

Agents for Issue Solving

Daniel Fried

11-891: Neural Code Generation

<https://cmu-codegen.github.io/f2025/>



Language
Technologies
Institute

What is an Agent?

What is An Agent?

- ▶ Russell and Norvig: “an intelligent agent **perceives** its environment via sensors and **acts rationally** upon that environment with its **effectors**”
- ▶ LLM agents:
 - ▶ “perceive” via text (& sometimes visual) observations of the state
 - ▶ “effectors”: tools (APIs) that take actions in the world
 - ▶ “act rationally”: ???

LLMs as Agents

► Warmup: text worlds with ReAct (reasoning + acting) prompting

Goal:

You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 1, a coffee machine 1, a countertop 3, a stove burner 1, and a toaster 1.

Your task is to: Put some pepper shaker on a drawer.

(2b) ReAct (Reason + Act)

Act 1: **Think**[First I need to find a pepper shaker...more likely to appear in cabinets (1-6), countertops (1-3), ...]

Act 2: Go to cabinet 1

Obs 2: On the cabinet 1, you see a vase 2.

(Here the agent go to cabinet 1, then cabinet 2, 3, then countertop 1 and 2)

Act 6: Go to countertop 3

Obs 6: On the countertop 3, you see a apple 1, a bread 1, a pepper shaker 1, and a vase 3.

Act 7: Take pepper shaker 1 from countertop 3

Obs 7: You pick up the pepper shaker 1 from the countertop 3.

Act 8: **Think**[Now I find a pepper shaker 1. Next, I need to put it in/on drawer 1.]

Act 9: Go to drawer 1

Obs 9: Drawer 1 is closed.

Act 10: Open drawer 1

Obs 10: You open Drawer 1 ...

Act 11: Put pepper shaker 1 in/on drawer 1

Obs 11: You put pepper shaker 1 in/on the drawer 1.



LLMs as Agents

Goal:

hackermd opened on Aug 4, 2021

Contributor ...

Describe the bug

The NumPy pixel data handler currently [requires the Pixel Representation attribute](#). This is problematic, because in case of Float Pixel Data or Double Float Pixel Data the attribute shall be absent. Compare [Floating Point Image Pixel Module Attributes](#) versus [Image Pixel Description Macro Attributes](#)

LLM

The script has successfully reproduced the bug, as it raised an `AttributeError` ...

The next step is to locate the `numpy_handler.py` file where the error is raised and modify the code to handle cases where the `PixelRepresentation` attribute is not required...






Observations

O_t :

```
{"open_file": "/pydicom__pydicom/reproduce_bug.py", "working_dir":  
"/pydicom__pydicom"}  
...  
Traceback (most recent call last):  
File "/pydicom__pydicom/pydicom/pixel_data_handlers/numpy_handler.py",  
line 293, in get_pixeldata  
    raise AttributeError(  
AttributeError: Unable to convert the pixel data as the following required  
elements are missing from the dataset: PixelRepresentation
```

Action a_t : `find_file(
 "numpy_handler.py")`

Tools Are Functions (APIs)

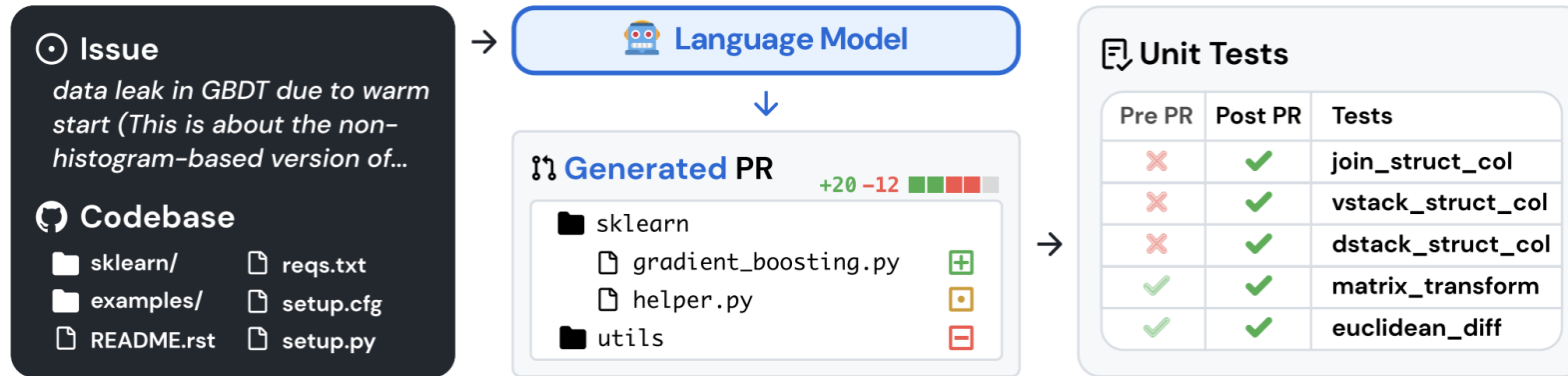
Category	Example Tools
 Knowledge access	<code>sql_executor(query: str) -> answer: any</code> <code>search_engine(query: str) -> document: str</code> <code>retriever(query: str) -> document: str</code>
 Computation activities	<code>calculator(formula: str) -> value: int float</code> <code>python_interpreter(program: str) -> result: any</code> <code>worksheet.insert_row(row: list, index: int) -> None</code>
 Interaction w/ the world	<code>get_weather(city_name: str) -> weather: str</code> <code>get_location(ip: str) -> location: str</code> <code>calendar.fetch_events(date: str) -> events: list</code> <code>email.verify(address: str) -> result: bool</code>
 Non-textual modalities	<code>cat_image.delete(image_id: str) -> None</code> <code>spotify.play_music(name: str) -> None</code> <code>visual_qa(query: str, image: Image) -> answer: str</code>
 Special-skilled LMs	<code>QA(question: str) -> answer: str</code> <code>translation(text: str, language: str) -> text: str</code>

What's Important in an LLM Agent?

- ▶ Understanding the observations and state
- ▶ Knowledge about the domain => implicit planning ability
- ▶ Using history of past actions
- ▶ General ability to call tools

SWE-Bench: An Issue-Solving Benchmark

SWE-Bench



- ▶ GitHub issues and pull-requests that fix them
- ▶ Model has to generate a patch (multi-file diff) to fix the issue
- ▶ Two types of tests to check correctness:
 - ▶ Fail-to-pass tests (reproduction)
 - ▶ Pass-to-pass tests (regression)
- ▶ Issue, code-base, and execution environment are all available

SWE-Bench: Data Collection

1  **Scrape PRs**
 12 popular repositories
 >90% Python Code

2  **Attribute Filter**
✓ Resolves an issue
✓ Contributes tests

3 **Execution Filter**
✓ Installs successfully
✓ PR passes all tests

- ▶ 12 popular repositories (django, flask, matplotlib, scikit-learn, ...), with 90K PRs
- ▶ Attribute filter: Select merged PRs that (1) resolve an issue and (2) contributed tests
- ▶ Execution filter: code runs and at least one test is pass-to-fail
- ▶ 2.3K PRs after filtering

SWE-Bench: Instances

Metadata

Repo	sympy / sympy	Issue #s	[17006]
Instance ID	sympy__sympy-17022	Pull Number	17022
Created At	Jun 9, 2019	Base Commit	b6fbc76

Problem Statement

Using lambdify on an expression containing an identity matrix gives us an unexpected result:

```
>>> import numpy as np
>>> n = symbols('n', integer=True)
>>> A = MatrixSymbol("A", n, n)
>>> a = np.array([[1, 2], [3, 4]])
>>> f = lambdify(A, A + Identity(n))
>>> f(a)
array([[1.+1.j, 2.+1.j],
       [3.+1.j, 4.+1.j]])
```

Instead, the output should be `array([[2, 2], [3, 5]])`, since we're adding an identity matrix to the array. Inspecting the globals and source code of `f` shows us why we get the result:

```
>>> import inspect
>>> print(inspect.getsource(f))
def _lambdifygenerated(A):
    return (I + A)
>>> f.__globals__['I']
1j
```

The code printer prints `I`, which is currently being interpreted as a Python built-in complex number. Printer should support identity matrices ...

Test Patch

sympy/printing/tests/test_pycode.py [...]

```
9  from sympy.logic import And, Or
10 - from sympy.matrices import SparseMatrix, MatrixSymbol
11 + from sympy.matrices import SparseMatrix, MatrixSymbol, Identity
12  from sympy.printing.pycode import (
...
46  def test_NumPyPrinter():
47      p = NumPyPrinter()
48      assert p.doprint(sign(x)) == 'numpy.sign(x)'
49      A = MatrixSymbol("A", 2, 2)
50      assert p.doprint(A**(-1)) == "numpy.linalg.inv(A)"
51      assert p.doprint(A**5) == "numpy.linalg.matrix_power(A, 5)"
52 + assert p.doprint(Identity(3)) == "numpy.eye(3)"
```

Gold Patch

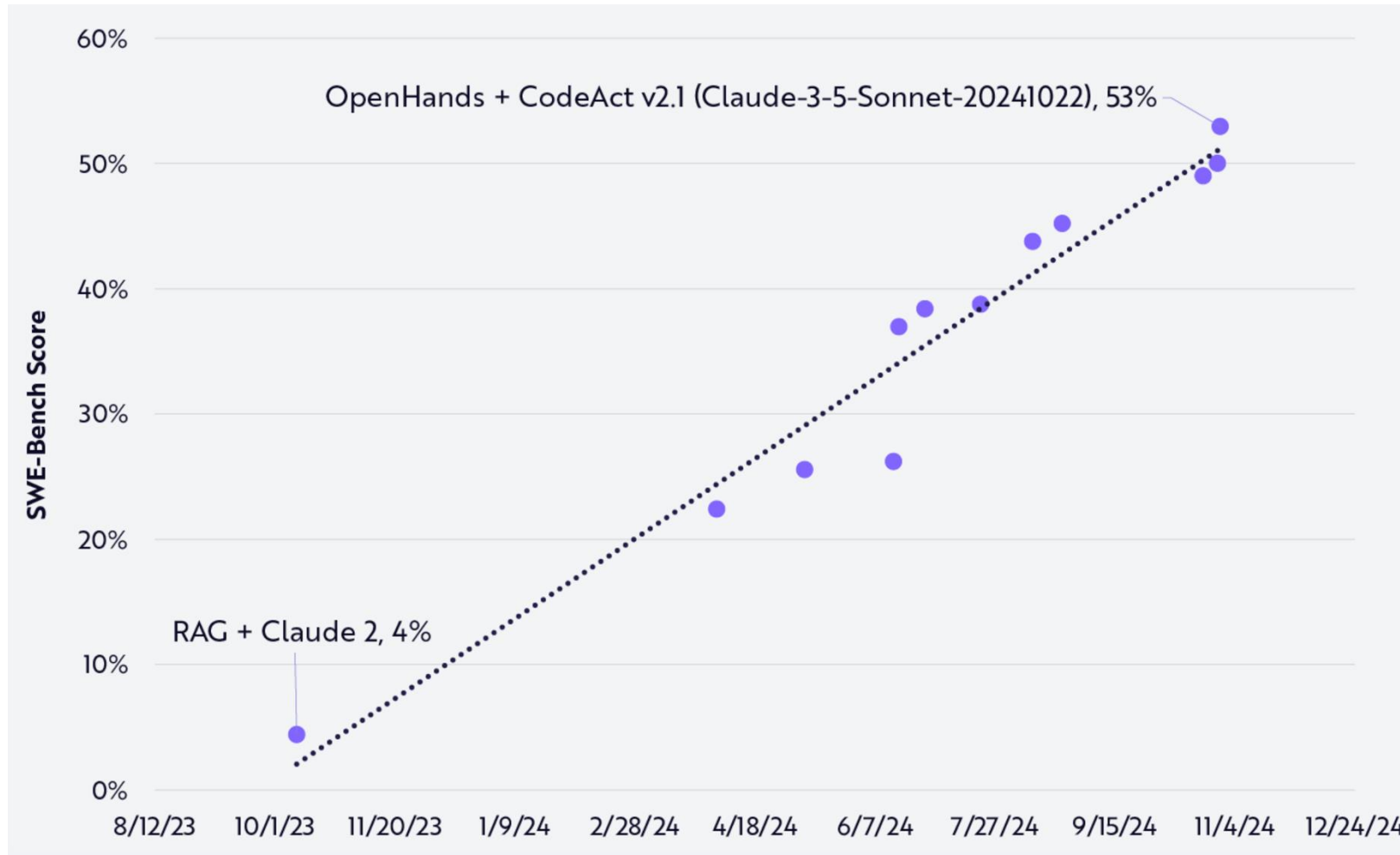
sympy/printing/pycode.py

```
609         return "%s(%s)" % (func, self._print(expr.tolist()))
610
611 +     def _print_Identity(self, expr):
612 +         shape = expr.shape
613 +         if all([dim.is_Integer for dim in shape]):
614 +             return "%s(%s)" % (self._module_format('numpy.eye'),
615 + self._print(expr.shape[0]))
616 +         else:
617 +             raise NotImplementedError("Symbolic matrix dimensions are not
618 + yet supported for identity matrices")
619
620 +     def _print_BlockMatrix(self, expr):
```

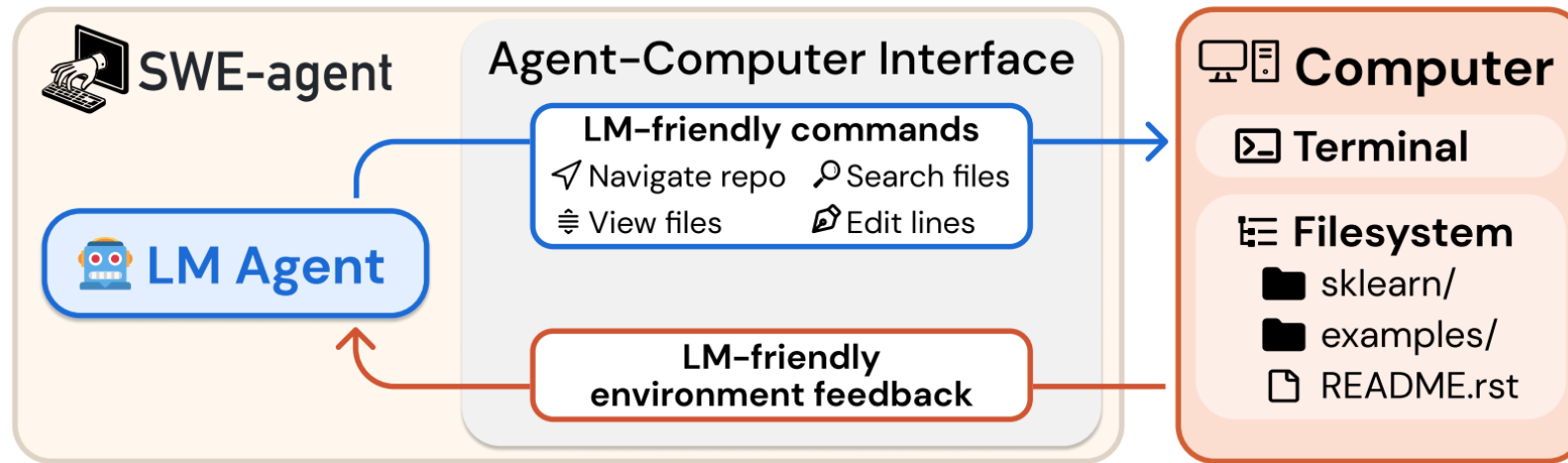
SWE-Bench Variants

- ▶ [SWE-Bench Lite](#): easier subset of 300 tasks with short, single-file edits
- ▶ [SWE-Bench Verified](#): annotate samples and remove (1) underspecified issue descriptions (2) fail-to-pass tests filter out valid solutions. 500 tasks after filtering
- ▶ SWE-Bench in other languages
 - ▶ Multimodal (JS): <https://www.swebench.com/multimodal.html>
 - ▶ Multilingual [[Khandpur et al.](#)], [[Rashid et al.](#)], [[Zan et al.](#)]

SWE-Bench: Progress



SWE-Agent



- ▶ Design a set of tools for the agent to use in a ReAct-style loop
 - ▶ Bash commands, file viewer, search tools, file editing, submit
- ▶ Observations of the state come through these tools
- ▶ Tailor the tools and their outputs to be easily usable by LLMs

Example Observations and Actions

Observation (File Viewer)

```
[File: solvers/diophantine.py (3093 lines total)]  
(400 lines above)  
401:     elif total_degree == 3:  
402:         diop_type = "cubic_thue"  
403:  
404:     elif (total_degree > 3):  
405:         if all(coeff[k] == 1):  
406:             diop_type = 'general_sum_even'  
407:  
408:     if diop_type is not None:  
409:         return var, coeff, diop_type  
(2684 lines below)
```

(a) Observation from the file viewer.

Response (Thought + Edit Action)

We need to add a check to ensure that all powers are the same. Let's edit the code to include this check.

```
edit 404:407  
    elif (total_degree > 3 and  
          len(set(k.exp[k.is_Pow])) == 1):  
        if all(coeff[k] == 1):  
            diop_type = 'general_sum_even'  
end_of_edit
```

(b) Action using the edit interface.

Minimal Code for the Agentive Loop

```
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)
```


Example Traces

Thought: The `reproduce_bug.py` script has been updated with the code provided in the issue. Now, let's run the script to see if we can reproduce the bug.

Action: `python reproduce_bug.py`

Observation:

Traceback (most recent call last):

```
File "/pydicom__pydicom/reproduce_bug.py", line 17, in <module>
    result = np.array_equal(ds.pixel_array, pixel_array)
```

...

```
File "/pydicom__pydicom/pydicom/pixel_data_handlers/numpy_handler.py", line 293, in get_pixeldata
    raise AttributeError(
```

AttributeError: Unable to convert the pixel data as the following required elements are missing from the dataset: PixelRepresentation

Example Traces

Thought: The script has successfully reproduced the bug, as it raised an ``AttributeError`` due to the missing ``PixelRepresentation`` attribute when attempting to access the ``pixel_array`` property. This confirms the issue described.

The next step is to locate the ``numpy_handler.py`` file where the error is raised and modify the code to handle cases where the ``PixelRepresentation`` attribute is not required, such as with Float Pixel Data or Double Float Pixel Data.

Let's use the ``find_file`` command to locate the ``numpy_handler.py`` file within the repository.

Action: ``find_file "numpy_handler.py"``

Observation:

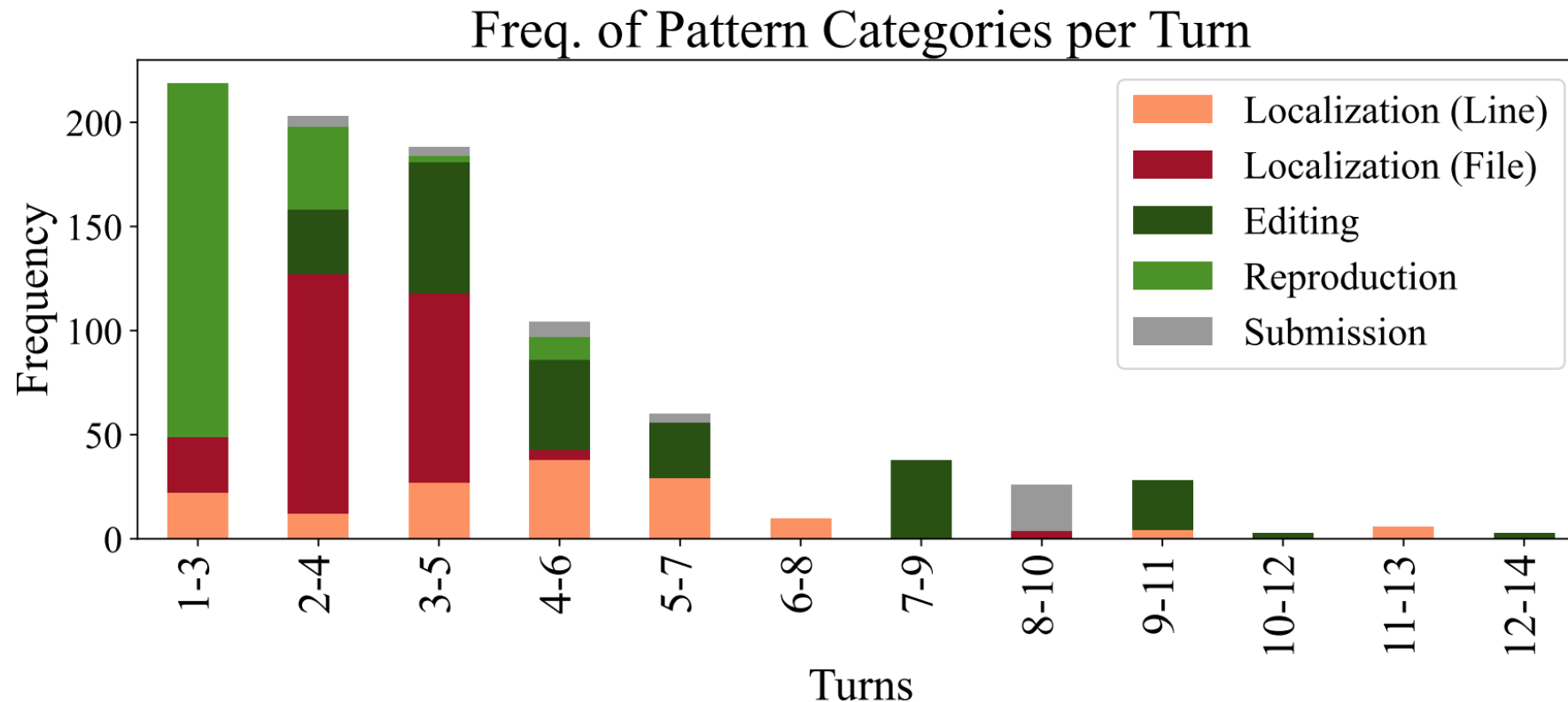
Found 3 matches for "numpy_handler.py" in /pydicom__pydicom:
/pydicom__pydicom/pydicom/overlays/numpy_handler.py
/pydicom__pydicom/pydicom/pixel_data_handlers/numpy_handler.py
/pydicom__pydicom/pydicom/waveforms/numpy_handler.py

SWE-Agent

Model	SWE-bench		SWE-bench Lite	
	% Resolved	\$ Avg. Cost	% Resolved	\$ Avg. Cost
RAG				
w/ GPT-4 Turbo	1.31	0.13	2.67	0.13
w/ Claude 3 Opus	3.79	0.25	4.33	0.25
Shell-only agent				
w/ GPT-4 Turbo	-	-	11.00	1.46
w/o Demonstration	-	-	7.33	0.79
SWE-agent				
w/ GPT-4 Turbo	12.47	1.59	18.00	1.67
w/ Claude 3 Opus	10.46	2.59	13.00	2.18






















SWE-Agent

- ▶ Agent tends to first reproduce the issue. Then, find the file and edit it.



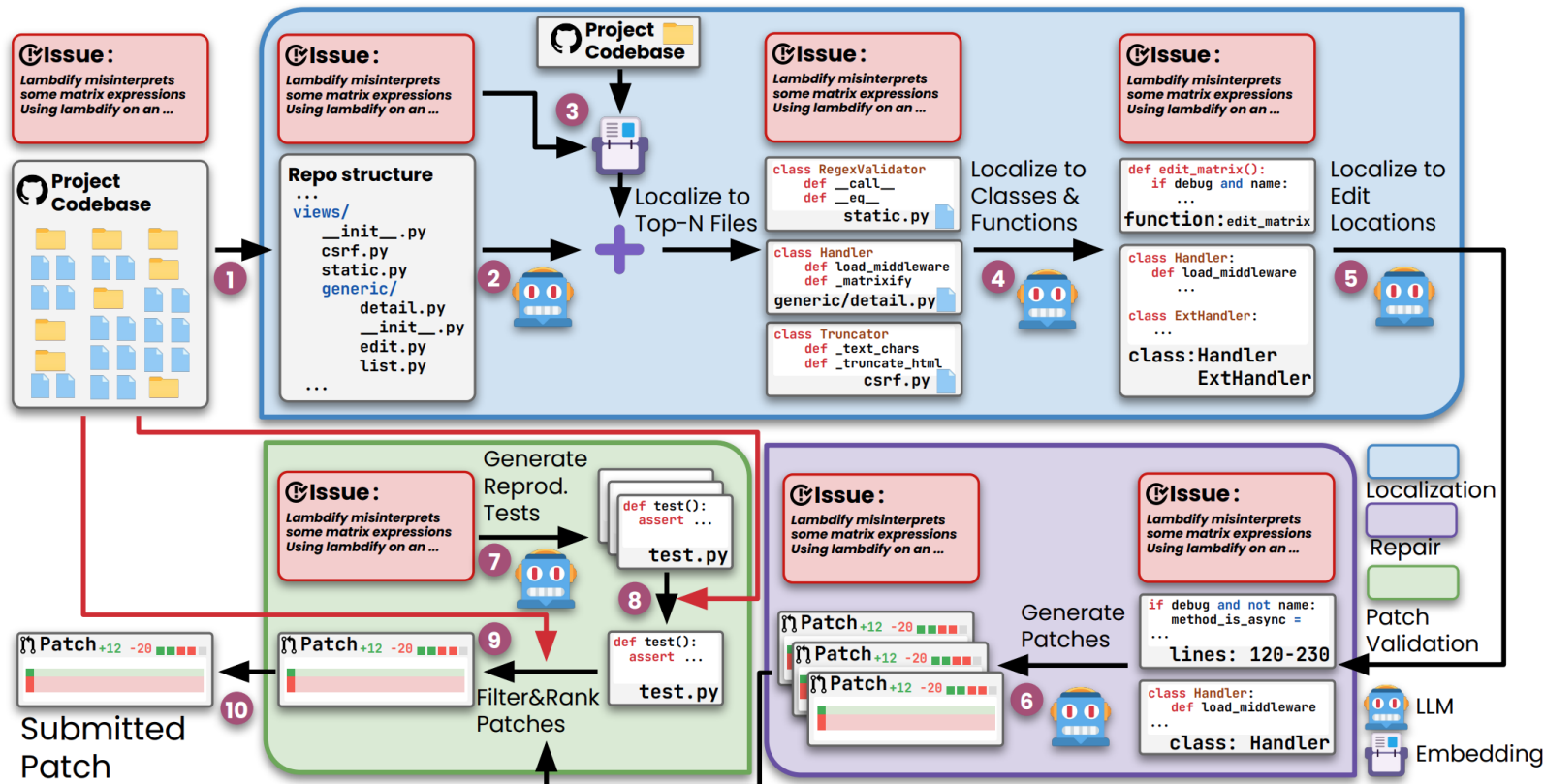
Do we really need all those tools? Approach 1

- ▶ Recent LLMs can do even better with just the shell
- ▶ E.g. “mini-SWE agent” has a 100-line implementation (vs thousands for SWE agent)

Bash Only Verified Lite Full Multimodal						
Bash Only evaluates all LMs with a minimal agent on SWE-bench Verified (details)						
Filters: Open Scaffold ▾ All Tags ▾						
Model	% Resolved	Avg. \$	Org	Date	Release	
  Claude 4.5 Sonnet (20250929)	70.60	\$0.56		2025-09-29	1.13.3	
  Claude 4 Opus (20250514)	67.60	\$1.13		2025-08-02	1.0.0	
  GPT-5 (2025-08-07) (medium reasoning)	65.00	\$0.28		2025-08-07	1.7.0	
  Claude 4 Sonnet (20250514)	64.93	\$0.37		2025-07-26	1.0.0	
  GPT-5 mini (2025-08-07) (medium reasoning)	59.80	\$0.04		2025-08-07	1.7.0	
  o3 (2025-04-16)	58.40	\$0.33		2025-07-26	1.0.0	
  Qwen3-Coder 480B/A35B Instruct	55.40	\$		2025-08-02	1.0.0	

Do we really need all those tools? Approach 2

- ▶ “Agentless” [Xia et al. 2024]: develop a workflow very tailored to resolving issues
- ▶ Codifies a common pattern from SWE-agent: reproduce, localize, repair, validate



Do we really need all those tools? Approach 2

- ▶ Each step in Agentless can generate multiple possible outputs and aggregate them (similar to MBR-exec, CodeT)
- ▶ Good fit for reasoning models that can take in a lot of context and produce complex outputs in a single shot
- ▶ SOTA results on SWE-Bench verified were briefly from o1/o3 + Agentless

SWE-Bench: Discussion

- ▶ How much of software engineering does this task capture?
- ▶ How much have LLMs memorized these particular repos?

Training SWE Agents

SWE Gym

- ▶ Build a set of 2.4K SWE-bench-like instances from 11 repos to use as a training set
- ▶ SWE-bench came with a training set, but instances were not executable
 - ▶ Ensuring executability is time-consuming! 200 human annotation hours, in this work
 - ▶ In Yiqing's lecture, we'll see how to automate this
- ▶ Executable environments allow training with execution feedback
 - ▶ This work: uses rejection sampling fine-tuning
 - ▶ Could also use RL

SWE Gym: Rejection Sampling Fine-Tuning

- ▶ Sample agentive trajectories from a teacher model
- ▶ Choose ones that pass test cases
- ▶ Use these as SFT data for a student model
 - ▶ In this work, distillation: GPT-4o is the teacher and Qwen-2.5 is the student
 - ▶ Could also do self-improvement, where the same model is teacher and student. Pass@k -> pass@1

SWE Gym: Results

Model Size	Empty Patch (% , ↓)			Stuck in Loop (% , ↓)			Avg. Turn(s)			Resolve Rate (% , ↑)		
	zero-shot	fine-tuned	Δ	zero-shot	fine-tuned	Δ	zero-shot	fine-tuned	Δ	zero-shot	fine-tuned	Δ
SWE-Bench Lite (300 instances)												
7B	40.3	29.7	-10.7	47.0	31.0	-16.0	20.3	22.2	+1.9	1.0 (± 1.0)	10.0 (± 2.4)	+9.0
14B	49.7	18.1	-31.6	31.7	27.1	-4.6	23.2	21.4	-1.8	2.7 (± 1.9)	12.7 (± 2.3)	+10.0
32B	27.0	18.1	-8.9	16.7	18.1	+1.5	15.5	29.3	+13.9	3.0 (± 1.4)	15.3 (± 2.5)	+12.3
SWE-Bench Verified (500 instances)												
7B	45.8	33.8	-12.0	39.6	21.0	-18.6	21.9	35.3	+13.4	1.8 (± 1.1)	10.6 (± 2.1)	+8.8
14B	44.9	14.5	-30.4	32.1	21.3	-10.7	25.5	30.1	+4.6	4.0 (± 1.6)	16.4 (± 2.0)	+12.4
32B	9.5	13.8	+4.3	29.4	23.8	-5.6	24.6	31.6	+7.0	7.0 (± 1.3)	20.6 (± 2.1)	+13.6

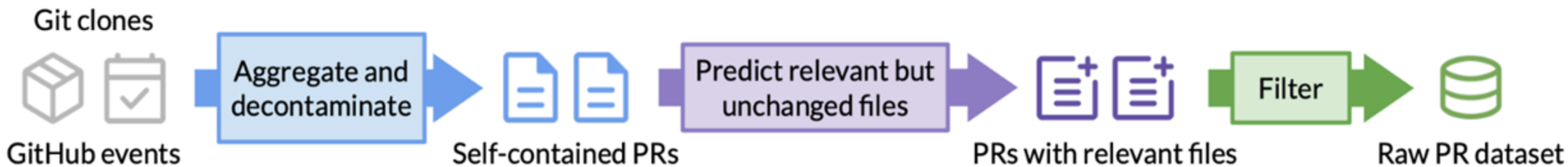
SWE-RL

- ▶ Task: SWE-bench. Generate a diff to solve an issue in a GitHub repo
- ▶ Use a simple, scalable RL reward that doesn't actually require executing the code. Train this at large scale.
- ▶ We'll find that it scales and generalizes better than SFT!

Data Curation

- GitHub events from GHArchive, Jan 2015 to Aug 2024
- 4.6M repos; 24M aggregated PR instances; 11M after filtering
- Relevant files prediction -> **reduce model bias**

Without this: editing every file or unable to handle noisy files in the context



Reward Modeling: Prompt

[System] A user will ask you to solve a task. You should first draft your thinking process (inner monologue). Then, generate the solution. Your response format must follow the template below:

<think>

Your thoughts or/and draft, like working through an exercise on scratch paper. Be as casual and as long as you want until you are confident to generate a correct solution.

</think>

<solution>

Final solution presented to the user.

</solution>



[User] ...here is the issue text: {issue}

...here is the code context: {context}

...generate *search/replace* edits to fix the issue...



[Assistant]



Reward Modeling: Output

<think>

We need to import ...

</think>

<solution>

mathweb/flask/app.py

<<<<<< SEARCH

from flask import Flask

=====

import math

from flask import Flask

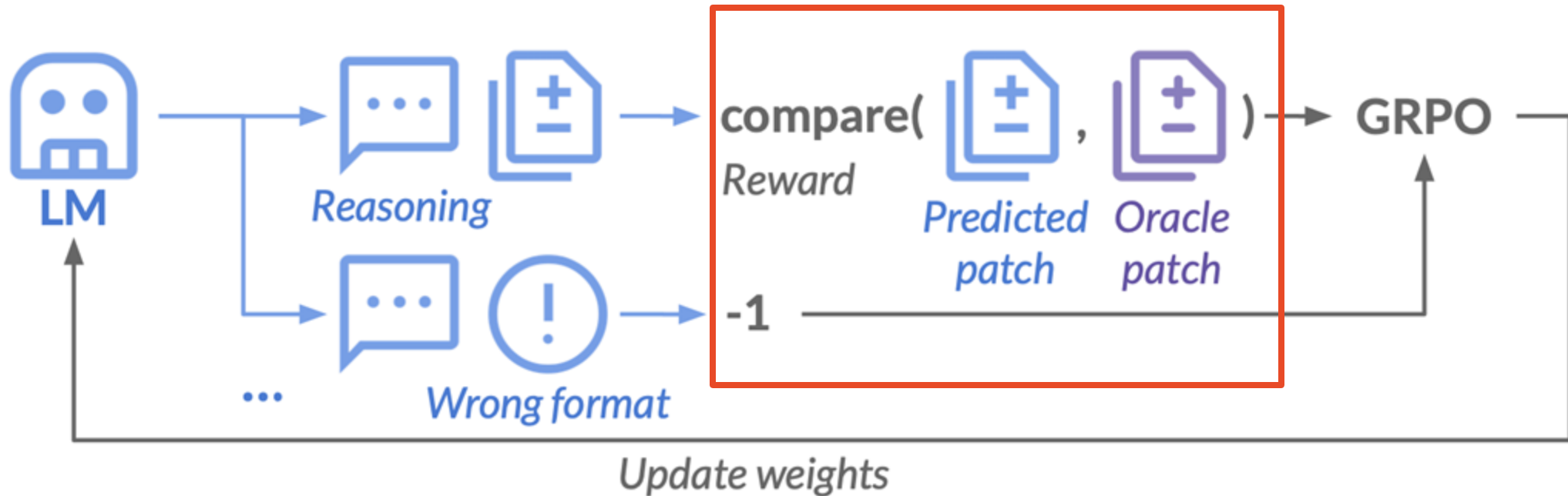
>>>>>> REPLACE

</solution>

Reward Modeling: Rewards

- **Reward function**

- **-1**: Wrong format or `search == replace`
- **[0,1]**: based on sequence similarity from `difflib.SequenceMatcher`



Reward Modeling: Aha Moments

- “Aha moments” emerge in both **ID** and **OOD** tasks
- Self Reflection, Exploring Multiple approaches, Divide-and-conquer

GitHub issue solving (in domain)

We need to identify where the issue of not preserving single quotes in TODO comments occurs... In the `ToDoItem` class, the description is being extracted...

But wait, in the `ToDoItem` constructor, there's this line:
`_type = description.Substring(0, idxDelimiter);`
However, the `idxDelimiter` is being found using `description.IndexOf(":", StringComparison.InvariantCulture);` ...but that's not the issue here.

But then I saw it: in the second constructor of `ToDoItem`, `idxDelimiter = description.IndexOf(":", StringComparison.InvariantCulture);`...

The actual issue is likely due to... And then it hits me... I think I've got it now.

Self reflection

Simple function implementation (out-of-domain)

To solve this problem, we need to create a function that takes a list of numbers and a delimiter as input... We can start by checking if the input list is empty....

If the list is not empty...

However, a more Pythonic way to solve this problem would be to use a list comprehension...

But before that, let's think about the edge cases...

Alternatively, we can use a list comprehension with the `zip` function...

But the most straightforward way to solve this problem is to use a simple loop...

Exploring multiple approaches

Mathematics (out-of-domain)

To solve this problem, we first need to find the x-intercepts of the two given ellipses...

For the first ellipse... For the second ellipse...

Now, let's find the intersection points... Now, let's use the distance formula...

The length of side AB is... The length of side BC is... The length of side CD is... The length of side DA is...

Since we want ABCD to be a square, all sides must have the same length...

Squaring both sides... Expanding and simplifying... Combining like terms... Multiplying both sides by 36... Adding 42 to both sides...

Divide-and-conquer

Evaluation: Setup

- **Model:** Llama3-SWE-RL-70B
 - base model: Llama-3.3-70B-Instruct
 - no SFT, just RL
- **SFT baseline:** Llama3-SWE-SFT-70B
 - same base model and seed PR dataset as the RL model
 - additional mixed data for diversity and generalizability
 - reasoning traces produced by Llama3
- **Benchmark:** SWE-bench Verified
- **Scaffold:** Agentless Mini

Evaluation: Baseline Comparison

- RL shows best repair performance despite slightly lower correct format rate
- SFT also improves format and patch quality over base model

	Correct Format	Repair Success
Llama-3.3-70B-Instruct	12.2%	5.4%
Llama-3-SWE-SFT-70B	96.2%	29.6%
Llama-3-SWE-RL-70B	95.6%	34.8%

Evaluation: Main Results

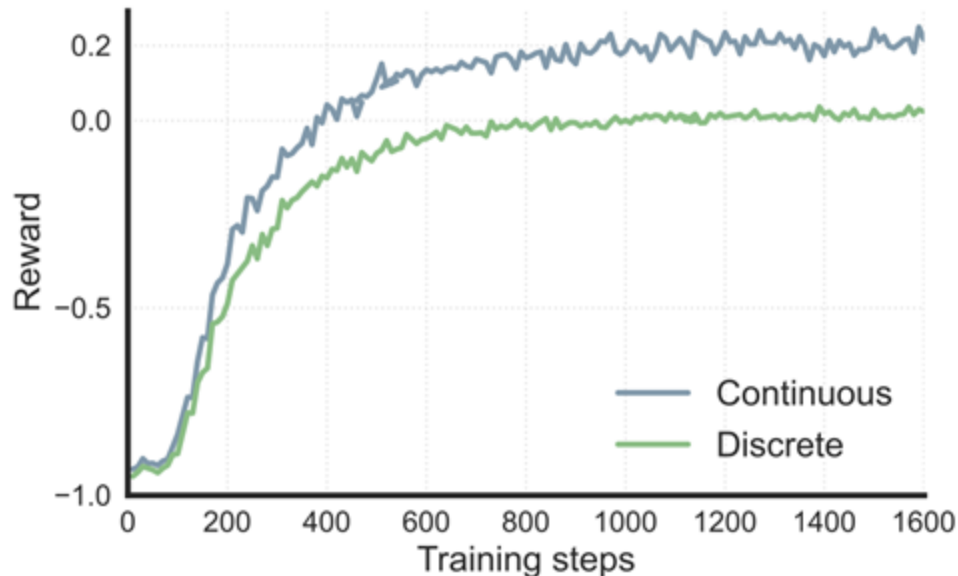
- **Llama3-SWE-RL-70B + Agentless Mini**

- **41.0%** solve rate on SWE-bench Verified
- Best among <100B models and comparable to **o1-preview + Agentless**

Model	Scaffold	SWE-bench Verified
Model closed-source or size \gg 100B		
GPT-4o	SWE-agent	23.2
Claude-3.5-Sonnet	SWE-agent	33.6
GPT-4o	Agentless	38.8
o1-preview	Agentless	41.3
DeepSeek-V3 ¹	Agentless	42.0
Claude-3.5-Sonnet	AutoCodeRover-v2.0	46.2
Claude-3.5-Sonnet	Tools	49.0
DeepSeek-R1 ¹	Agentless	49.2
Claude-3.5-Sonnet	Agentless	50.8
Claude-3.5-Sonnet	OpenHands	53.0
Model size \leq 100B		
SWE-Llama-13B	RAG	1.2
SWE-Llama-7B	RAG	1.4
Lingma-SWE-GPT-7B	SWE-SynInfer	18.2
Lingma-SWE-GPT-72B	SWE-SynInfer	28.8
SWE-Fixer-72B	SWE-Fixer	30.2
Llama3-Midtrain-8B (beta)²	Agentless Mini	31.0
SWE-Gym-32B	OpenHands	32.0
Llama3-SWE-RL-70B	Agentless Mini	41.0

Evaluation: Reward Ablation

- **Continuous reward:** sequence similarity
- **Discrete reward:** exact match (?better if execution correctness)
- Faster learning and better repair performance in continuous reward



Reward Type	Correct Format	Repair Success
Discrete	94.2%	29.0%
Continuous	95.6%	34.8%

Evaluation: Generalizability

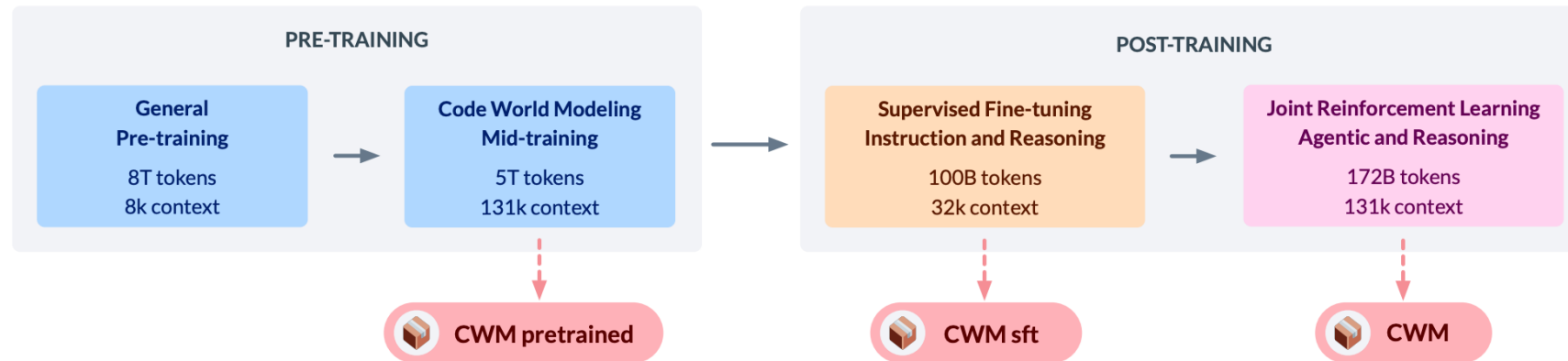
- Despite training only on SWE tasks, RL model outperforms SFT baseline and base model in all **OOD** tasks
- SFT generally degrades performance in **OOD** tasks

Benchmark	Llama-3.3-70B-Instruct	Llama-3-SWE-SFT-70B	Llama-3-SWE-RL-70B
HumanEval+	76.2	73.2	79.9
BCB-Hard(I/C)	28.4 / 29.1	25.7 / 24.3	28.4 / 29.1
CRUXEVAL(I/O)	60.5 / 61.9	68.4 / 75.1	71.6 / 75.5
Math(s/I)	63.2 / 70.9	54.0 / 71.7	73.7 / 73.7
MMLU	86.49	85.26	86.82

SWE-RL Takeaways

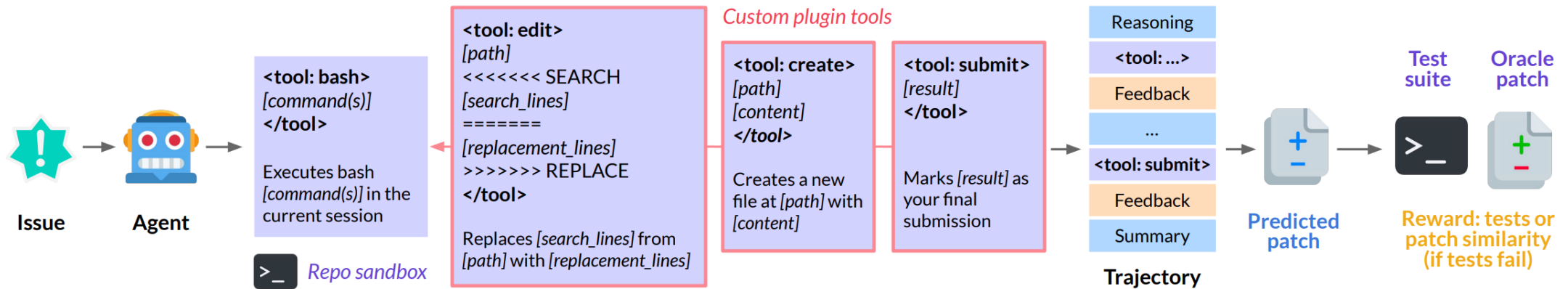
- ▶ Simple reward allows scalable RL even without executable environments
- ▶ Some generalization to other tasks/domains as well: RL seems to overfit less than SFT
- ▶ But, hard to scale beyond

RL with Execution: FAIR's Code World Model



- ▶ Construct 35K executable repository images, using a combination of an agent and GitHub continuous integration
- ▶ Repos have executable, human-written test cases
- ▶ What can you do with lots of executable repos?
 - ▶ Mid-training (see the paper)
 - ▶ RL on issues with execution reward

RL with Execution: FAIR's Code World Model



- ▶ Agent has a similar set of tools to SWE-Agent and OpenHands
- ▶ Two –part reward:
 - ▶ [execution] if all hidden test cases pass => 1
 - ▶ [similarity] else if patch similarity (from SWE-RL) is above threshold => 0
 - ▶ else => -1
- ▶ Training involves cold starts, hints, and multi-task RL with other verifiable tasks

RL with Execution: FAIR's Code World Model

Solve the following issue by implementing the necessary code changes and submitting a patch file:

`<issue_description> {issue} </issue_description>`

The `[result]` argument of `<tool: submit>` should be the path to a patch file that resolves the issue ... Once you've submitted at least once and are confident in your solution, provide a final response summarizing your work ...

Your primary objective is to ensure patch correctness above all else - thoroughly explore the codebase, think hard, and leverage significant execution to verify correctness by writing comprehensive tests to validate your solution and running existing tests to prevent regressions. Only submit when you are genuinely confident in your patch's correctness ...

<code><think> ...</code>	<code><tool: bash> ls ...</code>	...	<code><think> ...</code>	<code><tool: edit> ...</code>	...	<code><think> ...</code>	<code><tool: bash> pytest ...</code>	...	<code><think> ...</code>
<code><think> ... </think></code>	<code><tool: bash> git diff ... > submission.patch </tool></code>					(No output)					
<code><think> ... </think></code>	<code><tool: submit> submission.patch </tool></code>					The following patch content is marked as your final submission ...					
<code><think> ... Now, I'll provide a summary ... </think></code>					## Issue summary ... ## Reproduction ... ## Code changes ...						

RL with Execution: FAIR's Code World Model

- ▶ Taking away tools, or switching to another agentive harness, does decrease performance
- ▶ But it still does well

Harness	Configuration	Resolve Rate (%)
Mini-SWE-Agent	250 turns	37.6
OpenHands	40 turns	36.0
	128 turns	42.6
	500 turns	40.8
Ours (bash-only)	128 turns	42.1
Ours	128 turns	53.9