Process-level Trajectory Evaluation for Environment Configuration in Software Engineering Agents

Jiayi Kuang, Yinghui Li, Xin Zhang, Yangning Li, Di Yin, Xing Sun, Ying Shen, Philip S. Yu
[1]Sun Yat-sen University, [2]Tencent Youtu Lab, [3]The Hong Kong Polytechnic University, [4]Peng Cheng Laboratory, [5]University of Illinois Chicago, [6]Guangdong Provincial Key Laboratory of Fire Science and Intelligent Emergency Technology

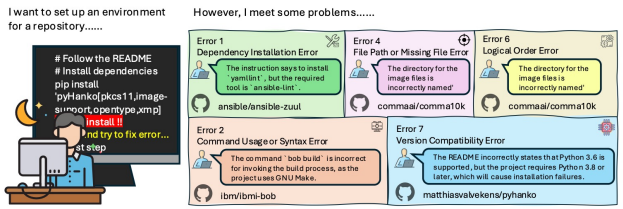
ICLR

Process in Environment Configuration

Configuring a runnable **execution environment** is the most fundamental and critical first step for software engineering, yet it remains challenging for both human engineers and current LLMs, requiring substantial manual effort. This burden constrains large-scale, high-quality dataset production, making rigorous evaluation of agents' environment configuration capabilities essential for progress in SWE.

We focus on ***process-level evaluation*** along the agent's configuration trajectory. Specifically, we investigate:

- how agents apply ***planning*** to devise reasonable configuration steps and strategies given the task requirements;
- how they use ***perception*** to accurately localize the causes of errors when failures occur (e.g., version incompatibilities or missing dependencies);
- how they utilize ***feedback*** to analyze the errors and try to fix them;
- how they translate precise feedbacks into ***actions*** that correct these errors, complete environment configuration, and ensure that subsequent code runs and passes evaluation.



The illustration of common problems in environment configuration.

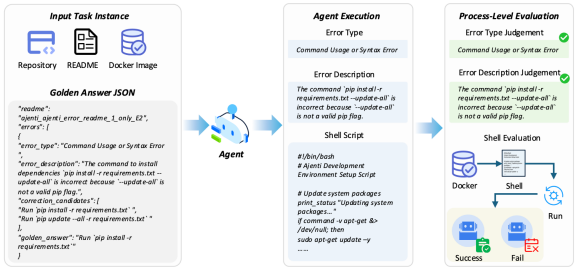
Key Insights in Enconda-bench

Inspired by how human engineers configure environments—typically following **README** steps first, then analyzing the causes of failures and attempting fixes—we consider editing an originally correct README by ***injecting erroneous commands or confusing steps***. As the model configures the environment based on such a README, it must locate and repair these errors.

- We propose a ***trajectory-based EnConda-Bench*** for process-level evaluation of environment configuration in SWE, enabling detailed assessment of the capabilities agents exhibit during environment configuration.
- We introduce automated data construction pipeline, which reduces manual labor and ***supplies large-scale training data*** for agents and LLMs.
- Our evaluation across multiple LLMs/agents finds basic error localization/classification abilities but ***limited environmental interaction and feedback utilization***, often yielding ineffective repairs, providing valuable findings and inspiration for future research.

Tasks and Benchmark Construction

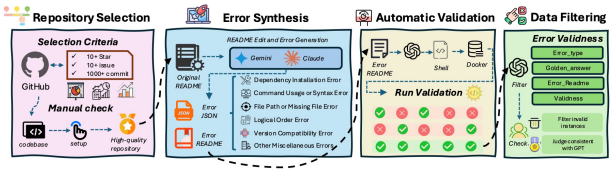
overall workflow of process-level environment configuration.



This figure illustrates an example of overall workflow

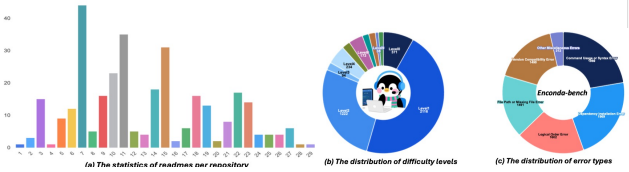
Data construction pipeline

- Select ***high-quality repositories*** via strict criteria;
- Employ advanced LLMs to ***edit*** key environment READMEs with common ***error types*** and annotate categories and suggested fixes;
- ***Validate and filter*** for effective errors via an automated framework to obtain high-quality task instances.



The illustration of our overall pipeline of benchmark construction.

Data statistics



Benchmark	Instances	Metric	Process-level
INSTALLAMATIC(Milliken et al., 2025)	40	Success build	✗
EXECUTIONAGENT(Bouzenia & Pradel, 2025)	50	Success build & test	✗
EnvBench(Eliseeva et al.)	994	Success build & test, missing imports	✗
SetupBench(Vergopoulos et al., 2025)	93	Success build & test,	✗
EnConda-Bench	4,201	Success build & test, error detection and fix	✓

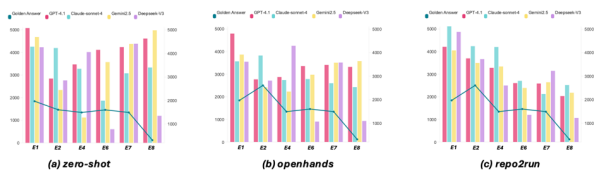
Experiment Results and Insights

Main evaluation results

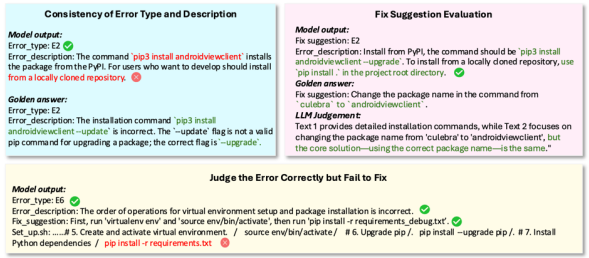
Table 4: Evaluation results across agents and LLMs. **Bold**: Optimal performance for every setting.

Framework	LLM	Perception: Error type Pre.	Perception: Error type Rec.	Perception: Error type F1	Feedback: Error description ACC.	Feedback and Action: Fix suggestion ACC.	Planning and Action: Execution Pass@1
Base Model							
Zero-Shot	GPT-4.1	33.4	**90.6**	48.8	39.6	18.2	1.5
	Claude-4	**37.1**	80.6	**50.8**	45.1	**28.5**	3.1
	Gemini2.5-pro	35.2	77.8	48.5	**45.2**	25.2	1.8
	DeepSeek-V3	33.2	65.8	44.2	39.7	22.3	**3.3**
	GPT-5-Codex	40.5	88.2	55.5	49.8	32.4	6.3
Code Agent							
SWE-Agent	GPT-4.1	43.7	83.2	55.3	49.8	30.7	7.2
	Claude-4	46.4	85.6	58.2	52.4	34.5	9.4
	Gemini2.5-pro	45.1	92.5	56.8	50.2	32.2	7.8
	DeepSeek-V3	41.2	70.3	51.9	44.5	27.8	7.4
	GPT-5-Codex	52.1	91.4	66.4	58.7	41.2	11.5
OpenHands	GPT-4.1	42.5	72	53.2	46.0	29.1	8.5
	Claude-4	**48.0**	87.5	**60.1**	**54.2**	**36.1**	**10.6**
	Gemini2.5-pro	45.2	85.2	57.5	51.8	32.3	8.5
	DeepSeek-V3	46.7	**93.6**	58.7	51.9	33.8	9.1
Environment Configuration Agent							
INSTALLAMATIC	GPT-4.1	37.5	70.4	48.9	45.3	29.1	5.6
	Claude-4	41.9	75.3	53.8	50.7	34.1	7.9
	Gemini2.5-pro	39.0	72.1	50.6	47.5	30.8	6.4
	DeepSeek-V3	40.7	76.8	53.2	49.3	32.5	7.1
Repo2Run	GPT-4.1	44.2	72.3	54.8	48.5	38.6	14.1
	Claude-4	**49.5**	77.5	**60.6**	52.2	**47.3**	**22.9**
	Gemini2.5-pro	48.6	**79.3**	60.1	**54.2**	45.6	17.8
	DeepSeek-V3	46.3	74.2	56.8	44.6	41.2	16.2
Gemini-CLI	Gemini-2.5-Pro	48.5	94.2	64.0	54.6	38.9	13.2

Errors of the golden label and the model's prediction



Case Study



We select some typical cases and provide an analysis of observed phenomena.