

573 A Details of OSWORLD Environment

574 A.1 Environment Infrastructure

575 As compared to core commonly used techniques like Docker¹, virtual machines can operate their
576 own kernel and system, enabling compatibility with a wide variety of operating systems (such as
577 Windows, macOS, Linux, etc.) across different CPU hardware types (x64, ARM, etc.), and supports
578 training and evaluation in a multiprocess manner on both headless servers and personal computers.

579 A.2 Observation Space

580 We implement three kinds of observation: **complete screenshot**, **accessibility tree** and **terminal**
581 **output**. We also implement a video recorder of the environment but don't put it into our modeling
582 due to the agent's ability limitations. OSWORLD supports observation refactoring and extending if
583 needed, such as getting data from certain opening applications that we want to focus on.

584 A.2.1 Screenshot

585 To align with the perception of a human user, we capture a screenshot of the entire computer screen.
586 Including the mouse cursor also proves helpful in certain cases where mouse information is crucial.
587 For screen resolution, we default to 1920×1080, as it is the most commonly used screen resolution
588 according to Internet Users Screen Resolution Realtime Statistics for 2023². This resolution also
589 offers a 16:9 aspect ratio. OSWORLD also supports modifying the resolution of virtual machines to
590 avoid potential memorization of absolute pixel values and to assist studies on topics like generalization
591 across different resolutions.

592 A.2.2 Accessibility Tree

593 An accessibility tree (or a11y tree, same logic to kubernetes and k8s), refers to an intricate structure
594 generated by the browser or OS accessibility APIs that renders a representative model of the web
595 content, providing a means of interaction for assistive technologies. Each node within the accessibility
596 tree hosts important information about a UI element. This could range from the nature of the object
597 (a button, checkbox, or paragraph of text), its current state (checked or unchecked, for checkboxes),
598 and even its spatial orientation on the screen.

599 Different operating systems employ varied accessibility APIs and tools to construct and manipulate
600 the accessibility tree. These include Microsoft Active Accessibility (MSAA) and User Interface
601 Automation (UIA) for Windows, NSAccessibility Protocol and macOS Accessibility Inspector for
602 macOS, and Assistive Technology Service Provider Interface (ATSPI)³ for GNOME desktop used
603 on Ubuntu. We adopt pyatspi to get the accessibility tree on Ubuntu and pywinauto on Windows.
604 We further convert it into XML format for message passing. Partial pieces of the XML formatted
605 accessibility tree are shown in Figure 4.

606 Tools such as Accerciser can help visualize the corresponding relationship of tree nodes and GUI
607 components in the accessibility tree as shown in Figure 5.

608 A.3 Action Space

609 We implement two kinds of action space: pyautogui and computer_v13. We mainly use pyautogui
610 action space, since it saves tokens for describing action space definition in prompting, compared with
611 self-designed actions.

¹<https://www.docker.com/>

²<https://www.screenresolution.org/year-2023/>

³<https://docs.gtk.org/atspi2/>


```

import pyautogui

# move mouse to XY coordinates over num_second seconds
pyautogui.moveTo(x, y, duration=num_seconds)

# drag mouse to XY
pyautogui.dragTo(x, y, duration=num_seconds)

pyautogui.click(x=moveToX, y=moveToY, clicks=num_of_clicks, button='left')
pyautogui.rightClick(x=moveToX, y=moveToY)
pyautogui.middleClick(x=moveToX, y=moveToY)
pyautogui.doubleClick(x=moveToX, y=moveToY)
pyautogui.tripleClick(x=moveToX, y=moveToY)
pyautogui.scroll(amount_to_scroll, x=moveToX, y=moveToY)
pyautogui.mouseDown(x=moveToX, y=moveToY, button='left')
pyautogui.mouseUp(x=moveToX, y=moveToY, button='left')

```

623 Keyboard Controlling Functions

```

import pyautogui

# useful for entering text, newline is Enter
pyautogui.typewrite('Hello world!\n', interval=secs_between_keys)

pyautogui.typewrite(['a', 'b', 'c', 'left', 'backspace', 'enter', 'f1'], interval=secs_between_keys)
pyautogui.hotkey('ctrl', 'c') # ctrl-c to copy
pyautogui.hotkey('ctrl', 'v') # ctrl-v to paste
pyautogui.keyDown(key_name)
pyautogui.keyUp(key_name)

```

624 **pyautogui as an Action Space** Given the various controls it provides, pyautogui can readily be used as
 625 an action space in building automation software or testing interfaces with minor adjustments. More formally, an
 626 action is within the action space when it meets the syntax of pyautogui or is one of three special actions WAIT,
 627 FAIL, and DONE. This might include actions like clicking at a certain location, entering text or key sequences,
 628 or even resting for a span (Pause). Each action could be mapped to an identifying label or number, forming a
 629 discrete action space. For example:

```

import pyautogui

def perform_action(action):
    if action == 0:
        pyautogui.moveTo(100, 100)
    elif action == 1:
        pyautogui.write('Hello world!', interval=0.25)
    else:
        pyautogui.pause(1)

```

630 In this scheme, the "perform_action" function constitutes the action space, where each unique action is associated
 631 with a unique integer (its action ID). The function interprets these action IDs and performs the corresponding
 632 action, forming a rudimentary discrete action space.

633 One interesting finding is that language models generate screenshot locate functions like:

```
pyautogui.locateOnScreen('Apple.png')
```

634 When there is insufficient grounding evidence (such as when no screenshot is inputted, the accessibility tree lacks
 635 a specific element, or the multimodal model cannot comprehend the user interface), employing this function to
 636 retrieve the correct icon image could present an interesting method.

637 A.3.2 COMPUTER_13

638 To facilitate potential reinforcement learning applications, we have created a variant of pyautogui, which
 639 we named computer_13. In this variant, we wrap pyautogui into a finite action class with parameterized
 640 enumeration, such that it features 13 action types, excluding three special ones for task process control. Utilizing
 641 this structured approach allows more effective reinforcement learning by providing a distinct and finite set of

actions to be learned and optimized. As summarized in Table 5, each action type has certain parameters, detailed in the collection, confirming the type, range, and whether each parameter is optional for that action.

Table 5: Action types and parameters defined in action space `computer_13`, a variance we created for the potential reinforcement learning research based on our environment.

Action Type	Parameters	Note
MOVE_TO	<i>x, y</i>	<i>Move the cursor to the specified position</i>
CLICK	<i>button,</i> <i>x, y,</i> <i>num_clicks</i>	<i>Click the left button if the button not specified, otherwise click the specified button; click at the current position if x and y are not specified, otherwise click at the specified position</i>
MOUSE_DOWN	<i>button</i>	<i>Press the left button if the button not specified, otherwise press the specified button</i>
MOUSE_UP	<i>button</i>	<i>Release the left button if the button not specified, otherwise release the specified button</i>
RIGHT_CLICK	<i>x, y</i>	<i>Right click at the current position if x and y are not specified, otherwise right click at the specified position</i>
DOUBLE_CLICK	<i>x, y</i>	<i>Double click at the current position if x and y are not specified, otherwise double click at the specified position</i>
DRAG_TO	<i>x, y</i>	<i>Drag the cursor to the specified position with the left button pressed</i>
SCROLL	<i>dx, dy</i>	<i>Scroll the mouse wheel up or down</i>
TYPING	<i>text</i>	<i>Type the specified text</i>
PRESS	<i>key</i>	<i>Press the specified key and release it</i>
KEY_DOWN	<i>key</i>	<i>Press the specified key</i>
KEY_UP	<i>key</i>	<i>Release the specified key</i>
HOTKEY	<i>keys</i>	<i>Press the specified key combination</i>
WAIT	-	<i>Wait until the next action</i>
FAIL	-	<i>Decide the task cannot be performed</i>
DONE	-	<i>Decide the task is done</i>

B Details of OSWORLD Benchmark

B.1 Operating System Selection

As of 2023, the most popular desktop operating systems are Windows (69.5%), macOS (20.4%), ChromeOS (3.2%), and Linux (3.1%)⁷. While Windows and macOS dominate the market share and boast the richest software ecosystems, their closed-source nature raises potential copyright concerns for direct usage. ChromeOS, being a web-based operating system, heavily depends on a Google account for its functionalities, rendering it less suitable for a public benchmark.

In contrast, Linux desktops offer a wealth of open-source software for most daily tasks, supported by an active community for both basic and advanced use. Essential applications such as Office Suite, browsers, email clients, multimedia apps, and thousands of games and applications are either pre-installed or readily available through the software center of the distribution. Consequently, we select Ubuntu, the most representative Linux desktop OS, as the foundation for the main part of our benchmark intended for public use. Additionally, we have developed components to facilitate agent interaction on a Windows virtual machine and have created a relatively small set of examples focusing on the Microsoft Office suite, including Excel, PowerPoint, and Word. This serves as a counterpart to the LibreOffice suite available on Ubuntu. These components can be utilized in-house or officially with the purchase of a license. Regarding macOS, theoretically, it is illegal to install macOS on non-Apple devices, which leads us to refrain from developing our benchmark on this platform to avoid copyright issues.

B.2 Software Selection

Due to the high cost of obtaining operation and evaluation script annotation data, we have chosen a representative set of software for the examples of Ubuntu part. We adopt standards that consider: 1) Availability - the software must be available on Ubuntu 22.04; 2) Open-source - the software should be open-sourced with an appropriate license to prevent copyright issues; 3) Popularity - the software should take a high download number and frequency of recommendations in blogs and tutorials; 4) Strong user community and good support resources

⁷<https://www.statista.com/statistics/218089/global-market-share-of-windows-7/>

667 - it is preferable to have an active and robust user community as well as official documents, which can serve
668 as ample resources for task collection and agent learning; 5) Diversity of categories - the software should be
669 diverse to adequately represent and cover a wide range of real-world cases.

670 As a result, we have shortlisted the software into two categories: general usage and professional usage. For
671 general usage, we have VLC for media playback, Chrome for web browsing, and Thunderbird for email
672 management. For professional usage, we have VS Code as a coding IDE, and LibreOffice (Calc, Writer, and
673 Impress) for handling spreadsheets, documents, and presentations respectively, along with GIMP for image
674 editing. This brings our total to eight different types of software.

675 B.3 Task Example Sources

676 We detail the task example sources in Table 6.

Table 6: Task Example Resources

App	Resources	Link
OS	Ubuntu Documentations	https://help.ubuntu.com/
	Ask Ubuntu	https://askubuntu.com/
	Super User	https://superuser.com/
	Stack Overflow	https://stackoverflow.com
	YouTube	https://www.youtube.com/
Calc	LibreOffice Help	https://help.libreoffice.org/
	Microsoft Tech Community	https://techcommunity.microsoft.com/
	libreofficehelp.com	https://www.libreofficehelp.com/
	Reddit r/LibreOfficeCal	https://www.reddit.com/r/LibreOfficeCalc/
	Reddit r/Excel	https://www.reddit.com/r/Excel/
	Super User	https://superuser.com/
	Medium	https://medium.com/
	Quora	https://www.quora.com/
	YouTube	https://www.youtube.com/
Writer	Ryan and Debi & Toren Personal Site	https://www.ryananddebi.com/
	LibreOffice Help	https://help.libreoffice.org/
	LibreOffice Forum	https://ask.libreoffice.org/
	libreofficehelp.com	https://www.libreofficehelp.com/
	Super User	https://superuser.com/
	Stack Overflow	https://stackoverflow.com
	Ask Ubuntu	https://askubuntu.com/
	Quora	https://www.quora.com/
	YouTube	https://www.youtube.com/
Impress	SearchStar Personal Site	https://seekstar.github.io/
	LibreOffice Help	https://help.libreoffice.org/
	libreofficehelp.com	https://www.libreofficehelp.com/
	Reddit r/LibreOffice	https://www.reddit.com/r/LibreOffice/
	Super User	https://superuser.com/
	Stack Overflow	https://stackoverflow.com
	Technical Tips	https://technical-tips.com/
	Just Click Here	https://justclickhere.co.uk/
	TikTok	https://www.tiktok.com/
VLC	VLC Documentation	https://docs.videolan.me
	VLCHelp.com	https://www.vlchelp.com/
	VideoLAN's Wiki	https://wiki.videolan.org/
	Ubuntu Documentations	https://help.ubuntu.com/
	Reddit r/Fedora	https://www.reddit.com/r/Fedora/
	Super User	https://superuser.com/
	Medium	https://medium.com/
Thunderbird	YouTube	https://www.youtube.com/
	Dedoimedo	https://www.dedoimedo.com/index.html
	Thunderbird Support	https://support.mozilla.org/en-US/products/thunderbird
	Reddit r/Thunderbird	https://www.reddit.com/r/Thunderbird/
	Reddit r/Automation	https://www.reddit.com/r/automation/
	Super User	https://superuser.com/
	WikiHow	https://www.wikihow.com/
	Quora	https://www.quora.com/

Continued on next page

Table 6 – continued from previous page

App	Resources
Chrome	BitRecover https://www.bitrecover.com/
	AdSigner https://www.adsigner.com/
	Google Chrome Help https://support.google.com/chrome
	Reddit r/Chrome https://www.reddit.com/r/Chrome/
	Super User https://superuser.com/
	WikiHow https://www.wikihow.com/
	in5steps.com https://in5stepstutorials.com/
	How-To Geek https://www.howtogeek.com/
	Medium https://medium.com/
	Quora https://www.quora.com/
VS Code	YouTube https://www.youtube.com/
	Laptop Mag https://www.laptopmag.com
	Super User https://superuser.com/
	Stack Overflow https://stackoverflow.com
	Quora https://www.quora.com/
GIMP	YouTube https://www.youtube.com/
	Campbell Muscle Lab GitHub https://campbell-muscle-lab.github.io/
	Reddit r/GIMP https://www.reddit.com/r/GIMP/
	Super User https://superuser.com/
	Stack Overflow https://stackoverflow.com
Workflow	Quora https://www.quora.com/
	Make-Use-Of https://www.makeuseof.com/
	YouTube https://www.youtube.com/
	UniPath Marketplace https://marketplace.uipath.com/
	sync.blue https://www.sync.blue/
	Device Tests https://devicetests.com/
	Make Tech Easier https://www.maketecheasier.com/
	Unix & Linux Stack Exchange https://unix.stackexchange.com/
	Geeks for Geeks https://www.geeksforgeeks.org/
	I Love Free Software https://www.ilovefreesoftware.com/
	The Geek Diary https://www.thegeekdiary.com/
	Zyxware https://www.zyxware.com/
	GNOME Discourse https://discourse.gnome.org/
	It's FOSS https://itsfoss.com/
	Super User https://superuser.com/
	Stack Overflow https://stackoverflow.com
	LibreOffice Forum https://ask.libreoffice.org/
	ImpressExtractNotes https://github.com/danielrcollins1/ImpressExtractNotes
	Medium https://medium.com/
	YouTube https://www.youtube.com/
	Kelvin Smith Library https://case.edu/library/

B.4 Task Examples Collection

Here we show the detailed statistics of OSWORLD benchmark, including the main set on Ubuntu (369 in total) and the analytic set on Windows (43 in total).

Table 7: Detailed statistics of OSWORLD benchmark suite about examples number, average instruction tokens, infeasible instructions and integrated instructions.

	OS	Calc	Impress	Writer	VLC	Thunderbird	Chrome	VSCode	GIMP	Workflow	Overall
Examples	24	47	47	23	17	15	46	23	26	101	369
Avg. Inst. Tokens	22.38	33.30	25.19	35.30	35.82	34.07	22.07	20.78	16.23	51.24	33.36
#Infeasible	5	1	0	1	3	1	3	5	10	1	30
#Integrated	7	19	30	0	0	0	26	0	0	2	84

We also visualize the intent distribution (We obtain sentence embeddings for instructions using OpenAI’s embedding model, and then apply t-SNE to reduce the dimensionality to two dimensions for visualization,) and compare it with other benchmarks which also focus on the digital agent. We randomly sample 300 examples from each dataset and randomly choose three different hyperparameters for t-SNE. Visualization results are shown in Figure 6. From the figure, we can observe that the semantic distribution of the instructions alone

Table 8: Detailed statistics of Windows analytic set benchmark suite. This set contains no infeasible tasks and integrated tasks.

	Excel	Word	PPT	Workflow	Overall
Examples	11	9	7	16	43
Avg. Inst. Tokens	19.45	21.44	21.86	47.57	32.48

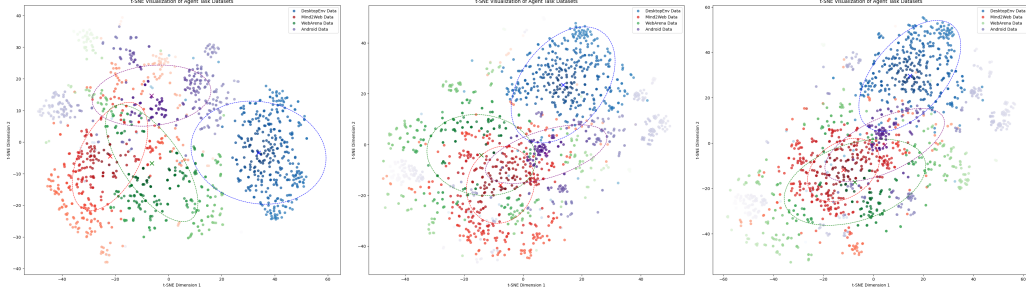


Figure 6: Comparison of instructions distribution. All datasets are sampled to 300 to make a fair comparison. The hyper-parameters of t-SNE are randomly sampled for each plot.

has reached the most comprehensive level. Additionally, our environment remains controllable and executable, offering a more reliable evaluation. It is also noticeable that the clustering centers of the other three are closely positioned, whereas the points in our distribution approaches are inconsistent with theirs, indicating that we can serve as a unique choice for a more comprehensive assessment of the capabilities of future intelligent agents.

B.5 Initial State Setup Details

The setup of the initial state contains three stages: 1) Start emulator. The specified virtual machine is activated and automatically reverted to the corresponding snapshot, which records the initial system settings of the machine. 2) Prepare files (Optional). The file or software that specifies the initial state of the task to be executed is downloaded to the virtual machine and opened. The system is configured to first download the files to the host through a direct link and then upload them to the VM via a LAN connection. Specifically, some initial files are set up for OS-related tasks by manipulating the file system directly from the command line. 3) Execute reprocessing commands (Optional). For tasks that require additional preprocessing, task-specific operations are executed after the completion of the first two phases. For example, taking the currently open LibreOffice Impress file to page five, clicking in the center of the screen to return to the main interface, *etc.* We provide convenient APIs to configure initial conditions and world settings, standardizing our tasks to make this process user-friendly and easily extendable.

B.6 Evaluation Configuration Details

In this section, we will show details of preparations for the evaluation of the selected apps (LibreOffice – Calc, Writer and Impress, Thunderbird, VLC Media Player, Chrome, VS Code, GIMP) and OS (Ubuntu and Windows).

B.6.1 Ubuntu

LibreOffice: Calc, Writer, and Impress LibreOffice is a popular open-source fully-featured office suite for Linux desktops. Our benchmark is built upon version 7.3.7.2, the version pre-installed in Ubuntu 22.04. Calc, Writer, and Impress are selected to build tasks on them. As the majority of tasks are to conduct a little revision to a file, we evaluate these tasks mainly by checking the final result file (in xlsx, docx, or pptx format). The check can be done by comparing the result file with a golden reference, or inspecting some particular attributes or settings of the file, *e.g.*, page style, freezing, and locale. Usually, the xlsx, docx, and pptx files are mainly accessed through `openpyxl`⁸, `python-docx`⁹, and `python-pptx`¹⁰. For some properties not supported by the current libraries, we also look them up directly via parsing the Office Open XML format¹¹.

⁸<https://openpyxl.readthedocs.io/en/stable/>

⁹<https://github.com/python-openxml/python-docx>

¹⁰<https://github.com/scanny/python-pptx>

¹¹<https://learn.microsoft.com/en-us/office/open-xml/about-the-open-xml-sdk>

713 **Thunderbird** Thunderbird is a popular open-source fully-featured email client for Linux desktops. Version
714 115.6.0 of Thunderbird is pre-installed in Ubuntu 22.04. We crafted an account profile to set up a feasible initial
715 state. Evaluation for Thunderbird configurations is mainly performed by reading various configurations or data
716 files in the profile folder. An open-source reverse engineering tool Firefox Decrypt¹² is adopted to decrypt the
717 stored account information for evaluation. The involved account information is just for examples and contains
718 no information about the real person. Besides, there are tasks instructing to help to compose a new email. In
719 these cases, the accessibility tree is leveraged to inspect the contents in the composing window before really
720 sending it.

721 **VLC Media Player** VLC Media Player is a popular open-source cross-platform multimedia player and
722 framework that plays most multimedia files. The evaluation for VLC Media Player is multifold, ranging from
723 utilizing VLC HTTP interface¹³, reading the VLC configuration file, comparing final result files, and leveraging
724 accessibility tree to inspect the desired content.

725 **Chrome** Google Chrome is one of the most popular and powerful cross-platform web browsers developed
726 on Google's free and open-source software project Chromium. The evaluation of Chrome is mainly based on
727 the utilization of Playwright¹⁴, a browser automation library to control Chromium, Firefox, and WebKit with
728 a single API. To connect Playwright running on host machine with Chrome running on virtual machine, port
729 transferring tool socat¹⁵ is leveraged. Additional information such as the HTML source codes of websites is also
730 leveraged in the evaluation of some tasks.

731 **VS Code** VS Code is a popular open-source multi-functional cross-platform editor for source-code editing.
732 The evaluation of VS Code tasks is primarily divided into two different categories. One subset of tasks is
733 predominantly oriented towards file manipulation. In the context of these tasks, a comparative analysis is
734 conducted between the resultant file and an anticipated reference gold file. Another subset of tasks is centered
735 around how to utilize the intrinsic functionalities of the VS Code software itself, such as modifying color themes,
736 initiating workspace sessions, and modifying settings. In these instances, it becomes important to extract relevant
737 internal information and configurations from the VS Code environment.

738 In the course of this research, we principally leverage the capabilities offered by the VS Code Extension API¹⁶
739 and information in the settings JSON file¹⁷ to obtain the requisite internal signal for the evaluation process. Our
740 methodology involves the development of a custom VS Code extension, its installation within the VS Code
741 software deployed on our virtual machine, and the subsequent invocation of the extension's command each time
742 an evaluation is required, as well as checking whether the settings JSON has the correct value for a specific key.

743 **GIMP** GIMP is an open-source raster graphics editor used for image manipulation, editing, free-form drawing,
744 format transcoding, and more specialized tasks. The evaluation for GIMP tasks is also mainly divided into two
745 different categories, just like the VS Code evaluation. One type of task is mainly oriented to file operations.
746 In these tasks, the resulting files are compared and analyzed with the expected reference golden files, mainly
747 relying on some powerful image processing libraries such as pillow¹⁸. Another category of tasks revolves
748 around taking advantage of the inherent capabilities of the GIMP software itself. In these instances, we primarily
749 read GIMP's configuration files to obtain internal information to evaluate the tasks.

750 B.6.2 Windows

751 **Microsoft Office: Excel, Word, and PowerPoint** Microsoft Office is the most popular office suite
752 on Windows desktops. These three components share the same functions with the corresponding LibreOffice
753 components by and large. They are used to edit xlsx, docx, and pptx files, respectively. Thus, the evaluation for
754 LibreOffice tasks can be reused for Microsoft Office tasks.

755 **Thunderbird** Thunderbird is a cross-platform email client. Only the structure of profile folder on Windows
756 is slightly different from that on Linux. We thus revised the account profile and reuse it to set up the same initial
757 state on Windows.

¹²https://github.com/unode/firefox_decrypt

¹³https://wiki.videolan.org/Control_VLC_via_a_browser/

¹⁴<https://playwright.dev/>

¹⁵<http://www.dest-unreach.org/socat/>, <https://linux.die.net/man/1/socat>

¹⁶<https://code.visualstudio.com/api>

¹⁷https://code.visualstudio.com/docs/getstarted/settings#_settingsjson

¹⁸<https://pypi.org/project/pillow/>

758 **Chrome** Chrome is a cross-platform web browser. To evaluate tasks on Chrome, only the port transferring
 759 tool needs to be replaced with Ncat¹⁹. Other configurations and the evaluations can be shared with Linux-version
 760 tasks.

761 B.7 Human Performance

762 We conduct human evaluations on each example in
 763 our dataset, with annotators being computer science
 764 major college students who possess basic software
 765 usage skills but have not been exposed to the samples
 766 or software before. We recorded the time required to
 767 complete each example and whether their completion
 768 of the example was correct. For comparison, we also
 769 sampled 100 examples from WebArena [66] under the
 770 same evaluation setup.

771 As illustrated, tasks from our dataset generally re-
 772 quired more time to complete, with a median completion
 773 time of 111.94 seconds (compared to 35.38 seconds in WebArena), and a significant number of
 774 examples distributed at 900 seconds or even more.
 775 In terms of accuracy, the human performance on our
 776 tasks was approximately 72.36%, significantly lower
 777 than the 88% observed on the pure web task dataset.
 778 These findings highlight the complexity and challenge
 779 of tasks in our dataset, which demand more time and
 780 effort. The lower accuracy rate further indicates that our tasks require a higher level of understanding and
 781 proficiency, underscoring the need for advanced models and techniques to tackle them effectively.

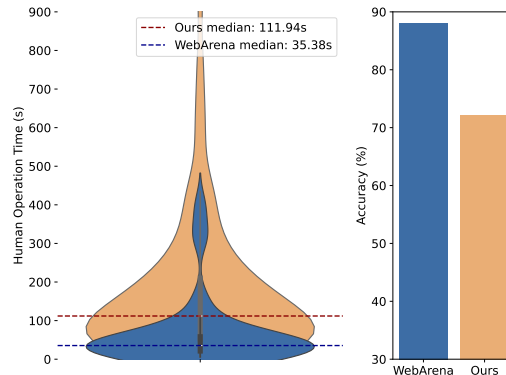


Figure 7: Human operation time and accuracy on OSWORLD and WebArena.

783 B.8 More Task Examples

784 In this section, we curate a collection of examples from various app sets, each characterized by distinct operational
 785 logic and requiring different capabilities. These examples are carefully chosen to illustrate the diverse challenges
 786 and requirements encountered when interacting with different types of applications.

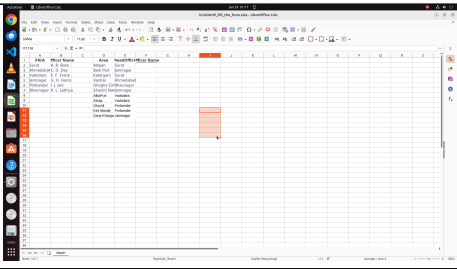
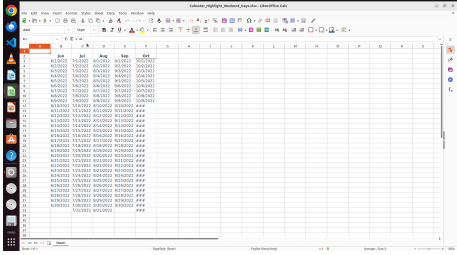
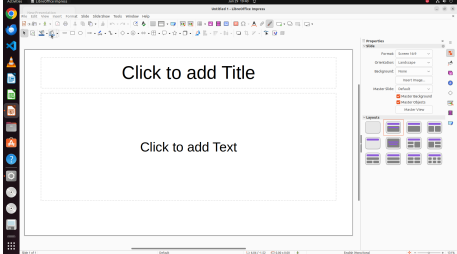
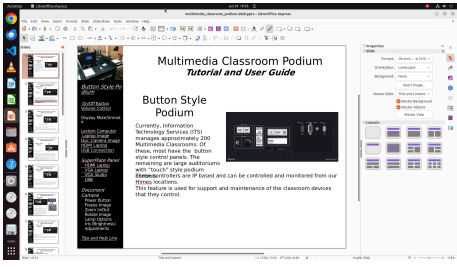
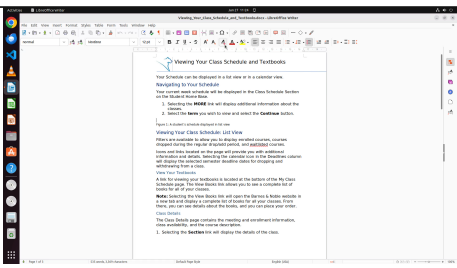
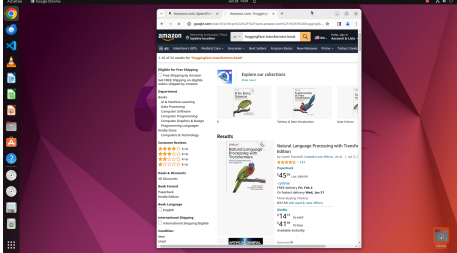
Table 9: More Example Showcase from Each Subset of Domains.

Related App(s)	Instruction(s)	Screenshot	Abilities Needed
OS	<i>I want to install Spotify on my current system. Could you please help me?</i>		knowledge of OS; omit distractions
Calc	<i>Check the names in column "Names with duplicates" and put the unique ones in column "Unique Names". Keep the original order.</i>		massive elements; knowledge tricks or reasoning over long actions

Continued on next page

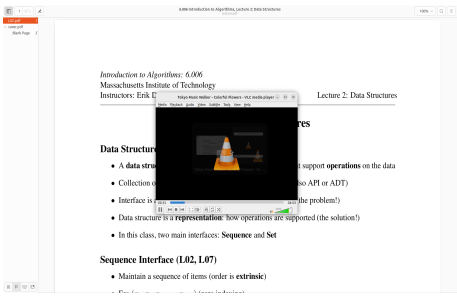
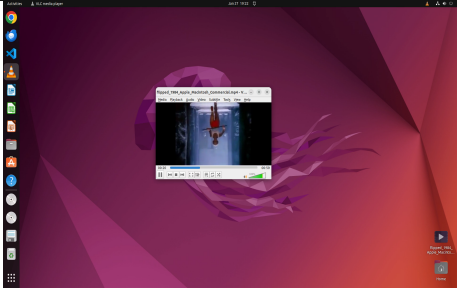
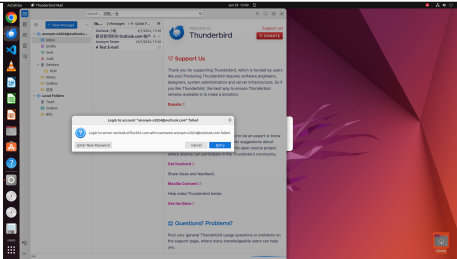
¹⁹<http://www.dest-unreach.org/socat/>

Table 9 – continued from previous page

Related App(s)	Task Instruction	Screenshot of Initial State	Abilities Needed
Calc	<i>I have a lookup table for the officers of each branch. Please, here is another table in which I need to fill with the officer names according the headoffice (i.e., the branch name). Help me to complete this.</i>		massive elements; knowledge of formulas and functions
Calc	<i>Given a partial calendar, please highlight all the weekends (Saturday & Sunday) by setting the cell background as red (#ff0000).</i>		massive elements; commonsense reasoning; software tricks
Impress	<i>I closed the slide pannel on the left and idk how to get it back please help</i>		software knowledge; imagine about UI layouts; overcome typos in instruction
Impress	<i>On it Whenever I launch a LibreOffice Impress, it uses both screens, one for current slide and next slide and another for actual presentation. What I want is to use only one monitor which shows presentation. I dont want the screen with Current slide and Next slide so that it can be used for other purposes. How should I achieve this?</i>		reason from unprofessional phenomenon expression
Writer	<i>Copy the screenshot 1.png from the desktop to where my cursor is located</i>		locate the position of cursor; switch from desktop and app
Chrome	<i>Can you help me clean up my computer by getting rid of all the tracking things that Amazon might have saved? I want to make sure my browsing is private and those sites don't remember me.</i>		understanding the unprofessional expression

Continued on next page

Table 9 – continued from previous page

Related App(s)	Task Instruction	Screenshot of Initial State	Abilities Needed
VLC	<i>I am reading lecture note in PDF while a music video is running in VLC media player. But I find I need to switch to the player every time I need to pause/start. Could you help me change the setting to allow pausing the video using keyboard shortcut without minimizing the PDF reader? I want to focus on the lecture note and don't be disturbed by the app switching.</i>		understanding the reference from unprofessional expression; software knowledge
VLC	<i>Hey, could you turn this video the right way up for me? And once it's flipped around, could you save it for me with the name '1984_Apple.mp4' on the main screen where all my files are?</i>		software knowledge; spatial judgment ability
Thunderbird	<i>Create a local folder called "Promotions" and create a filter to auto move the inbox emails whose subject contains "discount" to the new folder</i>		software knowledge
Thunderbird	<i>Attach the my AWS bill to the email. The bill is stored at /aws-bill.pdf. Don't close it or send it. I haven't finish all the contents.</i>		file management; extra requirement
Thunderbird	<i>I've got a bunch of email accounts in Thunderbird, and it's a hassle to check them one by one. Can you show me how to set up a unified inbox so I can see all my emails in one place?</i>		deep-hidden feature, need to be explored even by human users; pop-up window

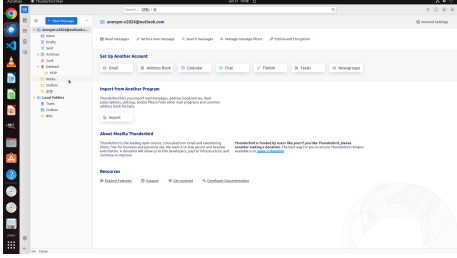
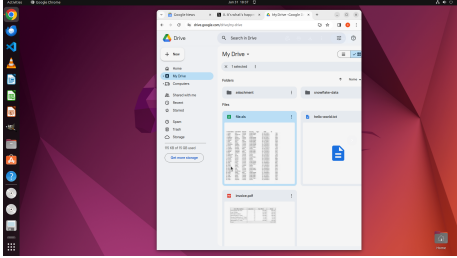
Continued on next page

Table 9 – continued from previous page

Related App(s)	Task Instruction	Screenshot of Initial State	Abilities Needed
VS Code	<i>Please modify VS Code's settings to disable error reporting for Python missing imports.</i>		software knowledge to deal with settings; reasoning to understand the cause and solution of the error
VS Code	<i>Please help me install the autoDocstring extension in VS Code.</i>		software knowledge to deal with Extensions; reasoning to search and install the extension successfully
GIMP	<i>Could you make the background of this image transparent for me?</i>		precise and intricate operations
GIMP	<i>Help me choose the yellow triangle and position it at the center of my picture.</i>		spatial perception and reasoning, as well as precise control of actions
Multiple (VLC+GIMP)	<i>Could you help me create an Animated GIF from a video file using VLC and GIMP from the source of video "src.mp4", 5-second clip beginning at 00:03?</i>		software knowledge to undergo sophisticated processes and ability to process multi-step procedure successfully

Continued on next page

Table 9 – continued from previous page

Related App(s)	Task Instruction	Screenshot of Initial State	Abilities Needed
Multiple (ThunderBird+Writer+Chrome)	<i>Help me export charts, graph or other images from docx files received in email "Lecture Document" in Notes folder and upload these png files to the figures/ folder in Google Drive for later use (use numbers to name them).</i>		ability to selectively export charts, graphs and images from docx file; software knowledge for google drive file upload
Multiple (Chrome+Calc)	<i>Could you help me extract data in the table from a new invoice uploaded to my Google Drive, then export it to a Libreoffice calc .xlsx file in the desktop?</i>		ability to do table data extraction; export data to .xlsx file

787 C Details of Baseline Methods

788 C.1 Hyper-Parameter of the Baseline Agents

789 We utilize the versions of gpt-3.5-turbo-16k, gpt-4-0125-preview, and gpt-4-vision-preview, re-
 790 spectively for GPT results, need to be noted that result could be changed from time since it is close-sourced.
 791 We also employ the gemini-pro and gemini-pro-vision versions for the Gemini models For all language
 792 models, we set the temperature parameter to 1.0, and top_p to 0.9, and the maximum number of tokens for
 793 generation is set to 1500. We set the maximum steps of interaction to 15 and the maximum time limits to 30
 794 minutes for all tasks since the agent could lead to a stuck environment under some unexpected cases.

795 C.2 Prompt Details

796 C.2.1 Prompt for A11y Tree, Screenshot and their Combination Setting

You are an agent which follow my instruction and perform desktop computer
 → tasks as instructed.
 You have good knowledge of computer and good internet connection and assume
 → your code will run on a computer for controlling the mouse and keyboard.
 For each step, you will get an observation of an image, which is the
 → screenshot of the computer screen and you will predict the action of the
 → computer based on the image.

You are required to use `pyautogui` to perform the action grounded to the
 → observation, but DONOT use the `pyautogui.locateCenterOnScreen` function
 → to locate the element you want to operate with since we have no image of
 → the element you want to operate with. DONOT USE `pyautogui.screenshot()`
 → to make screenshot.
 Return one line or multiple lines of python code to perform the action each
 → time, be time efficient. When predicting multiple lines of code, make
 → some small sleep like `time.sleep(0.5);` interval so that the machine
 → could take; Each time you need to predict a complete code, no variables
 → or function can be shared from history

797

You need to to specify the coordinates of by yourself based on your
 ↳ observation of current observation, but you should be careful to ensure
 ↳ that the coordinates are correct.
 You ONLY need to return the code inside a code block, like this:
 ```python  
 # your code here  
 ```

Specially, it is also allowed to return the following special code:
 When you think you have to wait for some time, return ```WAIT```;
 When you think the task can not be done, return ```FAIL``` , don't easily say
 ↳ ```FAIL``` , try your best to do the task;
 When you think the task is done, return ```DONE```.

My computer's password is 'password', feel free to use it when you need sudo
 ↳ rights.
 First give the current screenshot and previous things we did a short
 ↳ reflection, then RETURN ME THE CODE OR SPECIAL CODE I ASKED FOR. NEVER
 ↳ EVER RETURN ME ANYTHING ELSE.

798

799 For all y tree setting and all y tree + screenshot setting, the prompts are basically the same, just replace the
 800 screenshot words with all y tree words.

801 C.2.2 Prompt for SoM Setting

You are an agent which follow my instruction and perform desktop computer
 ↳ tasks as instructed.
 You have good knowledge of computer and good internet connection and assume
 ↳ your code will run on a computer for controlling the mouse and keyboard.
 For each step, you will get an observation of the desktop by 1) a screenshot
 ↳ with interact-able elements marked with numerical tags; and 2)
 ↳ accessibility tree, which is based on AT-SPI library. And you will
 ↳ predict the action of the computer based on the image and text
 ↳ information.

You are required to use `pyautogui` to perform the action grounded to the
 ↳ observation, but DONOT use the `pyautogui.locateCenterOnScreen` function
 ↳ to locate the element you want to operate with since we have no image of
 ↳ the element you want to operate with. DONOT USE `pyautogui.screenshot()`
 ↳ to make screenshot.
 You can replace x, y in the code with the tag of the element you want to
 ↳ operate with. such as:
 ```python  
 pyautogui.moveTo(tag\_3)  
 pyautogui.click(tag\_2)  
 pyautogui.dragTo(tag\_1, button='left')  
 ```

When you think you can directly output precise x and y coordinates or there
 ↳ is no tag on which you want to interact, you can also use them directly.
 But you should be careful to ensure that the coordinates are correct.
 Return one line or multiple lines of python code to perform the action each
 ↳ time, be time efficient. When predicting multiple lines of code, make
 ↳ some small sleep like `time.sleep(0.5);` interval so that the machine
 ↳ could take; Each time you need to predict a complete code, no variables
 ↳ or function can be shared from history
 You need to to specify the coordinates of by yourself based on your
 ↳ observation of current observation, but you should be careful to ensure
 ↳ that the coordinates are correct.
 You ONLY need to return the code inside a code block, like this:
 ```python  
 # your code here  
 ```

802

```

Specially, it is also allowed to return the following special code:
When you think you have to wait for some time, return ````WAIT````;
When you think the task can not be done, return ````FAIL````, don't easily say
→ ````FAIL````, try your best to do the task;
When you think the task is done, return ````DONE````.

My computer's password is 'password', feel free to use it when you need sudo
→ rights.
First give the current screenshot and previous things we did a short
→ reflection, then RETURN ME THE CODE OR SPECIAL CODE I ASKED FOR. NEVER
→ EVER RETURN ME ANYTHING ELSE.

```

803

804 C.3 Input Settings

805 **Accessibility tree** We evaluate reasoning capabilities of LLM agents on textual descriptions of the observa-
806 tion in the form of accessibility trees. To make accessibility trees (usually millions of tokens) tractable for LLM
807 contexts, we filter out non-essential elements by their tag, visibility, availability, *etc* as described in C.4. Only
808 the *tag*, *name*, *text*, *position*, and *size* of the remaining elements are kept and concatenated by tab character in the
809 input. As the raw coordinates are provided within the accessibility tree, the LLM is required to ground its action
810 predictions to accurate coordinates.

811 **Screenshot** To evaluate the visual reasoning ability of VLMs, we give the raw screenshot of the VM directly
812 to the VLM, which is understand the screenshot and predict correct actions with precise coordinates. The raw
813 resolution of the screen is set to 1920×1080 . To investigate robustness to resolution changes, we perform
814 ablation studies by manually downsampling the screenshot.

815 **Screenshot + accessibility tree** We test whether the combination of screenshots with the accessibility tree
816 can improve the capacity of VLM for spatial grounding.

817 **Set-of-Marks** Set-of-Marks (SoM) [56] enhances the grounding capabilities of VLMs marking image seg-
818 ments with annotations like alphanumerics, masks, or boxes. We leverage the information from the filtered
819 accessibility tree and mark the elements on the screenshot with a numbered bounding box. Following VisualWe-
820 bArena [22] and UFO [59], we further combine the annotated screenshot with the text metadata from accessibility
821 tree, including the *index*, *tag*, *name*, and *text* of the elements. Instead of predicting precise coordinates, the VLM
822 is supposed to specify the action object by its number index, which will be mapped into our action space by
823 post-processing. Ablation studies are also conducted with different resolutions for SoM setting.

824 C.4 Accessibility Tree Filtering

825 Since the original tree is large (usually over 1 million tokens in XML format), we filter the accessibility tree
826 nodes by their tags, visibilities, availabilities, *etc*. The concrete rules are illustrated in the following Table 10.

Table 10: Criteria for keeping all y tree nodes on Ubuntu and Windows platforms

Condition	Ubuntu	Windows
Node Tags	document*, *item, *button, *heading, *label, *scrollbar, *searchbox, *textbox, *link, *tabelement, *textfield, *textarea, *menu, alert, canvas, check- box, combo-box, entry, icon, image, paragraph, scroll-bar, section, slider, static, table-cell, terminal, text, netuiribbontab, start, trayclockwclass, tray- dummysearchcontrol, uiimage, uiproperty, uiribboncommandbar	Same as Ubuntu
Showing	True	Not Applicable
Visible	True	True

Condition	Ubuntu	Windows
Enabled or Editable or Expandable or Checkable	True	True
Has Name or Text or Image	True	True
Position	≥ 0	≥ 0
Size	> 0	> 0

C.5 Set-of-Mark Implementation Details

Our methodology involves an initial transformation of the original observational image acquired from our environment into the standardized *SoM* format and be putted into VLMs together with a table of the marks with metadata information such as tags and names. This format consists of bounding boxes that bound the sub-images of interest, each associated with a corresponding integer mark. Notably, our approach incorporates the utilization of the accessibility tree to identify the bounding boxes associated with all clickable buttons within the current image observation, instead of using segmentation models like the original *SoM* paper. Some examples of screenshots after applying SoM are shown in Figure 8. We can observe the emergence of some low-quality, unmodelable tasks, and even misleading bounding boxes, depending on the level of support from the software ecosystem. This could be another reason for the poor performance of SoM. Future improvements could be made in this regard.

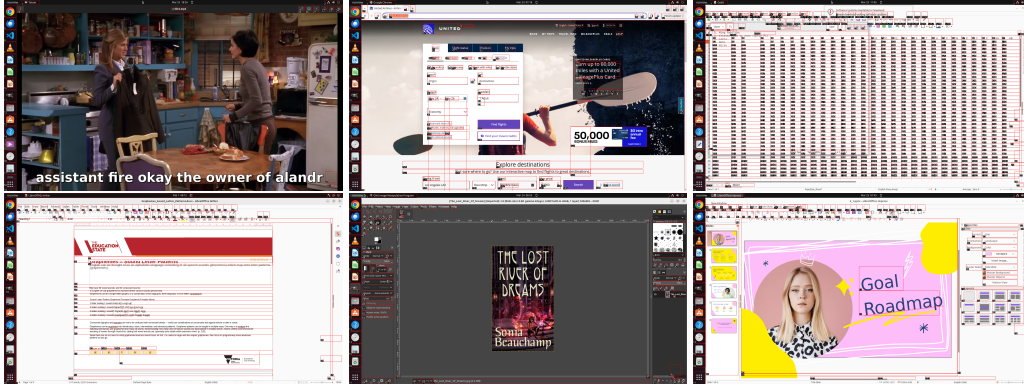


Figure 8: Showcase of example screenshots marked by SoM across different app GUI.

C.6 Full Results of Baseline Methods

Here we show the break-down results of baseline methods from different LLMs and VLMs for follow-up reference.

We have also compiled the distribution of steps taken by the GPT-4V model under our four settings: Accessibility Tree (A11y Tree), Screenshot, Screenshot combined with Accessibility Tree (Screenshot+A11y Tree), and Set-of-Mark. This data (as shown in Fig. 9) provides potential clues for future work. Overall, there are observable differences in how many steps the agent chooses to execute and when it decides to terminate under different settings. More detailed control and analysis of these behaviors can be explored in subsequent research efforts.

C.7 Running Time and Cost

We calculate and monitor the running time and cost for a portion of our experiments, which use the APIs of GPT-4V, Gemini-ProV, and Claude-3 Opus under different settings in April 2024. These results are shown in Table 12.

D Further Analysis

D.1 Performance by Task Difficulty, Feasibility and App Involved

We analyze the success rate across several additional subsets of tasks, as summarized in Tab. 13 and will be discussed in the following sections.

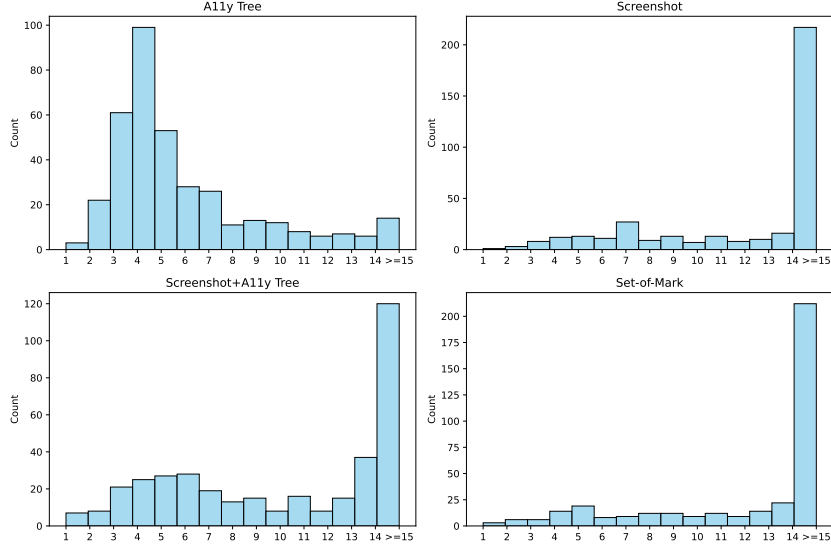


Figure 9: Distribution of steps taken by the GPT-4V based agents across four different settings.

Task difficulty We categorize the tasks based on the time required for human completion into three groups: 0~60s (Easy), 60s~180s (Medium), and greater than 180 seconds (Hard), as an indicator of difficulty. Across these groups, the model’s success rate drops as the required time increases, with tasks taking longer than 180 seconds becoming almost impossible to complete (considering we have infeasible examples for agent’s luckiness), whereas human performance across these three groups is 84.91%, 81.08% and 49.57%, showing a slight decline of the same trend but not to the extent of being unachievable.

Feasibility We also divide tasks into groups of tasks infeasible (e.g., deprecated features or hallucinated features) and tasks feasible, which requires the agents to have the ability to judge based on their own knowledge and exploration results. As shown in Tab. 13, we observe that agents currently perform slightly better in terms of infeasibility (16.67% to 13.34%), but overall, they are at a relatively low level. It is noteworthy that we also observe in some methods and settings (such as under the pure screenshot setting with the Gemini-Pro model), agents tend to easily output FAIL and refuse to continue trying. This situation leads to some false positives in infeasible tasks. The focus needs to be on improving overall performance.

Number of apps involved We also examined the performance based on whether the task involved apps software or within a single app. As shown in Tab. 13, the average performance for tasks involving a single app is low, at 13.74%, but still more than double the 6.57% observed for subsets of tasks involving workflows across multiple apps. Within single-app scenarios, tasks involving GUI-intensive Office apps generally performed the worst, with subsets such as LibreOffice Calc often scoring zero (we show more detailed results in App. C.6). These findings highlight the need for improved collaboration capabilities between software and enhanced proficiency in specific scenarios.

D.2 Allly tree-based Observation Length Distribution

The main experiment revealed the decisive role of the allly tree in performance within the current technological context. Even when we retain key attribute elements based on heuristic rules (keep nodes with tags of the document, item, button, heading, label, etc.), LLMs still require a sufficiently large context to process this information effectively. To further understand this, we sample some allly tree observations from OSWORLD and conducted the statistical analysis, as shown in Figure 10. The analysis indicates that a context length of 6000 is needed to accommodate about 90% of cases for a single observation.

Table 13: Success rate (SR) of GPT-4V (SoM) across different types of tasks.

Task Subset	% of Total	SR (↑)
Easy	28.72%	16.78%
Medium	40.11%	13.12%
Hard	30.17%	4.59%
Infeasible	8.13%	16.67%
Feasible	91.87%	13.34%
Single-App	72.63%	13.74%
Multi-App Workflow	27.37%	6.57%

Table 11: Detailed success rates of baseline LLM and VLM agents on OSWORLD, divided by apps (domains): OS, LibreOffice Calc, LibreOffice Impress, LibreOffice Writer, Chrome, VLC Player, Thunderbird, VS Code, GIMP and Workflow which is comprehensive with multiple apps, for gaining insights from interfaces and operation logics.

Inputs	Model	Success Rate (\uparrow)									
		OS	Calc	Impress	Writer	VLC	TB	Chrome	VSC	GIMP	Workflow
A11y	Mixtral-8x7B	12.50	0.00	0.39	4.34	10.22	6.67	2.17	8.69	3.85	0.10
	GPT-3.5	4.17	2.13	6.77	4.35	6.53	0.00	2.17	0.00	0.00	1.62
	Gemini-Pro	4.17	0.00	2.13	4.35	12.41	0.00	2.17	0.00	7.69	0.63
	GPT-4	20.83	0.00	6.77	4.35	23.53	26.67	26.09	30.43	23.08	2.97
	Gemini-Pro-1.5	12.50	2.13	2.13	4.35	6.53	0.00	10.87	8.70	0.00	3.60
	Llama-3-70B	4.17	0.00	0.39	8.70	6.53	0.00	2.17	0.00	0.00	0.63
	GPT-4o	20.83	6.38	6.77	8.69	12.41	20.00	17.39	21.74	11.54	7.56
	Qwen-Max	29.17	0.00	2.52	13.04	8.95	0.00	10.87	8.70	11.54	2.61
Screen	CogAgent	4.17	0.00	0.00	4.34	6.53	0.00	2.17	0.00	0.00	0.00
	Gemini-ProV	8.33	0.00	6.77	4.35	12.41	0.00	6.52	8.70	23.08	2.08
	GPT-4V	12.50	0.00	2.52	4.35	18.34	0.00	6.52	0.00	7.69	6.04
	Claude-3-Opus	4.17	0.00	2.52	4.34	6.53	0.00	2.17	0.00	3.84	2.61
	Gemini-Pro-1.5	12.50	0.00	13.16	8.70	6.53	0.00	2.17	0.00	11.54	3.60
	GPT-4o	8.33	0.00	6.77	4.35	16.10	0.00	4.35	4.35	3.85	5.58
Screen + A11y	CogAgent	4.17	2.17	0.00	4.35	6.53	0.00	2.17	0.00	0.00	0.10
	Gemini-ProV	4.17	2.13	6.77	4.35	18.30	0.00	4.35	0.00	0.00	1.52
	GPT-4V	16.67	0.00	6.77	21.73	24.18	33.33	21.74	21.74	15.38	4.59
	Claude-3-Opus	12.50	2.13	4.65	4.34	18.30	0.00	2.17	8.69	7.69	0.99
	Gemini-Pro-1.5	12.50	0.00	4.65	8.70	12.41	0.00	8.70	4.35	11.54	1.56
	GPT-4o	41.67	4.26	6.81	8.70	9.50	6.67	15.22	30.43	0.00	7.46
SoM	CogAgent	4.17	2.17	0.00	4.34	6.53	0.00	2.17	0.00	0.00	0.00
	Gemini-ProV	4.17	0.00	0.39	4.34	6.53	0.00	0.00	0.00	0.00	0.63
	GPT-4V	8.33	8.51	6.38	13.04	10.66	33.33	23.91	17.39	11.54	6.57
	Claude-3-Opus	12.50	2.13	0.39	8.70	6.53	13.33	17.39	0.00	11.54	4.49
	Gemini-Pro-1.5	16.67	0.00	10.64	4.35	24.18	13.33	8.70	4.35	15.38	3.60
	GPT-4o	20.83	0.00	6.77	4.35	6.53	0.00	4.35	4.35	0.00	3.60
Human Performance		75.00	61.70	80.85	73.91	70.59	46.67	78.26	73.91	73.08	73.27

Table 12: Summary of expected time and budget cost for different settings. Calculated in April 2024; these values may change with updates from the API providers.

Setting	Expected Time	Budget Cost (Full Test Set/Small Test Set)
GPT-4V (screenshot)	10h	\$100 (\$10)
Gemini-ProV (screenshot)	15h	\$0 (\$0)
Claude-3 Opus (screenshot)	15h	\$150 (\$15)
GPT-4V (a11y tree, SoM, etc.)	30h	\$500 (\$50)

887 D.3 Performance across Different Operating Systems

888 Another key challenge in building universal digital agents
889 is ensuring that these agents can maintain efficient and con-
890 sistent performance across different operating system envi-
891 ronments. The differences between OS and their software
892 ecosystems can significantly impact an agent’s observation
893 and action spaces, leading to performance uncertainties. Here,
894 we explore and analyze the correlation between the success
895 of agents in completing tasks on Windows after migrating
896 from Ubuntu using examples from OSWORLD.

Table 14: Comparison of model performance and correlation across operating systems.

OS	SR (%)	Correlation Coefficient
Ubuntu	4.88	0.7
Windows	2.55	

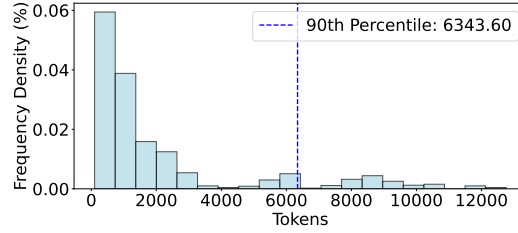


Figure 10: The length distribution of a11y tree as observation from sampled trajectories.

We enhance the functionality of the OSWORLD environment to support setting up initial experiment states, final evaluations, and obtaining observations such as the a11y tree and screenshots in Windows OS. Additionally, we have made example-wise fine-tuning modifications to the existing subset in OSWORLD for migration to Windows. We conduct evaluations using the GPT-4V screenshot-only method and present the correlation of performance across the two operating systems. As shown in Tab. 14, the model’s performance on Ubuntu and Windows is 4.88% and 2.55%, respectively, with a correlation coefficient of 0.7, despite the differences in their observation spaces. This implies that insights and methodologies developed within the OSWORLD framework can be effectively transferred to Windows environments with a high degree of reliability.

D.4 Qualitative Analysis

In this section we highlight representative examples of success, failure, and surprising outcomes, alongside a comparative study between GPT-4V and Claude-3 agents, to elucidate the unique challenges and insights our environment introduces. See App. D.5 for more details.

Success and failure cases We find agents, particularly based on GPT-4V, can successfully solve tasks that involve complex problem-solving or creative thinking, showcasing the advanced understanding and processing capabilities of the model already. One successful task is shown in the first row of Figure 11. The agent is requested to extract subtitle files from the video stream and save them locally. The agent first divides the screen into two parts, with the VLC application window on the left and the terminal window open on the right, and uses the ffmpeg command twice. The first use removes the subtitles embedded in the original video, and the second use saves the extracted subtitles locally.

Task Instruction: I downloaded an episode of Friends to practice listening, but I don't know how to remove the subtitles. Please help me remove the subtitles from the video and export it as "subtitles.srt" and store it in the same directory as the video.

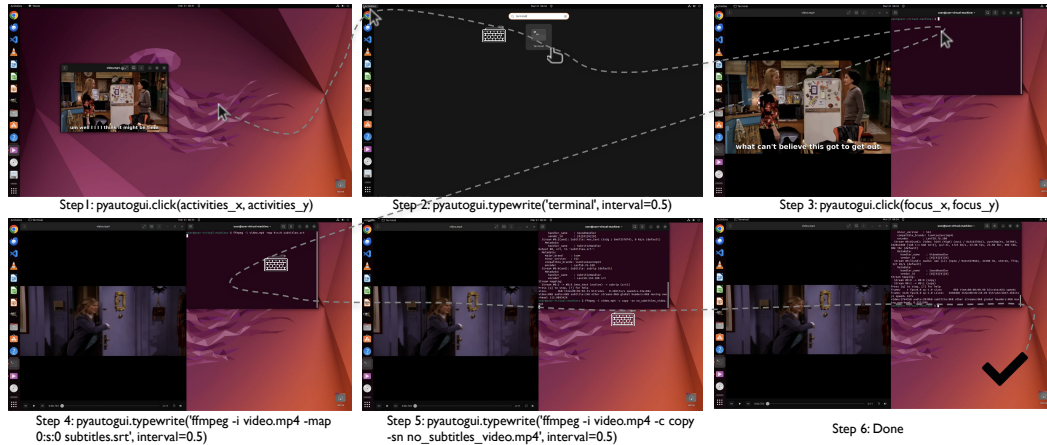


Figure 11: The agent successfully understood the complex task instructions, extracted the subtitle file from the video, and generated a pure video without embedded subtitles.

Despite the successes, there are notable failures that highlight the limitations of current models. In the task of “center-aligning the title of the document” (Fig. 12 line 1), the agent fails to ground the relatively simple requirement of “center alignment of texts”, performing many useless actions such as selecting irrelevant words, opening irrelevant menus, *etc.*

Moreover, we find that the agent lacks prior knowledge in using software, performing poorly in many specialized tasks (as shown in Fig. 13, with GIMP, LibreOffice Calc, and Chrome selected). Taking GIMP as an example,

for the instruction “reduce brightness” the agent does not know which menu in the toolbar is for brightness adjustment and instead randomly tries until exhausting the maximum number of steps.

Common errors by GPT-4V agents Among the 550 failed examples from different settings in our sample, more than 75% exist *mouse click inaccuracies*, which is the most common error. The agent fails to click the correct coordinates despite planning detailed and accurate steps in their code comments, indicating strong planning but weak execution capabilities. Mouse click inaccuracies lead to two other frequent errors: *repetitive clicks* and *environmental noise dilemma*. Repetitive clicks occur when the agent repeatedly misclicks, adjusts, and fails, consuming too many steps. Environmental noise arises from clicking unintended objects, causing pop-ups, or opening unrelated applications. Due to a lack of prior knowledge about most professional software, it falls into a mismatch dilemma between the actions taken and the current state, and don’t know how to get back to normal. Moreover, the agent lacks basic human-like cognition of web pages, such as not closing pop-ups in real-world web pages or being attracted by advertisement content, which affects its original correct judgment. Failures also arise from *misinterpretation of instructions* and *visual oversight*, highlighting the need for improvement in language and visual processing. See App. D.5.2 for the specific execution process.

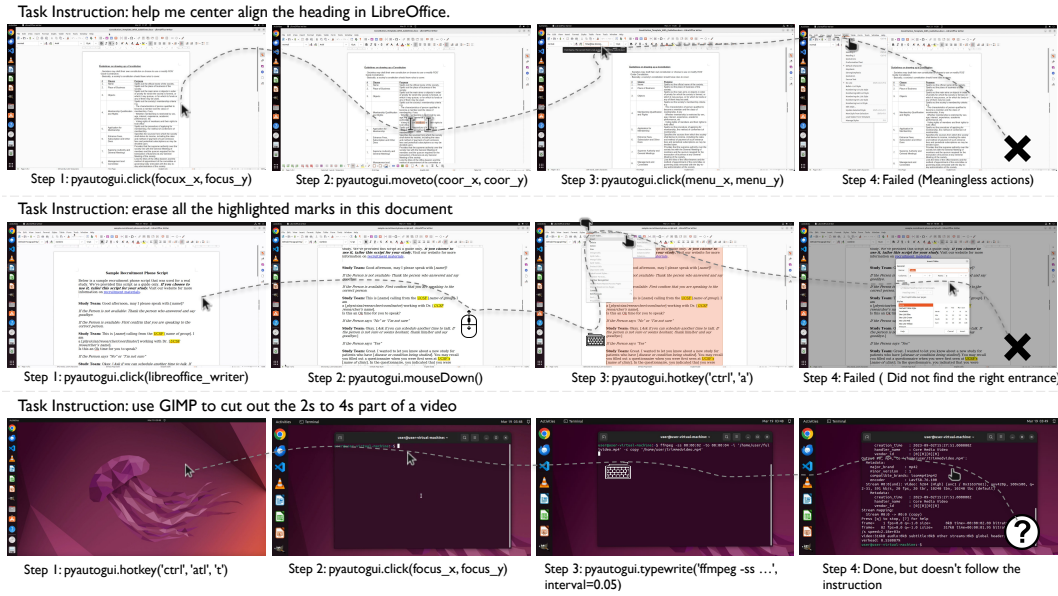


Figure 12: Screenshots of the three examples mentioned in the quality analysis. The first line is an example of GPT-4V failing at a very simple task, the second line is one example where agents face more difficulty than humans, and the third line is one example that is more difficult for humans than for agents.

Discrepancies in task difficulty between agent and human We identify notable disparities in the perceived difficulty of tasks between humans and AI agents. Tasks that are intuitively simple for humans often present substantial challenges to agents, and conversely, tasks that humans find demanding can be more straightforward for agents to execute. You can find more details in Fig. 16 and App. D.5.3.

Tasks where humans outperform agents These tasks mainly involve text-based and design-related work, such as “bold the font on this slide and add notes” or “erase all the highlighted marks in this document” (Fig. 12 Line 2). Since the Internet lacks such fine-grained data as the software execution process, the agent also lacks the corresponding training process, so its grounding ability is not good enough. The lack of understanding of GUI logic also causes poor performance on operations like selecting and scrolling.

Tasks where agents outperform humans Tasks that the agent considers simple but humans find difficult are concentrated in “code solvability tasks”, such as “monitor the system CPU for 30s and output the results” and “force close a process”. These tasks require little or no GUI interaction and can be completed by executing complex codes and instructions. It’s worth noting that completing through code sometimes mismatches with human instructions. In the task “use GIMP to cut out the 2s to 4s part of a video,(Fig. 12 Line 3)” the agent used “ffmpeg” command to complete the video cropping, ignoring the “use GIMP” requirement in the instructions.

Surprisingly, we discovered that agents are as prone to inefficiency in mechanically repetitive tasks, such as copying, pasting, and batch editing of Excel sheets, as humans. Humans frequently commit careless errors during execution. The shortcomings in agents stem either from the absence of an API or from insufficient training data

954 related to the API, hindering their ability to efficiently process tasks in batches. Furthermore, sluggish response
955 times can cause tasks to either time out or surpass the maximum allowed steps.

956 **Comparative analysis: Claude-3 vs. GPT-4V** Although Claude outperforms GPT-4 in many benchmarks
957 such as GSM8K, HumanEval, *etc.*, in our main experiment, we find that Claude has an average lower accuracy
958 rate compared to GPT-4V by 2.84% to 7.76%. We find that Claude can provide satisfactory high-level solutions,
959 but its grounding ability contains hallucinations in detail. For instance, Claude would interpret double-clicking a
960 file as selecting it instead of opening it, treat column B in LibreOffice Calc software as column C, and enter
961 text in the VS Code text replacement box without clicking on global replace. This shows that Claude can align
962 well with human planning in problem-solving, but lacks excellent grounding ability when it comes to execution.
963 Details can be seen in Fig. 17 and App. D.5.4.

964 D.5 Qualitative Analysis Examples

965 Here we give the specific code execution process for the examples in the qualitative analysis in §D.4 and add
966 more error cause analysis, trying to get insights for further improvement.

967 D.5.1 Success and Failure Cases

Success Task: *I downloaded an episode of Friends to practice listening, but I don't know how to remove the subtitles. Please help me remove the subtitles from the video and export it as "subtitles.srt" and store it in the same directory as the video.*

```
# Action 1
import pyautogui
import time
# Move the mouse to the top bar of the media player window to access the menu
pyautogui.moveTo(800, 10)
# Assuming the menu is at the top of the screen
pyautogui.click()
time.sleep(0.5)

# Action 2
# Move the mouse to the Activities corner to open the GNOME Activities Overview
pyautogui.moveTo(10, 10)
# Assuming the Activities corner is at the top-left
pyautogui.click()
# Type 'terminal' to search for the Terminal application
pyautogui.write('terminal')
time.sleep(0.5)
# Press Enter to open the Terminal application
pyautogui.press('enter')

# Action 3
# Focus the Terminal window and type the command to extract subtitles
pyautogui.click(640, 360)
# Clicking in the center of the screen to focus the Terminal
time.sleep(0.5)
pyautogui.write('ffmpeg -i video.mp4 -map 0:s:0 subtitles.srt')
pyautogui.press('enter')

# Action 4
# Focus the Terminal window and type the command
# to create a new video without subtitles
pyautogui.click(1100, 350)
# Clicking inside the Terminal window to focus it
pyautogui.write('ffmpeg -i video.mp4 -c copy -sn no_subtitles_video.mp4')

Done!
```

968

Failure Task: *Help me center align the heading in LibreOffice.*

```
# Action 1
import pyautogui
import time
# Move the mouse to the LibreOffice Writer document icon and double-click
pyautogui.moveTo(1100, 720, duration=1)
pyautogui.doubleClick()

# Action 2
# Move the mouse to the beginning of the heading text
and click and drag to select the heading
pyautogui.moveTo(200, 230, duration=1)
pyautogui.mouseDown()
pyautogui.moveTo(800, 230, duration=1)
pyautogui.mouseUp()
# Move the mouse to the center align button in the toolbar and click
pyautogui.moveTo(400, 150, duration=1)
pyautogui.click()

# Action 3
# Assuming the text is still selected,
move the mouse to the center align button in the toolbar and click
pyautogui.moveTo(400, 150, duration=1)
pyautogui.click()

# Action 4
# Move the mouse to the center align button in the toolbar and click
pyautogui.moveTo(340, 80, duration=1)
pyautogui.click()

Failed.
```

969

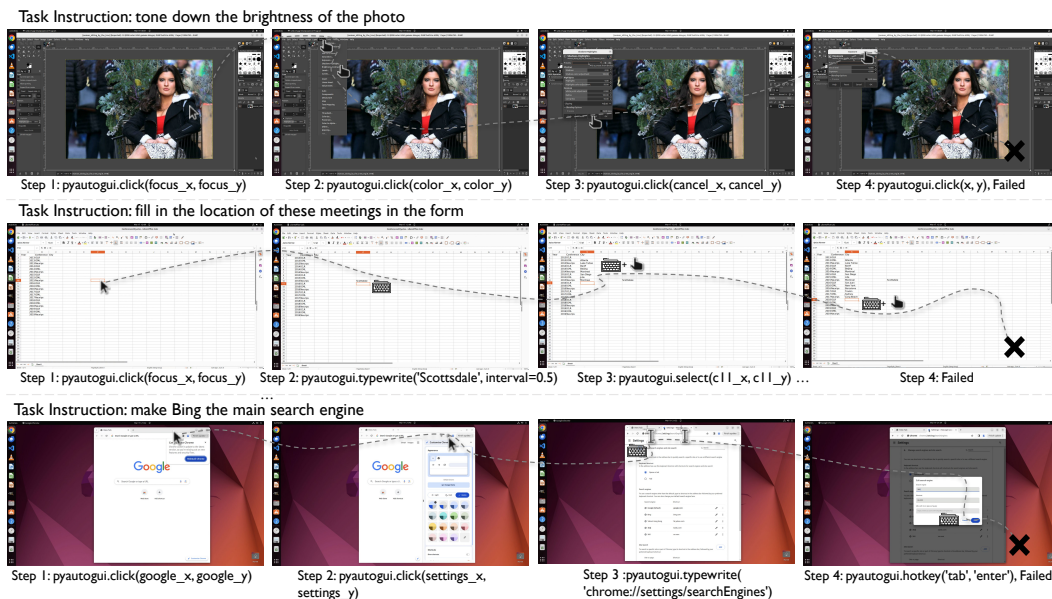


Figure 13: The figure shows examples of usage errors in three types of software. The first row is for GIMP, the second row is for LibreOffice Calc, and the third row is for Google Chrome. Detailed code execution processes are provided below.

GIMP Task: *Could you tone down the brightness of my photo?*

```
# Action 1
import pyautogui
import time
# Coordinates for the image icon on the desktop
# Adding half of the width and height to get the center
image_icon_x, image_icon_y = 1793 + 60, 920 + 17
# Double-click on the image icon
pyautogui.doubleClick(image_icon_x, image_icon_y)

# Action 2
# Coordinates for the `Colors` menu
# Adding half of the width and height to get the center
colors_menu_x, colors_menu_y = 344 + 28, 64 + 12
# Click on the `Colors` menu
pyautogui.click(colors_menu_x, colors_menu_y)

# Action 3
# Coordinates for the `Brightness-Contrast...` option in the `Colors` dropdown
# Assuming each menu item is approximately 28 pixels apart
brightness_contrast_option_x, brightness_contrast_option_y
= 344 + 28, 64 + 12 + 28 * 6
# Click on the `Brightness-Contrast...` option
pyautogui.click(brightness_contrast_option_x, brightness_contrast_option_y)

# Action 4
# Coordinates for the `Cancel` button in the `Shadows-Highlights` dialog
# Adding half the width and height to get the center
cancel_button_x, cancel_button_y = 375 + 42, 625 + 16
# Click the `Cancel` button
pyautogui.click(cancel_button_x, cancel_button_y)
# Coordinates for the `Brightness-Contrast...` option in the `Colors` dropdown
# Adjusted index for Brightness-Contrast option
brightness_contrast_option_x, brightness_contrast_option_y
= 344 + 28, 64 + 12 + 28 * 5
pyautogui.click(brightness_contrast_option_x, brightness_contrast_option_y)

# Action 5
# Coordinates for the `Cancel` button in the `Exposure` dialog
# Adding half the width and height to get the center
cancel_button_x, cancel_button_y
= 375 + 42, 391 + 16
pyautogui.click(cancel_button_x, cancel_button_y)
# Coordinates for the `Brightness-Contrast...` option in the `Colors` dropdown
# Adjusted index for Brightness-Contrast option
brightness_contrast_option_x, brightness_contrast_option_y
= 344 + 28, 64 + 12 + 28 * 6
pyautogui.click(brightness_contrast_option_x, brightness_contrast_option_y)

# Action 6
# Coordinates for the `Cancel` button in the `Color Balance` dialog
# Adding half the width and height to get the center
cancel_button_x, cancel_button_y
= 375 + 42, 588 + 16
pyautogui.click(cancel_button_x, cancel_button_y)
# Since the `Brightness-Contrast...` option was not correctly opened before,
we will use an estimated position based on the dropdown menu.
# Assuming `Brightness-Contrast...` is the second option in the `Colors` menu.
brightness_contrast_option_x, brightness_contrast_option_y
= colors_menu_x, colors_menu_y + 28 * 2
pyautogui.click(brightness_contrast_option_x, brightness_contrast_option_y)
```

```

# Action 7
# Coordinates for the 'Cancel' button in the 'Color Temperature' dialog
# Adding half the width and height to get the center
cancel_button_x, cancel_button_y = 375 + 42, 403 + 16
pyautogui.click(cancel_button_x, cancel_button_y)
# Since the 'Brightness-Contrast...' option was not correctly opened before,
we will use an estimated position based on the dropdown menu.
# Assuming 'Brightness-Contrast...' is the second option in the 'Colors' menu.
brightness_contrast_option_x, brightness_contrast_option_y
= colors_menu_x, colors_menu_y + 28 * 2
# Click on the 'Brightness-Contrast...' option
pyautogui.click(brightness_contrast_option_x, brightness_contrast_option_y)

# ... More Wrong Duplicate Actions; Can not find the right coordination.

# Failed

```

971

Libreoffice calc Task: *I now want to count the meeting cities of the three machine learning conferences in the past ten years from 2013 to 2019(including 2013 and 2019). I have listed the names and years of the conferences in excel. Please fill in the vacant locations.*

```

# Action 1
# Move to cell C2 which is the first empty cell under 'City' for ICLR 2013
pyautogui.click(319, 222)
# Coordinates of cell C2
time.sleep(0.5)
# Type the city name for ICLR 2013
pyautogui.write('Scottsdale')
# Press Enter to move to the next cell
pyautogui.press('enter')

# Action 2
# Move to cell C3 which is the city cell for ICML 2013
pyautogui.click(319, 268)
# Coordinates of cell C3
# Type the city name for ICML 2013
pyautogui.write('Atlanta')
# Press Enter to move to the next cell
pyautogui.press('enter')

# Action 3
# Move to cell C4 which is the city cell for NeurIPS 2013
pyautogui.click(319, 289)
# Coordinates of cell C4
# Type the city name for NeurIPS 2013
pyautogui.write('Lake Tahoe')
# Press Enter to move to the next cell
pyautogui.press('enter')

... More copy-and-paste operations

# Action 15
# Move to cell C16 which is the city cell for NeurIPS 2017
pyautogui.click(319, 541)
# Coordinates of cell C16
# Type the city name for NeurIPS 2017
pyautogui.write('Long Beach')
# Press Enter to move to the next cell
pyautogui.press('enter')

Failed.

```

972

Chrome Task: *Can you make Bing the main search thingy when I look stuff up on the internet?*

```
# Action 1
pyautogui.click(x=1280, y=50)
# Approximate coordinates for the `More` button
pyautogui.move(0, 150)
# Moving the cursor down to `Settings` in the drop-down menu
pyautogui.click()
# Clicking on `Settings`

# Action 2
pyautogui.click(tag_31)
# Click on the `Finish update` button to
approximate the location of `More` button
pyautogui.move(-50, 0)
# Move left from the `Finish update` button to approximate the `More` button
pyautogui.click()
# Click on the `More` button
time.sleep(0.5)
pyautogui.move(0, 150)
# Move down to the `Settings` option in the drop-down menu
pyautogui.click()
# Click on `Settings`

# Action 3
# Click on the `Customise Chrome` button which should be the `More` button
time.sleep(1)
# Wait for the dropdown menu to appear
# The `Settings` option is usually located towards the bottom of
the dropdown menu, so I'll move the cursor down to reach it
pyautogui.move(0, 200)
# Move down to the `Settings` option in the drop-down menu
pyautogui.click()
# Click on `Settings`

# Action 4
pyautogui.click(tag_47)
# Click on the `Close` button to close the `Customise Chrome` menu
pyautogui.click(tag_25)
# Click on the `New Tab` button to open a new tab
time.sleep(1)
# Wait for the new tab to open
pyautogui.typewrite('chrome://settings/searchEngines')
# Type the direct URL to the search engine settings page
pyautogui.press('enter')

# Action 5
# Simulate pressing the `Tab` key multiple times to navigate through the options
for _ in range(10):
    # Assuming it might take around 10 tabs to reach the search engine list
    pyautogui.press('tab')
    time.sleep(0.5)
# Wait for the focus to move to the next element
# Simulate pressing the `Enter` key to select an option,
which would be done once we reach the Bing option
pyautogui.press('enter')

# Action 6
pyautogui.click(tag_42)
# Click on the `Cancel` button to exit the edit mode for the `360` search engine

Failed.
```

973

974 **D.5.2 Common Errors by GPT-4V Agents**

975 In the following two figures (Figure 15 and Figure 14), we take the web page tasks as examples to show the two
976 most common types of errors in GPT4-V: *mouse click inaccuracies* and *inadequate handling of environmental noise*.

Task Instruction: On next Monday, look up a flight from Mumbai to Stockholm.

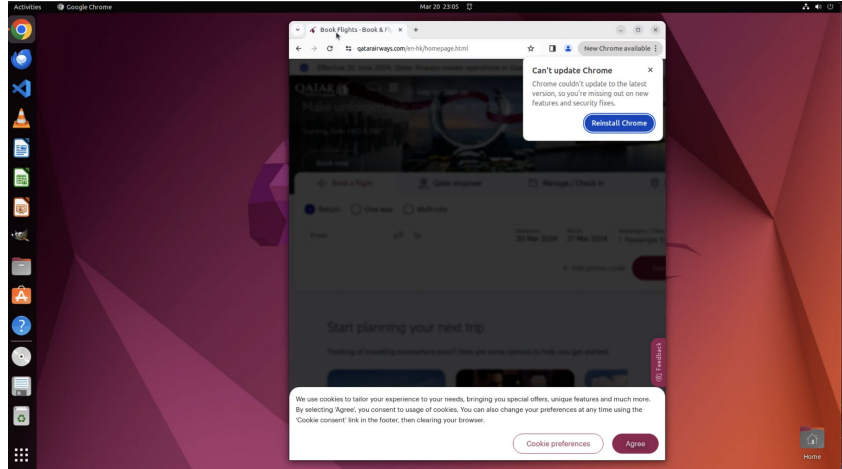


Figure 14: The error shown in the figure is due to mouse click inaccuracies. The agent was supposed to click on the product category images of the Nike official store and follow the instructions to search for women’s jerseys priced over 60 dollars. However, due to a coordinate calculation error, it mistakenly clicked on the favorite button, creating a significant discrepancy between the current state and the target state. Under these circumstances, the agent is unable to backtrack to the previous state and start over.

977

Task Instruction: Browse the list of women's Nike jerseys over \$60.

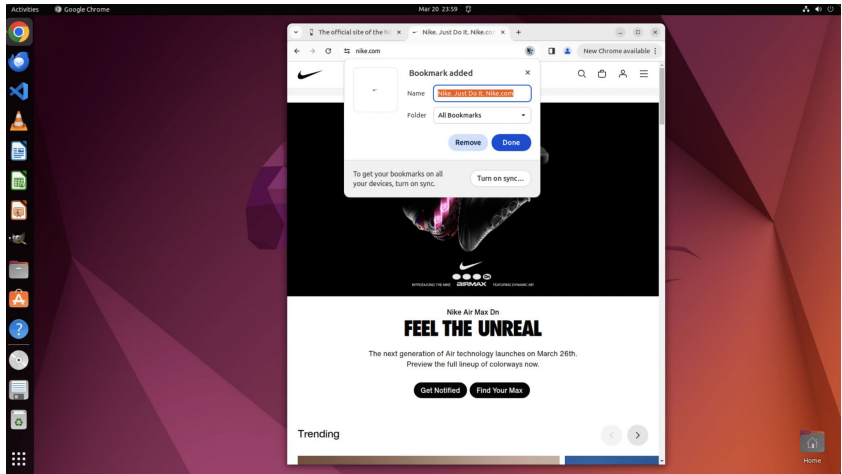


Figure 15: The error shown in the figure is due to inadequate handling of environmental noise. When the agent encounters web pages with cookies or pop-up ads, its unfamiliarity with the structure of such pages leads to attempts to interact directly with the web page without bypassing the cookies or pop-ups. Since the pop-ups or cookie requests are not closed, all actions are ineffective.

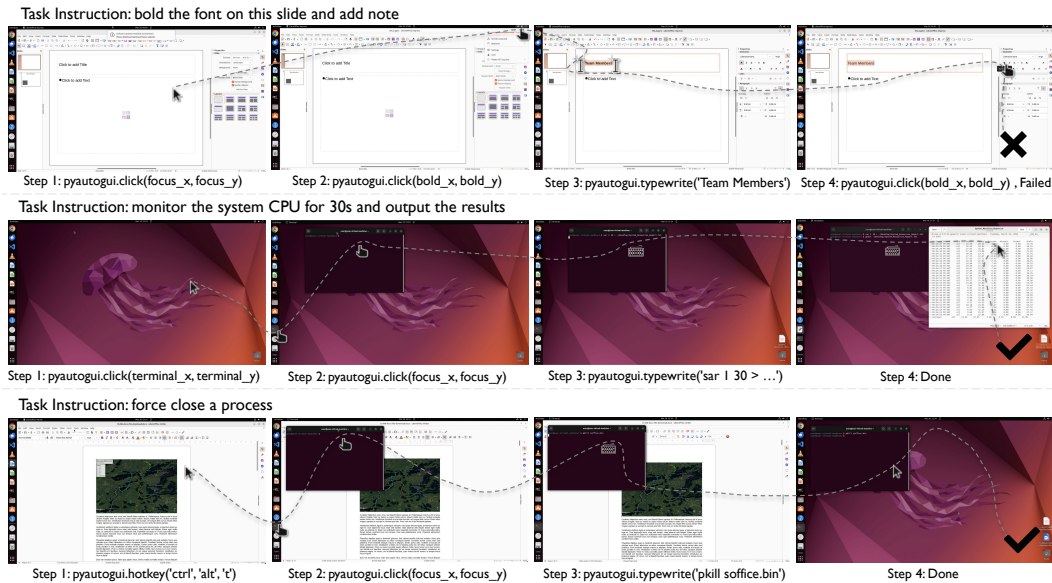


Figure 16: Supplementary examples of tasks performed by humans and agents.

Human > Agent Task: *erase all the highlighted marks in this document*

```
# Action 1
import pyautogui
import time
# Move to the 'Character Highlighting Color' button and click
pyautogui.moveTo(1020, 122)
pyautogui.click()

# Action 2
# Assuming the 'No Color' option is the first option in the dropdown
# Move the mouse slightly below the 'Character Highlighting Color' button
# to click on 'No Color'
pyautogui.moveTo(1020, 180)
time.sleep(0.5)
pyautogui.click()

# Action 3
# Move to the 'Save' button and click
pyautogui.moveTo(184, 108)
time.sleep(0.5)
pyautogui.click()

# Failed.
```

979

Human > Agent Task: *On page 2, add a note same to the title into the slide. Make the font of title bold.*

```
# Action 1
# Click on the vertical scrollbar to scroll down and reveal more slides
pyautogui.click(tag_43)

# Action 2
# Click on the second slide thumbnail to select it
```

980

```

pyautogui.click(tag_2)

# Action 3
# Click on the text box to add text
pyautogui.click(tag_42)
pyautogui.typewrite('Team Members', interval=0.1)
# Type the title
pyautogui.hotkey('ctrl', 'a')
# Select all text
pyautogui.hotkey('ctrl', 'b')
# Make the text bold

# Action 4
# Click on the Bold button to apply bold formatting
pyautogui.click(tag_68)

Failed.

```

981

Agent > Human Task: *use GIMP to cut out the 2s to 4s part of a video*

```

# Action 1
import pyautogui
import time
# Move the mouse to the assumed terminal icon position
# at the bottom of the dock and click to open it
# Assuming the terminal icon is at the bottom of the dock
pyautogui.moveTo(70, 1350, duration=1)
pyautogui.click()
# Type the command to use ffmpeg to trim the video
ffmpeg_command = `ffmpeg -ss 00:00:02 -to 00:00:04 -i /home/user/fullvideo.mp4
-c copy /home/user/trimmedvideo.mp4`
pyautogui.write(ffmpeg_command, interval=0.05)
pyautogui.press('enter')

# Action 2
# Close the search dialog
pyautogui.press('esc')
# Open the terminal using a keyboard shortcut (Ctrl+Alt+T is a common shortcut
# for opening a terminal in many Linux distributions)
pyautogui.hotkey('ctrl', 'alt', 't')

# Action 3
# Move the mouse to the terminal and click to focus on the terminal window
pyautogui.moveTo(260 + 722/2, 103 + 434/2, duration=1)
pyautogui.click()
# Type the command to use ffmpeg to trim the video
pyautogui.write(ffmpeg_command, interval=0.05)
pyautogui.press('enter')

# Done, but not follow the demanding in instruction.

```

982

Agent > Human Task: *I want to understand the resource usage of my Ubuntu system under normal workloads. Please use the 'sar' command in the 'sysstat' toolkit to monitor system activity, evaluate the status once every second for 30 seconds, output the results to "System Resources Report.txt" under Desktop.*

```

# Action 1
import pyautogui
import time

```

983


```

# Press the keyboard shortcut to open the terminal
pyautogui.hotkey('ctrl', 'alt', 't')
time.sleep(1)
# Wait a moment for the terminal to open

# Action 2
# Click inside the terminal to focus on it
pyautogui.click(600, 300)
# Assuming the terminal is in the upper left area of the screen
n# Type the command to monitor system activity
pyautogui.typewrite('sar 1 30 > ~/Desktop/System Resources Report.txt')

# Action 3
# Click on the 'System Resources Report.txt' file to select it
pyautogui.click(1850, 940)
# Coordinates taken from the label position and size
time.sleep(0.5)
# Right-click to open the context menu
pyautogui.click(1850, 940, button='right')
# Move to the 'Open With Other Application' option in the context menu
pyautogui.move(0, 70)
# Assuming 'Open With Other Application' is 70 pixels
down from the right-click position
# Click on 'Open With Other Application'
pyautogui.click()
time.sleep(0.5)
# Type 'gedit' to select the text editor
pyautogui.typewrite('gedit')
# Press 'Enter' to open the file with gedit
pyautogui.press('enter')

# Action 4
# Click inside the terminal to ensure it's in focus
pyautogui.click(500, 200)
# Coordinates are estimated to be within the terminal window
# Type the command to open the file with gedit
pyautogui.typewrite('gedit ~/Desktop/System Resources Report.txt')

Done!

```

984

Agent > Human Task: *Hey, my LibreOffice Writer seems to have frozen and I can't get it to close normally. Can you help me force quit the application from the command line? I'm on Ubuntu and I don't want to restart my computer or lose any other work I have open.*

```

# Action 1
import pyautogui
import time
# Open a terminal window
pyautogui.hotkey('ctrl', 'alt', 't')
time.sleep(1)
# Type the command to kill all instances of LibreOffice Writer
pyautogui.typewrite('killall soffice.bin')

# Action 2
# Open a terminal window
pyautogui.hotkey('ctrl', 'alt', 't')
time.sleep(1)
# Type the command to kill all instances of LibreOffice Writer
pyautogui.typewrite('killall soffice.bin')

# Action 3

```

985

```
# Ensure the terminal window is focused
pyautogui.click(500, 200)
# Assuming the terminal is at this position
# Type the pkll command to forcefully kill the LibreOffice Writer application
pyautogui.typewrite('pkll -9 soffice.bin')
```

Done!

986

987 D.5.4 Comparative Analysis: Claude-3 vs. GPT-4V

Task Instruction: find the path of the file named "secret.docx" and copy it to the clipboard.

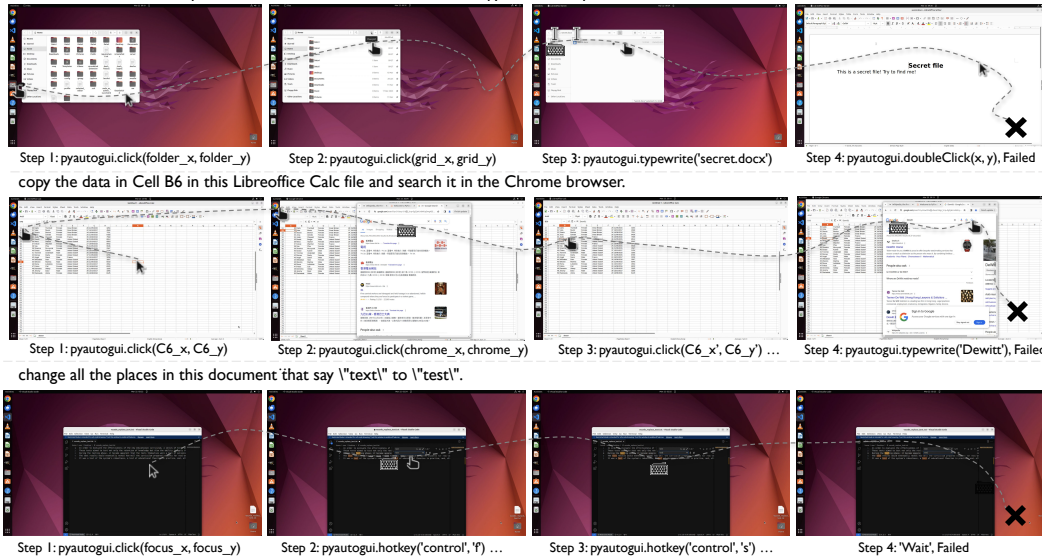


Figure 17: Screenshots of error example analysis of Claude-3 Opus.

Claude3 Failed Task1: *I remember there is a file named "secret.docx" on this computer, but I can't remember where it is. Please find the path where this file is stored and copy it to the clipboard.*

```
# Action 1
pyautogui.click(tag_10)
# click the Files icon to open the file manager

# Action 2
pyautogui.click(tag_21)
# click the search icon to open the search bar

# Action 3
pyautogui.typewrite('secret.docx')
# type the file name to search for
pyautogui.press('enter')
# press enter to start the search

# Action 4
pyautogui.doubleClick(tag_52)
# double click the secret.docx file to select it
pyautogui.hotkey('ctrl', 'c')
# press Ctrl+C to copy the selected file path to clipboard

Failed.
```

988

Claude3 Failed Task2: *copy the data in Cell B6 in this Libreoffice Calc file and search it in the Chrome browser.*

```
# Action 1
import pyautogui
import time
# Scroll down to view more search results if needed
pyautogui.press('pagedown')
time.sleep(1)
pyautogui.press('pagedown')
time.sleep(2)
# The answer appears to be in the snippet under the first search result
# `TV 31. u76f4u64adu4e2d.` which translates to `TV 31. Live.`
# So channel 31 in Hong Kong is RTHK TV 31
print(`Found the answer. Channel 31 in Hong Kong is RTHK TV 31.`)
# The original task of copying cell B36 from the spreadsheet
and searching it in the browser is complete
print(`Task complete.`)

# Action 2
# Activate the Chrome browser window
pyautogui.click(tag_3)
time.sleep(1)
# Click on the address/search bar
pyautogui.click(tag_30)
# Paste the copied value `Dewitt`
pyautogui.hotkey('ctrl', 'v')
# Press Enter to perform the search
pyautogui.press('enter')
print(`Searched for `Dewitt` in the Chrome browser.`)

# Action 3
print(`DONE`)

# Action 4
print(`DONE`)

Failed.
```

989

Claude3 Failed Task3: *Please help me change all the places in this document that say 'text' to 'test'.*

```
# Action 1
pyautogui.hotkey('ctrl', 'h')
pyautogui.write('text')
pyautogui.press('tab')
pyautogui.write('test')
pyautogui.press('enter')

# Action 2
pyautogui.press('enter')

# Action 3
pyautogui.hotkey('ctrl', 's')

Failed.
```

990

Task w/o SoM: turning off the code checking function of the python plugin in vscode

```

992 # Action 1
import pyautogui
import time
# Move the mouse to the gear icon (Manage) and click it
pyautogui.moveTo(472, 1034, duration=0.5)
pyautogui.click()

# Action 2
# Move the mouse to the 'Settings' menu item and click it
# A rough central point of the 'Settings' menu item
pyautogui.moveTo(497 + 385/2, 832 + 25/2, duration=0.5)
pyautogui.click()

# Action 3
# Move the mouse to the search settings bar and click it
# A rough central point of the search settings bar
pyautogui.moveTo(783 + 587/2, 390 + 19/2, duration=0.5)
pyautogui.click()
# Type the query to search for Python-related settings
pyautogui.write('python', interval=0.05)

# Action 4
# Move the mouse to the 'Edit in settings.json' button and click it
# A rough central point of the button
pyautogui.moveTo(990 + 116/2, 624 + 15/2, duration=0.5)
pyautogui.click()

# Action 5
# Move the cursor to the correct position
pyautogui.moveTo(889 + 1/2, 432 + 1/2, duration=0.5)
pyautogui.click()
# Type the configuration to disable Python missing imports error reporting
pyautogui.write('reportMissingImports: `none`', interval=0.05)
pyautogui.press('enter')
# Save the changes
pyautogui.hotkey('ctrl', 's')

# Done

```

993

Task w/ SoM: turning off the code checking function of the python plugin in vscode

```

# Action 1
import pyautogui
import time
pyautogui.hotkey('ctrl', 'shift', 'p')

# Action 2
pyautogui.write('Preferences: Open Settings (UI)', interval=0.05)
time.sleep(0.5)
pyautogui.press('enter')

# Action 3
# Click on the search settings input field
pyautogui.click(x=600, y=175)
# These coordinates may need to be adjusted for your screen resolution
# Type 'Python' into the search field
pyautogui.write('Python', interval=0.05)

```

994

```

pyautogui.press('enter')

# Action 4
# Click on the search settings bar and type in the query again
pyautogui.click(x=800, y=150)
# Click on the search bar, the coordinates may need adjustment
pyautogui.typewrite('Python missing imports', interval=0.25)

# ... Can not find the right click area

# Failed

```

995

996 D.6.1 Analysis on SoM Effectiveness

997 The SoM, effective for web tasks focusing only on labeled bounding boxes, **did not** excel in our diverse tasks
 998 that required more expertise and flexibility. It shortens the action space of the agent and thus hinders the
 999 agent's exploration and adaptability. For example in Figure 18, in the task of "turning off the code checking
 1000 function of the python plugin." in VS Code, agents without SoM succeeded by editing the settings.json, whereas
 1001 SoM-equipped agents struggled with finding and deselecting the checkbox in the settings. The latter has a longer
 1002 action path and fails due to incorrect mouse clicks. Furthermore, SoM's labeling of elements in professional
 1003 software also has such errors, that agents without SoM can use a lly tree to calculate and adjust coordinates,
 1004 while the elements' blocks under SoM can be hard to change during the task.

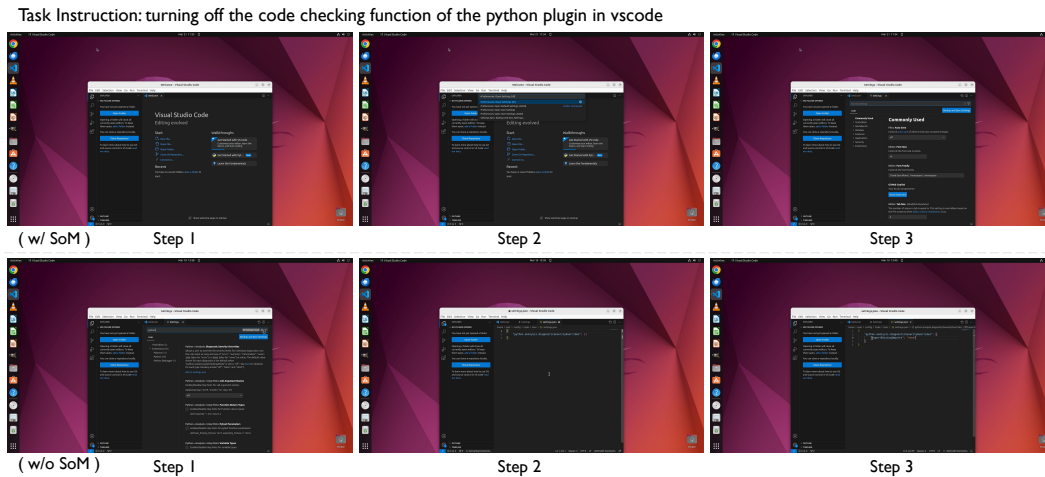


Figure 18: A task showcase where the SoM-equipped agent tried to find interactive settings, while the screen-a11ytree-equipped agents directly modified the value in the setting.json file.

1005 E Limitations and Future Work

1006 We identify several potential directions for community development and progress toward general-purpose agents
 1007 for computer operation:

1008 **Enhancing VLM capabilities for efficient and robust GUI interactions** For foundation model
 1009 development, we need to boost the efficiency of our models, enabling them to process much longer contexts and
 1010 perform inference computations efficiently, akin to the robotics community [6, 7] to better handle real-world
 1011 cases. Enhancements in VLMs' GUI grounding capabilities that is robust to application windows changes and
 1012 are also sought, focusing on the accurate understanding and generation of precise actions aligned with given
 1013 instructions. Moreover, amplifying VLMs' ability to comprehend context in the form of images is a pivotal goal,
 1014 since it is crucial to enable history encoding using images so that we can build memory and reflection upon that.
 1015 These improvements may require more efforts in the upstream pre-training stage, downstream fine-tuning stage,
 1016 and even in the model structure itself, as pointed out in previous work [9, 17, 33].

1017 **Advancing agent methodologies for exploration, memory, and reflection** The next-level approach
 1018 encompasses designing more effective agent architectures that augment the agents' abilities to explore au-
 1019 tonomously and synthesize their findings. The agents face challenges in leveraging lengthy raw observation and

1020 action records. It’s fascinating to explore novel methods for encoding this history, incorporating efficient memory
 1021 and reflection solutions to condense contextual information and aid the agent in extracting key information.
 1022 Additionally, integrating knowledge grounding into (V)LLM agents through memory mechanisms is a promising
 1023 avenue as well. Moreover, practice GUI assistants also require features of personalization and customization.
 1024 These features rely on techniques such as user profiling and retaining memories from long-term user-assistant
 1025 interactions. Additionally, crafting protocols specifically for digital agents operating within GUI and CLI
 1026 interfaces aims at facilitating efficient actions is also an essential thing for the feasibility of general-purpose
 1027 digital agents in the mid-short term.

1028 **Addressing the safety challenges of agents in realistic environments** The safety of agents is a
 1029 critical issue if applying a built agent in fully realistic environments, the developed universal digital agent could
 1030 potentially be used to bypass CAPTCHA systems in the future, as noted in [42]. However, due to the currently
 1031 limited capabilities of agents, we have not observed any harmful and damaging behaviors during our experiments,
 1032 an automatic agent has the opportunity to damage patent rights, abuse accounts, attempt to exploit software
 1033 vulnerabilities to create viruses, or engage in attacks. Currently, we adopt virtual machines to make it difficult
 1034 for developing digital agents to cause irreversible damage to our host machines. However, there still lacks a
 1035 reliable metric to assess the safety of an agent developed in an isolated environment. The current evaluation
 1036 functions mainly focus on the results closely regarding the task instructions, assess only the correctness of
 1037 task completion, and pay little attention to potential unnecessary damaging actions of agents. Owing to the
 1038 complexity of a complete computer environment, we didn’t work out an efficient way to detect the latent side
 1039 effects of the agent. Consequently, how to assess and control potential behaviors in open and real environments
 1040 through environmental constraints and agent training is an important further direction of research.

1041 **Expanding and refining data and environments for agent development** In terms of datasets and
 1042 environments, we can broaden the scope to cover more specialized domains, including real-sector needs in
 1043 healthcare, education, industry, transportation, and personalized requirements. Efforts can be made to ensure our
 1044 environment’s seamless deployment across various hardware and software settings. The variance of a11y tree
 1045 quality across different applications is also noticed. Although the problem is not remarkable in the applications
 1046 currently included, there is no guarantee of that the application developers obey the a11y convention and offer
 1047 clear and meaningful descriptions for GUI elements. More intelligent approaches to filter redundant a11y tree
 1048 elements and to handle latently missing elements deserve careful investigation as well. We also highlight the
 1049 necessity of a painless data collection method, allowing for the effortless acquisition of computer operation data
 1050 and its transformation into agent capabilities.

1051 F Distribution & License

1052 The proposed OSWORLD platform and task set are open-sourced under the Apache-2.0 license and are available
 1053 at <https://os-world.github.io/> and <https://github.com/xlang-ai/OSWorld>. The authors claim
 1054 that OSWORLD will be permanently maintained for future environments.

1055 A non-exhaustive list of artifacts used in the development of OSWORLD environment includes: pyautogui²⁰,
 1056 flask²¹, python-pptx²², python-docx²³, OpenCV²⁴, EasyOCR²⁵, *etc.* They are released under licenses
 1057 BSD-3-Clause, MIT, and Apache-2.0. A non-exhaustive list of artifacts used in the experiments includes:
 1058 openai-python²⁶, google-generativeai²⁷, groq-python²⁸, *etc.* They are released under licenses Apache-2.0. The
 1059 authors claim that the usage completely obeys the licenses.

1060 G Datasheet

1061 G.1 Motivation

1062 • **For what purpose was the dataset created?** Was there a specific task in mind? Was there a specific
 1063 gap that needed to be filled? Please provide a description.

²⁰<https://github.com/asweigart/pyautogui>

²¹<https://github.com/pallets/flask>

²²<https://github.com/scanny/python-pptx>

²³<https://github.com/python-openxml/python-docx>

²⁴<https://github.com/opencv/opencv-python>

²⁵<https://github.com/JaidedAI/EasyOCR>

²⁶<https://github.com/openai/openai-python>

²⁷<https://github.com/GoogleCloudPlatform/generative-ai>

²⁸<https://github.com/groq/groq-python>

The environment and dataset for OSWORLD are created to address a critical need for a comprehensive benchmark capable of evaluating the performance and capabilities of universal digital agents across real-world, operating system-level tasks. The motivation stemmed from the recognized gaps in existing benchmarks focusing predominantly on isolated applications or single-domain tasks. No existing dataset adequately represents the range of actions, coupled with real-world complexities, faced by operational software agents in a truly interactive operating system environment. This dataset enables the holistic assessment of agents in multi-application contexts and multi-modal interactions, with tasks requiring both comprehension and navigation of complex user interfaces across various commonly used software and operating systems.

- **Who created the dataset (e.g., which team, research group) and on behalf of which entity (e.g., company, institution, organization)?**

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu from the XLANG Lab²⁹ from University of Hong Kong, Carnegie Mellon University, Salesforce Research and University of Waterloo create the environment and the task set.

G.2 Composition

- **What do the instances that comprise the dataset represent (e.g., documents, photos, people, countries)?** Are there multiple types of instances (e.g., movies, users, and ratings; people and interactions between them; nodes and edges)? Please provide a description.

The instances within the OSWORLD dataset comprise a diverse collection of tasks simulated within real operating system environments, specifically focusing on Ubuntu and Windows systems. These tasks represent a variety of real-world computer usage scenarios, encompassing multiple applications and interfaces. Each instance includes detailed natural language instructions, setup configurations with corresponding files, and setup actions for environment initialization. The instances further encompass a variety of operation types such as document editing, file operations, email activities, and multimedia management. This design ensures a comprehensive evaluation of digital agents' ability to perform highly integrated, multi-modal interactions across different software applications and user interfaces. The dataset, therefore, offers a rich milieu of complex tasks requiring both cognitive understanding and interaction capabilities from the executing agents.

- **How many instances are there in total (of each type, if appropriate)?**

There are a total of 369 tasks for the Ubuntu operating system and 43 tasks designed for Windows within the OSWORLD benchmark. These tasks encompass a variety of task types including multi-app workflow tasks, single-application tasks, and integrated tasks from related datasets. Specifically, the dataset includes 268 single-app tasks (72.6%), 101 multi-app workflow tasks (27.4%), and 30 infeasible tasks (8.1%). Additionally, 84 tasks (22.8%) are integrated from other datasets.

- **Does the dataset contain all possible instances or is it a sample (not necessarily random) of instances from a larger set?** If the dataset is a sample, then what is the larger set? Is the sample representative of the larger set (e.g., geographic coverage)? If so, please describe how this representativeness was validated/verified. If it is not representative of the larger set, please describe why not (e.g., to cover a more diverse range of instances, because instances were withheld or unavailable).

The dataset in OSWORLD is a curated sample from a larger set of potential real-world tasks. These samples were selected based on their popularity, helpfulness, and diversity, as judged by view counts and user votes on various platforms including official guidelines, tutorials, forums, and educational courses. This selection process intends to cover a broad spectrum of typical and critical tasks that users perform using computer systems across different operating systems and software environments. While this dataset aims to cover various practical and high-impact scenarios, it does not claim to represent all possible tasks due to the vastness and evolving nature of computer operations. It emphasizes tasks with significant educational and practical utility to ensure that the benchmark provides both comprehensive and challenging objectives for developing advanced digital agents. The representativeness has not been systematically verified statistically due to the qualitative selection criteria based on task relevance and utility rather than mere statistical representation.

- **What data does each instance consist of?** "Raw" data (e.g., unprocessed text or images) or features? In either case, please provide a description.

Each instance within the OSWORLD dataset consists of both "raw" data and structured annotations. The main components of each task instance are:

- **Natural language instruction:** Each task is accompanied by a natural language description that guides the agent on how to execute the task. These instructions are crafted from real-world guidelines and scenarios.

²⁹<https://www.xlang.ai/>

1123 – **Setup configuration:** This includes necessary files and setup actions required for initializing the
1124 task environment as the cases of the real world.

1125 – **Evaluation scripts:** Scripts designed to programmatically assess whether the task has been
1126 successfully completed by the agent. These scripts function as objective measures of task
1127 performance.

1128 – **Accessibility Trees and Screenshots:** For GUI-based tasks, environments from instances include
1129 accessibility trees that provide structured data regarding the GUI elements, and screenshots that
1130 offer visual context. These components are crucial for agents that rely on visual and structural
1131 interpretations to navigate and interact within the tasks.

1132 These data elements combine raw and processed formats to deliver comprehensive task-specific
1133 information that supports both the execution and the evaluation of automated agents across varied
1134 operating systems and interfaces.

1135 • **Is there a label or target associated with each instance?** If so, please provide a description.

1136 Yes, each task in the OSWORLD benchmark is associated with a specific label describing the objective
1137 of the task. The label is a natural language instruction that indicates what the computer agent needs
1138 to achieve. Additionally, each task is accompanied by a setup configuration and an evaluation script,
1139 which are used to verify if the task has been accomplished successfully according to the predefined
1140 criteria. This structured approach ensures that agents are assessed accurately on their ability to perform
1141 a variety of real-world computing tasks.

1142 • **Is any information missing from individual instances?** If so, please provide a description, explaining
1143 why this information is missing (e.g., because it was unavailable). This does not include intentionally
1144 removed information, but might include, e.g., redacted text.

1145 Yes, some instances may lack certain details that were not available from the source materials. These
1146 sources include forums, tutorials, how-to websites, and video content from platforms like YouTube
1147 and TikTok. This missing information could be due to incomplete guidelines or advice given in the
1148 source materials, which were not exhaustively detailed. Additionally, although extensive efforts were
1149 made to create comprehensive annotations, some specifics might be inherently unobtainable due to the
1150 nature of the source content.

1151 • **Are there any errors, sources of noise, or redundancies in the dataset?** If so, please provide a
1152 description.

1153 Yes, our dataset may include various sources of noise or redundancies which typically arise from
1154 the variety of different real-world sources used for task examples, such as forums, tutorials, and
1155 guidelines. Additionally, the dataset includes deliberately collected infeasible tasks due to feature
1156 deprecation or hallucinated features, as highlighted in the design process. These instances are used to
1157 test the robustness and error-handling capabilities of digital agents but may be perceived as noise if
1158 not correctly accounted for in training. Redundancies may occur as multiple tasks may involve similar
1159 applications or operations, albeit with different specific instructions or end goals.

1160 • **Is the dataset self-contained, or does it link to or otherwise rely on external resources (e.g.,
1161 websites, tweets, other datasets)?** If it links to or relies on external resources, a) are there guarantees
1162 that they will exist, and remain constant, over time; b) are there official archival versions of the
1163 complete dataset (i.e., including the external resources as they existed at the time the dataset was
1164 created); c) are there any restrictions (e.g., licenses, fees) associated with any of the external resources
1165 that might apply to a dataset consumer? Please provide descriptions of all external resources and any
1166 restrictions associated with them, as well as links or other access points, as appropriate.

1167 The OSWORLD dataset is not entirely self-contained as it integrates tasks and guidelines derived from
1168 various external resources such as forums, tutorials, how-to websites, and public guidelines. These
1169 include websites like WikiHow, Reddit, Quora, StackOverflow, and video tutorials from platforms like
1170 YouTube and TikTok, among others.

1171 a) There are no guarantees that the external resources will exist indefinitely or remain constant over
1172 time, which could impact the dataset’s reliability and the reproducibility of the tasks.

1173 b) There are no official archival versions of the complete dataset that include external resources as
1174 they existed at the time the dataset was created. Consumers of the dataset will need to rely on
1175 the current versions of these resources, which may be subject to changes.

1176 c) Some of the external resources may have their own copyright or licensing restrictions which
1177 might apply to a dataset consumer. Each resource’s terms of service and copyright laws should
1178 be reviewed prior to use to ensure compliance.

1179 External resources vary in nature and can often have different terms of use; it is advised for users of
1180 the OSWORLD dataset to verify any potential restrictions or licensing requirements individually.

1181 • **Does the dataset contain data that might be considered confidential (e.g., data that is protected by
1182 legal privilege or by doctor–patient confidentiality, data that includes the content of individuals’
1183 non-public communications)?** If so, please provide a description.

No, the dataset does not contain any data that could be considered confidential. The tasks and examples within the OSWORLD dataset are created from widely accessible sources such as public forums, official tutorials, and openly available guidelines. All data in use, including task instructions and configurations, are derived from non-confidential, non-privileged, and public communications aimed at demonstrating real-world computer usage scenarios.

- **Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety?** If so, please describe why.
No, the dataset primarily consists of tasks defined and executed on operating systems using data from tutorials, guides, and how-to information that is publicly available and widely considered to be educational and informational. Therefore, it is unlikely to contain content that could be viewed as offensive, insulting, or threatening.
- **Does the dataset identify any subpopulations (e.g., by age, gender)?** If so, please describe how these subpopulations are identified and provide a description of their respective distributions within the dataset.
No, the dataset does not identify or differentiate any subpopulations based on age, gender, or other demographic factors. The focus is on the diversity of tasks related to computer usage across different operating systems and software environments.
- **Is it possible to identify individuals (i.e., one or more natural persons), either directly or indirectly (i.e., in combination with other data) from the dataset?** If so, please describe how.
No, it is not possible to identify individuals directly or indirectly from the dataset. The tasks and examples in OSWORLD are derived from general public sources such as forums, tutorials, and guidelines that are widely accessible and do not contain personal information. Furthermore, any task setup or descriptive data is specifically designed or curated to avoid inclusion of any personal identifiers or sensitive information that could lead to the re-identification of individuals.
- **Does the dataset contain data that might be considered sensitive in any way (e.g., data that reveals race or ethnic origins, sexual orientations, religious beliefs, political opinions or union memberships, or locations; financial or health data; biometric or genetic data; forms of government identification, such as social security numbers; criminal history)?** If so, please provide a description.
The OSWORLD dataset does not contain data that might be considered sensitive. The dataset is constructed from publicly available sources such as forums, tutorials, and guidelines and solely focuses on operational tasks within computer environments. It does not include any personal data categories or information related to individuals. The tasks are generated and anonymized without incorporating any sensitive or personal data elements.

1218 G.3 Collection Process

- **How was the data associated with each instance acquired?** Was the data directly observable (e.g., raw text, movie ratings), reported by subjects (e.g., survey responses), or indirectly inferred/derived from other data (e.g., part-of-speech tags, model-based guesses for age or language)? If the data was reported by subjects or indirectly inferred/derived from other data, was the data validated/verified? If so, please describe how.
The data for each task in OSWORLD was meticulously collected from a variety of credible sources including forums, official tutorials, instructional video content from platforms like YouTube and TikTok, and Q&A websites such as Reddit and Quora. The tasks were inspired by real-world computer usage scenarios which are often discussed in these sources. Each example was carefully annotated with natural language instructions, and detailed setup configurations, and included manually crafted evaluation scripts to ensure the task accurately represents real-world functionalities. To validate the realism and feasibility of these tasks, every task was cross-verified by at least two other authors. In particular, these reviewers checked the instructions for clarity, the setup for correctness, and the evaluation scripts for comprehensiveness. Additionally, some tasks inherently deemed infeasible due to outdated features or user misconceptions were also included to represent the full spectrum of real-world challenges.
- **What mechanisms or procedures were used to collect the data (e.g., hardware apparatuses or sensors, manual human curation, software programs, software APIs)?** How were these mechanisms or procedures validated?
Data collection for the OSWORLD benchmark involved a combination of manual human curation and usage of software programs. Specifically, task instructions and scenarios were gathered from diverse sources including official guidelines, online tutorials, how-to websites, public forums, and personal blogs. These were then annotated manually by the authors and supplemented with setup actions and evaluation scripts using software tools designed for task setup and verification.

1243 The validation of these mechanisms and procedures was conducted through a rigorous peer-review
1244 process within the research team. Each task and its associated data were cross-checked by multiple
1245 authors to ensure feasibility, clarity, and alignment with the baseline sources. Additionally, the
1246 examples gathered were tested by independent evaluators not involved in the initial annotation,
1247 acting as agents to execute the tasks, and providing feedback which was used to refine the tasks and
1248 annotations further.

- 1249 • **If the dataset is a sample from a larger set, what was the sampling strategy (e.g., deterministic,**
1250 **probabilistic with specific sampling probabilities)?**

1251 The tasks in the OSWORLD represent a carefully curated subset of potential real-world computing
1252 tasks. The selection was made based on a combination strategy that included both deterministic and
1253 probabilistic elements. We chose examples based on their popularity, helpfulness, and diversity, with
1254 popularity assessed by view counts and votes on various platforms such as forums, tutorials, and online
1255 courses. This involved deterministic selection of application types to ensure coverage across a broad
1256 spectrum of daily, professional, and workflow scenarios. Additionally, we incorporated tasks that were
1257 inherently challenging or infeasible due to software limitations or feature deprecations, enhancing the
1258 dataset’s practical relevance and complexity.

- 1259 • **Who was involved in the data collection process (e.g., students, crowdworkers, contractors) and**
1260 **how were they compensated (e.g., how much were crowdworkers paid)?**

1261 All the development of platform, app data collection, and task definition creation are completed by the
1262 authors.

- 1263 • **Over what timeframe was the data collected?** Does this timeframe match the creation timeframe of
1264 the data associated with the instances (e.g., recent crawl of old news articles)? If not, please describe
1265 the timeframe in which the data associated with the instances was created.

- 1266 • **Did you collect the data from the individuals in question directly, or obtain it via third parties or**
1267 **other sources (e.g., websites)?**

1268 The data for OSWORLD were primarily collected via third parties and other sources. This included
1269 a diverse array of platforms such as official guidelines and tutorials, video tutorials on TikTok and
1270 YouTube, how-to websites like WikiHow, Q&A forums such as Reddit, Quora, Superuser, and
1271 StackOverflow, formal online courses from Coursera and Udemy, and publicly available personal
1272 blogs and guidelines.

- 1273 • **Were the individuals in question notified about the data collection?** If so, please describe (or show
1274 with screenshots or other information) how notice was provided, and provide a link or other access
1275 point to, or otherwise reproduce, the exact language of the notification itself.

1276 No, the nature of the data collected for the OSWORLD benchmark did not involve any individual-based
1277 data that would require the notification of individuals. The tasks and environment setups used in
1278 OSWORLD were sourced from publicly available materials such as forums, tutorials, and guidelines,
1279 which do not contain personally identifiable information. Thus, notification to individuals was not
1280 applicable in this context.

- 1281 • **Did the individuals in question consent to the collection and use of their data?** If so, please
1282 describe (or show with screenshots or other information) how consent was requested and provided,
1283 and provide a link or other access point to, or otherwise reproduce, the exact language to which the
1284 individuals consented.

1285 The data collected for the OSWORLD includes tasks sourced from publicly accessible forums, tutorials,
1286 and community-driven platforms, which inherently involve contributions from public individuals.
1287 However, the paper does not specify that these individuals provided direct consent for the use of such
1288 data in the context of creating the benchmark. Typically, these public platforms have their own terms
1289 of service that users agree to, which may cover data usage for research, but exact consent for this
1290 specific study was not detailed. Hence, more explicit consent mechanisms would potentially enhance
1291 ethical compliance.

- 1292 • **If consent was obtained, were the consenting individuals provided with a mechanism to revoke**
1293 **their consent in the future or for certain uses?** If so, please provide a description, as well as a link
1294 or other access point to the mechanism (if appropriate).

1295 As the data collection involved publicly available sources and no specific documentation of direct
1296 consent is provided, there is also no mention of a mechanism for the individuals to revoke their consent
1297 in the future or for certain uses of the data. Implementing a consent revocation mechanism would be
1298 an important step towards enhancing the ethical handling of the data used in OSWORLD.

1299 G.4 Uses

- 1300 • **Has the dataset been used for any tasks already?** If so, please provide a description.

1301 Yes, the OSWORLD dataset has been utilized to support the development and evaluation of automated
1302 computer agents across various operating systems. It includes a set of 369 tasks on Ubuntu and 43
1303 tasks on Windows, which are designed to assess various capabilities such as commonsense reasoning,
1304 software navigation, and precise input control. These tasks have been employed in comprehensive
1305 testing and performance evaluation scenarios, which have involved comparing different LLM and
1306 VLM agents' abilities to handle multi-app workflows and other operation types within real computing
1307 environments. Consequently, the dataset has facilitated research into creating more effective and
1308 versatile digital agents. Detailed results and analyses stemming from these uses are documented in the
1309 paper.

1310 • **Is there a repository that links to any or all papers or systems that use the dataset?** If so, please
1311 provide a link or other access point.

1312 Yes, the OSWORLD dataset is accompanied by a repository that includes links to all papers and systems
1313 utilizing the dataset for research and development purposes. It serves as a comprehensive resource for
1314 tracing the impact and applications of the dataset across various studies and implementations. The
1315 repository can be accessed via <https://github.com/xlang-ai/OSWorld>.

1316 • **Is there anything about the composition of the dataset or the way it was collected and prepro-**
1317 **cessed/cleaned/labeled that might impact future uses?** For example, is there anything that a dataset
1318 consumer might need to know to avoid uses that could result in unfair treatment of individuals or
1319 groups (*e.g.*, stereotyping, quality of service issues) or other risks or harms (*e.g.*, legal risks, financial
1320 harms)? If so, please provide a description. Is there anything a dataset consumer could do to mitigate
1321 these risks or harms?

1322 No.

1323 G.5 Distribution

1324 • **Will the dataset be distributed to third parties outside of the entity (*e.g.*, company, institution,**
1325 **organization) on behalf of which the dataset was created?** If so, please provide a description.

1326 Yes, the dataset created as part of the OSWORLD project is made available publicly in <https://github.com/xlang-ai/OSWorld>. It is open-sourced under the Apache-2.0 license, and it can
1327 be accessed via the official project URLs provided on GitHub and the OSWORLD website. This
1328 facilitates wide accessibility and utilization across different sectors and by various entities beyond the
1329 originating organization, promoting broader research and development in digital agent technology.

1330 • **How will the dataset will be distributed (*e.g.*, tarball on website, API, GitHub)?** Does the dataset
1331 have a digital object identifier (DOI)?

1332 The platform and dataset are open-sourced at GitHub. We do not apply for a DOI.

1333 • **When will the dataset be distributed?**

1334 Both the environment and the task set have already been made public.

1335 • **Will the dataset be distributed under a copyright or other intellectual property (IP) license,**
1336 **and/or under applicable terms of use (ToU)?** If so, please describe this license and/or ToU, and
1337 provide a link or other access point to, or otherwise reproduce, any relevant licensing terms or ToU, as
1338 well as any fees associated with these restrictions.

1339 Both the environment and dataset are open-sourced under Apache-2.0 license.

1340 • **Have any third parties imposed IP-based or other restrictions on the data associated with the**
1341 **instances?** If so, please describe these restrictions, and provide a link or other access point to, or
1342 otherwise reproduce, any relevant licensing terms, as well as any fees associated with these restrictions.

1343 No third-party IP-based or other restrictions are imposed on the primary data utilized by the OS-
1344 WORLD benchmark. The components comprising the tasks are derived from widely accessible and
1345 openly available sources, such as forums, tutorials, and free-to-use websites. All efforts have been
1346 made to ensure the tasks do not infringe upon proprietary data. However, the platform for virtual
1347 machines and tasks adapted for Windows require activation due to copyright considerations post-
1348 implementation, which can involve associated fees, although optional. For more detailed licensing
1349 terms and specific conditions, refer to the original sources and legal advice where relevant.

1350 • **Do any export controls or other regulatory restrictions apply to the dataset or to individual**
1351 **instances?** If so, please describe these restrictions, and provide a link or other access point to, or
1352 otherwise reproduce, any supporting documentation.

1353 No.

1355 G.6 Maintenance

1356 • **Who will be supporting/hosting/maintaining the dataset?**

1357 The authors will support, host, and maintain permanently.

1358 • **How can the owner/curator/manager of the dataset be contacted (e.g., email address)?**
1359 Issues and discussions on GitHub and Hugging Face are welcome. One can also seek help from
1360 Tianbao Xie (tbxie@cs.hku.hk), Danyang Zhang (zhang-dy20@sjtu.edu.cn), and Tao Yu
1361 (taoyds@hku.hk).

1362 • **Is there an erratum?** If so, please provide a link or other access point.
1363 Currently, no. Errata will be announced if there is any in the future.

1364 • **Will the dataset be updated (e.g., to correct labeling errors, add new instances, delete instances)?**
1365 If so, please describe how often, by whom, and how updates will be communicated to dataset
1366 consumers (e.g., mailing list, GitHub)?
1367 Yes, the OSWORLD dataset will be periodically updated to correct any labeling errors, add new
1368 instances, and delete outdated or incorrect instances as necessary. The updates will be conducted
1369 annually by the core team of maintainers comprising computer science researchers and student
1370 contributors. Notice of updates, including details of changes made, will be communicated to dataset
1371 consumers through a dedicated mailing list and updates will be posted on the project's GitHub
1372 repository at <https://github.com/xlang-ai/OSWorld>.

1373 • **If the dataset relates to people, are there applicable limits on the retention of the data associated**
1374 **with the instances (e.g., were the individuals in question told that their data would be retained**
1375 **for a fixed period of time and then deleted)?** If so, please describe these limits and explain how they
1376 will be enforced.
1377 No.

1378 • **Will older versions of the dataset continue to be supported/hosted/maintained?** If so, please
1379 describe how. If not, please describe how its obsolescence will be communicated to dataset consumers.
1380 Older versions of the OSWORLD dataset will be available for use, but they may not receive updates
1381 or support once a new version is released. We aim to keep archival versions available for research
1382 reproducibility and comparison purposes. However, these archival versions will be hosted without
1383 active maintenance, meaning that issues or bugs identified in these versions will only be addressed in
1384 new releases. Obsolescence of any version will be communicated through our official website and
1385 version-release notes associated with the OSWORLD dataset.

1386 • **If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for**
1387 **them to do so?** If so, please provide a description. Will these contributions be validated/verified?
1388 If so, please describe how. If not, why not? Is there a process for communicating/distributing these
1389 contributions to dataset consumers? If so, please provide a description.
1390 Yes, the OSWORLD framework encourages the academic and developer community to extend, augment,
1391 and contribute to the dataset. Contributions can be made via GitHub, where the project and its datasets
1392 are hosted. Contributors can submit pull requests with their enhancements or new task datasets. All
1393 contributions undergo a rigorous review process by the existing project maintainers to ensure that they
1394 meet the quality standards of the dataset and are consistent with the goals of OSWORLD. This review
1395 process includes verifying the accuracy of the information, the relevance, and the usability of the
1396 tasks. Once approved and merged, contributions are incorporated into the main branch of the dataset
1397 repository and are made available for immediate access and use. Updates to the dataset are regularly
1398 communicated through the project's website and GitHub repository, as well as through mailing lists
1399 and social media to ensure that dataset consumers and contributors stay informed. Additionally,
1400 periodic releases of the dataset include detailed changelogs that summarize new additions, changes,
1401 and improvements, thus ensuring transparency and ease of access for all users interested in building
1402 upon or utilizing the OSWORLD dataset.