OpenHands: An Open Platform for AI Software Developers as Generalist Agents

Xingyao Wang[1,10], **Boxuan Li**[2], **Yufan Song**[2], **Frank F. Xu**[2], **Xiangru Tang**[3], **Mingchen Zhuge**[6], **Jiayi Pan**[4], **Yueqi Song**[2], **Bowen Li**, **Jaskirat Singh**[7], **Hoang H. Tran**[8], **Fuqiang Li**, **Ren Ma**, **Mingzhang Zheng**, **Bill Qian**[3], **Yanjun Shao**[3], **Niklas Muennighoff**[5], **Yizhe Zhang**, **Binyuan Hui**[9], **Junyang Lin**[9], **Robert Brennan**[10], **Hao Peng**[1], **Heng Ji**[1], **Graham Neubig**[2,10]
[1]UIUC [2]CMU [3]Yale [4]UC Berkeley [5]Contextual AI [6]KAUST [7]ANU
[8]HCMUT [9]Alibaba [10]All Hands AI
xingyao6@illinois.edu, gneubig@cs.cmu.edu

Abstract

Software is one of the most powerful tools that we humans have at our disposal; it allows a skilled programmer to interact with the world in complex and profound ways. At the same time, thanks to improvements in large language models (LLMs), there has also been a rapid development in AI agents that interact with and affect change in their surrounding environments. In this paper, we introduce OpenHands (*f.k.a.* OpenDevin), a platform for the development of powerful and flexible AI agents that interact with the world in similar ways to those of a human developer: by writing code, interacting with a command line, and browsing the web. We describe how the platform allows for the implementation of new agents, safe interaction with sandboxed environments for code execution, coordination between multiple agents, and incorporation of evaluation benchmarks. Based on our currently incorporated benchmarks, we perform an evaluation of agents over 15 challenging tasks, including software engineering (*e.g.*, SWE-bench) and web browsing (*e.g.*, WebArena), among others. Released under the permissive MIT license, OpenHands is a community project spanning academia and industry with more than 2.1K contributions from over 188 contributors.

	Code	https://github.com/All-Hands-AI/OpenHands
	Benchmark	https://hf.co/spaces/OpenHands/evaluation
	Slack	http://bit.ly/OpenHands-Slack

1 Introduction

Powered by large language models (LLMs; OpenAI 2024b; Team et al. 2023; Jiang et al. 2024; Chang et al. 2024), user-facing AI systems (such as ChatGPT) have become increasingly capable of performing complex tasks such as accurately responding to user queries, solving math problems, and generating code. In particular, AI *agents*, systems that can perceive and act upon the external environment, have recently received ever-increasing research focus. They are moving towards performing complex tasks such as developing software (Jimenez et al., 2024), navigating real-world websites (Zhou et al., 2023a), doing household chores (Ahn et al., 2022), or even performing scientific research (Boiko et al., 2023; Tang et al., 2024a).

As AI agents become capable of tackling complex problems, their development and evaluation have also become challenging. There are numerous recent efforts in creating open-source frameworks that facilitate the development of agents (Hong et al., 2023; Chen et al., 2024; Wu et al., 2023). These agent frameworks generally include: 1) **interfaces** through which agents interact with the world (such as JSON-based function calls or code execution), 2) **environments** in which agents operate, and 3) **interaction mechanisms** for human-agent or agent-agent communication. These frameworks streamline and ease the development process in various ways (Tab. 1, §C).

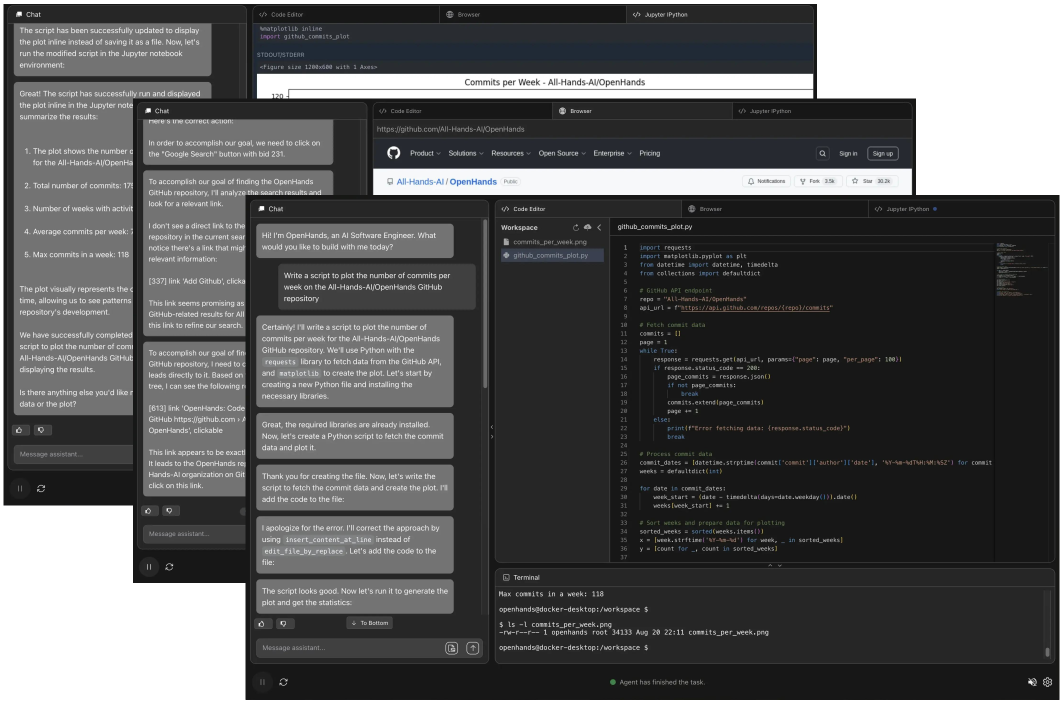

Figure 1: OpenHands User Interface (UI, §D) allows users to view files, check executed bash commands/Python code, observe the agent's browser activity, and directly interact with the agent.

When designing AI agents, we can also consider how *human* interacts with the world. The most powerful way in which humans currently interact with the world is through *software* – software powers every aspect of our life, supporting everything from the logistics for basic needs to the advancement of science, technology, and AI itself. Given the power of software, as well as the existing tooling around its efficient development, use, and deployment, it provides the ideal interface for AI agents to interact with the world in complex ways. However, building agents that can effectively develop software comes with its own unique challenges. How can we enable agents to effectively *create and modify code in complex software systems*? How can we provide them with tools to *gather information on-the-fly* to debug problems or gather task-requisite information? How can we ensure that development is *safe and avoids negative side effects* on the users' systems?

In this paper, we introduce OpenHands (*f.k.a.* OpenDevin), a community-driven platform designed for the development of generalist and specialist AI agents that interact with the world through software.[1] It features:

(1) An **interaction mechanism** which allows user interfaces, agents, and environments to interact through an *event stream* architecture that is powerful and flexible (§2.1).
(2) A **runtime environment** that consists of a docker-sandboxed operating system with a bash shell, a web browser, and IPython server that the agents can interact with (§2.2).
(3) An **interface** allowing the agent to interact with the environment in a manner similar to actual software engineers (§2.3). We provide the capability for agents to a) create and edit complex software, b) execute arbitrary code in the sandbox, and c) browse websites to collect information.
(4) **Multi-agent delegation**, allowing multiple specialized agents to work together (§2.4).
(5) **Evaluation framework**, facilitating the evaluation of agents across a wide range of tasks (§4).

Importantly, OpenHands is not just a conceptual framework, but it also includes a comprehensive and immediately usable implementation of agents, environments, and evaluations. As of this writing, OpenHands includes an agent hub with over 10 implemented agents (§3), including a strong generalist agent implemented based on the CodeAct architecture (Wang et al., 2024a), with additions for web browsing (ServiceNow) and code editing specialists (Yang et al., 2024). Interaction with users is

[1]While initially inspired by AI software engineer Devin (Cognition.ai), OpenHands has quickly evolved to support much wider range of applications beyond software engineering through diverse community contributions.

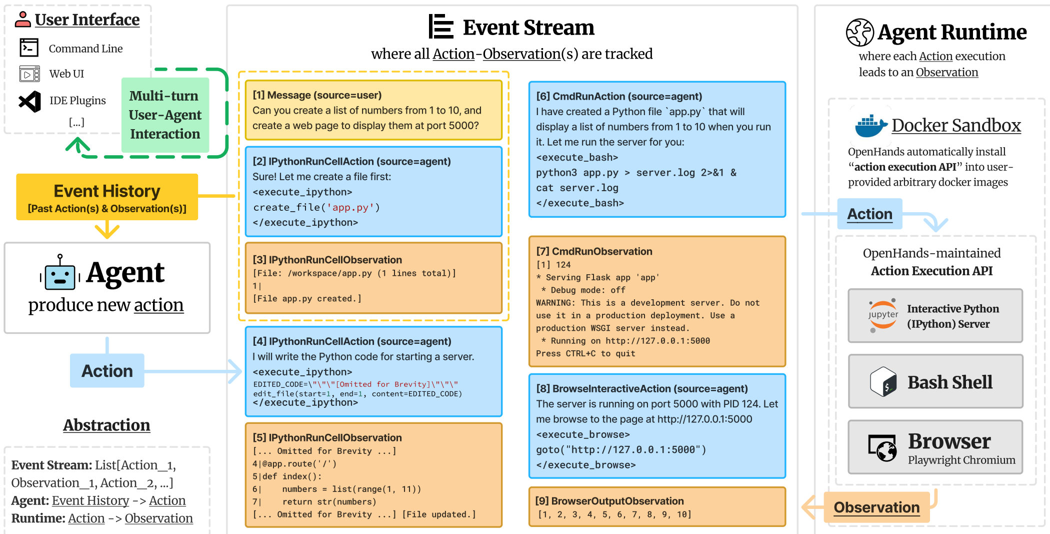

Figure 2: OpenHands consists of 3 main components: 1) **Agent abstraction** where community can contribute different implementation of agents (§2.1) into agenthub (§3); 2) **Event stream** for tracking history of actions and observations; 3) **Runtime** to execute all actions into observations (§2.2).

implemented through a chat-based user interface that visualizes the agent's current actions and allows for real-time feedback (Fig. 1, §D). Furthermore, the evaluation framework currently supports 15 benchmarks, which we use to evaluate our agents (§4).

Released under a permissive MIT license allowing commercial use, OpenHands is poised to support a diverse array of research and real-world applications across academia and industry. OpenHands has gained significant traction, with 32K GitHub stars and more than 2.1K contributions from over 188 contributors. We envision OpenHands as a catalyst for future research innovations and diverse applications driven by a broad community of practitioners.

2 OpenHands Architecture

We next describe using OpenHands in detail. In particular, we discuss 1) how to define and implement an agent (§2.1), 2) how each action execution leads to an observation (§2.2), 3) how to reliably manage and extend commonly used skills for agents (§2.3), and 4) how to compose multiple agents together for task solving (§2.4). Fig. 2 provides an overview.

2.1 Agent Definition and Implementation

An **agent** can perceive the **state** of the environment (*e.g.*, prior actions and observations) and produce an **action** for execution while solving a user-specified task.

The State and Event Stream. In OpenHands, the state is a data structure that encapsulates all relevant information for the agent's execution. A key component of this state is the **event stream**, which is a chronological collection of past actions and observations, including the agent's own actions and user interactions (*e.g.*, instructions, feedback). In addition to the event stream, the state incorporates auxiliary information for agent's operation, such as the accumulative cost of LLM calls, metadata to track multi-agent delegation (§2.4), and other execution-related parameters.

Actions. Inspired by CodeAct (Wang et al., 2024a), OpenHands connects an agent with the environment through a core set of general actions. Actions `IPythonRunCellAction` and `CmdRunAction` enable the agent to execute *arbitrary* Python code and bash commands inside the sandbox environment (*e.g.*, a securely isolated Linux operating system). `BrowserInteractiveAction` enables interaction with a web browser with a domain-specific language for browsing introduced by BrowserGym (Drouin et al., 2024). These actions were chosen to provide a comprehensive yet flexible set of primitives covering most tasks performed by human software engineers and analysts. The action space based on programming languages (PL) is powerful and flexible enough to perform any task with tools in different forms (*e.g.*, Python function, REST API, *etc.*) while being reliable and easy to maintain (Wang et al., 2024a) .

This design is also compatible with existing tool-calling agents that require a list of pre-defined tools (Chase, 2022). That is, users can easily define tools using PL supported in primitive actions (*e.g.*, write a Python function for calculator) and make those tools available to the agent through JSON-style function-calling experiences (Qin et al., 2023). Moreover, the framework's powerful PL-based primitives further make it possible for the agents to create tools by themselves (*e.g.*, by generating Python functions, Yuan et al. 2023) when API to complete the task is unavailable. Refer to §2.3 for how these core PL-based actions can be composed into a diverse set of tools.

Figure 3: Minimal example of implementing an agent in OpenHands.

```python
class MinimalAgent:
    def reset(self) -> None:
        self.system_message = "You are a helpful assistant ..."

    def step(self, state: State):
        messages: list[dict[str, str]] = [
            {'role': 'system', 'content': self.system_message}
        ]
        for prev_action, obs in state.history:
            action_message = get_action_message(prev_action)
            messages.append(action_message)
            obs_message = get_observation_message(obs)
            messages.append(obs_message)

        # use llm to generate response (e.g., thought, action)
        response = self.llm.do_completion(messages)

        # parse and execute action in the runtime
        action = self.parse_response(response)
        if self.is_finish_command(action):
            return AgentFinishAction()
        elif self.is_bash_command(action):
            return CmdRunAction(command=action.command)
        elif self.is_python_code(action):
            return IPythonRunCellAction(code=action.code)
        elif self.is_browser_action(action):
            return BrowseInteractiveAction(code=action.code)
        else:
            return MessageAction(content=action.message)
```

Observations. Observations describe the environmental changes (*e.g.*, execution result of prior actions, text messages from the human user *etc.*) that the agent observes.

Implement a New Agent. The agent abstraction is designed to be simple yet powerful, allowing users to create and customize agents for various tasks easily. The core of the agent abstraction lies in the `step` function, which takes the current state as input and generates an appropriate action based on the agent's logic. Simplified example code for the agent abstraction is illustrated in Fig. 3. By providing this abstraction, OpenHands allows the users to focus on defining desired agent behavior and logic without worrying about the low-level details of how actions are executed (§2.2).

2.2 Agent Runtime: How Execution of Actions Results in Observations

Agent Runtime provides a general environment that equips the agent with an action space comparable to that of human software developers, enabling OpenHands agents to tackle a wide range of software development and web-based tasks, including complex software development workflows, data analysis projects, web browsing tasks, and more. It allows the agent to access a bash terminal to run code and command line tools, utilize a Jupyter notebook for writing and executing code on-the-fly, and interact with a web browser for web-based tasks (*e.g.*, information seeking).

Docker Sandbox. For each task session, OpenHands spins up a securely isolated docker container sandbox, where all the actions from the event stream are executed. OpenHands connects to the sandbox through a REST API server running inside it (i.e., the OpenHands action execution API), executes arbitrary actions (e.g., bash command, python code) from the event stream, and returns the execution results as observations. A configurable workspace directory containing files the user wants the agent to work on is mounted into that secure sandbox for OpenHands agents to access.

OpenHands Action Execution API. OpenHands maintains an API server that runs *inside the docker sandbox* to listen for action execution requests from the event stream. The API server maintains:

(1) A bash shell that connects with the operating system environment (specified by the docker image) for command execution.
(2) A Jupyter IPython server to handle interactive *python* (IPython) code execution requests and return the execution results back to the event stream.
(3) A Chromium browser based on Playwright. The provider provides a set of action primitives defined by BrowserGym (ServiceNow; Drouin et al., 2024), such as navigation, clicking, typing, and scrolling. The full set of actions is detailed in §J. After executing these actions, the browser

runtime provides a rich set of observations about the current state of the browser, including HTML, DOM, accessibility tree (Mozilla), screenshot, opened tabs, *etc*.

Arbitrary Docker Image Support. OpenHands allows agents to run on arbitrary operating systems with different software environments by supporting runtime based on arbitrary docker images. OpenHands implements a build mechanism that takes a user-provided arbitrary docker image and installs OpenHands action execution API into that image to allow for agent interactions. We include a detailed description of OpenHands agent runtime in §F.

2.3 AGENT SKILLS: THE EXTENSIBLE AGENT-COMPUTER INTERFACE

SWE-Agent (Yang et al., 2024) highlights the importance of a carefully crafted Agent-Computer Interface (ACI, *i.e.*, specialized tools for particular tasks) in successfully solving complex tasks. However, creating, maintaining, and distributing a wide array of tools can be a daunting engineering challenge, especially when we want to make these tools available to different agent implementations (§3). To tackle these, we build an **AgentSkills library**, a toolbox designed to enhance the capabilities of agents, offering utilities not readily available through basic *bash* commands or *python* code.

Easy to create and extend tools. AgentSkills is designed as a Python package consisting of different utility functions (*i.e.*, tools) that are automatically imported into the Jupyter IPython environment (§2.2). The ease of defining a Python function as a tool lowers the barrier for community members to contribute new tools to the library. The generality of Python packages also allows different agent implementations to easily leverage these tools through one of our core action `IPythonRunCellAction` (§2.1).

Rigorously tested and maintained. We follow best practices in software engineering and write extensive unit tests for tools in AgentSkills to ensure their reliability and usability.

Inclusion criteria and philosophy. In the AgentSkills library, we do not aim to wrap every possible Python package and re-teach agents their usage (*e.g.*, LLM already knows `pandas` library that can read CSV file, so we don't need to re-create a tool that teaches the agent to read the same file format). We only add a new skill when: (1) it is not readily achievable for LLM to write code directly (*e.g.*, edit code and replace certain lines), and/or (2) it involves calling an external model (*e.g.*, calling a speech-to-text model, or model for code editing (Sanger)).

Currently supported skills. AgentSkills library includes file editing utilities adapted from SWE-Agent (Yang et al., 2024) and Aider (Gauthier) like `edit_file`, which allows modifying an existing file from a specified line; scrolling functions `scroll_up` and `scroll_down` for viewing a different part of files. It also contains tools that support reading multi-modal documents, like `parse_image` and `parse_pdf` for extracting information from images using vision-language models (*e.g.*, GPT-4V) and reading text from PDFs, respectively. A complete list of supported skills can be found in §I.

2.4 AGENT DELEGATION: COOPERATIVE MULTI-AGENT INTERACTION

OpenHands allows interactions between multiple agents as well. To this end, we use a special action type `AgentDelegateAction`, which enables an agent to delegate a specific subtask to another agent. For example, the generalist CodeActAgent, with limited support for web-browsing, can use `AgentDelegateAction` to delegate web browsing tasks to the specialized BrowsingAgent to perform more complex browsing activity (*e.g.*, navigate the web, click buttons, submit forms, *etc*.).

3 AGENTHUB: A HUB OF COMMUNITY-CONTRIBUTED AGENTS

Based on our agent abstraction (§2.1), OpenHands supports a wide range of community-contributed agent implementations for end users to choose from and act as baselines for different agent tasks.

CodeAct Agent. CodeActAgent is the default generalist agent based on the CodeAct framework (Wang et al., 2024a). At each step, the agent can (1) converse to communicate with humans in natural language to ask for clarification, confirmation, *etc*., or (2) to perform the task by executing code (*a.k.a.*, **CodeAct**), including executing bash commands, Python code, or browser-specific programming

Table 1: Comparison of different AI agent frameworks (§C). SWE refers to 'software engineering'. **Standardized tool library**: if framework contains reusable tools for different agent implementations (§2.3); **Built-in sandbox & code execution**: if it supports sandboxed execution of arbitrary agent-generated code; **Built-in web browser**: if it provides agents access to a fully functioning web browser; **Human-AI collaboration**: if it enables multi-turn human-AI collaboration (*e.g.*, human can interrupt the agent during task execution and/or provide additional feedback and instructions); **AgentHub**: if it hosts implementations of various agents (§3); **Evaluation Framework**: if it offers systematic evaluation of implemented agents on challenging benchmarks (§4); **Agent QC** (Quality Control): if the framework integrates tests (§E) to ensure overall framework software quality.

Framework	Domain	Graphic User Interface	Standardized Tool Library	Built-in Sandbox & Code Execution	Built-in Web Browser	Multi-agent Collaboration	Human-AI Collaboration	AgentHub	Evaluation Framework	Agent QC
AutoGPT Gravitas (2023)	General	✔	✗	✗	✗	✗	✗	✔	✗	✔
LangChain (Chase, 2022)	General	✗	✔	✗*	✗*	✗	✗	✔	✗	✗
MetaGPT (Hong et al., 2023)	General	✗	✔	✗	✔	✔	✗	✔	✗	✔
AutoGen (Wu et al., 2023)	General	✗	✔	✔	✔	✔	✔	✔	✔	✗
AutoAgents (Chen et al., 2024)	General	✗	✗	✗	✗	✔	✗	✗	✗	✗
Agents (Zhou et al., 2023b)	General	✗	✗	✗	✗	✔	✔	✗	✗	✗
Xagents (Team, 2023)	General	✔	✔	✗	✔	✔	✗	✔	✗	✗
OpenAgents (Xie et al., 2023)	General	✔	✗	✔	✔	✗	✗	✔	✗	✗
GPTSwarm (Zhuge et al., 2024)	General	✗	✔	✗	✗	✔	✔	✗	✗	✗
AutoCodeRover (Zhang et al., 2024b)	SWE	✗	✗	✔	✗	✗	✗	✗	✗	✗
SWE-Agent (Yang et al., 2024)	SWE	✗	✗	✔	✗	✗	✗	✗	✗	✗
OpenHands	General	✔	✔	✔	✔	✔	✔	✔	✔	✔

* No native support. Third-party commercial options are available.

language (§2.2). This general action space allows the agent (v1.5 and above) to perform various tasks, including editing files, browsing the web, running programs, etc.

Browsing Agent. We implemented a generalist web agent called Browsing Agent, to serve as a simple yet effective baseline for web agent tasks. The agent is similar to that in WebArena (Zhou et al., 2023a), but with improved observations and actions, with only zero-shot prompting. Full prompts are in §K.

GPTSwarm Agent. GPTSwarm (Zhuge et al., 2024) pioneers the use of optimizable graphs to construct agent systems, unifying language agent frameworks through modularity. Each node represents a distinct operation, while edges define collaboration and communication pathways. This design allows automatic optimization of nodes and edges, driving advancements in creating multi-agent systems.

Micro Agent(s). In addition, OpenHands enables the creation of **micro agent**, an agent *specialized* towards a particular task. A micro agent re-uses most implementations from an existing generalist agent (e.g., CodeAct Agent). It is designed to lower the barrier to agent development, where community members can share specialized prompts that work well for their particular use cases.

4 EVALUATION

To systematically track progress in building generalist digital agents, as listed in Tab. 2, we integrate 15 established benchmarks into OpenHands. These benchmarks cover software engineering, web browsing, and miscellaneous assistance. In this section, we compare OpenHands to open-source reproducible baselines that do not perform manual prompt engineering specifically based on the benchmark *content*. Please note that we use 'OH' as shorthand for OpenHands for the rest of this section for brevity reasons.

Table 2: Evaluation benchmarks in OpenHands.

Category	Benchmark	Required Capability
Software	SWE-Bench (Jimenez et al., 2024)	Fixing Github issues
	HumanEvalFix (Muennighoff et al., 2024)	Fixing Bugs
	BIRD (Li et al., 2023b)	Text-to-SQL
	BioCoder (Tang et al., 2024c)	Bioinformatics coding
	ML-Bench (Tang et al., 2024b)	Machine learning coding
	Gorilla APIBench (Patil et al., 2023)	Software API calling
	ToolQA (Zhuang et al., 2024)	Tool use
Web	WebArena (Zhou et al., 2023a)	Goal planning & realistic browsing
	MiniWoB++ (Liu et al., 2018)	Short trajectory on synthetic web
Misc. Assistance	GAIA (Mialon et al., 2023)	Tool-use, browsing, multi-modality
	GPQA (Rein et al., 2023)	Graduate-level Google-proof Q&A
	AgentBench (Liu et al., 2023)	Operating system interaction (bash)
	MINT (Wang et al., 2024b)	Multi-turn math and code problems
	Entity Deduction Arena (Zhang et al., 2024a)	State tracking & strategic planning
	ProofWriter (Tafjord et al., 2021)	Deductive Logic Reasoning

4.1 RESULT OVERVIEW

In OpenHands, our goal is to develop **general digital agents** capable of interacting with the world through software interfaces (as exemplified by the code actions described in §2.1). We recognize that

Table 3: Selected evaluation results for OpenHands agents (§4). See Tab. 4 (software), Tab. 5 (web), Tab. 6 (miscellaneous assistance) for full results across benchmarks.

Agent	Model	Software (§4.2) SWE-Bench Lite	Web (§4.3) WebArena	Misc. (§4.4) GPQA	GAIA
Software Engineering Agents					
SWE-Agent (Yang et al., 2024)	`gpt-4-1106-preview`	18.0	–	–	–
AutoCodeRover (Zhang et al., 2024b)	`gpt-4-0125-preview`	19.0	–	–	–
Aider (Gauthier)	`gpt-4o` & `claude-3-opus`	26.3	–	–	–
Moatless Tools (Örwall)	`claude-3.5-sonnet`	26.7	–	–	–
Agentless (Xia et al., 2024)	`gpt-4o`	27.3	–	–	–
Web Browsing Agents					
Lemur (Xu et al., 2023)	`Lemur-chat-70b`	–	5.3	–	–
Patel et al. (2024)	Trained 72B w/ synthetic data	–	9.4	–	–
AutoWebGLM (Lai et al., 2024)	Trained 7B w/ human/agent annotation	–	18.2	–	–
Auto Eval & Refine (Pan et al., 2024)	GPT-4 + Reflexion w/ GPT-4V	–	20.2	–	–
WebArena Agent (Zhou et al., 2023a)	`gpt-4-turbo`	–	14.4	–	–
Misc. Assistance Agents					
AutoGPT (Gravitas, 2023)	`gpt-4-turbo`	–	–	–	13.2
Few-shot Prompting + Chain-of-Thought (Rein et al., 2023)	`Llama-2-70b-chat`	–	–	28.1	–
	`gpt-3.5-turbo-16k`	–	–	29.6	–
	`gpt-4`		–	38.8	–
OpenHands Agents					
CodeActAgent `v1.8`	`gpt-4o-mini-2024-07-18`	6.3	8.3	–	–
	`gpt-4o-2024-05-13`	22.0	14.5	*53.1	–
	`claude-3-5-sonnet`	26.0	15.3	52.0	–
GPTSwarm `v1.0`	`gpt-4o-2024-05-13`	–	–	–	32.1

* Numbers are reported from CodeActAgent `v1.5`.

a software agent should excel not only in code editing but also in web browsing and various auxiliary tasks, such as answering questions about code repositories or conducting online research.

Tab. 3 showcases a curated set of evaluation results. While OpenHands agents may not achieve top performance in every category, they are designed with generality in mind. Notably, the same CodeAct agent, without any modifications to its system prompt, demonstrates competitive performance across three major task categories: software development, web interaction, and miscellaneous tasks. This is particularly significant when compared to the baseline agents, which are typically designed and optimized for specific task categories.

4.2 Software Engineering

Next, we report results specifically for software engineering benchmarks in Tab. 4.

SWE-Bench (Jimenez et al., 2024) is designed to assess agents' abilities in solving real-world GitHub issues, such as bug reports or feature requests. The agent interacts with the repository and attempts to fix the issue provided through file editing and code execution. The agent-modified code repository is tested against a test suite incorporating new tests added from human developers' fixes for the same issue. Each test instance accompanies a piece of "hint text" that consists of natural language suggestions for how to solve the problem. Throughout this paper, we report all results *without using hint text*. A canonical subset, SWE-bench Lite, is created to facilitate accessible and efficient testing. We default to use this subset for testing for cost-saving consideration.[2] **Result.** As shown in Tab. 4, our most recent version of CodeActAgent v1.8, using `claude-3.5-sonnet`, achieves a competitive resolve rate of 26% compared to other open-source software development specialists.

4.2.1 HumanEvalFix

HumanEvalFix (Muennighoff et al., 2024) tasks agents to fix a bug in a provided function with the help of provided test cases. The bugs are created to ensure one or more test cases fail. We focus on the Python subset of the benchmark and allow models to solve the bugs by self-debug over multiple turns, incorporating feedback from test execution. We follow the setup from Muennighoff et al. (2024) using pass@k (Chen et al., 2021). **Results.** In Tab. 4, OpenHands CodeActAgent successfully fixes 79.3% of bugs in the Python split. This is significantly better than all non-agentic approaches,

[2] Running the complete set of 2294 instances costs $6.9k, using a conservative estimate of $3 per instance.

Table 4: OpenHands Software Engineering evaluation results (§4.2).

Agent	Model	Success Rate (%)	$ Avg. Cost
SWE-Bench Lite (Jimenez et al., 2024), 300 instances, *w/o Hint*			
SWE-Agent (Yang et al., 2024)	`gpt-4-1106-preview`	18.0	1.67
AutoCodeRover (Zhang et al., 2024b)	`gpt-4-0125-preview`	19.0	–
Aider (Gauthier)	`gpt-4o` & `claude-3-opus`	26.3	–
OH CodeActAgent v1.8	`gpt-4o-mini-2024-07-18`	7.0	0.01
	`gpt-4o-2024-05-13`	22.0	1.72
	`claude-3-5-sonnet@20240620`	26.0	1.10
HumanEvalFix (Muennighoff et al., 2024), 164 instances			
Prompting, 0-shot	`BLOOMZ-176B`	16.6	–
	`OctoCoder-15B`	30.4	–
	`DeepSeekCoder-33B-Instruct`	47.5	–
	`StarCoder2-15B`	48.6	–
SWE-agent, 1-shot (Yang et al., 2024)	`gpt-4-turbo`	87.7	–
OH CodeActAgent v1.5, Generalist, 0-shot.	`gpt-3.5-turbo-16k-0613`	20.1	0.11
	`gpt-4o-2024-05-13`	79.3	0.14
BIRD (Li et al., 2023b), 300 instances			
Prompting, 0-shot	`CodeLlama-7B-Instruct`	18.3	-
	`CodeQwen-7B-Chat`	31.3	-
OH CodeActAgent v1.5	`gpt-4-1106-preview`	42.7	0.19
	`gpt-4o-2024-05-13`	47.3	0.11
ML-Bench (Tang et al., 2024b), 68 instances			
prompting + BM25, 0-shot	`gpt-3.5-turbo`	11.0	-
	`gpt-4-1106-preview`	22.1	-
	`gpt-4o-2024-05-13`	26.2	-
SWE-Agent (Yang et al., 2024)	`gpt-4-1106-preview`	42.6	1.91
Aider (Gauthier)	`gpt-4o`	64.4	-
OH CodeActAgent v1.5	`gpt-4o-2024-05-13`	76.5	0.25
	`gpt-4-1106-preview`	58.8	1.22
	`gpt-3.5-turbo-16k-0613`	13.2	0.12
BioCoder (Python) (Tang et al., 2024b), 157 instances			
prompting, 0-shot	`gpt-3.5-turbo`	11.0	-
	`gpt-4-1106-preview`	12.7	-
OH CodeActAgent v1.5	`gpt-4o-2024-05-13`	27.5	0.13
BioCoder (Java) (Tang et al., 2024b), 50 instances			
prompting, 0-shot	`gpt-3.5-turbo`	4.1	-
	`gpt-4-1106-preview`	6.4	-
OH CodeActAgent v1.5	`gpt-4o-2024-05-13`	44.0	0.11
Gorilla APIBench (Patil et al., 2023), 1775 instances			
Prompting, 0-shot	`claude-v1`	8.7	-
	`gpt-4-0314`	21.2	-
	`gpt-3.5-turbo-0301`	29.7	-
Gorilla, finetuned for API calls, 0-shot (Patil et al., 2023; Touvron et al., 2023)	`llama-7b`	75.0	-
OH CodeActAgent v1.5	`gpt-3.5-turbo-0125`	21.6	0.002
	`gpt-4o-2024-05-13`	36.4	0.04
ToolQA (Zhuang et al., 2024), 800 instances			
Prompting, 0-shot	`ChatGPT + CoT`	5.1	-
	`ChatGPT`	5.6	-
	`Chameleon`	10.6	-
ReAct, 0-shot (Yao et al., 2023; OpenAI, 2024a)	`gpt-3.5-turbo`	36.8	-
	`gpt-3`	43.1	-
OH CodeActAgent v1.5	`gpt-3.5-turbo-0125`	2.3	0.03
	`gpt-4o-2024-05-13`	47.2	0.91

almost doubling the performance of `StarCoder2-15B` (Lozhkov et al., 2024; Li et al., 2023c). While SWE-Agent achieves 87.7%, Yang et al. (2024) provides the model a full demonstration of a successful sample trajectory fixing one of the bugs in the test dataset ("1-shot"), whereas our evaluation of OpenHands is 0-shot. As HumanEvalFix has been created by humans and all bugs carefully validated, achieving 100% on this benchmark is entirely feasible, which we seek to do in future iterations of OpenHands.

ML-Bench (Tang et al., 2024b) evaluates agents' ability to solve machine learning tasks across 18 GitHub repositories. The benchmark comprises 9,641 tasks spanning 169 diverse ML problems, requiring agents to generate bash scripts or Python code in response to user instructions. In the sandbox environment, agents can iteratively execute commands and receive feedback, allowing them to understand the repository context and fulfill user requirements progressively. Following the setup from the original paper, we perform agent evaluation on the quarter subset of ML-Bench. **Results.** As shown in Table 4, OpenHands agents with GPT-4o achieve the highest success rate of 76.47% on ML-Bench, outperforming SWE-Agent (42.64%). Performance drops with less capable models. These results demonstrate the effectiveness of OpenHands agent in complex ML tasks. We notice that agents show potential in reducing hallucination and syntax errors compared to non-agent approaches in the ML-LLM-Bench settings (Tang et al., 2024b).

Gorilla APIBench (Patil et al., 2023) evaluates agents' abilities to use APIs. it incorporates tasks on TorchHub, TensorHub, and HuggingFace. During the evaluation, models are given a question related to API usage, such as "*identify an API capable of converting spoken language in a recording to text*." Correctness is evaluated based on whether the model's API call is in the correct domain. **Results.** As shown in Table 4, OpenHands using GPT-4o, with a success rate of 36.4%, outperforms baselines not specifically finetuned for API calling. While Gorilla shows higher performance on APIBench, Patil et al. (2023) finetune this model for API calling in particular.

ToolQA (Zhuang et al., 2024) evaluates agents' abilities to use external tools. This benchmark includes tasks on various topics like flight status, coffee price, Yelp data, and Airbnb data, requiring the use of various tools such as text tools, database tools, math tools, graph tools, code tools, and system tools. It features two levels: easy and hard. Easy questions focus more on single-tool usage, while hard questions emphasize reasoning. We adopt the easy subset for evaluation. **Results.** Compared to all baselines, OpenHands with GPT-4o shows the highest performance. We notice that agents perform better on tasks related to CSV and database tool usage but requires improvements on math and calculator tool usage.

BioCoder (Tang et al., 2024c) is a repository-level code generation benchmark that evaluates agents' performance on bioinformatics-related tasks, specifically the ability to retrieve and accurately utilize context. The original prompts contain the relevant context of the code; however, in this study, we have removed them to demonstrate the capability of OpenHands to perform context retrieval, self-debugging, and reasoning in multi-turn interactions. BioCoder consists of 157 Python and 50 Java functions, each targeting a specific area in bioinformatics, such as proteomics, genomics, and other specialized domains. The benchmark targets real-world code by generating code in existing repositories where the relevant code has been masked out. **Results.** Table 4 shows that OpenHands, using GPT-4o, achieves a success rate of 44.0%. This outperforms all prompting-based non-agent baselines, with GPT-4 alone only achieving 6.4%.

BIRD (Li et al., 2023b) is a benchmark for text-to-SQL tasks (*i.e.*, translate natural language into executable SQL) aimed at realistic and large-scale database environments. We select 300 samples from the dev set to integrate into OpenHands and evaluate on execution accuracy. Additionally, we extend the setting by allowing the agent to engage in multi-turn interactions to arrive at the final SQL query, enabling it to correct historical results by observing the results of SQL execution. **Results.** As shown in Table 4, OpenHands with GPT-4o achieves an execution accuracy of 47.3% on a subset of BIRD, showcasing the potential of OpenHands as a SQL agent. The result outperforms approaches utilizing prompting with code LLMs, such as CodeLlama-7B-Instruct (18.3%) and CodeQwen-7B-Chat (Bai et al., 2023) (31.3%).

4.3 Web Browsing

We report evaluation results for web browsing benchmarks in Tab. 5.

WebArena (Zhou et al., 2023a) is a self-hostable, execution-based web agent benchmark that allows agents to freely choose which path to take in completing their given tasks. WebArena comprises 812 human-curated task instructions across various domains, including shopping, forums, developer platforms, and content management systems. Each task is paired with a handwritten test case that verifies agent success, *e.g.*, by checking the status of a web page element against a reference or the textual answer returned by the agent. **Results.** From Tab. 5, we can see that our BrowsingAgent achieves competitive performance among agents that use LLMs with domain-general prompting techniques. Some agents (e.g., AutoWebGLM) require manual effort tailored to the WebArena task domain. This showcases the performance trade-off between a generalist vs. a domain-tailored specialist web agent, and we opt for a more general browsing agent as a building block in OpenHands.

MiniWoB++ (Liu et al., 2018) is an interactive web benchmark, with built-in reward functions. The tasks are synthetically initialized on 125 different minimalist web interfaces. Unlike WebArena, tasks are easier without page changes, require fewer steps, and provide low-level step-by-step task directions. Note that it contains a portion of environments that require vision capability to tackle successfully, and many existing work choose to focus only on a subset of the tasks (Kim et al., 2024; Li et al., 2023d; Shaw et al., 2023). Still, we report the performance on the full set and only include baselines that are evaluated on the full set. **Results.** From Tab. 5, we see that our BrowsingAgent finishes nearly half of the tasks without any adaptation to the environment. However, due to the synthetic nature of MiniWoB++, the state-of-the-art agents explicitly trained for the environments with reinforcement learning and/or human behavior cloning have almost saturated the performance.

Table 5: OpenHands Web Browsing Evaluation Results (§4.3).

Agent	Model	Success Rate (%)	$ Avg. Cost
WebArena (Zhou et al., 2023a), 812 instances			
Lemur (Xu et al., 2023)	`Lemur-chat-70b`	5.3	–
Patel et al. (2024)	Trained 72B with self-improvement synthetic data	9.4	–
AutoWebGLM (Lai et al., 2024)	Trained 7B with human/agent hybrid annotation	18.2	–
Auto Eval & Refine (Pan et al., 2024)	GPT-4 + Reflexion w/ GPT-4V reward model	20.2	–
WebArena Agent (Zhou et al., 2023a)	`Llama3-chat-8b`	3.3	–
	`Llama3-chat-70b`	7.0	–
	`gpt-3.5-turbo`	6.2	–
	`gpt-4-turbo`	14.4	–
OH BrowsingAgent v1.0	`gpt-3.5-turbo-0125`	5.2	0.02
	`gpt-4o-mini-2024-07-18`	8.5	0.01
	`gpt-4o-2024-05-13`	14.8	0.15
	`claude-3-5-sonnet-20240620`	15.5	0.10
OH CodeActAgent v1.8 via **delegation** to BrowsingAgent v1.0	`gpt-4o-mini-2024-07-18`	8.3	–
	`gpt-4o-2024-05-13`	14.5	–
	`claude-3-5-sonnet-20240620`	15.3	–
MiniWoB++ (Liu et al., 2018), 125 environments			
Workflow Guided Exploration (Liu et al., 2018)	Trained specialist model with environment exploration	34.6	–
CC-NET (Humphreys et al., 2022)	Trained specialist model with RL and human annotated BC	91.1	–
OH BrowsingAgent v1.0	`gpt-3.5-turbo-0125`	27.2	0.01
	`gpt-4o-2024-05-13`	40.8	0.05
OH CodeActAgent v1.8 via **delegation** to BrowsingAgent v1.0	`gpt-4o-2024-05-13`	39.8	–

4.4 Miscellaneous Assistance

Results for miscellaneous assistance benchmarks are reported in Tab. 6.

GAIA (Mialon et al., 2023) evaluates agents' general task-solving skills, covering different real-world scenarios. It requires various agent capabilities, including reasoning, multi-modal understanding, web browsing, and coding. GAIA consists of 466 curated tasks across three levels. Setting up GAIA is traditionally challenging due to the complexity of integrating various tools with the agent, but OpenHands's infrastructure (*e.g.*, runtime §2.2, tools §2.3) simplifies the integration significantly. **Results.** In our experiments, we achieved a score of 32.1 on the GAIA (level-1 val), significantly improving over the original AutoGPT (Gravitas, 2023). GAIA is sensitive to the support of multimodal input and web navigation skills, suggesting further score improvements as OpenHands's infrastructure improves.

GPQA (Rein et al., 2023) evaluates agents' ability for coordinated tool use when solving challenging graduate-level problems. It consists of 448 curated and difficult multiple-choice questions in biology, physics, and chemistry. Tool use (*e.g.*, python) and web search are often useful to assist agents in answering these questions since they provide accurate calculations that LLMs are often incapable of and access to information outside of the LLM's parametric knowledge base. **Results.** Results are shown in Tab. 6 and 7. We observe that OpenHands's integrated for supporting diverse tool use (*e.g.*, python for calculations) as well as web-search (for searching relevant facts) allows the resulting agent to better solve complex multi-step problems, surpassing the prior *state-of-the-art* by 9.6% and 12.3% on the main and diamond subsets respectively on GPQA (Rein et al., 2023).

AgentBench (Liu et al., 2023) evaluates agents' reasoning and decision-making abilities in a multi-turn, open-ended generation setting. We selected the code-grounded operating system (OS) subset with 144 tasks. Agents from OpenHands interact directly with the task-specific OS using bash commands in a multi-turn manner, combining interaction and reasoning to automate task completion. **Results.** In our experiments (Tab. 6), OpenHands CodeActAgent v1.5 achieves a score of 57.6% on the AgentBench using `gpt-4o`, outperforming the 42.4% baseline using `gpt-4` from the original paper. Interestingly, when employing weaker models such as `gpt-3.5-turbo`, OpenHands agents generally underperform compared to the original baseline agents. This finding suggests that generalist agents, like those implemented in OpenHands, require a certain threshold of foundation model capability - particularly instruction following - to function effectively.

MINT (Wang et al., 2024b) is a benchmark designed to evaluate agents' ability to solve challenging tasks through *multi-turn interactions* using *tools* and *natural language feedback* simulated by GPT-4. We use coding and math subsets used in Yuan et al. (2024). We follow the original paper and allow the agent to interact with up to five iterations with two chances to propose solutions. **Results.** As

Table 6: OpenHands miscellaneous assistance evaluation results (§4.4).

Agent	Model	Success Rate (%)	$ Avg. Cost
GAIA (Mialon et al., 2023), L1 validation set, 53 instances			
AutoGPT (Gravitas, 2023)	`gpt-4-turbo`	13.2	–
OH GPTSwarm v1.0	`gpt-4-0125-preview`	30.2	0.110
	`gpt-4o-2024-05-13`	32.1	0.050
GPQA (Rein et al., 2023), diamond set, 198 instances (refer to §G, Tab. 7 for other subsets)			
Human (Rein et al., 2023)	Expert human	81.3	–
	Non-expert human	21.9	–
Few-shot Prompting + Chain-of-Thought (Rein et al., 2023)	`gpt-3.5-turbo-16k`	29.6	–
	`gpt-4`	38.8	–
OH CodeActAgent v1.8	`claude-3-5-sonnet-20240620`	52.0	0.065
AgentBench (Liu et al., 2023), OS (bash) subset, 144 instances			
AgentBench Baseline Agent (Liu et al., 2023)	`gpt-4`	42.4	–
	`gpt-3.5-turbo`	32.6	–
OH CodeActAgent v1.5	`gpt-4o-2024-05-13`	57.6	0.085
	`gpt-3.5-turbo-0125`	11.8	0.006
MINT (Wang et al., 2024b): `math` subset, 225 instances			
MINT Baseline Agent	`gpt-4-0613`	65.8	–
OH CodeActAgent v1.5	`gpt-4o-2024-05-13`	77.3	0.070
	`gpt-3.5-turbo-16k-0613`	33.8	0.048
MINT (Wang et al., 2024b): `code` subset, 136 instances			
MINT Baseline Agent	`gpt-4-0613`	59.6	–
OH CodeActAgent v1.5	`gpt-4o-2024-05-13`	50.0	0.087
	`gpt-3.5-turbo-16k-0613`	5.2	0.030
ProofWriter (Tafjord et al., 2021), 600 instances			
Few-shot Prompting + Chain-of-Thought (Pan et al., 2023)	`gpt4`	68.1	–
Logic-LM (Pan et al., 2023)	`gpt4 + symbolic solver`	79.6	–
OH CodeActAgent v1.5	`gpt-4o-2024-05-13`	78.8	–
Entity Deduction Arena (Zhang et al., 2024a), 200 instances			
Human	-	21.0	–
Zero-shot Prompting (Zhang et al., 2024a)	`gpt-4-0314`	40.0	–
	`gpt-3.5-turbo-0613`	27.0	–
OH CodeActAgent v1.5	`gpt-4o-2024-05-13`	38.0	–
	`gpt-3.5-turbo-16k-0613`	24.0	–

shown in Tab. 6), OpenHands agents achieve comparable performance to the default agent in the original benchmark, with a performance improvement in the `math` subset.

ProofWriter (Tafjord et al., 2021) is a synthetic dataset created to assess deductive reasoning abilities of LLMs. Same as Logic-LM (Pan et al., 2023), we focus on the most challenging subset, which contains 600 instances requiring 5-hop reasoning. To minimize the impact of potential errors in semantic parsing, we use the logical forms provided by Logic-LM. **Results.** In Tab. 6, OpenHands agent employs a symbolic solver to solve the task, achieving performance comparable to the *state-of-the-art* neuro-symbolic model (*i.e.*, Logic-LM) (Pan et al., 2023).

Entity Deduction Arena (EDA) (Zhang et al., 2024a) evaluates agents' ability to deduce unknown entities through strategic questioning, akin to the 20 Questions game. This benchmark tests the agent's state tracking, strategic planning, and inductive reasoning capabilities over multi-turn conversations. We evaluate two datasets "Things" and "Celebrities", each comprising 100 instances, and report the average success rate over these two datasets. **Results.** Tab. 6 shows that CodeActAgent yields comparable performance comparing with the results reported in the original paper (Zhang et al., 2024a).

5 Conclusion

We introduce OpenHands, a community-driven platform that enables the development of agents that interact with the world through software interfaces. By providing a powerful interaction mechanism, a safe sandboxed environment, essential agent skills, multi-agent collaboration capabilities, and a comprehensive evaluation framework, OpenHands accelerates research innovations and real-world applications of agentic AI systems. Despite challenges in developing safe and reliable agents (§A), we are excited about our vibrant community and look forward to OpenHands's continued evolution.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. 1

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023. 9

Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023. 1

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, 2024. 1

Harrison Chase. LangChain, October 2022. URL https://github.com/langchain-ai/langchain. 4, 6, 19

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje F. Karlsson, Jie Fu, and Yemin Shi. Autoagents: A framework for automatic agent generation, 2024. 1, 6, 19

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. 7

Cognition.ai. Introducing devin, the first ai software engineer. URL https://www.cognition.ai/blog/introducing-devin. 2

Jiaxi Cui, Zongjian Li, Yang Yan, Bohua Chen, and Li Yuan. Chatlaw: Open-source legal large language model with integrated external knowledge bases, 2023. 19

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. Workarena: How capable are web agents at solving common knowledge work tasks?, 2024. 3, 4

Paul Gauthier. How aider scored sota 26.3% on swe bench lite | aider. https://aider.chat/2024/05/22/swe-bench-lite.html. Accessed: 2024-06-05. 5, 7, 8

Significant Gravitas. Auto-gpt: An autonomous gpt-4 experiment, 2023. *URL https://github.com/Significant-Gravitas/Auto-GPT*, 2023. 6, 7, 10, 11, 19

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2023. 1, 6, 19

Dong Huang, Qingwen Bu, Jie M. Zhang, Michael Luck, and Heming Cui. Agentcoder: Multi-agent-based code generation with iterative testing and optimisation, 2024. 19

Peter C Humphreys, David Raposo, Tobias Pohlen, Gregory Thornton, Rachita Chhaparia, Alistair Muldal, Josh Abramson, Petko Georgiev, Adam Santoro, and Timothy Lillicrap. A data-driven approach for learning to control computers. In *International Conference on Machine Learning*, pp. 9466–9482. PMLR, 2022. 10

IPython. Jupyter and the future of IPython — IPython. URL https://ipython.org. 4

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024. 1

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can Language Models Resolve Real-world Github Issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66. 1, 6, 7, 8, 18, 20

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. Language models can solve computer tasks. *Advances in Neural Information Processing Systems*, 36, 2024. 9

Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, et al. Autowebglm: Bootstrap and reinforce a large language model-based web navigating agent. *arXiv preprint arXiv:2404.03648*, 2024. 7, 10

Hareton K. N. Leung and Lee J. White. A study of integration testing and software regression at the integration level. In *Proceedings of the Conference on Software Maintenance, ICSM 1990, San Diego, CA, USA, 26-29 November, 1990*, pp. 290–301. IEEE, 1990. doi: 10.1109/ICSM.1990.131377. URL https://doi.org/10.1109/ICSM.1990.131377. 20

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large scale language model society. *arXiv preprint arXiv:2303.17760*, 2023a. 19

Jinyang Li, Binyuan Hui, GE QU, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can LLM already serve as a database interface? a BIg bench for large-scale database grounded text-to-SQLs. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023b. 6, 8, 9, 18

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. Starcoder: may the source be with you!, 2023c. URL https://arxiv.org/abs/2305.06161. 8

Tao Li, Gang Li, Zhiwei Deng, Bryan Wang, and Yang Li. A zero-shot language agent for computer control with structured reflection. *arXiv preprint arXiv:2310.08740*, 2023d. 9

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://arxiv.org/abs/1802.08802. 6, 9, 10, 18

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv: 2308.03688*, 2023. 6, 10, 11, 18

Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, Tianyang Liu, Max Tian, Denis Kocetkov, Arthur Zucker, Younes Belkada, Zijian Wang, Qian Liu, Dmitry Abulkhanov, Indraneil Paul, Zhuang Li, Wen-Ding Li, Megan Risdal, Jia Li, Jian Zhu, Terry Yue Zhuo, Evgenii Zheltonozhskii, Nii Osae Osae Dade, Wenhao Yu, Lucas Krauß, Naman Jain, Yixuan Su, Xuanli He, Manan

Dey, Edoardo Abati, Yekun Chai, Niklas Muennighoff, Xiangru Tang, Muhtasham Oblokulov, Christopher Akiki, Marc Marone, Chenghao Mou, Mayank Mishra, Alex Gu, Binyuan Hui, Tri Dao, Armel Zebaze, Olivier Dehaene, Nicolas Patry, Canwen Xu, Julian McAuley, Han Hu, Torsten Scholak, Sebastien Paquet, Jennifer Robinson, Carolyn Jane Anderson, Nicolas Chapados, Mostofa Patwary, Nima Tajbakhsh, Yacine Jernite, Carlos Muñoz Ferrandis, Lingming Zhang, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. Starcoder 2 and the stack v2: The next generation, 2024. URL https://arxiv.org/abs/2402.19173. 8

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. *CoRR*, abs/2311.12983, 2023. doi: 10.48550/ARXIV.2311.12983. URL https://doi.org/10.48550/arXiv.2311.12983. 6, 10, 11, 18

Mozilla. Accessibility tree - MDN Web Docs Glossary: Definitions of Web-related terms | MDN. URL https://developer.mozilla.org/en-US/docs/Glossary/Accessibility_tree. 5

Niklas Muennighoff, Qian Liu, Armel Zebaze, Qinkai Zheng, Binyuan Hui, Terry Yue Zhuo, Swayam Singh, Xiangru Tang, Leandro von Werra, and Shayne Longpre. Octopack: Instruction tuning code large language models, 2024. 6, 7, 8, 18

Y Nakajima. Babyagi. *URL https://github.com/yoheinakajima/babyagi*, 2023. 19

OpenAI. Chatgpt: May 2024 version. https://www.openai.com/chatgpt, 2024a. Accessed: 2024-05-29. 8, 19

OpenAI. Hello gpt-4o. https://openai.com/index/hello-gpt-4o/, 2024b. Accessed: 2024-05-15. 1

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel

Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. 19

Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. Autonomous evaluation and refinement of digital agents. *arXiv preprint arXiv:2404.06474*, 2024. 7, 10, 19

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *arXiv preprint arXiv:2305.12295*, 2023. 11

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior, 2023. 19

Ajay Patel, Markus Hofmarcher, Claudiu Leoveanu-Condrei, Marius-Constantin Dinu, Chris Callison-Burch, and Sepp Hochreiter. Large language models can self-improve at web agent tasks. *arXiv preprint arXiv:2405.20309*, 2024. 7, 10

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023. 6, 8, 9, 18

Playwright. Fast and reliable end-to-end testing for modern web apps | Playwright. URL https://playwright.dev/. 4

Chen Qian, Xin Cong, Wei Liu, Cheng Yang, Weize Chen, Yusheng Su, Yufan Dang, Jiahao Li, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. Communicative agents for software development, 2023. 19

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. *CoRR*, abs/2307.16789, 2023. doi: 10.48550/ARXIV.2307.16789. URL https://doi.org/10.48550/arXiv.2307.16789. 4

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. GPQA: A Graduate-Level Google-Proof Q&A Benchmark. *arXiv preprint arXiv:2311.12022*, 2023. 6, 7, 10, 11, 18, 24

Aman Sanger. Near-instant full-file edits. https://www.cursor.com/blog/instant-apply. Accessed: 2024-06-05. 5

ServiceNow. BrowserGym: a Gym Environment for Web Task Automation. URL https://github.com/ServiceNow/BrowserGym. 2, 4

Peter Shaw, Mandar Joshi, James Cohan, Jonathan Berant, Panupong Pasupat, Hexiang Hu, Urvashi Khandelwal, Kenton Lee, and Kristina N Toutanova. From pixels to ui actions: Learning to follow instructions via graphical user interfaces. *Advances in Neural Information Processing Systems*, 36: 34354–34370, 2023. 9

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024. 19

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3621–3634, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.317. URL https://aclanthology.org/2021.findings-acl.317. 6, 11, 18

Xiangru Tang, Qiao Jin, Kunlun Zhu, Tongxin Yuan, Yichi Zhang, Wangchunshu Zhou, Meng Qu, Yilun Zhao, Jian Tang, Zhuosheng Zhang, et al. Prioritizing safeguarding over autonomy: Risks of llm agents for science. *arXiv preprint arXiv:2402.04247*, 2024a. 1

Xiangru Tang, Yuliang Liu, Zefan Cai, Yanjun Shao, Junjie Lu, Yichi Zhang, Zexuan Deng, Helan Hu, Kaikai An, Ruijun Huang, Shuzheng Si, Sheng Chen, Haozhe Zhao, Liang Chen, Yan Wang, Tianyu Liu, Zhiwei Jiang, Baobao Chang, Yin Fang, Yujia Qin, Wangchunshu Zhou, Yilun Zhao, Arman Cohan, and Mark Gerstein. ML-Bench: Evaluating large language models and agents for machine learning tasks on repository-level code, 2024b. URL https://arxiv.org/abs/2311.09835. 6, 8, 18

Xiangru Tang, Bill Qian, Rick Gao, Jiakang Chen, Xinyun Chen, and Mark B Gerstein. BioCoder: a benchmark for bioinformatics code generation with large language models. *Bioinformatics*, 40 (Supplement_1):i266–i276, 06 2024c. ISSN 1367-4811. 6, 9, 18

Xiangru Tang, Anni Zou, Zhuosheng Zhang, Ziming Li, Yilun Zhao, Xingyao Zhang, Arman Cohan, and Mark Gerstein. Medagents: Large language models as collaborators for zero-shot medical reasoning, 2024d. 19

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. 1

XAgent Team. Xagent: An autonomous agent for complex task solving, 2023. 6, 19

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 8

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable Code Actions Elicit Better LLM Agents. In *ICML*, 2024a. 2, 3, 4, 5, 18

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. MINT: Evaluating LLMs in Multi-turn Interaction with Tools and Language Feedback. In *ICLR*, 2024b. 6, 10, 11, 18

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*, 2023. 1, 6, 19

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying llm-based software engineering agents. *arXiv preprint*, 2024. 7

Tianbao Xie, Fan Zhou, Zhoujun Cheng, Peng Shi, Luoxuan Weng, Yitao Liu, Toh Jing Hua, Junning Zhao, Qian Liu, Che Liu, et al. Openagents: An open platform for language agents in the wild. *arXiv preprint arXiv:2310.10634*, 2023. 6

Yiheng Xu, Hongjin Su, Chen Xing, Boyu Mi, Qian Liu, Weijia Shi, Binyuan Hui, Fan Zhou, Yitao Liu, Tianbao Xie, et al. Lemur: Harmonizing natural language and code for language agents. *arXiv preprint arXiv:2310.06830*, 2023. 7, 10

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering, 2024. 2, 5, 6, 7, 8, 19

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X. 8

Yining Ye, Xin Cong, Shizuo Tian, Jiannan Cao, Hao Wang, Yujia Qin, Yaxi Lu, Heyang Yu, Huadong Wang, Yankai Lin, et al. Proagent: From robotic process automation to agentic process automation. *arXiv preprint arXiv:2311.10751*, 2023. 19

Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi R. Fung, Hao Peng, and Heng Ji. CRAFT: customizing llms by creating and retrieving from specialized toolsets. *CoRR*, abs/2309.17428, 2023. doi: 10.48550/ARXIV.2309.17428. URL https://doi.org/10.48550/arXiv.2309.17428. 4

Lifan Yuan, Ganqu Cui, Hanbin Wang, Ning Ding, Xingyao Wang, Jia Deng, Boji Shan, Huimin Chen, Ruobing Xie, Yankai Lin, Zhenghao Liu, Bowen Zhou, Hao Peng, Zhiyuan Liu, and Maosong Sun. Advancing llm reasoning generalists with preference trees, 2024. 10

Yizhe Zhang, Jiarui Lu, and Navdeep Jaitly. Probing the multi-turn planning capabilities of llms via 20 question games. 2024a. 6, 11, 18

Yuntong Zhang, Haifeng Ruan, Zhiyu Fan, and Abhik Roychoudhury. Autocoderover: Autonomous program improvement, 2024b. 6, 7, 8, 19

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2023a. 1, 6, 7, 9, 10, 18

Wangchunshu Zhou, Yuchen Eleanor Jiang, Long Li, Jialong Wu, Tiannan Wang, Shi Qiu, Jintian Zhang, Jing Chen, Ruipu Wu, Shuai Wang, Shiding Zhu, Jiyu Chen, Wentao Zhang, Xiangru Tang, Ningyu Zhang, Huajun Chen, Peng Cui, and Mrinmaya Sachan. Agents: An open-source framework for autonomous language agents, 2023b. 6, 19

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. Toolqa: A dataset for llm question answering with external tools. *Advances in Neural Information Processing Systems*, 36, 2024. 6, 8, 9, 18

Mingchen Zhuge, Haozhe Liu, Francesco Faccio, Dylan R Ashley, Róbert Csordás, Anand Gopalakrishnan, Abdullah Hamdi, Hasan Abed Al Kader Hammoud, Vincent Herrmann, Kazuki Irie, et al. Mindstorms in natural language-based societies of mind. *arXiv preprint arXiv:2305.17066*, 2023. 19

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jurgen Schmidhuber. Language agents as optimizable graphs. *arXiv preprint arXiv:2402.16823*, 2024. 6, 18, 19

Albert Örwall. Moatless tools. URL https://github.com/aorwall/moatless-tools. 7