, extracting visual features using the Pix2Struct Vision Transformer [128] and aligning each element with corresponding "HTML text" following MindAct [129] for semantic grounding.

#### 3.2.2. Planning

Planning is the process of developing a sequence of actions to achieve a specific goal based on the current environment [55–57]. It enables OS Agents to break down complex tasks into smaller, manageable sub-tasks and solve them step by step. Unlike general agents, the environment of OS Agents is constantly evolving. For instance, dynamic web pages change over time, and GUIs also adapt after each action is executed. Therefore, feasible planning is crucial for OS Agents to effectively cope with these ongoing environmental changes. We categorize existing studies into two key approaches based on whether the planning is fixed or iterates in response to environmental changes: global planning and iterative planning, detailed as follows:

**Global**. OS Agents only generate a global plan once and execute it without making adjustments based on environmental changes. Chain-of-Thought (CoT) [130] prompts (M)LLMs to break down complex tasks into reasoning steps, which forms the foundation of global planning in most OS Agents [46]. Due to the one-time nature of global planning, research on global planning focuses on fitting the OS Agents' environment and tasks, proposing sufficiently feasible plans from the outset. For example, OS-Copilot [29] leverages LLMs to formalize the global plan into a directed acyclic graph, enabling parallel execution of independent sub-tasks, which minimizes execution time and improves efficiency. ACE [30] prompts LLMs to refine extracted steps in alignment with user queries. Agent S [109] proposes experience-augmented hierarchical planning, where plans are informed by integrating knowledge from memory and online sources. Similarly, AIA [117] utilizes Standard Operating Procedures (SOP) to break down complex tasks into manageable sub-tasks.

**Iterative.** In contrast to global planning, iterative planning allows OS Agents to continuously iterate their plans based on historical actions or changes in the environment, enabling them to adapt to ongoing environmental changes. This methodology is crucial for OS Agents to handle dynamic and unpredictable environments effectively. In specific, ReAct [59] builds on the concept of CoT by integrating reasoning with the outcome of actions, making planning more adaptable to changes in the environment. This approach has been widely applied in OS Agents [23,25,27,45,116] for iterative planning. Reflexion [131] builds upon ReAct by allowing access to previous actions and states, which enhances strategic planning of OS Agents in complex, time-sensitive scenarios [46,113,114]. In addition to these general iterative planning methods, some studies have proposed iterative planning approaches specifically tailored for OS Agents. For instance, Auto-GUI [44] employs a CoT technique, where a history of past actions is used to generate future plans iteratively after each step. OSCAR [28] introduces task-driven replanning, allowing the OS Agent to modify its plan based on real-time feedback from the environment. SheetCopilot [122] employs State Machine-based Task Planning, where proposed plans are revised using either a feedback-based mechanism or a retrieval-based approach, enhancing the OS Agent's ability to adapt to dynamic environments. RCI [58] prompts LLMs to find problems in their output and improve the output based on what they find, assisting the OS Agent in refining its reasoning process, which leads to more effective and accurate planning. CoAT [26] introduces a more complex and OS Agent-targeted reasoning method compared to ReAct. It prompts the LLMs to perform a reasoning process involving Screen Description, Action Thinking, and Next Action Description, ultimately leading to an Action Result.

 doi:10.20944/preprints202412.2294.v1



3.2.3. Memory

As the complexity of automated tasks in operating systems continues to increase, enhancing the intelligence and execution efficiency of OS Agents has become a key research focus. Among these studies, the memory module serves as one of the core components. Using memory effectively, OS Agents can continuously optimize their performance during task execution, adapt to dynamic environments, and perform tasks in various complex scenarios. In this section, we discuss current research advancements related to memory in OS Agents.

**Memory Sources.** Memory can be categorized into Internal Memory, External Memory, and Specific Memory, each serving distinct functions in task execution: immediate information storage, external knowledge support, and operation optimization, respectively. In recent years, research has increasingly focused on improving memory adaptability and diversity to meet the demands of more complex tasks [115,132–135]. For example, the introduction of dynamic memory management mechanisms optimizes memory retrieval and updates, while the integration of multimodal approaches further broadens the types and scope of memory data, enabling agents to access more diverse information sources when handling complex scenarios.

- **Internal Memory**. In the following, we introduce several components of Internal Memory. (1) *Action History.* By recording each step of operations, the action history helps OS Agents track task paths and optimize decisions. For instance, Auto-GUI [44] integrates historical and future action plans through the chain of previous action histories. (2) *Screenshots.* The storage of screenshots supports visual reasoning and the recognition of GUI components. For example, CoAT [26] semantically processes screenshots to extract interface information, enabling better understanding of the task scene. Wang and Liu [28], Rawles et al. [136] utilize screenshots annotated with Set-of-Mark (SoM) to support visual reasoning, accurately identify GUI components, and perform precise operations, while also aiding in task planning and validation. ToL [137] uses GUI screenshots as input to construct a Hierarchical Layout Tree and combines visual reasoning to generate descriptions of content and layout. (3) *State Data.* Dynamic information from the environment, such as page positions and window states, are stored to help OS Agents quickly locate task objectives and maintain high task execution accuracy in changing environments. Specifically, CoCo-Agent [138] records layouts and dynamic states through Comprehensive Environment Perception (CEP), while Abuelsaad et al. [113], Tao et al. [119] employ Document Object Model denoising techniques to dynamically store page information. In the following, we present the two forms of internal memory.
  Short-term Memory stores immediate information about the current task, including the action history of the agent, state information, and the execution trajectory of the task. It supports decision optimization and task tracking, providing contextual support for the ongoing task. Recent advances focus on improving the memory capabilities of OS Agents. For example, understanding the layout of objects in a scene through visual information enables multimodal agents to possess more comprehensive cognitive abilities when handling complex tasks.
  Long-term Memory stores historical tasks and interaction records, such as the execution paths of previous tasks, providing references and reasoning support for future tasks. For example, OS-Copilot [29] stores user preferences and the agent's historical knowledge, such as semantic knowledge and task history, as declarative memory. This is used to make personalized decisions and execute tasks, while dynamically generating new tools or storing task-related skill codes during task execution [114].
- **External Memory.** External memory provides long-term knowledge support, primarily enriching an agent's memory capabilities through knowledge bases, external documents, and online information. For instance, agents can retrieve domain-specific background information from external knowledge bases to make more informed judgments in tasks requiring domain expertise. Additionally, some agents dynamically acquire external knowledge by invoking tools such as Ap-

 doi:10.20944/preprints202412.2294.v1



plication Programming Interfaces (APIs) [49,139], integrating this knowledge into their memory to assist with task execution and decision optimization.

- **Specific Memory.** Specific memory focuses on storing information directly related to specific tasks and user needs while incorporating extensive task knowledge and optimized application functions, which can be stored internally or extended through external data sources [140]. Specific Memory can store task execution rules, subtask decomposition methods, and domain knowledge [116]. It provides agents with prior knowledge to assist in handling complex tasks. For instance, MobileGPT [118] adopts a three-tier hierarchical memory structure (task, sub-task, action) and organizes memory in the form of a transition graph, breaking tasks down into sub-tasks represented as function calls for quick access and efficient invocation, while CoCo-Agent [138] employs task decomposition and Conditional Action Prediction (CAP) to store execution rules and methods. In terms of interface element recognition and interaction, Wang and Liu [28], He et al. [50], Agashe et al. [109] enhance task understanding by parsing the Accessibility Tree to obtain information about all UI elements on the screen.
  Additionally, Specific Memory can also be used to record user profiles, preferences, and interaction histories to support personalized recommendations, demand prediction, and inference of implicit information. For example, OS-Copilot [29] records user preferences through user profiles, such as tool usage habits and music or video preferences, enabling personalized solutions and recommendation services. Moreover, Specific Memory also supports recording application function descriptions and page access history to facilitate cross-application operation optimization and historical task tracking. For instance, AppAgent [23] learns application functionality by recording operation histories and state changes, storing this information as documentation. Similarly, ClickAgent [45] improves understanding and operational efficiency in application environments by using GUI localization models to identify and locate GUI elements within applications, while also recording functionality descriptions and historical task information.

**Memory Optimization.** Memory optimization can enhance an agent's efficiency in operations and decision-making during complex tasks by effectively managing and utilizing memory resources. In the following, we introduce several key strategies.

- **Management.** For humans, memory information is constantly processed and abstracted in the brain. Similarly, the memory of OS Agents can be effectively managed to generate higher-level information, consolidate redundant content, and remove irrelevant or outdated information. Effective memory management enhances overall performance and prevents efficiency loss caused by information overload. In specific, Yan et al. [24], Tan et al. [114] introduce a multimodal self-summarization mechanism, generating concise historical records in natural language to replace directly storing complete screens or action sequences. WebAgent [17] understands and summarizes long HTML documents through local and global attention mechanisms, as well as long-span denoising objectives. On the other hand, WebVoyager [27] employs a Context Clipping method, retaining the most recent three observations while keeping a complete record of thoughts and actions from the history. However, for longer tasks, this approach may lead to the loss of important information, potentially affecting task completion. Additionally, Agent-E [113] optimizes webpage representations by filtering task-relevant content, compressing DOM structure hierarchies, and retaining key parent-child relationships, thereby reducing redundancy. AGENTOCCAM [108] optimizes the agent's workflow memory through a planning tree, treating each new plan as an independent goal and removing historical step information related to previous plans.
- **Growth Experience.** By revisiting each step of a task, the agent can analyze successes and failures, identify opportunities for improvement, and avoid repeating mistakes in similar scenarios [58]. For instance, MobA [140] introduces dual reflection, evaluating task feasibility before execution and reviewing completion status afterward. Additionally, In [120], the agent analyzes the sequence of actions after a task failure, identifies the earliest critical missteps, and generates structured

 doi:10.20944/preprints202412.2294.v1



recommendations for alternative actions. OS Agents can return to a previous state and choose an alternative path when the current task path proves infeasible or the results do not meet expectations, which is akin to classic search algorithms, enabling the agent to explore multiple potential solutions and find the optimal path. For example, LASER [25] uses a Memory Buffer mechanism to store intermediate results that were not selected during exploration, allowing the agent to backtrack flexibly within the state space. After taking an incorrect action, the agent can return to a previous state and retry. SheetCopilot [122] utilizes a state machine mechanism to guide the model in re-planning actions by providing error feedback and spreadsheet state feedback, while MobA [140] uses a tree-like task structure to record the complete path, ensuring an efficient backtracking process.

- **Experience Retrieval.** OS Agents can efficiently plan and execute by retrieving experiences similar to the current task from long-term memory, which helps to reduce redundant operations [115,121]. For instance, AWM [133] extracts similar task workflows from past tasks and reuses them in new tasks, minimizing the need for repetitive learning. Additionally, PeriGuru [46] uses the K-Nearest Neighbors algorithm to retrieve similar task cases from a task database and combines them with Historical Actions to enhance decision-making through prompts.

#### 3.2.4. Action

The action space defines the interfaces through which (M)LLM-based Agents engage with operating systems, spanning across platforms such as computers, mobile devices, and web browsers. We systematically categorized the action space of OS Agents into input operations, navigation operations, and extended operations.

**Input Operations.** Input operations encompass interactions via mouse/touch and keyboard, forming the foundation for OS Agents to interact with digital interfaces.

Mouse and touch operations encompass three primary types: (1) *click/tap* actions that are universally implemented across different platforms and serve as the most basic form of interaction [48,121,129], (2) *long press/hold* actions that are particularly crucial for mobile interfaces and context menu activation [23,34,46], and (3) *drag/move* operations that enable precise control and manipulation of interface elements [30,141,142].

Keyboard operations comprise two main categories: (1) *basic text input* capabilities that allow agents to enter alphanumeric characters and symbols [44,48,129], and (2) *special key* operations (e.g., shortcuts, function keys) [30,31,48] that enable agents to efficiently navigate and manipulate target applications through keyboard commands.

**Navigation Operations.** Navigation operations enable OS Agents to traverse targeted platforms and acquire sufficient information for subsequent actions. Navigation operations encompass both basic navigation and web-specific navigation features.

Basic navigation includes: (1) *scroll* operations that enable agents to explore content beyond the current viewport, particularly crucial for processing long documents or infinite-scroll interfaces [24,30,118], (2) *back/forward navigation* that allows agents to traverse through navigation history and return to previously visited states [23,44,48], and (3) *home function* that provides quick access to the initial or default state of applications, ensuring reliable reset points during task execution [23,44,116].

Web navigation extends these capabilities with (1) *tab management* that enables agents to handle multiple concurrent sessions and switches between different web contexts [27,49,143], and (2) *URL navigation* features that allow direct access to specific web resources and facilitate efficient web traversal [25,27,129].

**Extended Operations.** Extended Operations provide additional capabilities beyond standard interface interactions, enabling more flexible and powerful agent behaviors. These operations primarily include (1) *code execution* capabilities that allow agents to dynamically extend their action space beyond predefined operations, enabling flexible and customizable control through direct script execution and command interpretation [29,51,114], and (2) *API integration* features that expand agents' capabilities by accessing external tools and information resources, facilitating interactions with third-party services

 doi:10.20944/preprints202412.2294.v1



and specialized functionalities [29,51,114,122]. These operations fundamentally enhance the adaptability and functionality of OS Agents, allowing them to handle more complex and diverse tasks that may not be achievable through conventional interface-based interactions alone.

## 4. Evaluation of OS Agents

Evaluation plays a crucial role in developing OS Agents, as it helps assess their performance and effectiveness in various scenarios. The current literature features a multitude of evaluation techniques, which vary significantly according to the specific environment and application. For a clear display and summary of the evaluation framework, we will delve into a comprehensive overview of a generic evaluation framework for OS Agents, structured around evaluation protocols and benchmarking. At the same time, we have provided the recent benchmarks for OS Agents in Table 3.

**Table 3.** Recent benchmarks for OS Agents. We divided the Benchmarks into three sections based on the Platform (as mentioned in §4.2.1) and sorted them by release date. The following is an explanation of the abbreviations. BS: Benchmark Settings, M/P: Mobile, PC: Desktop, IT: Interactive, ST: Static, OET: Operation Environment Types, RW: Real-World, SM: Simulated, GG: GUI Grounding, IF: Information Processing, AT: Agentic, CG: Code Generation.

| Benchmark | Platform | BS | OET | Task | Date |
|---|---|---|---|---|---|
| AndroidControl [144] | M/P | ST | - | AT | 06/2024 |
| AndroidWorld [34] | M/P | IT | RW | AT | 05/2024 |
| Android-50 [36] | M/P | IT | RW | AT | 05/2024 |
| B-MoCA [43] | M/P | IT | RW | AT | 04/2024 |
| LlamaTouch [145] | M/P | IT | RW | AT | 04/2024 |
| AndroidArena [33] | M/P | IT | RW | AT | 02/2024 |
| AITW [136] | M/P | ST | - | AT | 07/2023 |
| UGIF-DataSet [33] | M/P | ST | - | AT | 11/2022 |
| MoTIF [146] | M/P | ST | - | AT | 02/2022 |
| PIXELHELP [147] | M/P | IT | RW | GG | 05/2020 |
| WindowsAgentArena [31] | PC | IT | RW | AT | 09/2024 |
| OfficeBench [148] | PC | IT | RW | AT | 07/2024 |
| OSWorld [149] | PC | IT | RW | AT | 04/2024 |
| OmniACT [32] | PC | ST | - | CG | 02/2024 |
| ASSISTGUI [30] | PC | IT | RW | AT | 12/2023 |
| Mind2Web-Live [150] | Web | IT | RW | IF, AT | 06/2024 |
| MMInA [151] | Web | IT | RW | IF, AT | 04/2024 |
| GroundUI [152] | Web | ST | - | GG | 03/2024 |
| TurkingBench [153] | Web | IT | RW | AT | 03/2024 |
| WorkArena [42] | Web | IT | RW | IF, AT | 03/2024 |
| WebLINX [41] | Web | ST | - | IF, AT | 02/2024 |
| Visualwebarena [40] | Web | IT | RW | GG, AT | 01/2024 |
| WebVLN-v1 [96] | Web | IT | RW | IF, AT | 12/2023 |
| WebArena [154] | Web | IT | RW | AT | 07/2023 |
| Mind2Web [129] | Web | ST | - | IF, AT | 06/2023 |
| WebShop [39] | Web | ST | - | AT | 07/2022 |
| PhraseNode [155] | Web | ST | - | GG | 08/2018 |
| MiniWoB [38] | Web | ST | - | AT | 08/2017 |
| FormWoB [38] | Web | IT | SM | AT | 08/2017 |

### *4.1. Evaluation Protocol*

This section is dedicated to outlining the comprehensive evaluation protocols. Central to the assessment of OS Agents are two pivotal concerns: (1) *Evaluation Principles*: how the evaluation process should be conducted, and (2) *Evaluation Metrics*: which aspects need to be assessed. We will now elaborate on the principles and metrics for evaluating OS Agents, focusing on these two issues.

 doi:10.20944/preprints202412.2294.v1



#### 4.1.1. Evaluation Principle

The evaluation of OS Agents requires a combination of multiple aspects and techniques to gain a comprehensive insight into their capabilities and limitations. The assessment process can be primarily divided into objective and subjective evaluations. This integration of objective and subjective evaluation methods not only secures the assessment of performance in controlled environments, but also prioritizes the agent's reliability and practical usability in real-world situations.

**Objective Evaluation.** Objective evaluation primarily measures the performance of OS Agents based on standardized numerical metrics, which are typically rule-based calculations or hardcoded assessments on standard benchmark datasets. This form of evaluation specifically targets the agent's accuracy in perception [156,157], the quality of its generated content [153,158], the effectiveness of its actions [91], and its operational efficiency [43,159]. Typically, the computation of specific metrics encompasses exact match [150,153], fuzzy match [151], and semantic matching for text, elements, and images. Through precise and efficient numerical analysis, objective evaluation enables quick and standardized measurement of the agent's performance.

**Subjective Evaluation.** Besides automated objective assessments, subjective evaluations are also essential. These human-centered subjective evaluations aim to measure how well the output matches human expectations [24,91,150], typically applied in scenarios that require a high level of comprehension and are difficult to quantify using traditional metrics. Such subjective evaluations are based on different subjective aspects, including relevance, coherence, naturalness, harmlessness, and overall quality. Early subjective evaluations were primarily based on direct human assessments [160], which, while yielding high-quality results, are expensive and difficult to reproduce. Later, LLMs were introduced as evaluators to substitute for human judgment [161,162], exploiting their strong instruction-following capabilities. Such LLM-as-a-judge evaluation method [163–165] can offer detailed explanations for annotation, providing a finer-grained understanding of the agent's strengths and weaknesses. Nevertheless, despite the gains in efficiency, there are still limitations regarding its reliability and controllability [19,155,166].

#### 4.1.2. Evaluation Metric

As mentioned in §2.2, the evaluation process of OS Agents mainly examines their abilities in terms of understanding, planning and action grounding. During evaluation, the agent, provided with task instructions and the current environment input, is expected to execute a sequence of continuous actions until the task is accomplished. By collecting the agent's observations, action outputs, and other environmental information during the process, specific metrics can be calculated. Specifically, the evaluation scope includes both granular step-level evaluations and a more holistic task-level assessment. The former focuses on whether each step in the process aligns with the predefined path, while the latter is concerned with whether the agent achieves the goal in the end.

**Step-level Evaluation.** Step-level evaluation centers on a detailed, step-by-step analysis of the planning trajectory, offering a fine-grained evaluation of the actions taken by the agent at each step. In step-level evaluation, the agent's output in response to instruction of each step is directly assessed, with a focus on the accuracy of action grounding and the matching of potential object elements (which refers to the target of the action). For action grounding, the predicted action at each step is typically compared directly with the reference action to obtain operation metrics, such as *operation accuracy and F1* [91,158]. For element matching of actions, different approaches are used depending on the type of action and elements, for example, comparing based on element ID or the element position, leading to *element accuracy and F1* [155]. In the case of specific tasks, such as those involving visual grounding in question-answering, there are dedicated metrics like *BLEU* [158], *ROUGE* [153], and *BERTScore* [167]. By aggregating all the relevant metrics for a single step, it is possible to assess the step's success, thereby obtaining the *step success rate (step SR)* [150]. Despite providing fine-grained comprehension, such step-level evaluation has limitations in assessing the performance of long, continuous action

 doi:10.20944/preprints202412.2294.v1



sequences [40,149,155], and a given task may have various valid paths. To boost the robustness [168] of the evaluation, it is usually necessary to integrate the final task outcome into the assessment.

**Task-level Evaluation.** Task-level evaluation centers on the final output and evaluates whether the agent reaches the desired final state. The two main criteria are task completion and resource utilization. The former assesses whether the agent has successfully fulfilled the assigned tasks as per the instructions, while the latter examines the agent's overall efficiency during task completion.

- **Task Completion Metrics.** Task Completion Metrics measure the effectiveness of OS Agents in successfully accomplishing assigned tasks. These metrics cover several key aspects. *Overall Success Rate (SR)* [38,40,42,44] provides a straightforward measure of the proportion of tasks that are fully completed. *Accuracy* [156,157,168] assesses the precision of the agent's responses or actions, ensuring outputs closely match with the expected outcomes. Additionally, *Reward function* [32,39,40,169] is another critical metric, which assigns numerical values to guide agents toward specific objectives in reinforcement learning.
- **Efficiency Metrics.** Efficiency Metrics evaluate how efficiently the agent completes assigned tasks, considering factors such as step cost, hardware expenses, and time expenditure. Specifically, *Step Ratio* [43,159,170] compares the number of steps taken by the agent to the optimal one (often defined by human performance). A lower step ratio indicates a more efficient and optimized task execution, while higher ratios highlight redundant or unnecessary actions. *API Cost* [168,171,172] evaluates the financial costs associated with API calls, which is particularly relevant for agents that use external language models or cloud services. Furthermore, *Execution Time* [173] measures the time required for the agent to complete a task, and *Peak Memory Allocation* [151] shows the maximum GPU memory usage during computation. These efficiency metrics are critical for evaluating the real-time performance of agents, especially in resource-constrained environments.

### *4.2. Evaluation Benchmark*

To comprehensively evaluate the performance and capabilities of OS Agents, researchers have developed a variety of benchmarks. These benchmarks construct various environments, based on different platforms and settings, and cover a wide range of tasks. This subsection offers a detailed overview of these benchmarks, organized by evaluation platforms, benchmark settings, and tasks.

#### 4.2.1. Evaluation Platform

The platform acts as an integrated evaluation environment, specifically encompassing the virtual settings in which benchmarks are performed. Different platforms present unique challenges and evaluation focuses. Some benchmarks also incorporate multiple platforms at the same time, which places greater demands on the agent's cross-platform transferability. Existing real-world platforms can primarily be categorized into three types: Mobile, Desktop, and Web. Each platform has its unique characteristics and evaluation focuses, which we will elaborate on as follows.

**Mobile.** Mobile platforms such as Android [33,35,36,43] or iOS [24] present unique challenges for OS Agents. While mobile GUI elements are simpler due to smaller screens, they require more complex actions, such as precise gestures for navigating widgets or zooming. The open nature of Android provides a wider action space, encompassing standard GUI interactions and function-calling APIs, such as sending text messages, which imposes higher demands on the agents' planning and action grounding capabilities.

**Desktop.** Desktop platform is more complex due to the diversity of operating systems and applications. Efficient desktop benchmarks [31,148,149] need to handle the wide variety and complexity of real-world computing environments, which span different operating systems, interfaces, and applications. As a result, the scope of manageable tasks and the scalability of testing agents are often constrained.

**Web.** Web platforms are essential interfaces to access online resources. Webpages [38–42] are open and built with HTML, CSS, and JavaScript, making them easy to inspect and modify in real-time.

 doi:10.20944/preprints202412.2294.v1



Since agents interact with the web interface in the same way humans do, it's possible to crowdsource human demonstrations of web tasks from anyone with access to a web browser, keyboard, and mouse, at a low cost. This accessibility has also attracted significant attention from researchers in the field.

#### 4.2.2. Benchmark Setting

Apart from the categorization of platforms, the environmental spaces for OS Agents to percept and take actions vary across different evaluation benchmarks. We have organized the existing benchmark environments, primarily dividing them into **static** and **interactive** categories, with the interactive environments further split into **simulated** and **real-world** settings.

**Static.** Static Environments, which are prevalent in early studies, are often created by caching website copies or static data, thereby establishing an offline context for evaluation. The process of setting up a static environment is quite simple, as it merely involves caching the content from real websites. Evaluations generally rely on the cached static content for tasks such as visual grounding, and only one-step action are supported. MiniWoB [38] is built on simple HTML/CSS/JavaScript pages and employs predefined simulation tasks. Mind2Web [129] captures comprehensive snapshots of each website along with complete interaction traces, enabling seamless offline replay. Owing to the lack of dynamic interaction and environmental feedback, such static evaluations tend to be less authentic and versatile, making them inadequate for a comprehensive assessment.

**Interactive.** Interactive Environments provide a more authentic scenario, characterized by their dynamism and interactivity. In contrast to static environments, OS Agents can execute a sequence of actions, receive feedback from the environment, and make corresponding adjustments. Interactive evaluation settings facilitate the evaluation of an agent's skills in more sophisticated settings. These interactive environments can be subdivided into simulated and real-world types. (1) For the *simulated environment*, FormWoB [38] created a virtual website to avoid the reproducibility issues caused by the dynamic nature of real-world environments, while Rawles et al. [136] developed virtual apps to assess the capabilities of OS Agents. However, these simulated environments are often overly simplistic by excluding unexpected conditions, thus failing to capture the complexity of real-world scenarios. (2) For the *real-world environment*, which is truly authentic and encompasses real websites and apps, one must consider the continuously updating nature of the environment, uncontrollable user behaviors, and diverse device setups. This scenario underscores the requirement for agents to exhibit strong generalization across real-world conditions. OSWorld [149], for example, constructed virtual machines running Windows, Linux, and MacOS to systematically evaluate the performance of OS Agents across different operating systems. Similarly, AndroidWorld [34], conducted tests on real apps using Android emulators, highlighting the importance of evaluating agents under diverse and realistic conditions.

#### 4.2.3. Task

To comprehensively assess the capabilities of OS Agents, a spectrum of specialized tasks has been integrated into the established benchmarks. These tasks span from system-level tasks such as installing and uninstalling applications to daily application such as sending emails and shopping online. These tasks are intended to measure how closely current agents can mimic human performance.

**Task Categorization.** In evaluating OS Agents, task categorization is critical for understanding their capabilities and limitations at a fine-grained level. Based on the capabilities required by the evaluation process, current benchmark tasks can primarily be categorized into three types: **GUI Grounding**, **Information Processing** and **Agentic Tasks**, details of which are described as follows.

- **GUI Grounding.** GUI grounding tasks aim to evaluate agent's abilities to transform instructions to various actionable elements. Grounding is fundamental for interacting with operation systems that OS Agents must possess. Early works, such as PIXELHELP [147], provide a benchmark that pairs English instructions with actions performed by users on a mobile emulator.
- **Information Processing.** In the context of interactive agents, the ability to effectively handle information is a critical component for addressing complex tasks. This encompasses not only

 doi:10.20944/preprints202412.2294.v1



retrieving relevant data from various sources but also summarizing and distilling information to meet specific user needs. Such capabilities are particularly essential in dynamic and diverse environments, where agents must process large volumes of information, and deliver accurate results. To explore these competencies, Information Processing Tasks can be further categorized into two main types: (1) *Information Retrieval Tasks* [42,150,151] examine agent's ability to process complex and dynamic information by understanding instructions and GUI interfaces, extracting the desired information or data. Browsers (either web-based or local applications) are ideal platforms for information retrieval tasks due to their vast repositories of information. Additionally, applications with integrated data services also serve as retrieval platforms. For instance, AndroidWorld [34] requires OS Agents to retrieve scheduled events from Simple Calendar Pro. (2) *Information Summarizing Tasks* are designed to summarize specified information from a GUI interface, testing agent's ability to comprehend and process information. For example, certain tasks in WebLinx [41] focus on summarizing web-based news articles or user reviews.

- **Agentic Tasks.** Agentic tasks are designed to evaluate an agent's core abilities (as mentioned in §2.2) and represent a key focus in current research. In these tasks, OS Agents are provided with an instruction or goal and tasked with identifying the required steps, planning actions, and executing them until the target state is reached, without relying on any explicit navigation guidance. For instance, WebLINX [41] offers both low-level and high-level instructions, challenging agents to complete single-step or multi-step tasks, thereby testing their planning capabilities. Similarly, MMInA [151] emphasizes multi-hop tasks, requiring agents to navigate across multiple websites to fulfill the given instruction.

## 5. Challenge & Future

### *5.1. Safety & Privacy*

A recent report [174] highlighted a notable case where a human player successfully outwitted the Freysa AI agent in a $47,000 crypto challenge, underscoring vulnerabilities even in advanced AI systems and emphasizing the need to address these security risks. This incident aligns with broader concerns as (M)LLMs are increasingly integrated into diverse domains, such as healthcare, education, and autonomous systems, where security has become a critical issue. This growing adoption has led to numerous studies [175–181] investigating the security risks associated with LLMs and their applications. In particular, some research has delved into the challenges faced by OS Agents regarding security risks. The following subsections discuss existing research on the security aspects of OS Agents. §5.1.1 analyzes various attack strategies targeting OS Agents, §5.1.2 explores existing defense mechanisms and limitations, and §5.1.3 reviews existing security benchmarks designed to assess the robustness and reliability of OS Agents.

#### 5.1.1. Attack

Several researchers have investigated adversarial attacks targeting OS Agents. Wu et al. [182] identified a novel threat called Web Indirect Prompt Injection (WIPI), in which adversaries indirectly control LLM-based Web Agents by embedding natural language instructions into web pages. Recent findings [183] further uncovered security risks for MLLMs, illustrating how adversaries can generate adversarial images that cause the captioner to produce adversarial captions, ultimately leading the agents to deviate from the user's intended goals. Similar vulnerabilities have been identified in other studies. Ma et al. [184] introduced an attack method called environmental injection, highlighting that advanced MLLMs are vulnerable to environmental distractions, which can cause agents to perform unfaithful behaviors. Expanding on the concept, Liao et al. [185] executed an environmental injection attack by embedding invisible malicious instructions within web pages, prompting the agents to assist adversaries in stealing users' personal information. Xu et al. [186] further advanced this approach by leveraging malicious instructions generated by an adversarial prompter model, trained

 doi:10.20944/preprints202412.2294.v1



on both successful and failed attack data, to mislead MLLM-based Web Agents into executing targeted adversarial actions.

Other studies have explored security issues in specific environments. Zhang et al. [187] explored adversarial pop-up window attacks on MLLM-based Web Agents, demonstrating how this method interferes with the decision-making process of the agents. Kumar et al. [188] investigated the security of refusal-trained LLMs when deployed as browser agents. Their study found that these models' ability to reject harmful instructions in conversational settings does not effectively transfer to browser-based environments. Moreover, existing attack methods can successfully bypass their security measures, enabling jailbreaking. Yang et al. [189] proposed a security threat matrix for agents running on mobile devices, systematically examining the security issues of MLLM-based Mobile Agents and identifying four realistic attack paths and eight attack methods.

#### 5.1.2. Defense

Although several security frameworks have been developed for LLM-based Agents [190–194], studies on defenses specific to OS Agents [195] remain limited. Bridging this gap requires the development of robust defense mechanisms tailored to the vulnerabilities of OS Agents, such as injection attacks, backdoor exploits, and other potential threats. Future research could prioritize these areas, focusing on developing comprehensive and scalable security solutions for OS Agents.

#### 5.1.3. Benchmark

Several security benchmarks [196,197] have been introduced to evaluate the robustness of OS Agents in various scenarios. The online benchmark ST-WebAgentBench [196] has been developed to systematically assess the safety and trustworthiness of web agents within enterprise environments. It focuses on six key dimensions of reliability, offering a comprehensive framework for evaluating agent behavior in high-risk contexts. Similarly, a benchmarking platform named MobileSafetyBench [197] has been developed to assess the security of LLM-based Mobile Agents, focusing on evaluating their performance in handling safety-critical tasks within Android environments, including interactions with messaging and banking applications.

### *5.2. Personalization & Self-Evolution*

Much like Jarvis as Iron Man's personal assistant in the movies, developing personalized OS Agents has been a long-standing goal in AI research. A personal assistant is expected to continuously adapt and provide enhanced experiences based on individual user preferences. OpenAI's memory feature[4] has made strides in this direction, but many (M)LLMs today still perform insufficient in providing personalized experience to users and self-evolving over user interactions.

Early works [198,199] allowed LLM-based Agents to interact with environments of games, summarizing experiences into text, thus accumulating memory and facilitating self-evolution [200]. For example, Wang et al. [198] demonstrated the potential for agents to adapt and evolve through experience. Later, researchers applied these principles to the OS Agent domain [23,29,201]. These efforts validated the feasibility of memory mechanisms in OS Agents. Although due to the limited resources available in academia and the difficulty of accessing real user data, much of the current research focuses on improving performance for specific tasks rather than personalization. The memory mechanism still shows potential for OS Agents to accumulate user data over time, thus improving user experience and performance.

Moreover, expanding the modalities of memory from text to other forms, such as images, voice, presents significant challenges. Managing and retrieving this memory effectively also remains an open issue. We believe that in the future, overcoming these challenges will enable OS Agents to provide more personalized, dynamic, and context-aware assistance, with more sophisticated self-evolution mechanisms that continually adapt to the user's needs and prefernces.

[4] https://openai.com/index/memory-and-new-controls-for-chatgpt/

 doi:10.20944/preprints202412.2294.v1



## 6. Related Work

(Multimodal) Large Language Models [202–206] have emerged as transformative tools in artificial intelligence, driving significant advancements across various domains. Zhao et al. [207] summarize a foundational overview of LLMs. Yin et al. [208], Zhang et al. [209] comprehensively reviews the progress of Multimodal LLMs. In addtion, Long et al. [210] explores the use of synthetic data for training. Zhang et al. [211] presents the current state of research on the field of instruction tuning for LLMs.

With the flourishing development of (M)LLM-based Agents, numerous comprehensive surveys have emerged, offering detailed insights into various aspects of these systems. Wang et al. [212], Cheng et al. [213], Gan et al. [214] provides an overview of general LLM-based Agents. For the agent frameworks, Zhou et al. [215], Zhang et al. [216], Li et al. [217] explore methods to enhance agents' capabilities of planning, memory and multi-agents interaction. Qiao et al. [218] presents comprehensive comparisons for LLM's reasoning abilities. Hou et al. [219], Hu et al. [220], Li et al. [221] summarizes studies in different application fields including software engineering, game and personal assistance. Some concurrent works [222–226] touch on concepts that share some features with OS Agents, such as personalized agents, GUI Agents and generalist virtual agents. This work aims to provide an integrated view on the construction and evaluation of OS Agents, that leverage environments and interfaces provided by operating systems, while identifying open challenges and future directions in this domain for forthcoming studies.

## 7. Conclusion

The development of (multimodal) large language models has created new opportunities for OS Agents, moving the idea of advanced AI assistants closer to being realized. In this survey, we have aimed to outline the fundamentals underlying OS Agents, including their key components and capabilities. We have also reviewed various approaches to their construction, with particular attention to domain-specific foundation models and agent frameworks. Through the evaluation protocols and benchmarks discussed, we have explored methods for assessing the performance of OS Agents across a variety of tasks. Looking ahead, we identify critical challenges, such as safety and privacy, personalization and self-evolution, as areas that require continued research and attention. This summary of the current state of the field, along with potential directions for future work, is intended to contribute to the ongoing development of OS Agents and support their relevance and utility in both academic and industrial settings.

## References

1. Apple Inc. Siri - apple, 2024. URL https://www.apple.com/siri/. Accessed: 2024-12-04.
2. Microsoft Research. Cortana research - microsoft research, 2024. URL https://www.microsoft.com/en-us/research/group/cortana-research/. Accessed: 2024-12-04.
3. Google. Google assistant, 2024. URL https://assistant.google.com/. Accessed: 2024-12-04.
4. Amazon. Alexa - amazon, 2024. URL https://alexa.amazon.com/. Accessed: 2024-12-04.
5. Amrita S Tulshan and Sudhir Namdeorao Dhage. Survey on virtual assistant: Google assistant, siri, cortana, alexa. In *Advances in Signal Processing and Intelligent Recognition Systems: 4th International Symposium SIRS 2018, Bangalore, India, September 19–22, 2018, Revised Selected Papers 4*, pages 190–201. Springer, 2019.
6. Google. Gemini - google. URL https://gemini.google.com/. Accessed: 2024-12-12.
7. OpenAI. Home - openai. URL https://openai.com/. Accessed: 2024-12-12.
8. xAI. x.ai. URL https://x.ai/. Accessed: 2024-12-12.
9. 01.AI. 01.ai. URL https://www.lingyiwanwu.com/. Accessed: 2024-12-12.
10. Anthropic. Anthropic. URL https://www.anthropic.com/. Accessed: 2024-12-12.
11. Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot arena: An open platform for evaluating llms by human preference, 2024.
12. Anthropic. 3.5 models and computer use - anthropic, 2024. URL https://www.anthropic.com/news/3-5-models-and-computer-use. Accessed: 2024-12-04.

13. Apple. Apple intelligence, 2024. URL https://www.apple.com/apple-intelligence/. Accessed: 2024-12-04.
14. Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, et al. Autoglm: Autonomous foundation agents for guis. *arXiv preprint arXiv:2411.00820*, 2024.
15. Google DeepMind. Project mariner, 2024. URL https://deepmind.google/technologies/project-mariner/. Accessed: 2024-12-04.
16. Anthropic. Claude model - anthropic, 2024. URL https://www.anthropic.com/claude. Accessed: 2024-12-04.
17. Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. *arXiv preprint arXiv:2307.12856*, 2023.
18. Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. Ferret-ui: Grounded mobile ui understanding with multimodal llms. In *European Conference on Computer Vision*, pages 240–255. Springer, 2025.
19. Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for gui agents. *arXiv preprint arXiv:2410.05243*, 2024.
20. Ziyang Meng, Yu Dai, Zezheng Gong, Shaoxiong Guo, Minglong Tang, and Tongquan Wei. Vga: Vision gui assistant–minimizing hallucinations through image-centric fine-tuning. *arXiv preprint arXiv:2406.14056*, 2024.
21. Xuetian Chen, Hangcheng Li, Jiaqing Liang, Sihang Jiang, and Deqing Yang. Edge: Enhanced grounded gui understanding with enriched multi-granularity synthetic data. *arXiv preprint arXiv:2410.19461*, 2024.
22. Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, et al. Os-atlas: A foundation action model for generalist gui agents. *arXiv preprint arXiv:2410.23218*, 2024.
23. Chi Zhang, Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. Appagent: Multimodal agents as smartphone users. *arXiv preprint arXiv:2312.13771*, 2023.
24. An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu Zhong, Julian McAuley, Jianfeng Gao, et al. Gpt-4v in wonderland: Large multimodal models for zero-shot smartphone gui navigation. *arXiv preprint arXiv:2311.07562*, 2023.
25. Kaixin Ma, Hongming Zhang, Hongwei Wang, Xiaoman Pan, Wenhao Yu, and Dong Yu. Laser: Llm agent with state-space exploration for web navigation. *arXiv preprint arXiv:2309.08172*, 2023.
26. Jiwen Zhang, Jihao Wu, Yihua Teng, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. Android in the zoo: Chain-of-action-thought for gui agents. *arXiv preprint arXiv:2403.02713*, 2024.
27. Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models. *arXiv preprint arXiv:2401.13919*, 2024.
28. Xiaoqiang Wang and Bang Liu. Oscar: Operating system control via state-aware reasoning and re-planning. *arXiv preprint arXiv:2410.18963*, 2024.
29. Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. Os-copilot: Towards generalist computer agents with self-improvement. *arXiv preprint arXiv:2402.07456*, 2024.
30. Difei Gao, Lei Ji, Zechen Bai, Mingyu Ouyang, Peiran Li, Dongxing Mao, Qinchen Wu, Weichen Zhang, Peiyi Wang, Xiangwu Guo, et al. Assistgui: Task-oriented desktop graphical user interface automation. *arXiv preprint arXiv:2312.13108*, 2023.
31. Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, et al. Windows agent arena: Evaluating multi-modal os agents at scale. *arXiv preprint arXiv:2409.08264*, 2024.
32. Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem Alshikh, and Ruslan Salakhutdinov. Omniact: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web. *arXiv preprint arXiv:2402.17553*, 2024.
33. Sagar Gubbi Venkatesh, Partha Talukdar, and Srini Narayanan. Ugif: Ui grounded instruction following. *arXiv preprint arXiv:2211.07615*, 2022.

 doi:10.20944/preprints202412.2294.v1




34. Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. Androidworld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*, 2024.
35. Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on computer control agents. *arXiv preprint arXiv:2406.03679*, 2024.
36. William E Bishop, Alice Li, Christopher Rawles, and Oriana Riva. Latent state estimation helps ui agents to reason. *arXiv preprint arXiv:2405.11120*, 2024.
37. Mingzhe Xing, Rongkai Zhang, Hui Xue, Qi Chen, Fan Yang, and Zhen Xiao. Understanding the weakness of large language model agents within a complex android environment. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6061–6072, 2024.
38. Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning*, pages 3135–3144. PMLR, 2017.
39. Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35::0 20744–20757, 2022.
40. Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *arXiv preprint arXiv:2401.13649*, 2024.
41. Xing Han Lù, Zdeněk Kasner, and Siva Reddy. Weblinx: Real-world website navigation with multi-turn dialogue. *arXiv preprint arXiv:2402.05930*, 2024.
42. Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, et al. Workarena: How capable are web agents at solving common knowledge work tasks? *arXiv preprint arXiv:2403.07718*, 2024.
43. Juyong Lee, Taywon Min, Minyong An, Dongyoon Hahm, Haeone Lee, Changyeon Kim, and Kimin Lee. Benchmarking mobile device control agents across diverse configurations. *arXiv preprint arXiv:2404.16660*, 2024.
44. Zhuosheng Zhang and Aston Zhang. You only look at screens: Multimodal chain-of-action agents. *arXiv preprint arXiv:2309.11436*, 2023.
45. Jakub Hoscilowicz, Bartosz Maj, Bartosz Kozakiewicz, Oleksii Tymoshchuk, and Artur Janicki. Clickagent: Enhancing ui location capabilities of autonomous agents. *arXiv preprint arXiv:2410.11872*, 2024.
46. Kelin Fu, Yang Tian, and Kaigui Bian. Periguru: A peripheral robotic mobile app operation assistant based on gui image understanding and prompting with llm. *arXiv preprint arXiv:2409.09354*, 2024.
47. Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v (ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*, 2024.
48. Liangtai Sun, Xingyu Chen, Lu Chen, Tianle Dai, Zichen Zhu, and Kai Yu. Meta-gui: Towards multi-modal conversational agents on mobile gui. *arXiv preprint arXiv:2205.11029*, 2022.
49. Yueqi Song, Frank Xu, Shuyan Zhou, and Graham Neubig. Beyond browsing: Api-based web agents. *arXiv preprint arXiv:2410.16464*, 2024.
50. Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Hongming Zhang, Tianqing Fang, Zhenzhong Lan, and Dong Yu. Openwebvoyager: Building multimodal web agents via iterative real-world exploration, feedback and optimization. *arXiv preprint arXiv:2410.19609*, 2024.
51. Kai Mei, Zelong Li, Shuyuan Xu, Ruosong Ye, Yingqiang Ge, and Yongfeng Zhang. Aios: Llm agent operating system. *arXiv e-prints, pp. arXiv–2403*, 2024.
52. Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, et al. Autowebglm: A large language model-based web navigating agent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5295–5306, 2024.
53. Songqin Nong, Jiali Zhu, Rui Wu, Jiongchao Jin, Shuo Shan, Xiutian Huang, and Wenhao Xu. Mobileflow: A multimodal llm for mobile gui agent. *arXiv preprint arXiv:2407.04346*, 2024.
54. Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14281–14290, 2024.
55. Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey, 2023. URL https://arxiv.org/abs/2212.10403.

 doi:10.20944/preprints202412.2294.v1




56. Yadong Zhang, Shaoguang Mao, Tao Ge, Xun Wang, Adrian de Wynter, Yan Xia, Wenshan Wu, Ting Song, Man Lan, and Furu Wei. Llm as a mastermind: A survey of strategic reasoning with large language models, 2024. URL https://arxiv.org/abs/2404.01230.
57. Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716*, 2024.
58. Geunwoo Kim, Pierre Baldi, and Stephen McAleer. Language models can solve computer tasks. *Advances in Neural Information Processing Systems*, 36, 2024.
59. Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.
60. Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.
61. Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*, 2023.
62. Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Le Hou, Kevin Clark, Stephen Pfohl, Heather Cole-Lewis, Darlene Neal, et al. Towards expert-level medical question answering with large language models. *arXiv preprint arXiv:2305.09617*, 2023.
63. Chaojun Xiao, Xueyu Hu, Zhiyuan Liu, Cunchao Tu, and Maosong Sun. Lawformer: A pre-trained language model for chinese legal long documents. *AI Open*, 2::0 79–84, 2021.
64. Harrison Chase. LangChain, October 2022. URL https://github.com/langchain-ai/langchain.
65. Significant Gravitas. AutoGPT. URL https://github.com/Significant-Gravitas/AutoGPT.
66. Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for a multi-agent collaborative framework, 2024. URL https://arxiv.org/abs/2308.00352.
67. Xueyu Hu, Ziyu Zhao, Shuang Wei, Ziwei Chai, Qianli Ma, Guoyin Wang, Xuwu Wang, Jing Su, Jingjing Xu, Ming Zhu, et al. Infiagent-dabench: Evaluating agents on data analysis tasks. *arXiv preprint arXiv:2401.05507*, 2024.
68. Zhangsheng Li, Keen You, Haotian Zhang, Di Feng, Harsh Agrawal, Xiujun Li, Mohana Prasad Sathya Moorthy, Jeff Nichols, Yinfei Yang, and Zhe Gan. Ferret-ui 2: Mastering universal user interface understanding across platforms. *arXiv preprint arXiv:2410.18967*, 2024.
69. Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for generalist gui agent. In *NeurIPS 2024 Workshop on Open-World Agents*, 2024.
70. Junpeng Liu, Tianyue Ou, Yifan Song, Yuxiao Qu, Wai Lam, Chenyan Xiong, Wenhu Chen, Graham Neubig, and Xiang Yue. Harnessing webpage uis for text-rich visual understanding. *arXiv preprint arXiv:2410.13824*, 2024.
71. Pawel Pawlowski, Krystian Zawistowski, Wojciech Lapacz, Marcin Skorupa, Adam Wiacek, Sebastien Postansque, and Jakub Hoscilowicz. Tinyclick: Single-turn agent for empowering gui automation. *arXiv preprint arXiv:2410.11871*, 2024.
72. Shikhar Murty, Dzmitry Bahdanau, and Christopher D Manning. Nnetscape navigator: Complex demonstrations for web agents without a demonstrator. *arXiv preprint arXiv:2410.02907*, 2024.
73. Tianyue Ou, Frank F Xu, Aman Madaan, Jiarui Liu, Robert Lo, Abishek Sridhar, Sudipta Sengupta, Dan Roth, Graham Neubig, and Shuyan Zhou. Synatra: Turning indirect knowledge into direct demonstrations for digital agents at scale. *arXiv preprint arXiv:2409.15637*, 2024.
74. Qinzhuo Wu, Weikai Xu, Wei Liu, Tao Tan, Jianfeng Liu, Ang Li, Jian Luan, Bin Wang, and Shuo Shang. Mobilevlm: A vision-language model for better intra-and inter-ui understanding. *arXiv preprint arXiv:2409.14818*, 2024.
75. Jiwen Zhang, Yaqi Yu, Minghui Liao, Wentao Li, Jihao Wu, and Zhongyu Wei. Ui-hawk: Unleashing the screen stream understanding for gui agents. *Preprints*, 2024.
76. Qinchen Wu, Difei Gao, Kevin Qinghong Lin, Zhuoyu Wu, Xiangwu Guo, Peiran Li, Weichen Zhang, Hengxu Wang, and Mike Zheng Shou. Gui action narrator: Where and when did that action take place? *arXiv preprint arXiv:2406.13719*, 2024.

 doi:10.20944/preprints202412.2294.v1




77. Quanfeng Lu, Wenqi Shao, Zitao Liu, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, Yu Qiao, and Ping Luo. Gui odyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. *arXiv preprint arXiv:2406.08451*, 2024.
78. Andrea Burns, Kate Saenko, and Bryan A Plummer. Tell me what's next: Textual foresight for generic ui representations. *arXiv preprint arXiv:2406.07822*, 2024.
79. Lucas-Andrei Thil, Mirela Popa, and Gerasimos Spanakis. Navigating webai: Training agents to complete web tasks with large language models and reinforcement learning. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, pages 866–874, 2024.
80. Moghis Fereidouni et al. Search beyond queries: Training smaller language models for web interactions via reinforcement learning. *arXiv preprint arXiv:2404.10887*, 2024.
81. Ajay Patel, Markus Hofmarcher, Claudiu Leoveanu-Condrei, Marius-Constantin Dinu, Chris Callison-Burch, and Sepp Hochreiter. Large language models can self-improve at web agent tasks. *arXiv preprint arXiv:2405.20309*, 2024.
82. Gilles Baechler, Srinivas Sunkara, Maria Wang, Fedir Zubach, Hassan Mansoor, Vincent Etter, Victor Cărbune, Jason Lin, Jindong Chen, and Abhanshu Sharma. Screenai: A vision-language model for ui and infographics understanding. *arXiv preprint arXiv:2402.04615*, 2024.
83. Jihyung Kil, Chan Hee Song, Boyuan Zheng, Xiang Deng, Yu Su, and Wei-Lun Chao. Dual-view visual contextualization for web navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14445–14454, 2024.
84. Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents. *arXiv preprint arXiv:2401.10935*, 2024.
85. Yue Jiang, Eldon Schoop, Amanda Swearngin, and Jeffrey Nichols. Iluvui: Instruction-tuned language-vision modeling of uis from machine conversations. *arXiv preprint arXiv:2310.04869*, 2023.
86. Zhizheng Zhang, Wenxuan Xie, Xiaoyi Zhang, and Yan Lu. Reinforced ui instruction grounding: Towards a generic ui task automation api. *arXiv preprint arXiv:2310.04716*, 2023.
87. Iat Long Iong, Xiao Liu, Yuxuan Chen, Hanyu Lai, Shuntian Yao, Pengbo Shen, Hao Yu, Yuxiao Dong, and Jie Tang. Openwebagent: An open toolkit to enable web agents on large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 72–81, 2024.
88. Hiroki Furuta, Kuang-Huei Lee, Ofir Nachum, Yutaka Matsuo, Aleksandra Faust, Shixiang Shane Gu, and Izzeddin Gur. Multimodal web navigation with instruction-finetuned foundation models. *arXiv preprint arXiv:2305.11854*, 2023.
89. Hiroki Furuta, Yutaka Matsuo, Aleksandra Faust, and Izzeddin Gur. Exposing limitations of language model agents in sequential-task compositions on the web. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024.
90. Longxi Gao, Li Zhang, Shihe Wang, Shangguang Wang, Yuanchun Li, and Mengwei Xu. Mobileviews: A large-scale mobile gui dataset. *arXiv preprint arXiv:2409.14337*, 2024.
91. Yifan Xu, Xiao Liu, Xueqiao Sun, Siyi Cheng, Hao Yu, Hanyu Lai, Shudan Zhang, Dan Zhang, Jie Tang, and Yuxiao Dong. Androidlab: Training and systematic benchmarking of android autonomous agents. *arXiv preprint arXiv:2410.24024*, 2024.
92. Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
93. Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.
94. Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
95. Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, et al. Guicourse: From general vision language models to versatile gui agents. *arXiv preprint arXiv:2406.11317*, 2024.
96. Qi Chen, Dileepa Pitawela, Chongyang Zhao, Gengze Zhou, Hsiang-Ting Chen, and Qi Wu. Webvln: Vision-and-language navigation on websites. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 1165–1173, 2024.
97. Richard S Sutton. Reinforcement learning: An introduction. *A Bradford Book*, 2018.
98. Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ryTp3f-0-.

 doi:10.20944/preprints202412.2294.v1




99. Izzeddin Gur, Ulrich Rueckert, Aleksandra Faust, and Dilek Hakkani-Tur. Learning to navigate the web. *arXiv preprint arXiv:1812.09195*, 2018.
100. Sheng Jia, Jamie Kiros, and Jimmy Ba. Dom-q-net: Grounded rl on structured language. *arXiv preprint arXiv:1902.07257*, 2019.
101. Maayan Shvo, Zhiming Hu, Rodrigo Toro Icarte, Iqbal Mohomed, Allan D Jepson, and Sheila A McIlraith. Appbuddy: Learning to accomplish tasks in mobile apps via reinforcement learning. In *Canadian AI*, 2021.
102. Peter C Humphreys, David Raposo, Tobias Pohlen, Gregory Thornton, Rachita Chhaparia, Alistair Muldal, Josh Abramson, Petko Georgiev, Adam Santoro, and Timothy Lillicrap. A data-driven approach for learning to control computers. In *International Conference on Machine Learning*, pages 9466–9482. PMLR, 2022.
103. FengPeiyuan, Yichen He, Guanhua Huang, Yuan Lin, Hanchong Zhang, Yuchen Zhang, and Hang Li. AGILE: A novel reinforcement learning framework of LLM agents. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=Ul3lDYo3XQ.
104. Peter Shaw, Mandar Joshi, James Cohan, Jonathan Berant, Panupong Pasupat, Hexiang Hu, Urvashi Khandelwal, Kenton Lee, and Kristina N Toutanova. From pixels to ui actions: Learning to follow instructions via graphical user interfaces. *Advances in Neural Information Processing Systems*, 36::0 34354–34370, 2023.
105. Hao Bai, Yifei Zhou, Mert Cemri, Jiayi Pan, Alane Suhr, Sergey Levine, and Aviral Kumar. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. *arXiv preprint arXiv:2406.11896*, 2024.
106. Taiyi Wang, Zhihao Wu, Jianheng Liu, Jianye Hao, Jun Wang, and Kun Shao. Distrl: An asynchronous distributed reinforcement learning framework for on-device control agents. *arXiv preprint arXiv:2410.14803*, 2024.
107. Hongru Cai, Yongqi Li, Wenjie Wang, Fengbin Zhu, Xiaoyu Shen, Wenjie Li, and Tat-Seng Chua. Large language models empowered personalized web agents. *arXiv preprint arXiv:2410.17236*, 2024.
108. Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. Agentoccam: A simple yet strong baseline for llm-based web agents. *arXiv preprint arXiv:2410.13825*, 2024.
109. Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent s: An open agentic framework that uses computers like a human. *arXiv preprint arXiv:2410.08164*, 2024.
110. Zeru Shi, Kai Mei, Mingyu Jin, Yongye Su, Chaoji Zuo, Wenyue Hua, Wujiang Xu, Yujie Ren, Zirui Liu, Mengnan Du, et al. From commands to prompts: Llm-based semantic file system for aios. *arXiv preprint arXiv:2410.11843*, 2024.
111. Mobina Shahbandeh, Parsa Alian, Noor Nashid, and Ali Mesbah. Naviqate: Functionality-guided web application navigation, 2024. URL https://arxiv.org/abs/2409.10741.
112. Husam Barham and Mohammed Fasha. Towards llmci-multimodal ai for llm-vision ui operation. 2024.
113. Tamer Abuelsaad, Deepak Akkil, Prasenjit Dey, Ashish Jagmohan, Aditya Vempaty, and Ravi Kokku. Agent-e: From autonomous web navigation to foundational design principles in agentic systems. *arXiv preprint arXiv:2407.13032*, 2024.
114. Weihao Tan, Wentao Zhang, Xinrun Xu, Haochong Xia, Gang Ding, Boyu Li, Bohan Zhou, Junpeng Yue, Jiechuan Jiang, Yewen Li, et al. Cradle: Empowering foundation agents towards general computer control. In *NeurIPS 2024 Workshop on Open-World Agents*.
115. Yang Deng, Xuan Zhang, Wenxuan Zhang, Yifei Yuan, See-Kiong Ng, and Tat-Seng Chua. On the multi-turn instruction following for conversational web agents. *arXiv preprint arXiv:2402.15057*, 2024.
116. Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. *arXiv preprint arXiv:2401.16158*, 2024.
117. Tinghe Ding. Mobileagent: enhancing mobile control via human-machine interaction and sop integration. *arXiv preprint arXiv:2401.04124*, 2024.
118. Sunjae Lee, Junyoung Choi, Jungjae Lee, Munim Hasan Wasi, Hojun Choi, Steven Y Ko, Sangeun Oh, and Insik Shin. Explore, select, derive, and recall: Augmenting llm with human-like memory for mobile task automation. *arXiv preprint arXiv:2312.03003*, 2023.
119. Heyi Tao, Sethuraman TV, Michal Shlapentokh-Rothman, and Derek Hoiem. Webwise: Web interface control and sequential exploration with large language models. *arXiv preprint arXiv:2310.16042*, 2023.
120. Tao Li, Gang Li, Zhiwei Deng, Bryan Wang, and Yang Li. A zero-shot language agent for computer control with structured reflection. *arXiv preprint arXiv:2310.08740*, 2023.

121. Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *The Twelfth International Conference on Learning Representations*, 2023.
122. Hongxin Li, Jingran Su, Yuntao Chen, Qing Li, and ZHAO-XIANG ZHANG. Sheetcopilot: Bringing software productivity to the next level through large language models. *Advances in Neural Information Processing Systems*, 36, 2024.
123. Bryan Wang, Gang Li, and Yang Li. Enabling conversational interaction with mobile ui using large language models. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–17, 2023.
124. Junting Lu, Zhiyang Zhang, Fangkai Yang, Jue Zhang, Lu Wang, Chao Du, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. Turn every application into an agent: Towards efficient human-agent-computer interaction with api-first llm-based agents. *arXiv preprint arXiv:2409.17140*, 2024.
125. Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v, 2023. URL https://arxiv.org/abs/2310.11441.
126. Srinivas Sunkara, Maria Wang, Lijuan Liu, Gilles Baechler, Yu-Chung Hsiao, Abhanshu Sharma, James Stout, et al. Towards better semantic understanding of mobile interfaces. *arXiv preprint arXiv:2210.02663*, 2022.
127. Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding dino: Marrying dino with grounded pre-training for open-set object detection, 2024. URL https://arxiv.org/abs/2303.05499.
128. Kenton Lee, Mandar Joshi, Iulia Raluca Turc, Hexiang Hu, Fangyu Liu, Julian Martin Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. Pix2struct: Screenshot parsing as pretraining for visual language understanding. In *International Conference on Machine Learning*, pages 18893–18912. PMLR, 2023.
129. Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36, 2024.
130. Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.
131. Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.
132. Wangchunshu Zhou, Yuchen Eleanor Jiang, Peng Cui, Tiannan Wang, Zhenxin Xiao, Yifan Hou, Ryan Cotterell, and Mrinmaya Sachan. Recurrentgpt: Interactive generation of (arbitrarily) long text, 2023. URL https://arxiv.org/abs/2305.13304.
133. Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024.
134. Tian Huang, Chun Yu, Weinan Shi, Zijian Peng, David Yang, Weiqi Sun, and Yuanchun Shi. Promptrpa: Generating robotic process automation on smartphones from textual prompts. *arXiv preprint arXiv:2404.02475*, 2024.
135. Jaekyeom Kim, Dong-Ki Kim, Lajanugen Logeswaran, Sungryull Sohn, and Honglak Lee. Auto-intent: Automated intent discovery and self-exploration for large language model web agents. *arXiv preprint arXiv:2410.22552*, 2024.
136. Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Androidinthewild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems*, 36, 2024.
137. Read Anywhere Pointed. Layout-aware gui screen reading with tree-of-lens grounding.
138. Xinbei Ma, Zhuosheng Zhang, and Hai Zhao. Coco-agent: A comprehensive cognitive mllm agent for smartphone gui automation. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 9097–9110, 2024.
139. Revanth Gangi Reddy, Sagnik Mukherjee, Jeonghwan Kim, Zhenhailong Wang, Dilek Hakkani-Tur, and Heng Ji. Infogent: An agent-based framework for web information aggregation. *arXiv preprint arXiv:2410.19054*, 2024.
140. Zichen Zhu, Hao Tang, Yansi Li, Kunyao Lan, Yixuan Jiang, Hao Zhou, Yixiao Wang, Situo Zhang, Liangtai Sun, Lu Chen, et al. Moba: A two-level agent system for efficient mobile task automation. *arXiv preprint arXiv:2410.13757*, 2024.
141. Runliang Niu, Jindong Li, Shiqi Wang, Yali Fu, Xiyu Hu, Xueyuan Leng, He Kong, Yi Chang, and Qi Wang. Screenagent: A vision language model-driven computer control agent. *arXiv preprint arXiv:2402.07945*, 2024.

 doi:10.20944/preprints202412.2294.v1




142. Junhee Cho, Jihoon Kim, Daseul Bae, Jinho Choo, Youngjune Gwon, and Yeong-Dae Kwon. Caap: Context-aware action planning prompting to solve computer tasks with front-end ui only. *arXiv preprint arXiv:2406.06947*, 2024.
143. Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. Tree search for language model agents. *arXiv preprint arXiv:2407.01476*, 2024.
144. Wei Li, William E Bishop, Alice Li, Christopher Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on ui control agents. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
145. Li Zhang, Shihe Wang, Xianqing Jia, Zhihan Zheng, Yunhe Yan, Longxi Gao, Yuanchun Li, and Mengwei Xu. Llamatouch: A faithful and scalable testbed for mobile ui task automation. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pages 1–13, 2024.
146. Andrea Burns, Deniz Arsan, Sanjna Agrawal, Ranjitha Kumar, Kate Saenko, and Bryan A Plummer. A dataset for interactive vision-language navigation with unknown command feasibility. In *European Conference on Computer Vision*, pages 312–328. Springer, 2022.
147. Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. Mapping natural language instructions to mobile ui action sequences. *arXiv preprint arXiv:2005.03776*, 2020.
148. Zilong Wang, Yuedong Cui, Li Zhong, Zimin Zhang, Da Yin, Bill Yuchen Lin, and Jingbo Shang. Officebench: Benchmarking language agents across multiple applications for office automation. *arXiv preprint arXiv:2407.19056*, 2024.
149. Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972*, 2024.
150. Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, et al. Webcanvas: Benchmarking web agents in online environments. *arXiv preprint arXiv:2406.12373*, 2024.
151. Ziniu Zhang, Shulin Tian, Liangyu Chen, and Ziwei Liu. Mmina: Benchmarking multihop multimodal internet agents. *arXiv preprint arXiv:2404.09992*, 2024.
152. Longtao Zheng, Zhiyuan Huang, Zhenghai Xue, Xinrun Wang, Bo An, and Shuicheng Yan. Agentstudio: A toolkit for building general virtual agents. *arXiv preprint arXiv:2403.17918*, 2024.
153. Kevin Xu, Yeganeh Kordi, Tanay Nayak, Ado Asija, Yizhong Wang, Kate Sanders, Adam Byerly, Jingyu Zhang, Benjamin Van Durme, and Daniel Khashabi. Tur [k] ingbench: A challenge benchmark for web agents. *arXiv preprint arXiv:2403.11905*, 2024.
154. Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.
155. Panupong Pasupat, Tian-Shun Jiang, Evan Zheran Liu, Kelvin Guu, and Percy Liang. Mapping natural language commands to web elements. *arXiv preprint arXiv:1808.09132*, 2018.
156. Maria Wang, Srinivas Sunkara, Gilles Baechler, Jason Lin, Yun Zhu, Fedir Zubach, Lei Shu, and Jindong Chen. Webquest: A benchmark for multimodal qa on web page sequences. *arXiv preprint arXiv:2409.13711*, 2024.
157. Kaining Ying, Fanqing Meng, Jin Wang, Zhiqian Li, Han Lin, Yue Yang, Hao Zhang, Wenbo Zhang, Yuqi Lin, Shuo Liu, et al. Mmt-bench: A comprehensive multimodal benchmark for evaluating large vision-language models towards multitask agi. *arXiv preprint arXiv:2404.16006*, 2024.
158. Yilun Jin, Zheng Li, Chenwei Zhang, Tianyu Cao, Yifan Gao, Pratik Jayarao, Mao Li, Xin Liu, Ritesh Sarkhel, Xianfeng Tang, et al. Shopping mmlu: A massive multi-task online shopping benchmark for large language models. *arXiv preprint arXiv:2410.20745*, 2024.
159. Luyuan Wang, Yongyu Deng, Yiwei Zha, Guodong Mao, Qinmin Wang, Tianchen Min, Wei Chen, and Shoufa Chen. Mobileagentbench: An efficient and user-friendly benchmark for mobile llm agents. *arXiv preprint arXiv:2406.08184*, 2024.
160. Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In *NeurIPS*, 2023.
161. Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*, 2023.

 doi:10.20944/preprints202412.2294.v1




162. Tu Vu, Kalpesh Krishna, Salaheddin Alzubi, Chris Tar, Manaal Faruqui, and Yun-Hsuan Sung. Foundational autoraters: Taming large language models for better automatic evaluation, 2024. URL https://arxiv.org/abs/2407.10817.
163. Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Yuanzhuo Wang, and Jian Guo. A survey on llm-as-a-judge, 2024. URL https://arxiv.org/abs/2411.15594.
164. Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing fine-grained evaluation capability in language models, 2024. URL https://arxiv.org/abs/2310.08491.
165. Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Prometheus 2: An open source language model specialized in evaluating other language models, 2024. URL https://arxiv.org/abs/2405.01535.
166. Tassnim Dardouri, Laura Minkova, Jessica López Espejel, Walid Dahhane, and El Hassane Ettifouri. Visual grounding for desktop graphical user interfaces. *arXiv preprint arXiv:2407.01558*, 2024.
167. Irene Weber. Large language models as software components: A taxonomy for llm-integrated applications. *CoRR*, abs/2406.10300, 2024.
168. Zekai Zhang, Yiduo Guo, Yaobo Liang, Dongyan Zhao, and Nan Duan. Pptc-r benchmark: Towards evaluating the robustness of large language models for powerpoint task completion. *arXiv preprint arXiv:2403.03788*, 2024.
169. Danyang Zhang, Lu Chen, and Kai Yu. Mobile-env: A universal platform for training and evaluation of mobile interaction. *arXiv preprint arXiv:2305.08144*, 2023.
170. Jingxuan Chen, Derek Yuen, Bin Xie, Yuhao Yang, Gongwei Chen, Zhihao Wu, Li Yixing, Xurui Zhou, Weiwen Liu, Shuai Wang, et al. Spa-bench: A comprehensive benchmark for smartphone agent evaluation. In *NeurIPS 2024 Workshop on Open-World Agents*, 2024.
171. Yiduo Guo, Zekai Zhang, Yaobo Liang, Dongyan Zhao, and Nan Duan. Pptc benchmark: Evaluating large language models for powerpoint task completion. *arXiv preprint arXiv:2311.01767*, 2023.
172. Shihan Deng, Weikai Xu, Hongda Sun, Wei Liu, Tao Tan, Jianfeng Liu, Ang Li, Jian Luan, Bin Wang, Rui Yan, et al. Mobile-bench: An evaluation benchmark for llm-based mobile agents. *arXiv preprint arXiv:2407.00993*, 2024.
173. Tianqi Xu, Linyao Chen, Dai-Jie Wu, Yanjun Chen, Zecheng Zhang, Xiang Yao, Zhiqiang Xie, Yongchao Chen, Shilong Liu, Bochen Qian, et al. Crab: Cross-environment agent benchmark for multimodal language model agents. *arXiv preprint arXiv:2407.01511*, 2024.
174. Danny Park. Human player outwits freysa ai agent in $47,000 crypto challenge, 2024. URL https://www.theblock.co/amp/post/328747/human-player-outwits-freysa-ai-agent-in-47000-crypto-challenge. Accessed: 2024-11-30.
175. Zehang Deng, Yongjian Guo, Changzhou Han, Wanlun Ma, Junwu Xiong, Sheng Wen, and Yang Xiang. Ai agents under threat: A survey of key security challenges and future pathways, 2024. URL https://arxiv.org/abs/2406.02630.
176. Yuyou Gan, Yong Yang, Zhe Ma, Ping He, Rui Zeng, Yiming Wang, Qingming Li, Chunyi Zhou, Songze Li, Ting Wang, Yunjun Gao, Yingcai Wu, and Shouling Ji. Navigating the risks: A survey of security, privacy, and ethics threats in llm-based agents, 2024. URL https://arxiv.org/abs/2411.09523.
177. Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, 4:0 (2)::0 100211, June 2024. ISSN 2667-2952. 10.1016/j.hcc.2024.100211. URL http://dx.doi.org/10.1016/j.hcc.2024.100211.
178. Erfan Shayegani, Md Abdullah Al Mamun, Yu Fu, Pedram Zaree, Yue Dong, and Nael Abu-Ghazaleh. Survey of vulnerabilities in large language models revealed by adversarial attacks, 2023. URL https://arxiv.org/abs/2310.10844.
179. Tianyu Cui, Yanling Wang, Chuanpu Fu, Yong Xiao, Sijia Li, Xinhao Deng, Yunpeng Liu, Qinglin Zhang, Ziyi Qiu, Peiyang Li, Zhixing Tan, Junwu Xiong, Xinyu Kong, Zujie Wen, Ke Xu, and Qi Li. Risk taxonomy, mitigation, and assessment benchmarks of large language model systems, 2024. URL https://arxiv.org/abs/2401.05778.
180. Shenzhi Wang, Chang Liu, Zilong Zheng, Siyuan Qi, Shuo Chen, Qisen Yang, Andrew Zhao, Chaofei Wang, Shiji Song, and Gao Huang. Boosting llm agents with recursive contemplation for effective deception handling. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 9909–9953, 2024.

 doi:10.20944/preprints202412.2294.v1




181. Seth Neel and Peter Chang. Privacy issues in large language models: A survey, 2024. URL https://arxiv.org/abs/2312.06717.
182. Fangzhou Wu, Shutong Wu, Yulong Cao, and Chaowei Xiao. Wipi: A new web threat for llm-driven web agents, 2024. URL https://arxiv.org/abs/2402.16965.
183. Chen Henry Wu, Jing Yu Koh, Ruslan Salakhutdinov, Daniel Fried, and Aditi Raghunathan. Adversarial attacks on multimodal agents, 2024. URL https://arxiv.org/abs/2406.12814.
184. Xinbei Ma, Yiting Wang, Yao Yao, Tongxin Yuan, Aston Zhang, Zhuosheng Zhang, and Hai Zhao. Caution for the environment: Multimodal agents are susceptible to environmental distractions, 2024. URL https://arxiv.org/abs/2408.02544.
185. Zeyi Liao, Lingbo Mo, Chejian Xu, Mintong Kang, Jiawei Zhang, Chaowei Xiao, Yuan Tian, Bo Li, and Huan Sun. Eia: Environmental injection attack on generalist web agents for privacy leakage, 2024. URL https://arxiv.org/abs/2409.11295.
186. Chejian Xu, Mintong Kang, Jiawei Zhang, Zeyi Liao, Lingbo Mo, Mengqi Yuan, Huan Sun, and Bo Li. Advweb: Controllable black-box attacks on vlm-powered web agents, 2024. URL https://arxiv.org/abs/2410.17401.
187. Yanzhe Zhang, Tao Yu, and Diyi Yang. Attacking vision-language computer agents via pop-ups, 2024. URL https://arxiv.org/abs/2411.02391.
188. Priyanshu Kumar, Elaine Lau, Saranya Vijayakumar, Tu Trinh, Scale Red Team, Elaine Chang, Vaughn Robinson, Sean Hendryx, Shuyan Zhou, Matt Fredrikson, Summer Yue, and Zifan Wang. Refusal-trained llms are easily jailbroken as browser agents, 2024. URL https://arxiv.org/abs/2410.13886.
189. Yulong Yang, Xinshan Yang, Shuaidong Li, Chenhao Lin, Zhengyu Zhao, Chao Shen, and Tianwei Zhang. Security matrix for multimodal agents on mobile devices: A systematic and proof of concept study, 2024. URL https://arxiv.org/abs/2407.09295.
190. Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of lm agents with an lm-emulated sandbox, 2024. URL https://arxiv.org/abs/2309.15817.
191. Wenyue Hua, Xianjun Yang, Mingyu Jin, Zelong Li, Wei Cheng, Ruixiang Tang, and Yongfeng Zhang. Trustagent: Towards safe and trustworthy llm-based agents, 2024. URL https://arxiv.org/abs/2402.01586.
192. Haishuo Fang, Xiaodan Zhu, and Iryna Gurevych. Inferact: Inferring safe actions for llm-based agents through preemptive evaluation and human feedback, 2024. URL https://arxiv.org/abs/2407.11843.
193. Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, Dawn Song, and Bo Li. Guardagent: Safeguard llm agents by a guard agent via knowledge-enabled reasoning, 2024. URL https://arxiv.org/abs/2406.09187.
194. Md Shamsujjoha, Qinghua Lu, Dehai Zhao, and Liming Zhu. Designing multi-layered runtime guardrails for foundation model based agents: Swiss cheese model for ai safety by design, 2024. URL https://arxiv.org/abs/2408.02205.
195. Rodrigo Pedro, Daniel Castro, Paulo Carreira, and Nuno Santos. From prompt injections to sql injection attacks: How protected is your llm-integrated web application?, 2023. URL https://arxiv.org/abs/2308.01990.
196. Ido Levy, Ben Wiesel, Sami Marreed, Alon Oved, Avi Yaeli, and Segev Shlomov. St-webagentbench: A benchmark for evaluating safety and trustworthiness in web agents, 2024. URL https://arxiv.org/abs/2410.06703.
197. Juyong Lee, Dongyoon Hahm, June Suk Choi, W. Bradley Knox, and Kimin Lee. Mobilesafetybench: Evaluating safety of autonomous agents in mobile device control, 2024. URL https://arxiv.org/abs/2410.17520.
198. Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.
199. Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, et al. Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. *arXiv preprint arXiv:2305.17144*, 2023.
200. Wangchunshu Zhou, Yixin Ou, Shengwei Ding, Long Li, Jialong Wu, Tiannan Wang, Jiamin Chen, Shuai Wang, Xiaohua Xu, Ningyu Zhang, Huajun Chen, and Yuchen Eleanor Jiang. Symbolic learning enables self-evolving agents, 2024. URL https://arxiv.org/abs/2406.18532.

 doi:10.20944/preprints202412.2294.v1




201. Yanda Li, Chi Zhang, Wanqi Yang, Bin Fu, Pei Cheng, Xin Chen, Ling Chen, and Yunchao Wei. Appagent v2: Advanced agent for flexible mobile interactions. *arXiv preprint arXiv:2408.11824*, 2024.
202. Alan Wake, Albert Wang, Bei Chen, CX Lv, Chao Li, Chengen Huang, Chenglin Cai, Chujie Zheng, Daniel Cooper, Ethan Dai, et al. Yi-lightning technical report. *arXiv preprint arXiv:2412.01253*, 2024.
203. Dongxu Li, Yudong Liu, Haoning Wu, Yue Wang, Zhiqi Shen, Bowen Qu, Xinyao Niu, Guoyin Wang, Bei Chen, and Junnan Li. Aria: An open multimodal native mixture-of-experts model, 2024. URL https://arxiv.org/abs/2410.05993.
204. Kaizhi Zheng, Xuehai He, and Xin Eric Wang. Minigpt-5: Interleaved vision-and-language generation via generative vokens, 2024. URL https://arxiv.org/abs/2310.02239.
205. Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond, 2023. URL https://arxiv.org/abs/2308.12966.
206. Yong Dai, Duyu Tang, Liangxin Liu, Minghuan Tan, Cong Zhou, Jingquan Wang, Zhangyin Feng, Fan Zhang, Xueyu Hu, and Shuming Shi. One model, multiple modalities: A sparsely activated approach for text, sound, image, video and code, 2022. URL https://arxiv.org/abs/2205.06126.
207. Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.
208. Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. A survey on multimodal large language models. *National Science Review*, page nwae403, 2024.
209. Duzhen Zhang, Yahan Yu, Jiahua Dong, Chenxing Li, Dan Su, Chenhui Chu, and Dong Yu. Mm-llms: Recent advances in multimodal large language models. *arXiv preprint arXiv:2401.13601*, 2024.
210. Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. On llms-driven synthetic data generation, curation, and evaluation: A survey. *arXiv preprint arXiv:2406.15126*, 2024.
211. Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*, 2023.
212. Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18:0 (6)::0 186345, 2024.
213. Yuheng Cheng, Ceyao Zhang, Zhengwen Zhang, Xiangrui Meng, Sirui Hong, Wenhao Li, Zihao Wang, Zekai Wang, Feng Yin, Junhua Zhao, et al. Exploring large language model based intelligent agents: Definitions, methods, and prospects. *arXiv preprint arXiv:2401.03428*, 2024.
214. Yuyou Gan, Yong Yang, Zhe Ma, Ping He, Rui Zeng, Yiming Wang, Qingming Li, Chunyi Zhou, Songze Li, Ting Wang, et al. Navigating the risks: A survey of security, privacy, and ethics threats in llm-based agents. *arXiv preprint arXiv:2411.09523*, 2024.
215. Wangchunshu Zhou, Yuchen Eleanor Jiang, Long Li, Jialong Wu, Tiannan Wang, Shi Qiu, Jintian Zhang, Jing Chen, Ruipu Wu, Shuai Wang, Shiding Zhu, Jiyu Chen, Wentao Zhang, Xiangru Tang, Ningyu Zhang, Huajun Chen, Peng Cui, and Mrinmaya Sachan. Agents: An open-source framework for autonomous language agents, 2023. URL https://arxiv.org/abs/2309.07870.
216. Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model based agents. *arXiv preprint arXiv:2404.13501*, 2024.
217. Xinyi Li, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth*, 1:0 (1)::0 9, 2024.
218. Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. Reasoning with language model prompting: A survey. *arXiv preprint arXiv:2212.09597*, 2022.
219. Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology*, 2023.
220. Sihao Hu, Tiansheng Huang, Fatih Ilhan, Selim Tekin, Gaowen Liu, Ramana Kompella, and Ling Liu. A survey on large language model-based game agents. *arXiv preprint arXiv:2404.02039*, 2024.

 doi:10.20944/preprints202412.2294.v1



221. Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, et al. Personal llm agents: Insights and survey about the capability, efficiency and security. *arXiv preprint arXiv:2401.05459*, 2024.
222. Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, Rui Kong, Yile Wang, Hanfei Geng, Jian Luan, Xuefeng Jin, Zilong Ye, Guanjing Xiong, Fan Zhang, Xiang Li, Mengwei Xu, Zhijun Li, Peng Li, Yang Liu, Ya-Qin Zhang, and Yunxin Liu. Personal llm agents: Insights and survey about the capability, efficiency and security, 2024. URL https://arxiv.org/abs/2401.05459.
223. Biao Wu, Yanda Li, Meng Fang, Zirui Song, Zhiwei Zhang, Yunchao Wei, and Ling Chen. Foundations and recent trends in multimodal mobile agents: A survey, 2024. URL https://arxiv.org/abs/2411.02006.
224. Shuai Wang, Weiwen Liu, Jingxuan Chen, Weinan Gan, Xingshan Zeng, Shuai Yu, Xinlong Hao, Kun Shao, Yasheng Wang, and Ruiming Tang. Gui agents with foundation models: A comprehensive survey, 2024. URL https://arxiv.org/abs/2411.04890.
225. Minghe Gao, Wendong Bu, Bingchen Miao, Yang Wu, Yunfei Li, Juncheng Li, Siliang Tang, Qi Wu, Yueting Zhuang, and Meng Wang. Generalist virtual agents: A survey on autonomous agents across digital platforms, 2024. URL https://arxiv.org/abs/2411.10943.
226. Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Minghua Ma, Qingwei Lin, Saravan Rajmohan, et al. Large language model-brained gui agents: A survey. *arXiv preprint arXiv:2411.18279*, 2024.