上海人工智能实验室
Shanghai Artificial Intelligence Laboratory

SCHOOL OF COMPUTING & DATA SCIENCE
The University of Hong Kong

ModelScope

Constructing Trajectory Data for Generalist GUI Agents

Qiushi Sun
qiushisun.github.io
𝕏 @qiushi_sun

Date: 19 Feb 2025

上海人工智能实验室
Shanghai Artificial Intelligence Laboratory

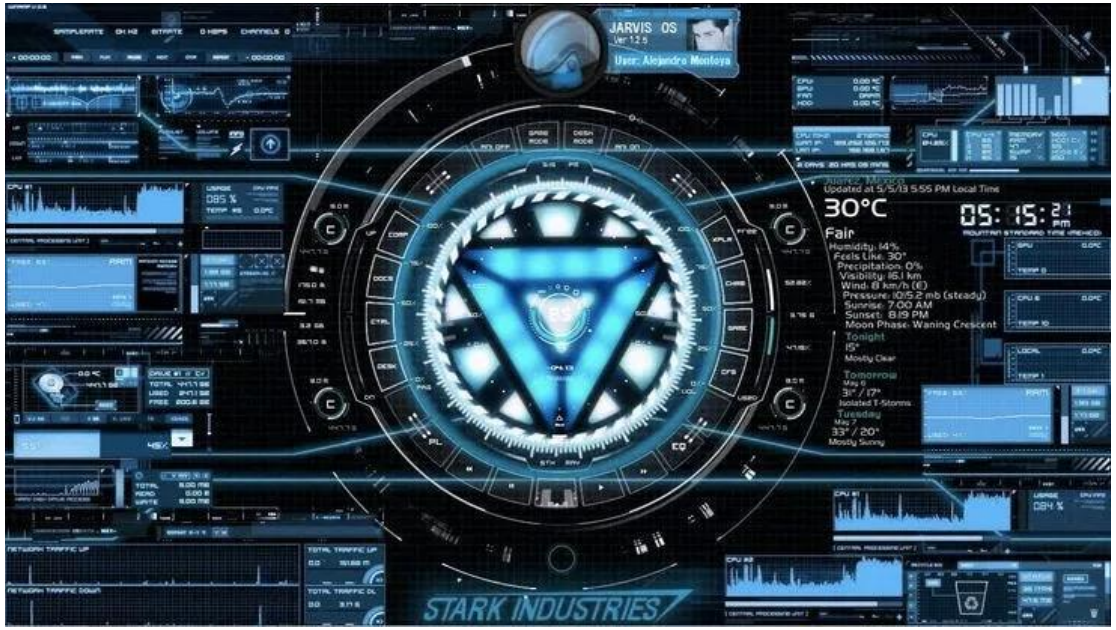
ModelScope

Today

1. Background of Computer Use Agents
2. Building GUI Agent Data with OS-Genesis
3. Future Directions and Early Attempts

Part1 | Computer Use Agents

Computer Use Agents



The Feasibility of Jarvis AI from Marvel in Real Life

[1] *OS-Copilot: Towards Generalist Computer Agents with Self-Improvement*

Computer Use Agents

Once out of reach, but we are turning it into reality.

Computer Use Agents

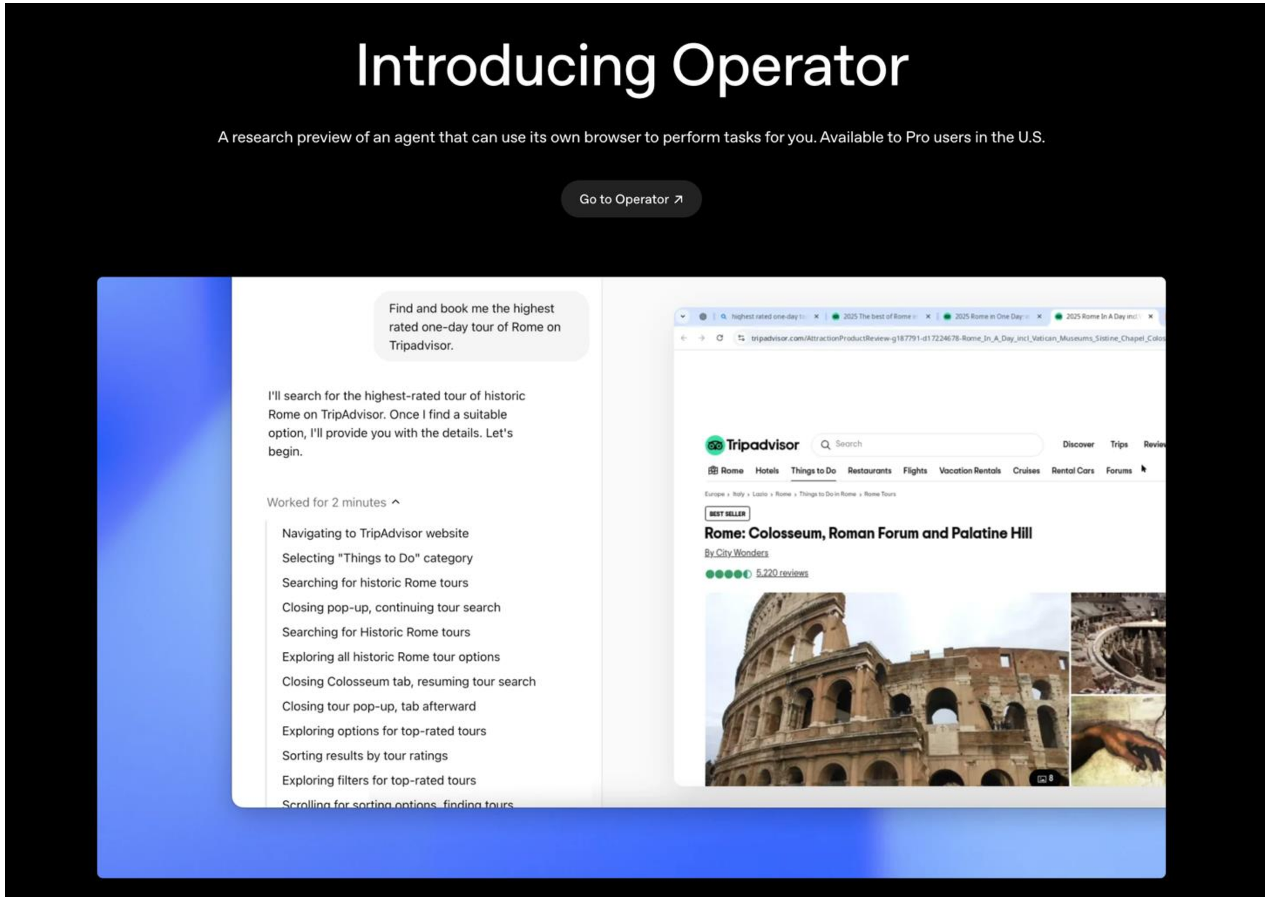
Both academia and industry are building computer use agents

Claude Computer Use

[2] *Introducing computer use, a new Claude 3.5 Sonnet, and Claude 3.5 Haiku, 22 Oct 2024*

Computer Use Agents

Both academia and industry are building computer use agents



OpenAI Operator

[3] *Introducing Operator: A research preview of an agent that can use its own browser to perform tasks for you., Jan 23, 2025*

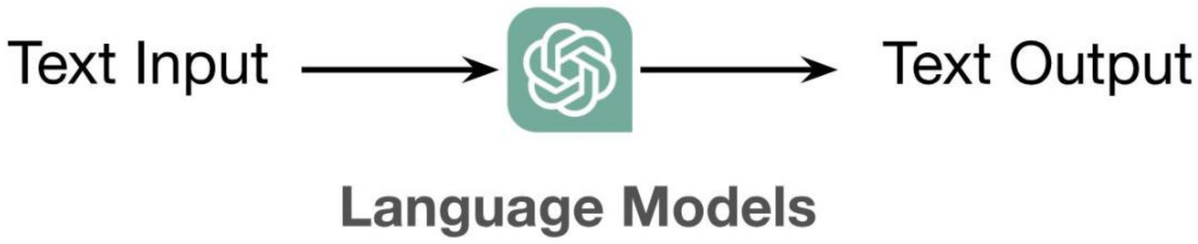
Computer Use Agents

They are quite promising for achieving Digital Automation.

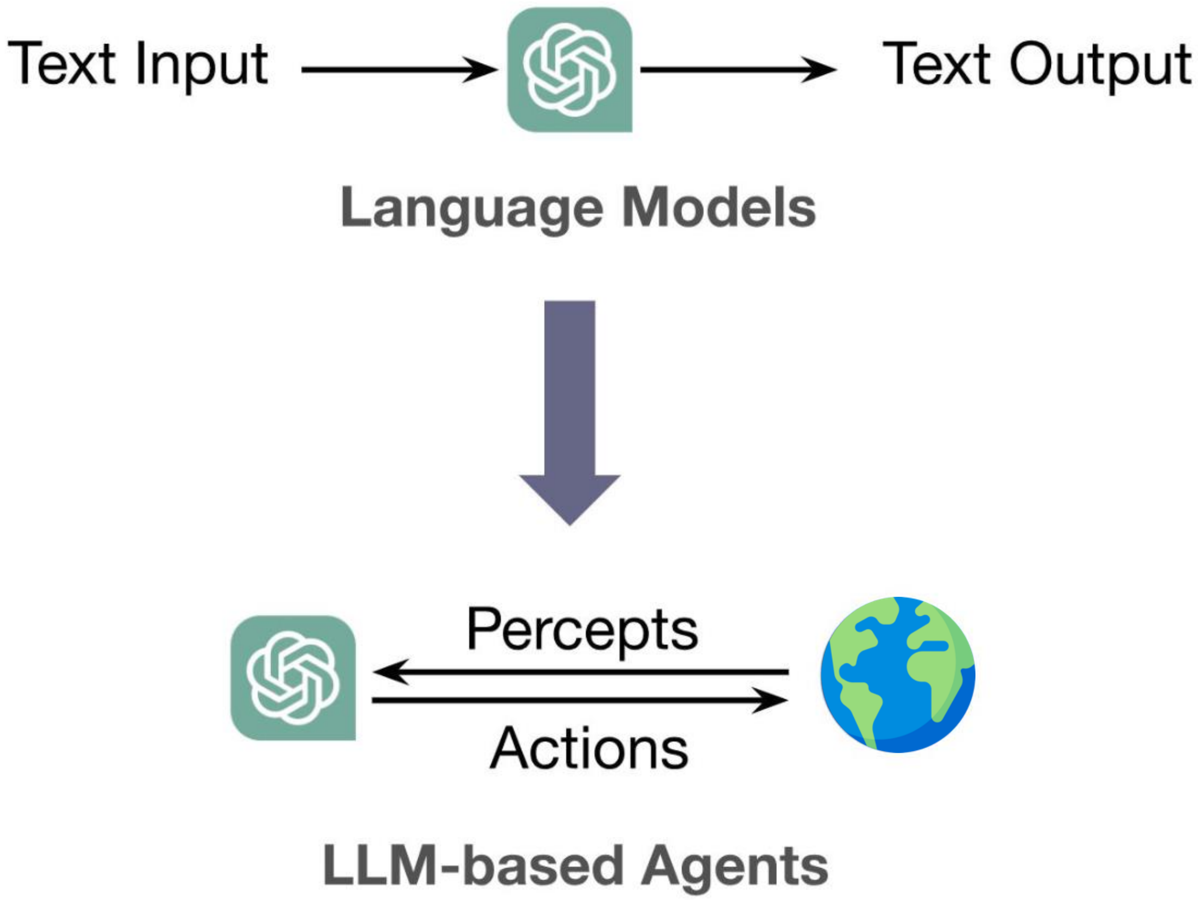
Can we transform a (V)LM into such computer / GUI agents?

Of course! But it is a non-trivial job!

Recap: Language Agents



Recap: Language Agents



But this is not enough.

Computer Use Agents

Agents are promising, but building powerful agents is challenging.

1. Agents need to follow human instructions.
2. Agents need to perform planning and action.
3. Agents need to perceive envs. and the applications they are interacting with.

Best Way to build Computer Use Agents

Behavioral Cloning / Imitation Learning.

Sounds good, but where is our data?

Data Scarcity

Data curation is much more expensive than you think.

Take Scale AI as an example.

Not to mention scenario/domain - specific data.

[4] *https://x.com/alexandr_wang*

Data Scarcity

How about having the machine collect data?

1. Pre-defined tasks are required, but they may not align with the environment.
2. Limited diversity and a poor success rate.😢

Data Scarcity

So, our goals are as follows:

1. Eliminate human involvement.
2. Obtain high-quality Trajectory data.
3. Diversity and Scalability.

Part2 | Building GUI Agent Data with OS-Genesis

OS-Genesis Automating GUI Agent Trajectory Construction via Reverse Task Synthesis

Qiushi Sun*, Kanzhi Cheng*, Zichen Ding*, Chuanyang Jin*, Yian Wang
Fangzhi Xu, Zhenyu Wu, Liheng Chen, Chengyou Jia, Zhoumianze Liu
Ben Kao, Guohao Li, Junxian He, Yu Qiao, Zhiyong Wu

上海人工智能实验室
Shanghai Artificial Intelligence Laboratory
SAPIENTIA·ET·VIRTUS
JOHNS HOPKINS UNIVERSITY
SHANGHAI JIAO TONG UNIVERSITY
UNIVERSITY OF OXFORD

GUI Trajectory Data

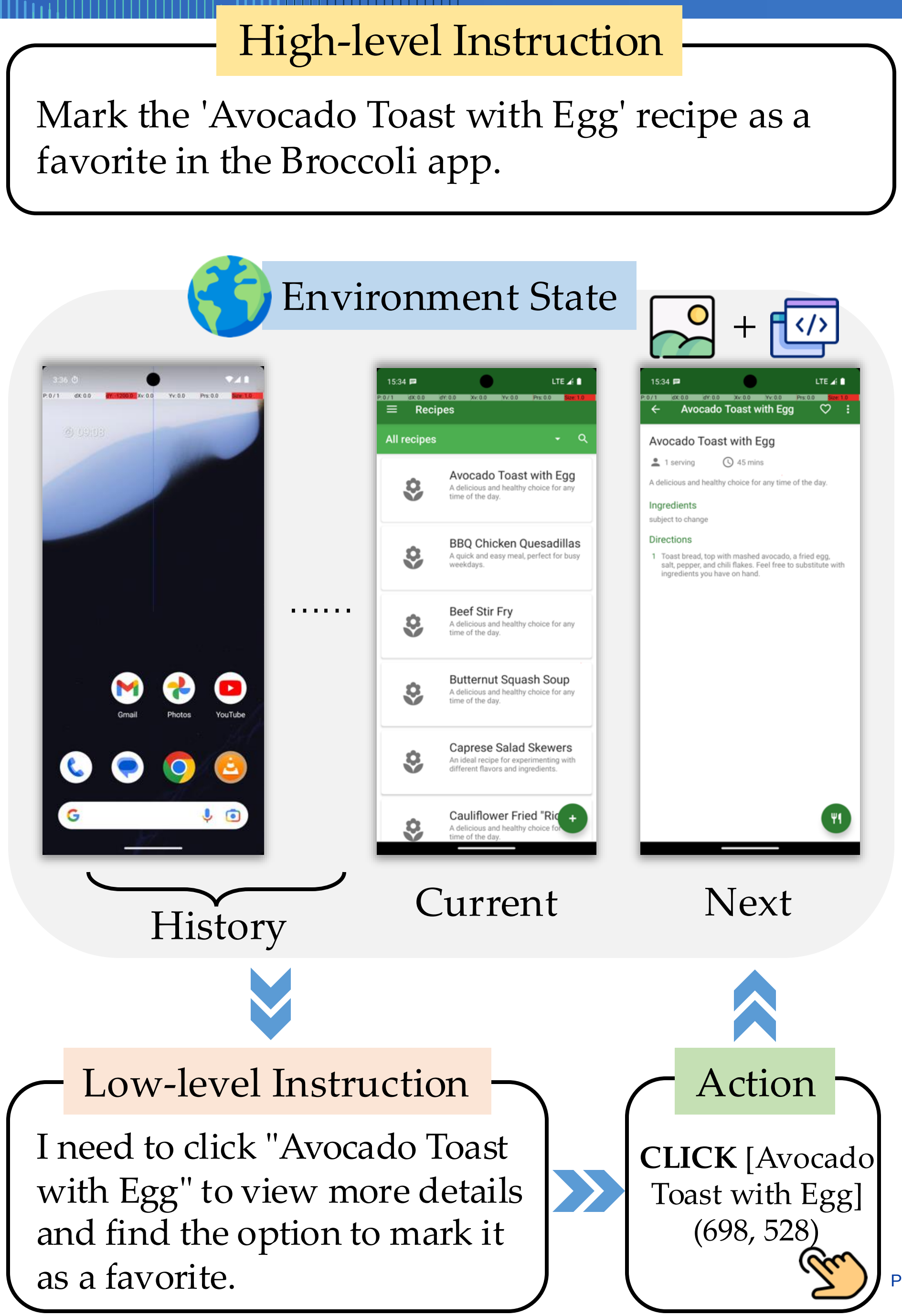
The best data format of GUI agents

1. A high-level instruction that defines the overall goal the agent aims to accomplish

2. A series of low-level instructions that each describe specific steps required

3. Actions (e.g., `CLICK`, `TYPE`)

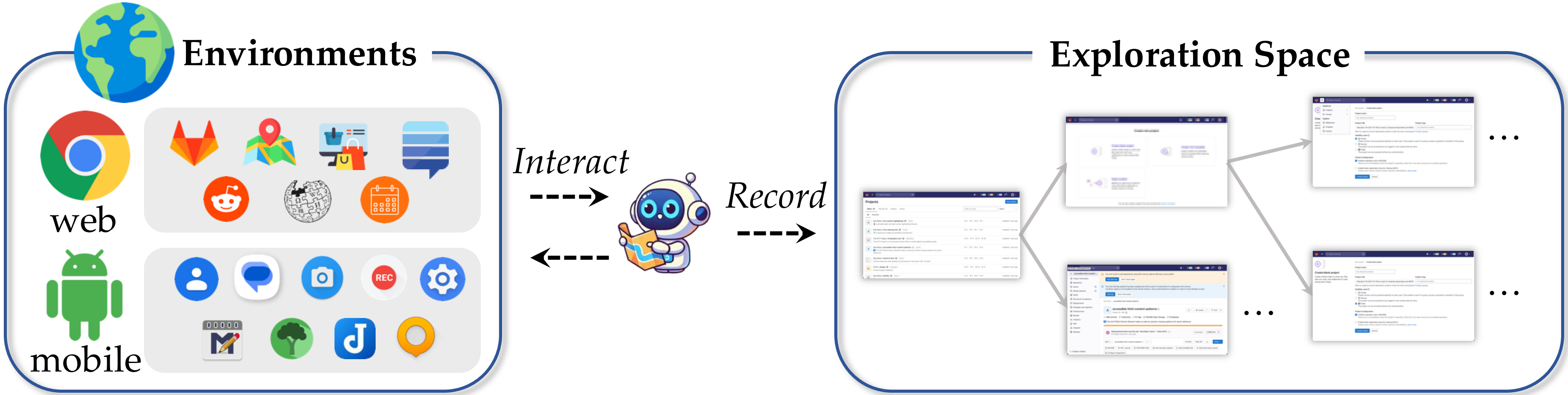
4. States, which include visual representations like screenshots and textual representations such as a11ytree



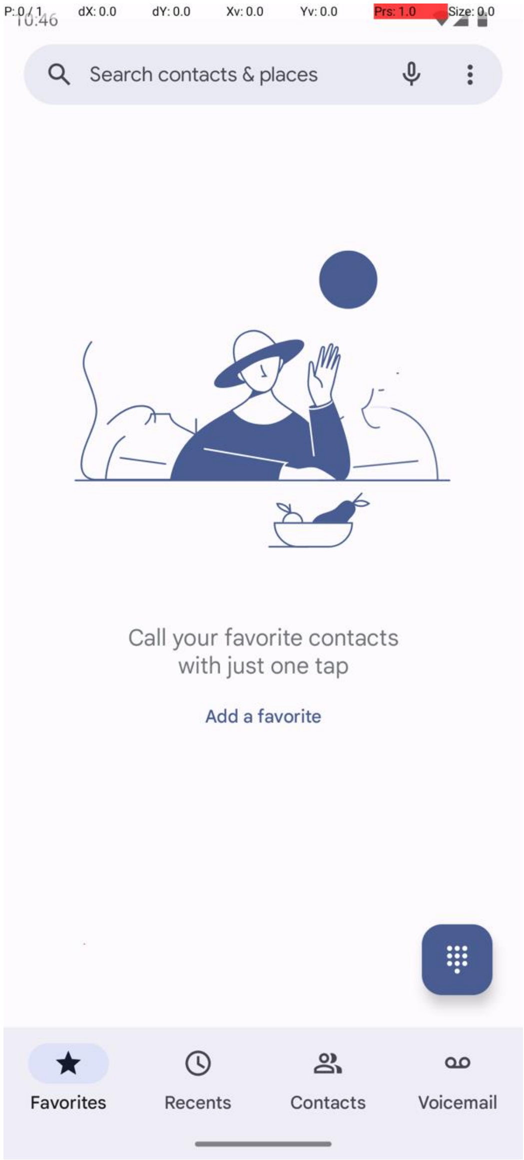
Reverse Task Synthesis

Interaction-Driven Functional Discovery is a rule-based process that explores dynamic GUI environments by interacting with UI elements. It uncovers functionalities through interaction triples

We collect: <Screen1, action, Screen2>

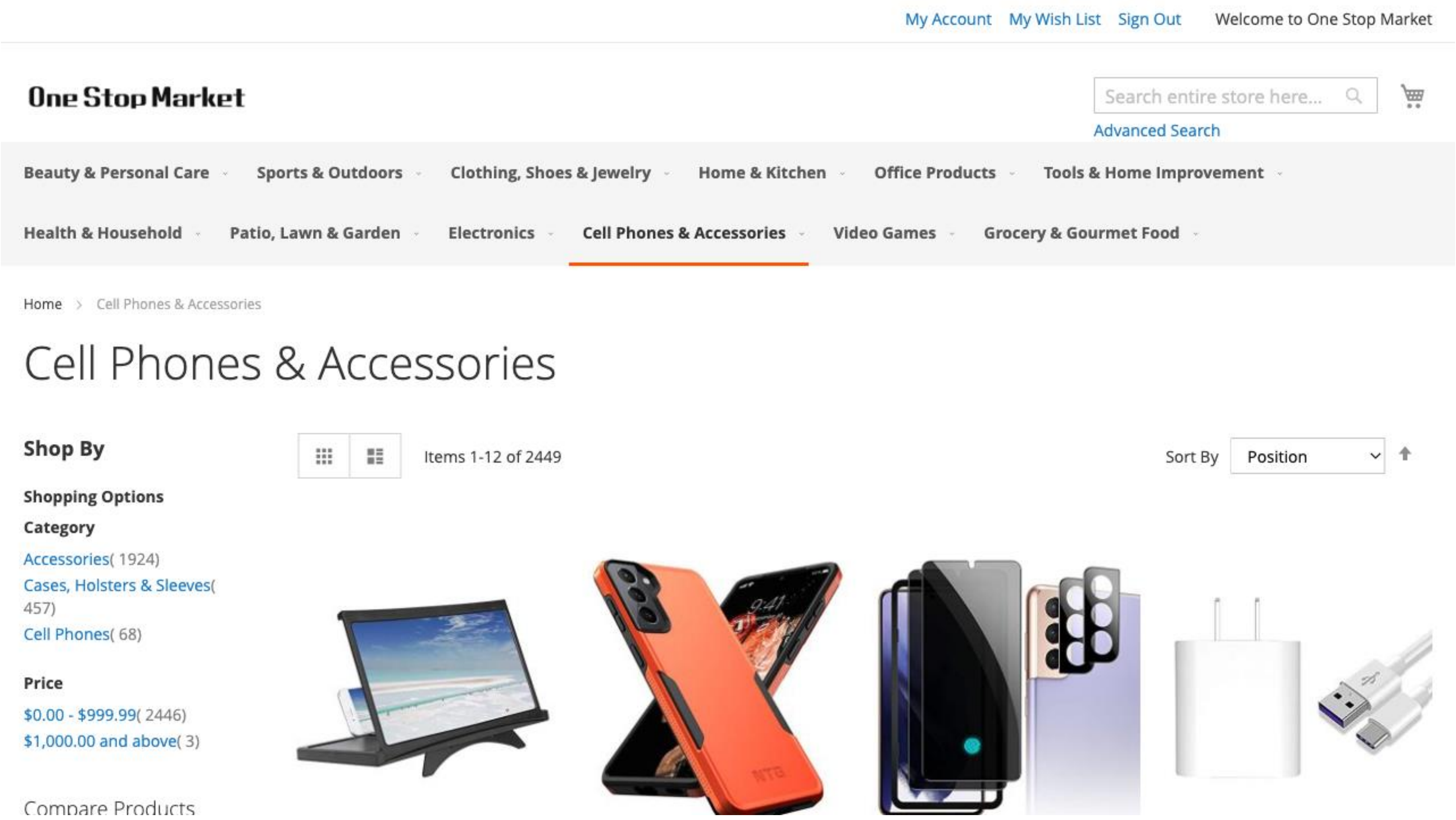


Dynamic Environments



[5] AndroidWorld: A Dynamic Benchmarking Environment for Autonomous Agents, ICLR 2025

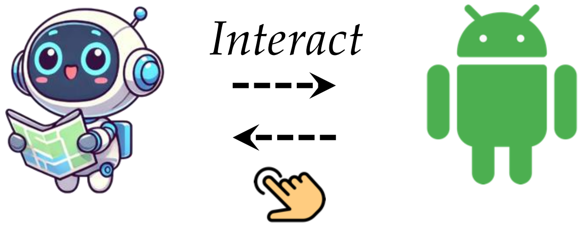
Dynamic Environments



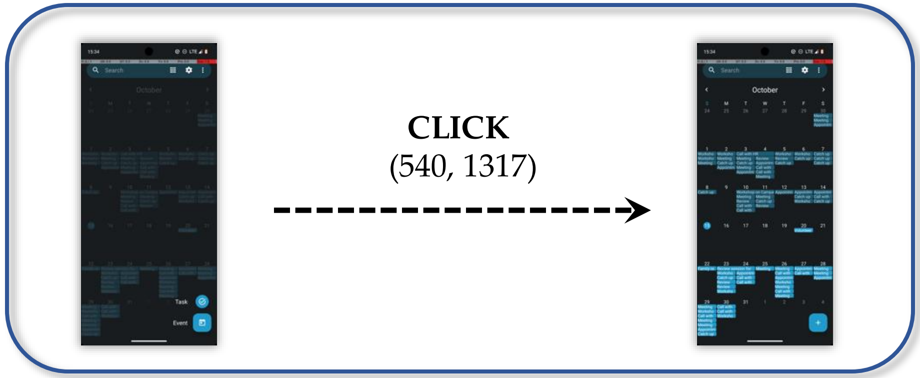
[6] *WebArena: A Realistic Web Environment for Building Autonomous Agents, ICLR 2024*

Reverse Task Synthesis

Retroactively interpreting changes in the GUI environment caused by actions.

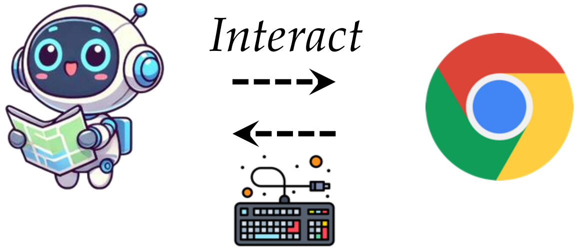


Screenshots & Actions

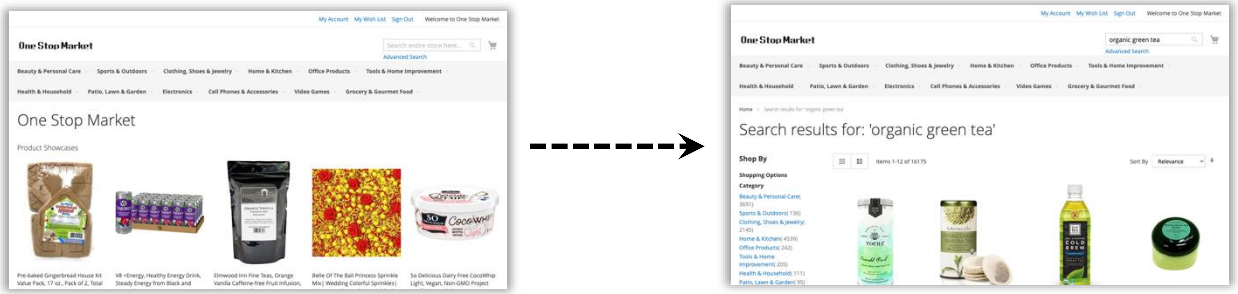


Reverse Task Synthesis

Retroactively interpreting changes in the GUI environment caused by actions.



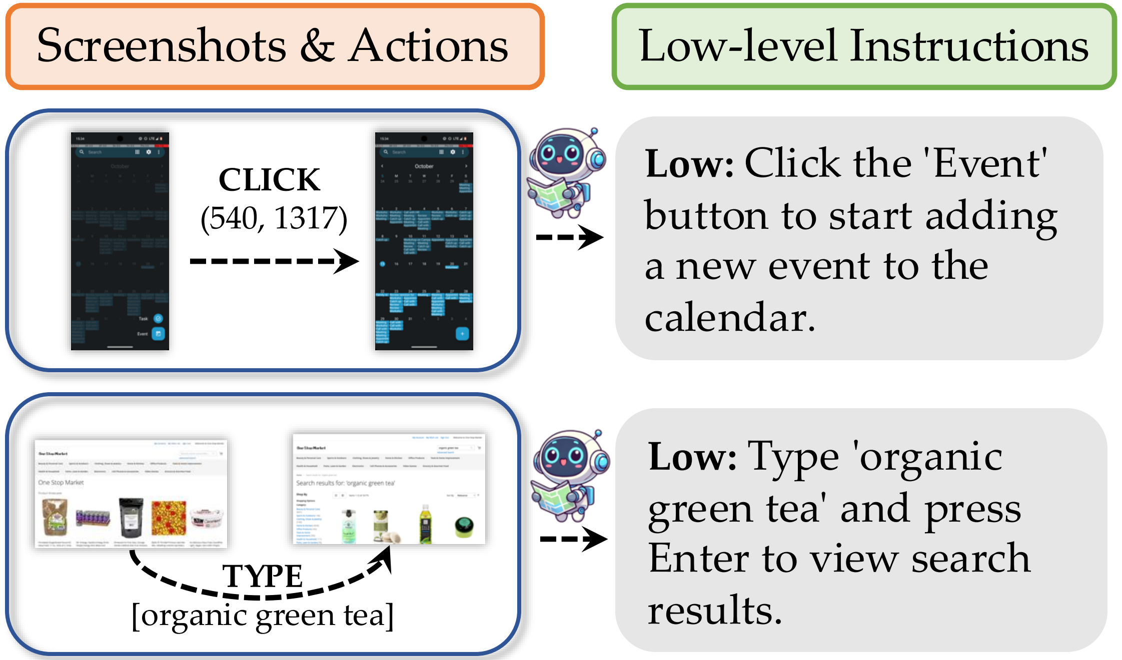
Screenshots & Actions



TYPE
[organic green tea]

Reverse Task Synthesis

Retroactively interpreting changes in the GUI environment caused by actions, this process generates executable low-level instructions



The data we synthesized:

1. Grounded

2. Actionable

Reverse Task Synthesis

Retroactively interpreting changes in the GUI environment caused by actions, this process generates executable low-level instructions, which are then transformed into broader, goal-oriented high-level tasks

Screenshots & Actions	Low-level Instructions	High-level Instruction
	**Low:** Click the 'Event' button to start adding a new event to the calendar.	**High**: In Simple Calendar Pro, create a new event titled 'Team Meeting' scheduled for October 15, 2023, at 10:00 AM. Save the event after filling in the details.
	**Low:** Type 'organic green tea' and press Enter to view search results.	**High**: Search for 'organic green tea' and filter the results to show only products under the 'Health & Household' category, sorted by price from lowest to highest.

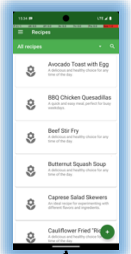
Reverse Task Synthesis

After reverse task synthesis generates task instructions, they are automatically executed in the GUI environment to build complete trajectories.

High: Mark the 'Avocado Toast with Egg' recipe as a favorite in the Broccoli app.

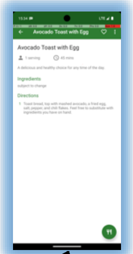
Low: I need to first open the Broccoli-Recipe App.

Action: OpenApp (“Broccoli”)



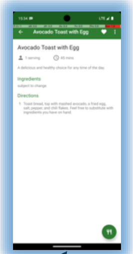
Low: Click on Avocado Toast with Egg to view more details.

Action: CLICK (698,528)



Low: Click “Mark as favorite”option.

Action: CLICK (912,201)

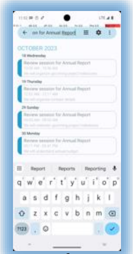


High: Set a reminder for the 'Review session for Annual Report' scheduled on October 18th in Simple Calendar Pro and save the changes.

Low: I need to first open the Simple Calendar Pro.

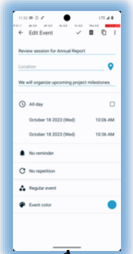
Action: OpenApp (“Simple Calendar Pro”)

……



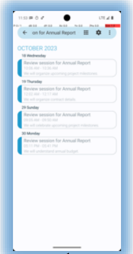
Low: Select the specific event “Review session for Annual Report”.

Action: CLICK (77, 596)



Low: Click “Save” button.

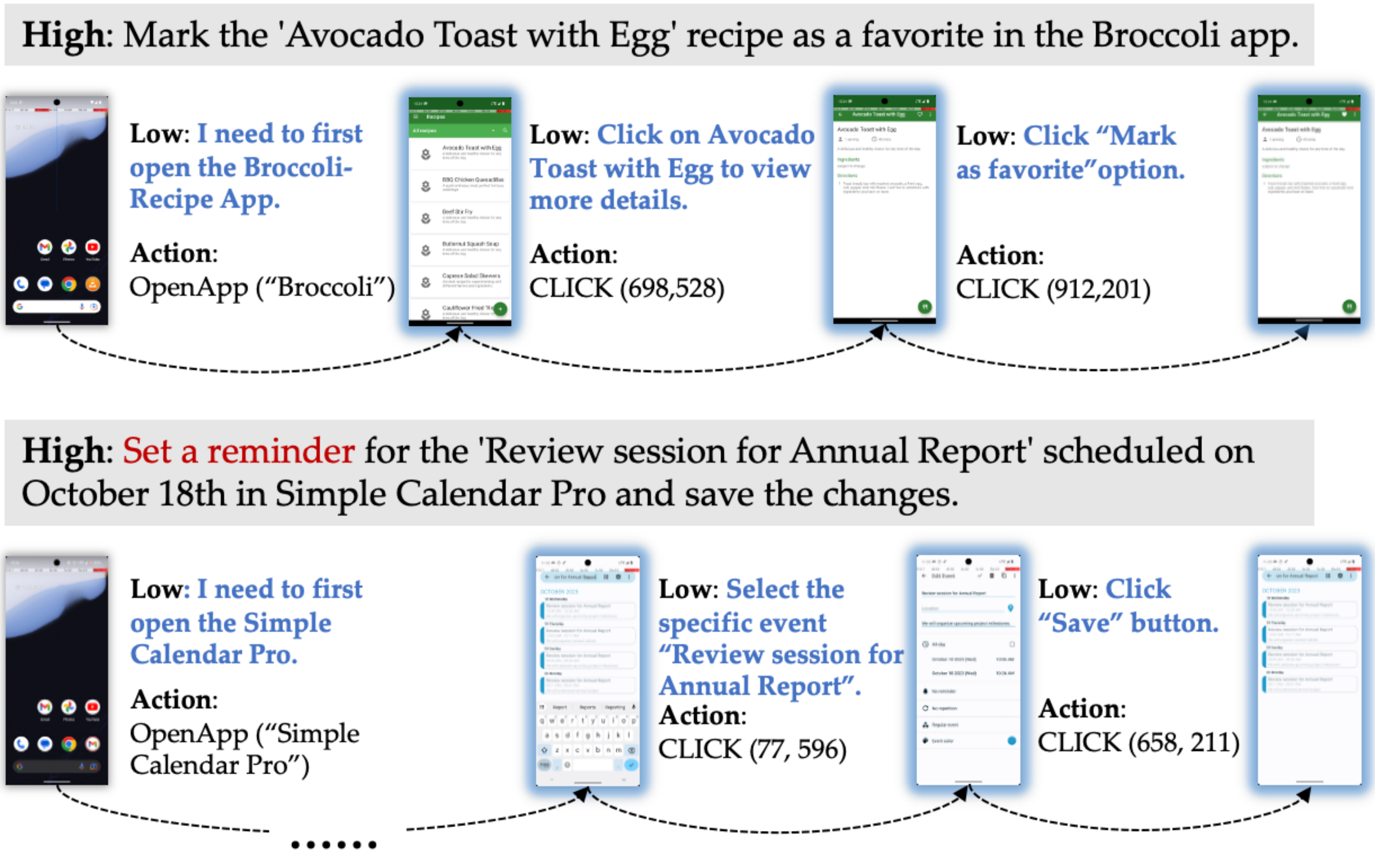
Action: CLICK (658, 211)

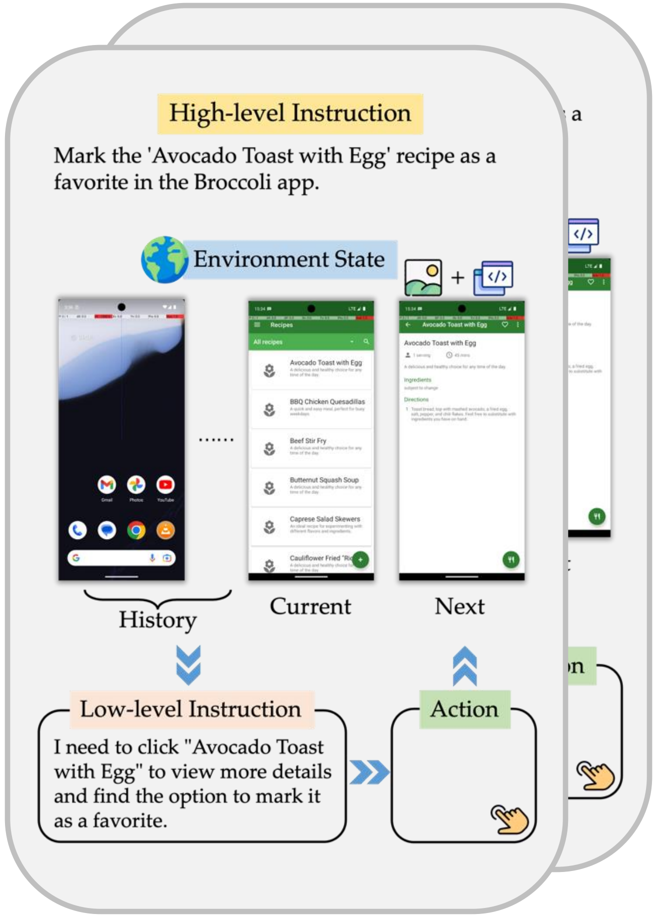


Reverse Task Synthesis

Trajectories collected! But is this all?

Let's consider data quality and synthesis efficiency.





Data Quality Control

Tasks are executed by machines, not all of them are successful.

Previous approach:

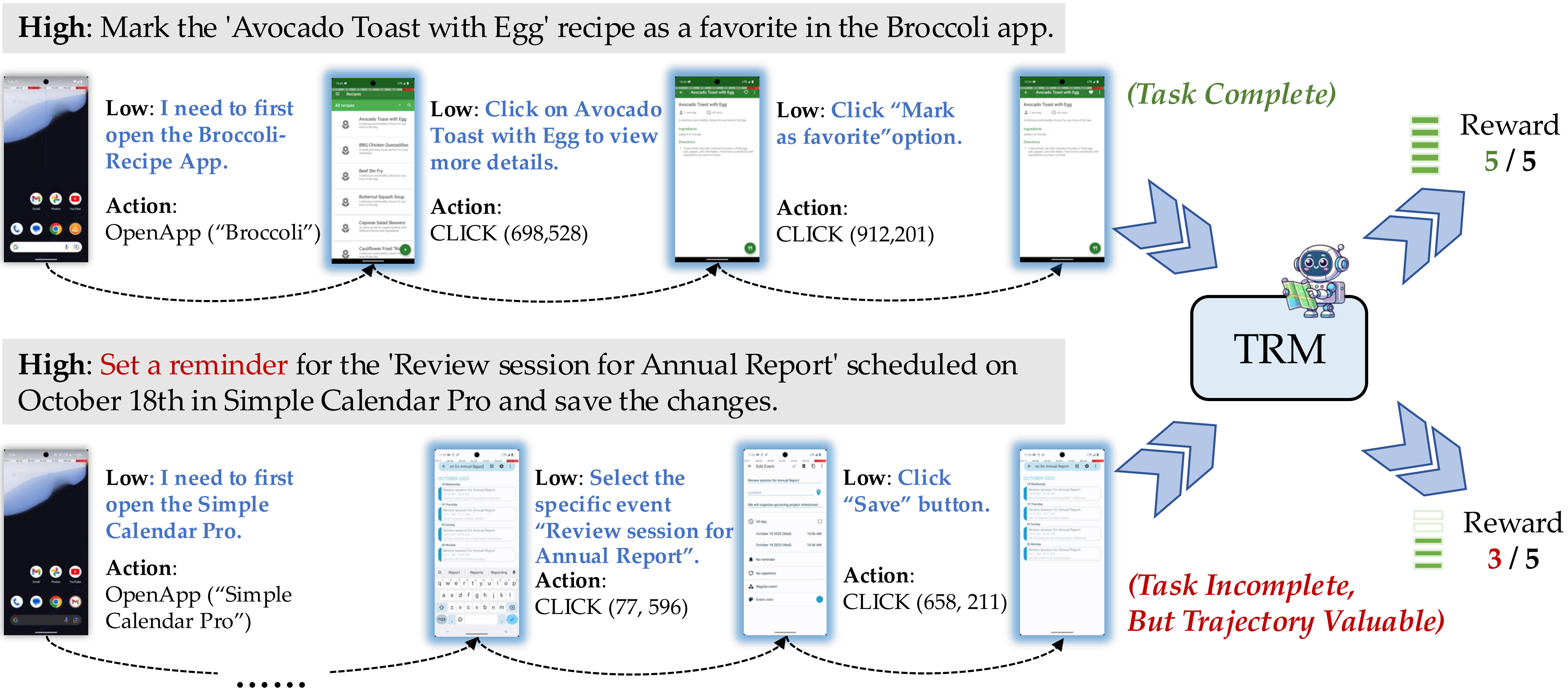
1. Training all data at once - what about the quality?

2. Discarding all incomplete Trajectories - what about the efficiency?

Thus, we introduce a Trajectory Reward Model to handle this.

Reward Modeling

We introduce a Trajectory Reward Model for weighted sampling in training.



Models

Data Synthesis

GPT-4o

Qwen-VL Qwen2-VL-72B-Instruct

Backbones

InternVL InternVL2-4B / 8B

Qwen-VL Qwen2-VL-7B-Instruct

Training Strategies

Leverage trajectory characteristics to train GUI agents with complete capabilities

1. Planning Training

High-level

Low-level

$$\mathcal{L}_1 = -\sum_{t_i \in \mathcal{T}} \log \Big(p_\theta(\ell \mid s, h_i, c) \cdot p_\theta(a \mid s, h_i, c, \ell) \Big)$$

Input

Actions

History

2. Action Training

$$\mathcal{L}_2 = -\sum_{t_i \in \mathcal{T}} \log p_\theta(a \mid s, c, \ell)$$

Training Strategies

After Training, our agents will generate ReACT-Style output

Examples:

Step 1: To create a new folder in Markor, I need to first open the Markor app.

action: `{"action_type":"open_app", "app_name":"Markor"}`

Step 2: To create a new folder, I need to click on the "Create a new file or folder" button, which is indicated by the plus icon.

action: `{"action_type":"click","x":964.5,"y":2074.5}`

Step 3: I need to change the folder name to folder_20241224. The current text field for the folder name is visible and editable.

action:

`{"action_type":"type","text":"folder_20241224","x":373.5,"y":552.0}`

…

Baselines

We adapt / build the following forward baselines

- **Zero-Shot.** Advanced prompting-based agents, such as M3A.
- **Task-Driven.** GUI Trajectories synthesized using pre-defined tasks. Given initial screenshots of the app/web page and task examples, use GPT-4 to generate high-level instructions and collect data.
- **Self-Instruct.** Builds on Task-Driven by adding self-instructed tasks.

Setting: Screenshot + A11ytree

[7] Self-Instruct: Aligning Language Models with Self-Generated Instructions, ACL 2023

Experiments: Mobile

Base Model	Strategies	AndroidWorld	AndroidControl-High		AndroidControl-Low	
			SR	Type	SR	Type
GPT-4o	Zero-Shot (M3A)	23.70	53.04	69.14	69.59	80.27
InternVL2-4B	Zero-Shot	0.00	16.62	39.96	33.69	60.65
	Task-Driven	4.02	27.37	47.08	66.48	90.37
	Task-Driven w. Self Instruct	7.14	24.95	44.27	66.70	90.79
	OS-Genesis	**15.18**	**33.39**	**56.20**	**73.38**	**91.32**
InternVL2-8B	Zero-Shot	2.23	17.89	38.22	47.69	66.67
	Task-Driven	4.46	23.79	43.94	64.43	89.83
	Task-Driven w. Self Instruct	5.36	23.43	44.43	64.69	89.85
	OS-Genesis	**16.96**	**35.77**	**64.57**	**71.37**	**91.27**
Qwen2-VL-7B	Zero-Shot	0.89	28.92	61.39	46.37	72.78
	Task-Driven	6.25	38.84	58.08	71.33	88.71
	Task-Driven w. Self Instruct	9.82	39.36	58.28	71.57	89.73
	OS-Genesis	**17.41**	**44.54**	**66.15**	**74.17**	**90.72**

Table 1: Performance on AndroidWorld and AndroidControl benchmarks.

Findings: OS-Genesis + Opensource VLM > Propriety Models + Complex Prompting

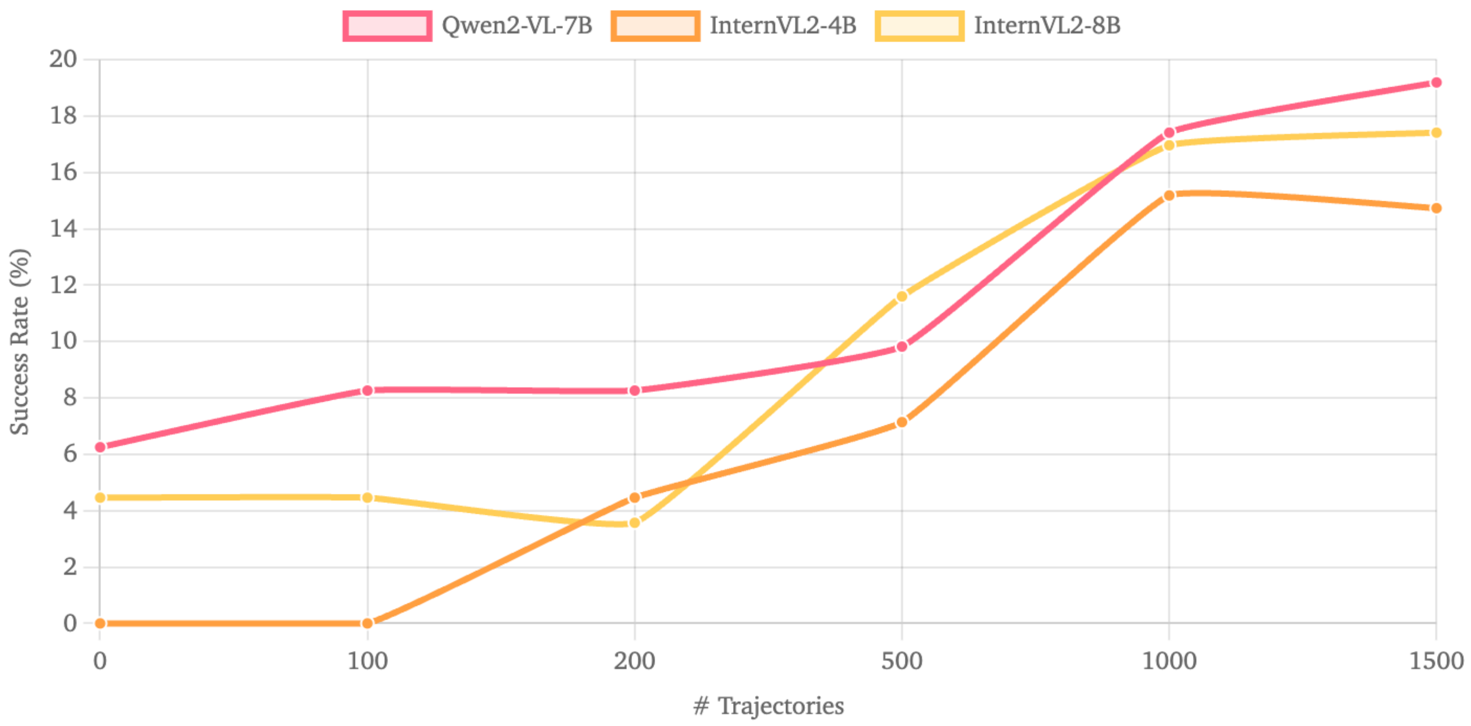
Experiments: Web

Base Model	Strategies	Shopping	CMS	Reddit	Gitlab	Maps	Overall
GPT-4o	Zero-Shot	14.28	21.05	6.25	14.29	20.00	16.25
InternVL2-4B	Zero-Shot	0.00	0.00	0.00	0.00	0.00	0.00
	Task-Driven	5.36	1.76	0.00	**9.52**	5.00	4.98
	Task-Driven w. Self Instruct	5.36	3.51	0.00	**9.52**	7.50	5.81
	OS-Genesis	**10.71**	**7.02**	**3.13**	7.94	**7.50**	**7.88**
InternVL2-8B	Zero-Shot	0.00	0.00	0.00	0.00	0.00	0.00
	Task-Driven	3.57	7.02	0.00	6.35	2.50	4.56
	Task-Driven w. Self Instruct	**8.93**	10.53	6.25	**7.94**	0.00	7.05
	OS-Genesis	7.14	**15.79**	**9.34**	6.35	**10.00**	**9.96**
Qwen2-VL-7B	Zero-Shot	**12.50**	7.02	6.25	6.35	5.00	7.47
	Task-Driven	8.93	7.02	6.25	6.35	5.00	7.05
	Task-Driven w. Self Instruct	8.93	1.76	3.13	4.84	**7.50**	5.39
	OS-Genesis	7.14	**8.77**	**15.63**	**15.87**	5.00	**10.79**

Table 2: Performance on WebArena benchmarks.

Analysis

How Scaling Trajectory Data Improves Agentic Ability?

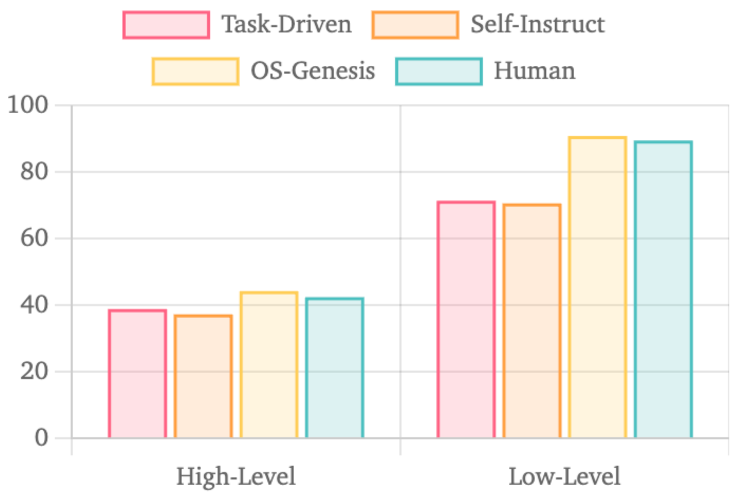


Insight: Generally improves, but will saturate.

Analysis

How Far are we from Human Data?

Let's first take a look at high-level instructions.

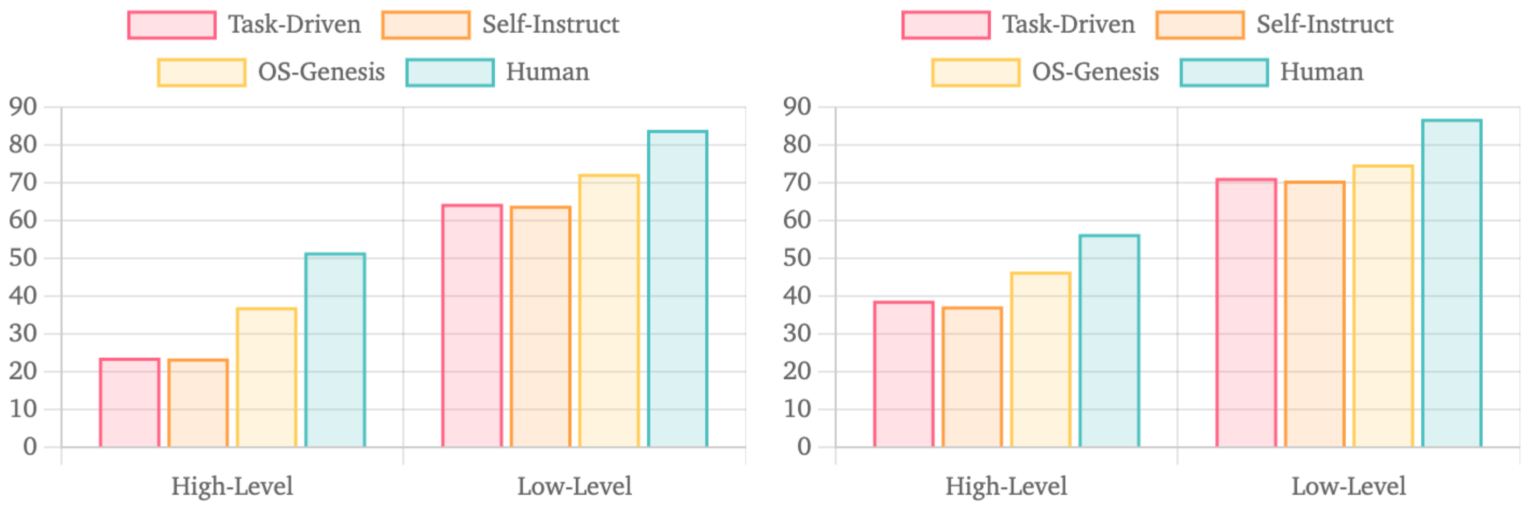


Insight: Reverse Task Synthesis Elicits Better Executability.

Analysis

How Far are we from Human Data?

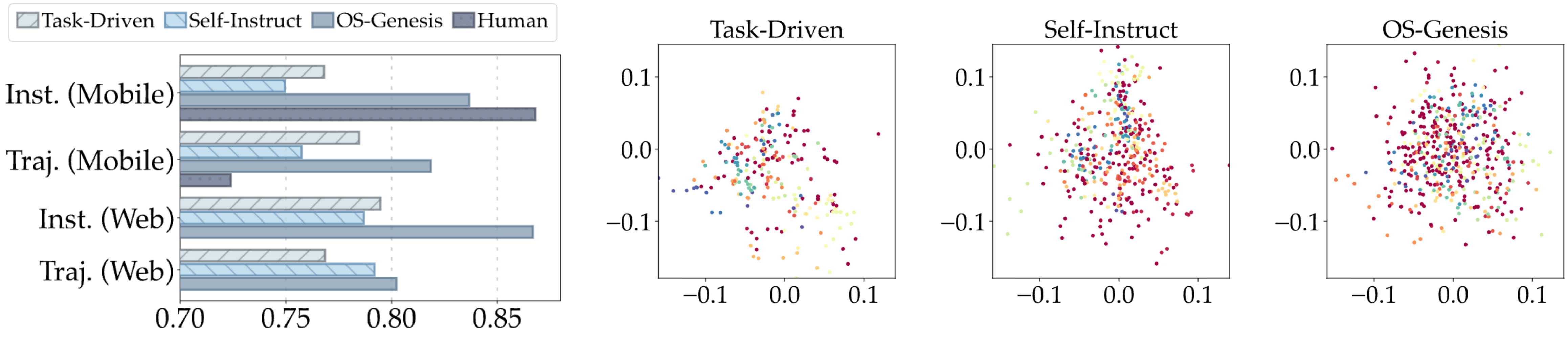
Then, OS-Genesis v.s. Human-annotated Trajectories.



Insight: OS-Genesis achieves ~80% of human data's effectiveness.

Analysis

How about our data diversity?



Insight: Significantly better than Forward methods and approaches the human level.

Checkpoints & Data Access

Available on ModelScope

Checkpoints & Data Access

Available on ModelScope

e.g., https://www.modelscope.cn/models/OS-Copilot/OS-Genesis-8B-AC

Our Project

OS-Genesis

Automating GUI Agent Trajectory Construction via Reverse Task Synthesis

Introducing OS-Genesis, a ***manual-free*** data pipeline for synthesizing GUI agent trajectory. OS-Genesis is characterized by the following core features:

- **Interaction-driven**: Agents actively explore GUI environments through stepwise interactions to discover functionalities and generate data.
- **Reverse Task Synthesis**: OS-Genesis retroactively derives meaningful low/high-level task instructions from observed interactions and state changes, enabling the construction of diverse and executable trajectories without pre-defined tasks.
- **Trajectory Data**: We construct and release high-quality mobile and web trajectories to accelerate GUI agents research.
- **Performance**: OS-Genesis significantly outperforms other synthesis methods on benchmarks like AndroidWorld and WebArena.

arXiv **Code** **Checkpoints** **Data**

https://qiushisun.github.io/OS-Genesis-Home/

Part3 | Future Directions and Early Attempts

We are just standing at the dawn of a long journey

There is still so much to do, such as:

1. Better action models
2. More advanced agent scheduling algorithms
3. Stronger planning capabilities
4. Safety, robustness and efficiency of agents

Let's look at some examples.

We are just standing at the dawn of a long journey

There is still so much to do, such as:

1. Better action models
2. More advanced agent scheduling algorithms
3. Stronger planning capabilities
4. Safety, robustness and efficiency of agents

Let's look at some examples.

ACL 2024
Bangkok, Thailand

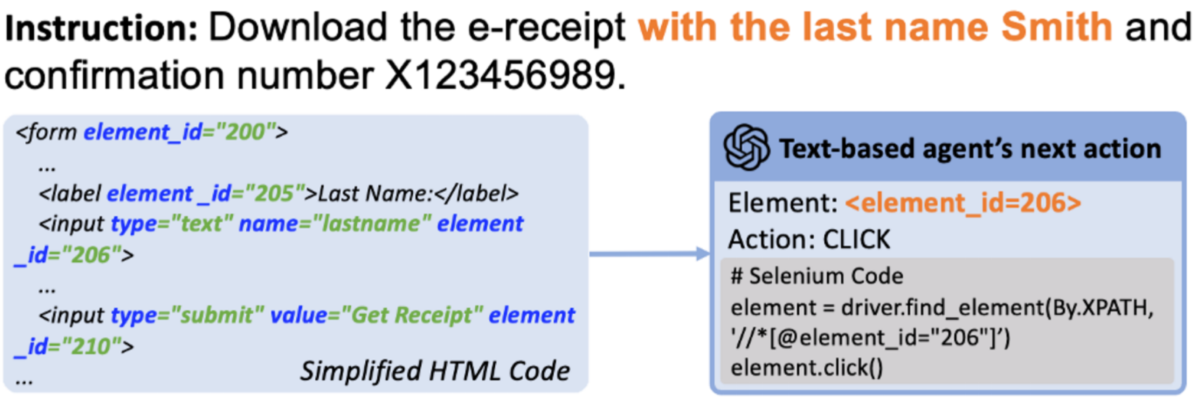
SeeClick: Harnessing GUI Grounding for Advanced Visual GUI Agents

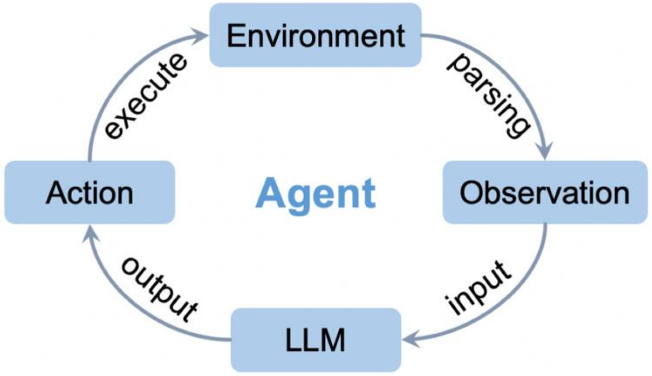
Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, Zhiyong Wu

上 海 人 工 智 能 实 验 室
Shanghai Artificial Intelligence Laboratory

NANJING UNIVERSITY 1902

GUI Agents depend on structured text face inherent limitations:

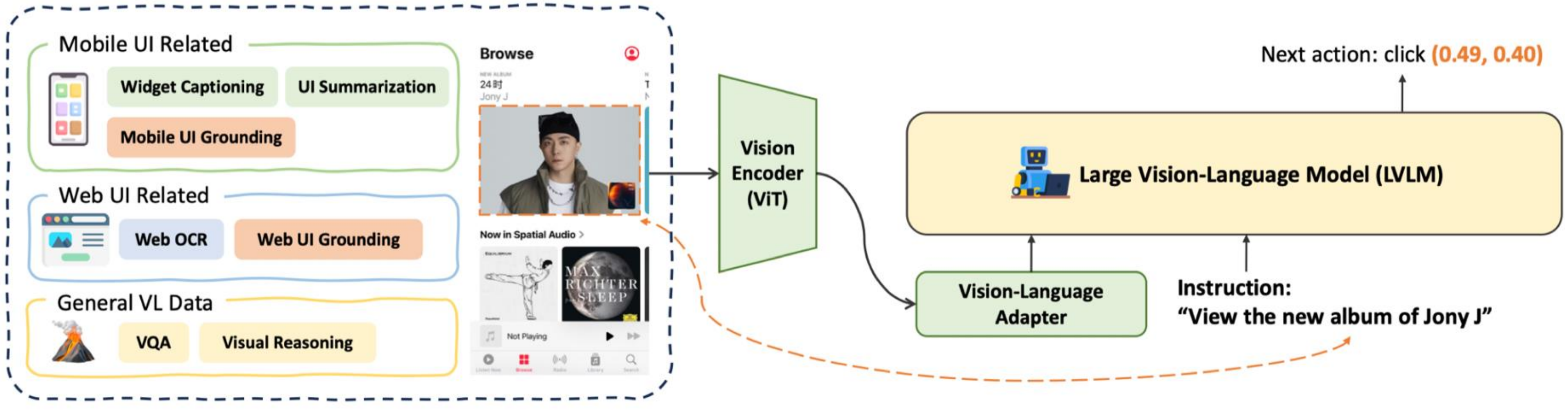




- Structured text representation is **not always available** (e.g. iOS and Desktop platform)
- Structured texts are **inconsistent**, with different representations across different platform (e.g., HTML, XLM , Accessibility Tree, …)

Our contribution:

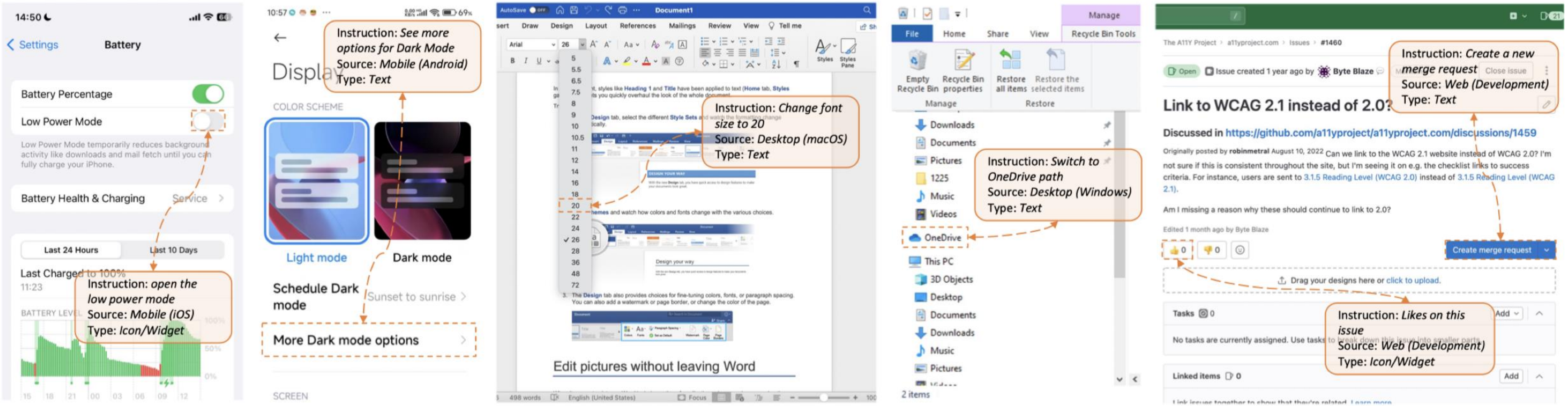
- GUI Grounding Pre-training: We applied GUI grounding continual pre-training to Qwen-VL to develop SeeClick
- An intuitive manner to perform element localization
- The first large-scale web grounding dataset



(a) Overview of SeeClick's framework and GUI grounding pre-training.

Figure 3: Example of two types of elements automatically collected from the webpage.

ScreenSpot: A Specialized GUI Grounding Benchmark



(b) Examples of the proposed GUI grounding benchmark *ScreenSpot*.

LVLMs	Model Size	GUI Specific	Mobile		Desktop		Web		Average
			Text	Icon/Widget	Text	Icon/Widget	Text	Icon/Widget	
MiniGPT-v2	7B	✗	8.4%	6.6%	6.2%	2.9%	6.5%	3.4%	5.7%
Qwen-VL	9.6B	✗	9.5%	4.8%	5.7%	5.0%	3.5%	2.4%	5.2%
GPT-4V	-	✗	22.6%	24.5%	20.2%	11.8%	9.2%	8.8%	16.2%
Fuyu	8B	✓	41.0%	1.3%	33.0%	3.6%	33.9%	4.4%	19.5%
CogAgent	18B	✓	67.0%	24.0%	**74.2%**	20.0%	**70.4%**	28.6%	47.4%
SeeClick	9.6B	✓	**78.0%**	**52.0%**	72.2%	**30.0%**	55.7%	**32.5%**	**53.4%**

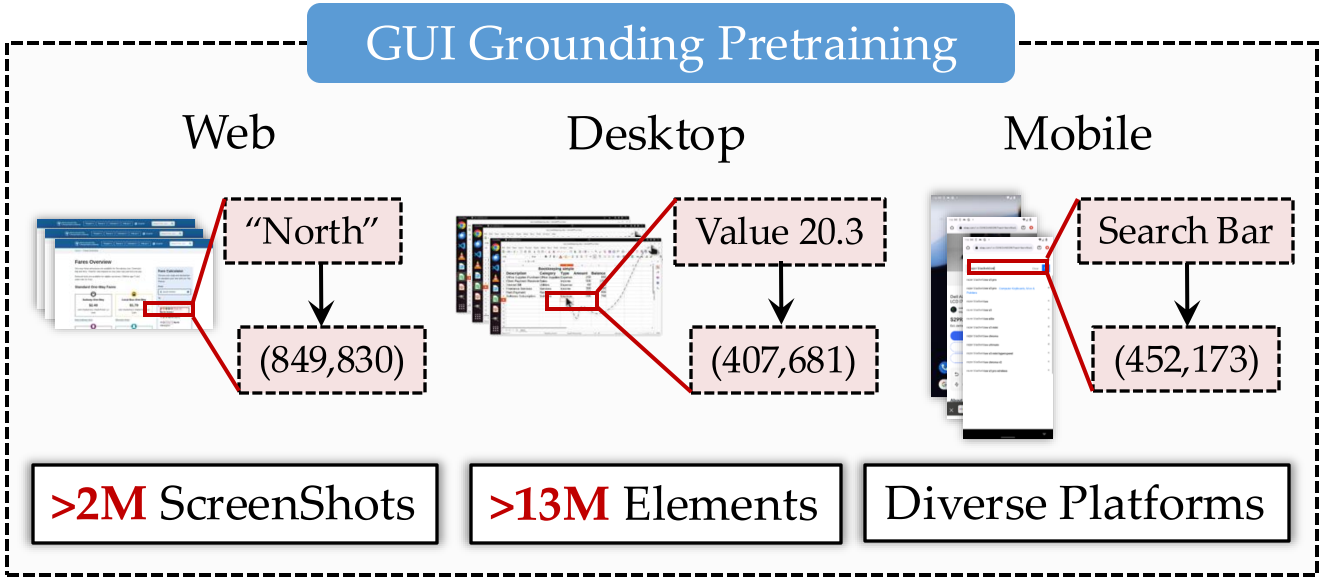
ICLR 2025 **Spotlight**

OS-ATLAS: A Foundation Action Model For Generalist GUI Agents

Zhiyong Wu[1*], Zhenyu Wu[1,2*], Fangzhi Xu[1*], Yian Wang[2*], Qiushi Sun[3], Chengyou Jia[1], Kanzhi Cheng[1], Zichen Ding[1], Liheng Chen[3], Paul Pu Liang[4], Yu Qiao[1]

[1]Shanghai AI Lab, [2]Shanghai Jiaotong University, [3]University of Hong Kong, [4]MIT

Infrastructure and Data Synthesis

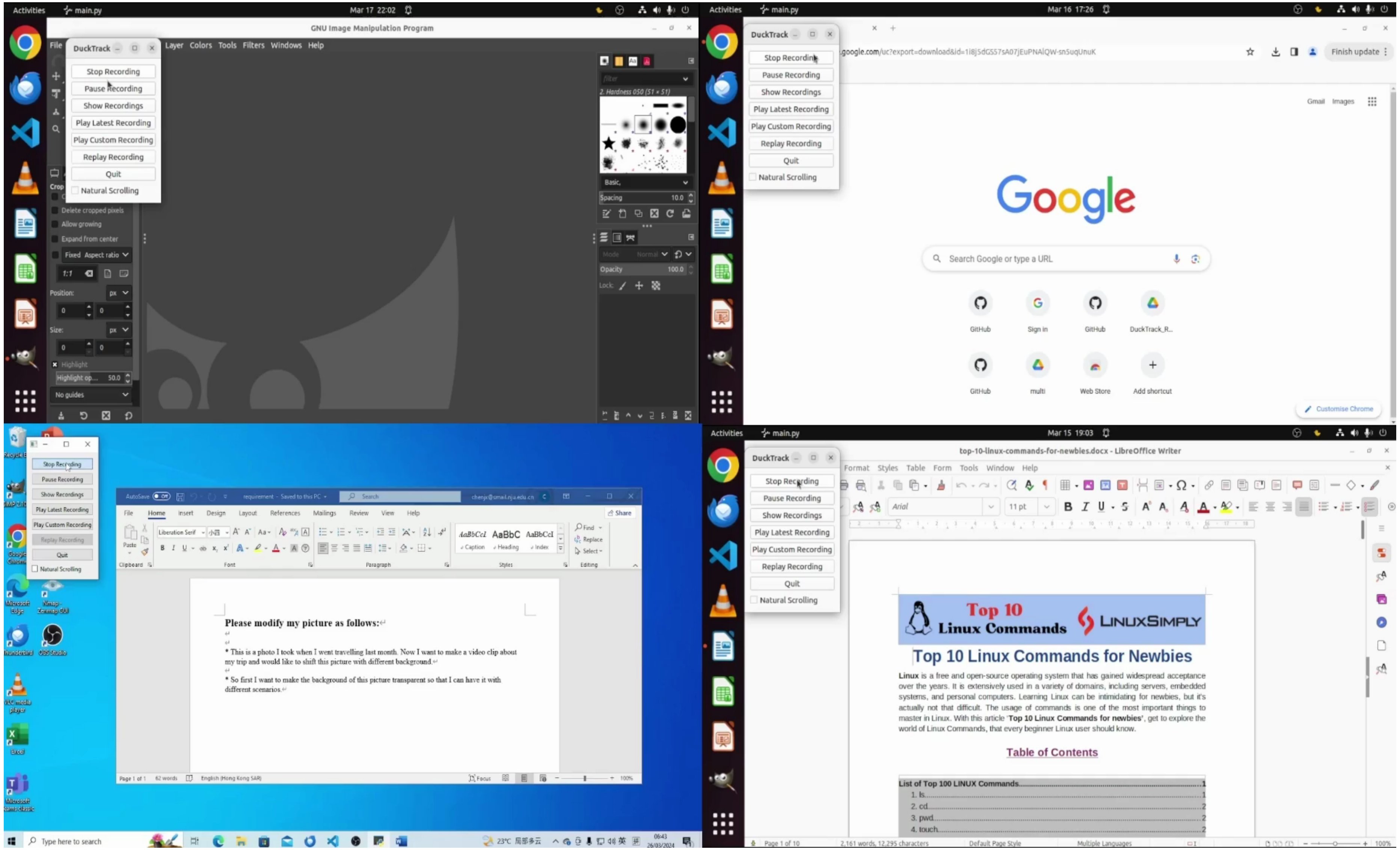


Dataset	#Screenshots Web	Mobile	Desktop	Open Source	#Elements
SeeClick	270K	94K	-	✓	3.3M
Ferret-UI	-	124K	-	✗	<1M
GUICourse	73K	9K	-	✓	10.7M
CogAgent	400K	-	-	✗	70M
OS-Atlas	1.9M	285K	54K	✓	13.58M

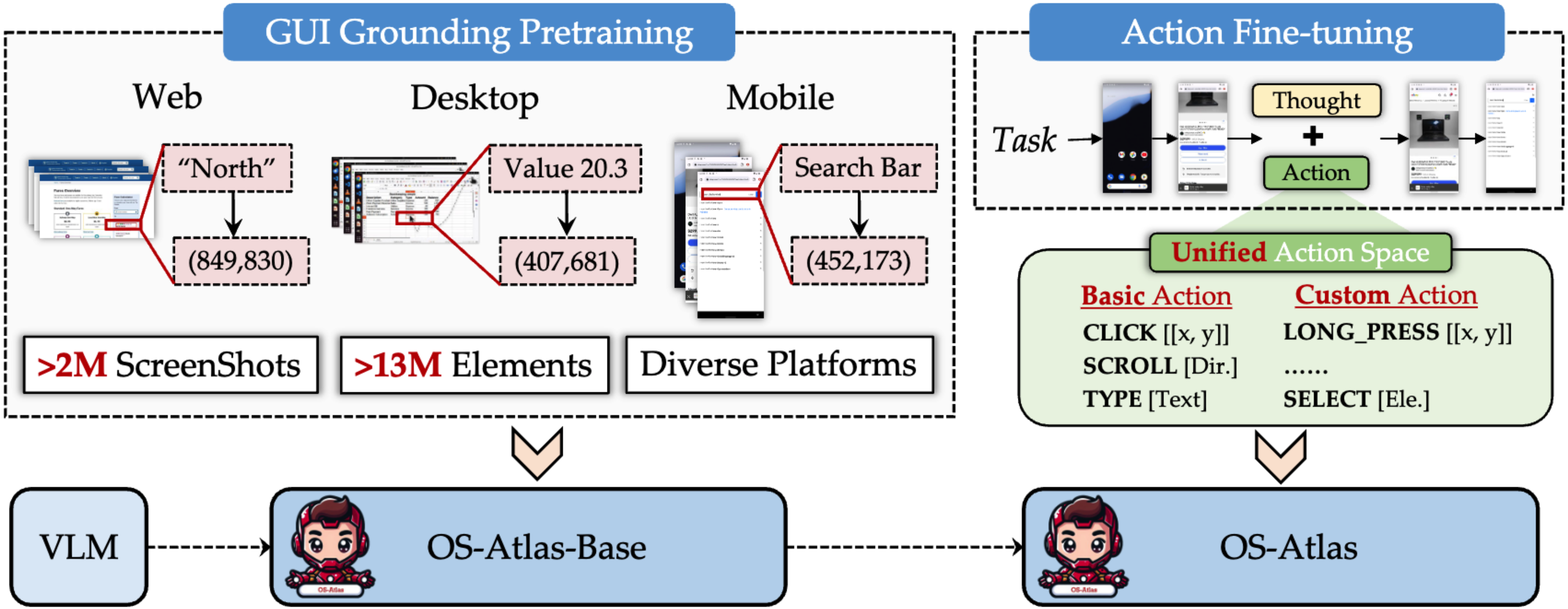
1. The first multi-platform GUI grounding data synthesis toolkit, including Windows, MacOS, Linux, Android, and the Web.

2. Comprises over 2.3M distinct screenshots and more than 13 million GUI elements.

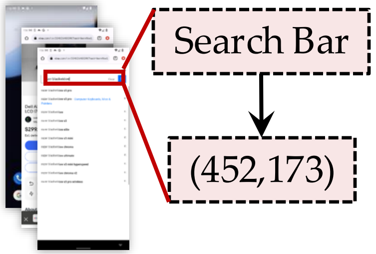
Data Synthesis with Random Walk



Two Stage Training



Visual Grounding Performance



Grounding Models	Mobile		Desktop		Web		Avg.
	Text	Icon/Widget	Text	Icon/Widget	Text	Icon/Widget	
Fuyu	41.00	1.30	33.00	3.60	33.90	4.40	19.50
CogAgent	67.00	24.00	74.20	20.00	70.40	28.60	47.40
SeeClick	78.00	52.00	72.20	30.00	55.70	32.50	53.40
InternVL-2-4B	9.16	4.80	4.64	4.29	0.87	0.10	4.32
Qwen2-VL-7B	61.34	39.29	52.01	44.98	33.04	21.84	42.89
UGround-7B	82.80	60.30	82.50	63.60	80.40	70.40	73.30
OS-Atlas-Base-4B	85.71	58.52	72.16	45.71	82.61	63.11	70.13
OS-Atlas-Base-7B	**93.04**	**72.93**	**91.75**	**62.86**	**90.87**	**74.27**	**82.47**

We are just standing at the dawn of a long journey

There is still so much to do, such as:

1. Better action models
2. More advanced agent scheduling algorithms
3. Stronger planning capabilities
4. Safety, robustness and efficiency of agents

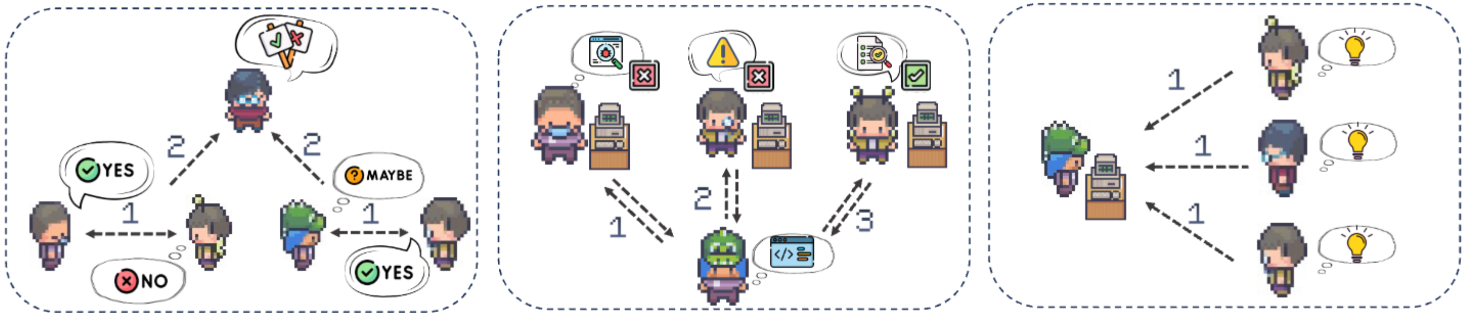
Let's look at some examples.

Multi-Agent Algorithms

Published as a conference paper at COLM 2024

Corex: Pushing the Boundaries of Complex Reasoning through Multi-Model Collaboration

Qiushi Sun[◇♡*] **Zhangyue Yin**[♣] **Xiang Li**[♠] **Zhiyong Wu**[◇†] **Xipeng Qiu**[♣] **Lingpeng Kong**[♡]
[◇]Shanghai AI Laboratory [♡]The University of Hong Kong
[♣]Fudan University [♠]East China Normal University
qiushisun@connect.hku.hk, yinzy21@m.fudan.edu.cn, xiangli@dase.ecnu.edu.cn
wuzhiyong@pjlab.org.cn, xpqiu@fudan.edu.cn, lpk@cs.hku.hk



Discuss **Review** **Retrieve**

How about multi-agent + GUI Agents

AgentStore: Scalable Integration of Heterogeneous Agents As Specialized Generalist Computer Assistant

Chengyou Jia[1,2], Minnan Luo[1,*], Zhuohang Dang[1], Qiushi Sun[2,3], Fangzhi Xu[1,2], Junlin Hu[2], Tianbao Xie[3] Zhiyong Wu[2,*]

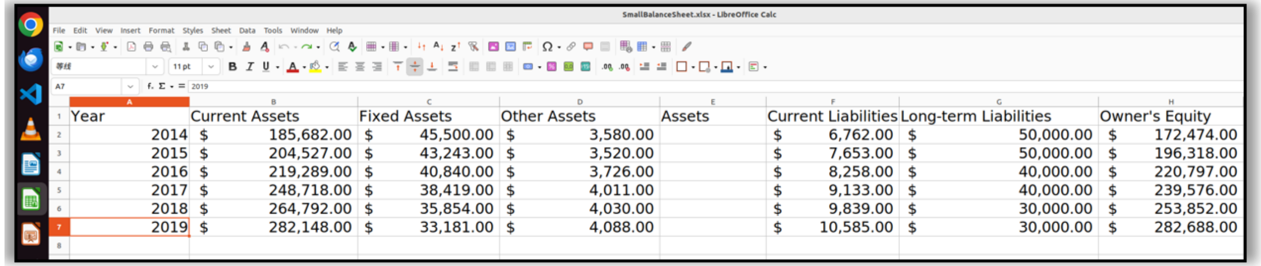
[1]Xi'an Jiaotong University [2]Shanghai Artificial Intellegence Laboratory [3]The University of Hong Kong

上海人工智能实验室
Shanghai Artificial Intelligence Laboratory

西安交通大學
XI'AN JIAOTONG UNIVERSITY

Can a Single Agent handle a variety of OS tasks?

Task_1: In a new sheet with 4 headers "Year", "CA changes", "FA changes", and "OA changes", calculate the annual changes for the Current Assets, Fixed Assets, and Other Assets columns.



Year	Current Assets	Fixed Assets	Other Assets	Assets	Current Liabilities	Long-term Liabilities	Owner's Equity
2014	$ 185,682.00	$ 45,500.00	$ 3,580.00		$ 6,762.00	$ 50,000.00	$ 172,474.00
2015	$ 204,527.00	$ 43,243.00	$ 3,520.00		$ 7,653.00	$ 50,000.00	$ 196,318.00
2016	$ 219,289.00	$ 40,840.00	$ 3,726.00		$ 8,258.00	$ 40,000.00	$ 220,797.00
2017	$ 248,718.00	$ 38,419.00	$ 4,011.00		$ 9,133.00	$ 40,000.00	$ 239,576.00
2018	$ 264,792.00	$ 35,854.00	$ 4,030.00		$ 9,839.00	$ 30,000.00	$ 253,852.00
2019	$ 282,148.00	$ 33,181.00	$ 4,088.00		$ 10,585.00	$ 30,000.00	$ 282,688.00

SheetAgent
specialize in sheet processing

Step 1: Install and locate

pip install openpyxl && lsof | grep '.xlsx'

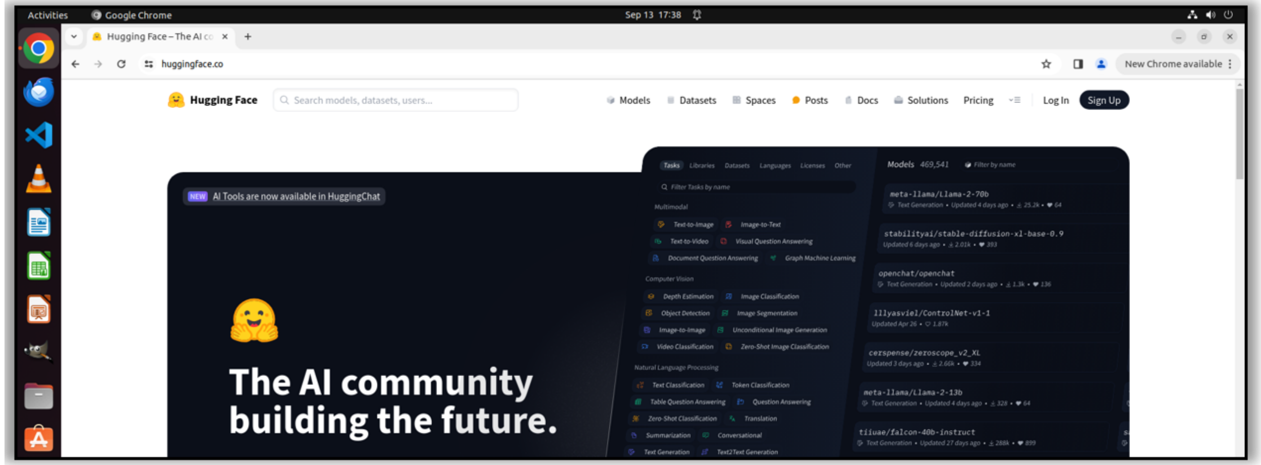
Step 2: Create new sheet and add headers

ws_new = wb.create_sheet(title=sheet_name)
ws_new.append(headers), wb.save(file_path)

Step 3: Insert table for the required data

for row in range(2, ws_original.max_row + 1):
year = ws_original.cell(arg).value,…
ws_new.append([year, …])

Task_2: Find the daily paper and take down the meta information of papers on 1st March, 2024 in the opened . pptx file. Please conform to the format and complete others.



WebAgent
specialize in web browsing

SildeAgent
specialize in slide editing

Different specialist agents are required to collaborate system-wide tasks

SubTask 1: Find papers and extract meta info

Step 1: Click daily papers to browsing
Step 2: Filter results by choosing1st March
Step 3: Extract info for selecting papers

subtask complete — message passing

SubTask 2: write meta info into pptx

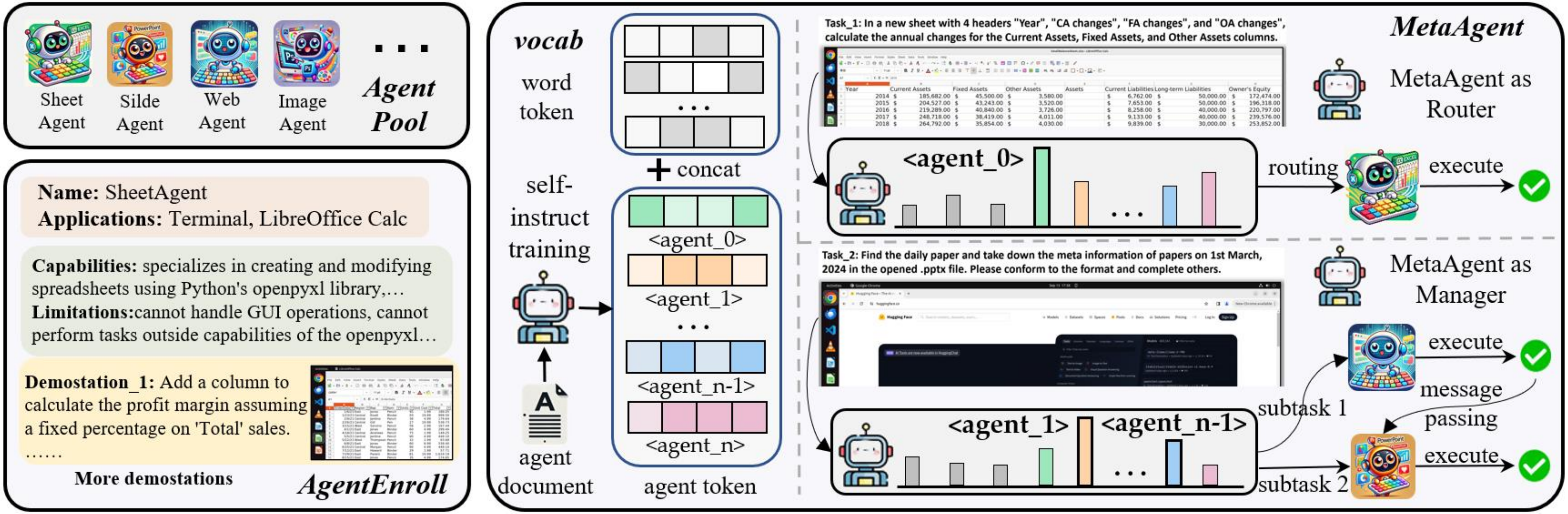
Step 1: Install package and locate .pptx file
Step 2: load content for current .pptx file
Step 3: Write info into corresponding file
Step 4: Save and overwrite the original file

Generalist Agent: lack of specialized abilities.
Specialized Agent: Unable to generalize to system-level tasks.

From APPStore to AgentStore:

Build an open and scalable platform for dynamically integrating various agents.

AgentStore

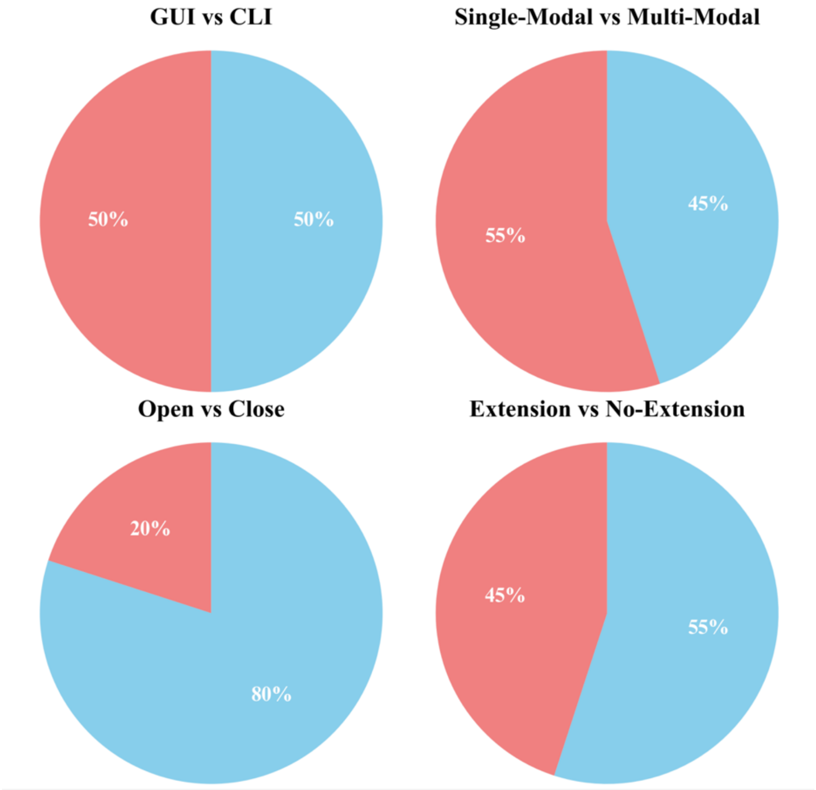


1. Quickly integrate their own specialized agents into the platform, similar to the functionality of the App store.
2. We introduce a novel MetaAgent with AgentToken strategy, to select the most suitable agent(s) to complete tasks.

Specialized agents in AgentStore

Table 6: The presentation of agents in the AgentPool.

	CLI or GUI?	Single or Multi Modal?	Open or Close Base Model?	Domain for OSworld	Support Extension?
OSAgent	GUI	Multi	Close	OS	✓
Friday (Wu et al., 2024)	CLI	Single	Close	OS	✓
SheetAgent	CLI	Single	Close	Calc	✗
CalcAgent	GUI	Multi	Close	Calc	✓
SlideAgent	CLI	Single	Close	Impress	✗
ImPressAgent	GUI	Multi	Close	Impress	✓
WordAgent	CLI	Single	Close	Writer	✗
WriterAgent	GUI	Multi	Close	Writer	✓
VLCAgent	GUI	Multi	Close	VLC	✓
MailAgent	GUI	Multi	Close	TB	✓
ChromeAgent	GUI	Multi	Close	Chrome	✓
WebAgent (He et al., 2024)	GUI	Multi	Close	Chrome	✗
VSAgent	GUI	Multi	Open	VSC	✗
VSGUIAgent	CLI	Single	Close	VSC	✓
GimpAgent	GUI	Multi	Close	GIMP	✓
ImageAgent	CLI	Single	Open	GIMP	✓
Searcher	CLI	Single	Close	-	✗
GoogleDrive	CLI	Single	Close	-	✗
CoderAgent	CLI	Single	Open	-	✗
VisionAgent	CLI	Multi	Open	-	✗



LLM/CLI-based model + VLM/GUI-based model

Performance

Agent	Base	Success Rate (%)									
		OS*	Calc	Impress	Writer	VLC	TB	Chrome	VSC	GIMP	AVG
CogAgent	GogVLM	1.60	2.17	0.00	4.35	6.53	0.00	2.17	0.00	0.00	1.32
MMAgent	GPT-4o	14.44	4.26	6.81	8.70	9.50	6.67	15.22	30.43	0.00	11.21
CRADLE	GPT-4o	8.00	0.00	4.65	8.70	6.53	0.00	8.70	0.00	38.46	7.81
Friday*	GPT-4o	15.20	25.50	0.00	21.73	0.00	0.00	0.00	17.39	15.38	11.11
Open-Inter*	GPT-4o	12.80	12.76	0.00	13.04	0.00	0.00	0.00	17.39	15.38	8.94
AgentStore(GT)	Hybrid	20.00	36.17	10.63	47.83	47.06	40.00	34.78	47.82	38.46	29.54
AgentStore(ICL)	Hybrid	9.60	0.00	2.13	4.34	35.29	33.33	30.43	30.43	15.38	13.55
AgentStore(FT)	Hybrid	8.80	27.65	4.26	13.04	41.17	40.00	34.78	8.60	15.38	17.34
AgentStore(AT)	Hybrid	13.86	31.91	8.51	39.13	47.06	40.00	32.61	39.13	30.77	23.85

AgentStore achieved a success rate of 23.85% on highly challenging OSWorld benchmark. (Claude 3.5 Sonnet: 22%)

Rank	Model
1 Oct 24, 2024	AgentStore (AgentToken) *Shanghai AI Lab* Shanghai AI Lab, '24
2 Oct 11, 2024	Agent S w/ GPT-4o *Simular Research* Simular Research, '24
3 Oct 11, 2024	Agent S w/ Claude-3.5 *Simular Research* Simular Research, '24
4 Oct 24, 2024	AgentStore (Fine-Tuning) *Shanghai AI Lab* Shanghai AI Lab, '24
5 Oct 24, 2024	AgentStore (In-Context Learning) *Shanghai AI Lab* Shanghai AI Lab, '24
6 Mar 20, 2024	GPT-4 Vision *OpenAI* OpenAI, '23

We are just standing at the dawn of a long journey

There is still so much to do, such as:

1. Better action models
2. More advanced agent scheduling algorithms
3. Stronger planning capabilities
4. Safety, robustness and efficiency of agents

...

Stay tuned!

Thanks for listening

Contact: qiushisun@connect.hku.hk