QCon
全球软件开发大会

InfoQ 极客传媒

veRL for Training Coding Agent

姓名

张驰　verl项目发起人

巫锡斌　verl核心maintainer

01

Introduction to verl

verl in 2025

– Same front–end code

– Auto–backend selection

https://github.com/volcengine/verl

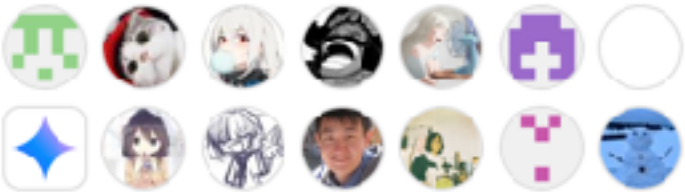
https://arxiv.org/abs/2409.19256

14.6k star, 2.3 fork

Awesome work using verl

- TinyZero: a reproduction of DeepSeek R1 Zero recipe for reasoning tasks
- SkyThought: RL training for Sky-T1-7B by NovaSky AI team.
- simpleRL-reason: SimpleRL-Zoo: Investigating and Taming Zero Reinforcement Learning for Open Base Models in the Wild
- Easy-R1: Multi-modal RL training framework
- OpenManus-RL: LLM Agents RL tuning framework for multiple agent environments.
- rLLM: async RL training with verl-pipeline
- RAGEN: a general-purpose reasoning agent training framework
- Search-R1: RL with reasoning and searching (tool-call) interleaved LLMs
- ReSearch: Learning to Reason with Search for LLMs via Reinforcement Learning
- Skywork-OR1: Skywork open reasoner series
- ToRL: Scaling tool-integrated RL
- Absolute Zero Reasoner: A no human curated data self-play framework for reasoning
- verl-agent: A scalable training framework for long-horizon LLM/VLM agents, along with a new algorithm GiGPO
- RL-Factory: An easy and efficient RL post-training framework for Agentic Learning
- ReTool: ReTool: reinforcement learning for strategic tool use in LLMs. Code release is in progress...
- verl-tool: An unified and easy-to-extend tool-agent training framework based on verl
- PRIME: Process reinforcement through implicit rewards
- MemAgent: MemAgent: Reshaping Long-Context LLM with Multi-Conv RL-based Memory Agent
- PURE-VLM: A post-training recipe for scaling RL on Advanced Reasoning models
- GUI-R1: GUI-R1: A Generalist R1-style Vision-Language Action Model For GUI Agents
- DeepRetrieval: RL Training of Search Agent with Search/Retrieval Outcome
- Code-R1: Reproducing R1 for Code with Reliable Rewards
- DeepResearcher: Scaling deep research via reinforcement learning in real-world environments
- VAGEN: Training VLM agents with multi-turn reinforcement learning

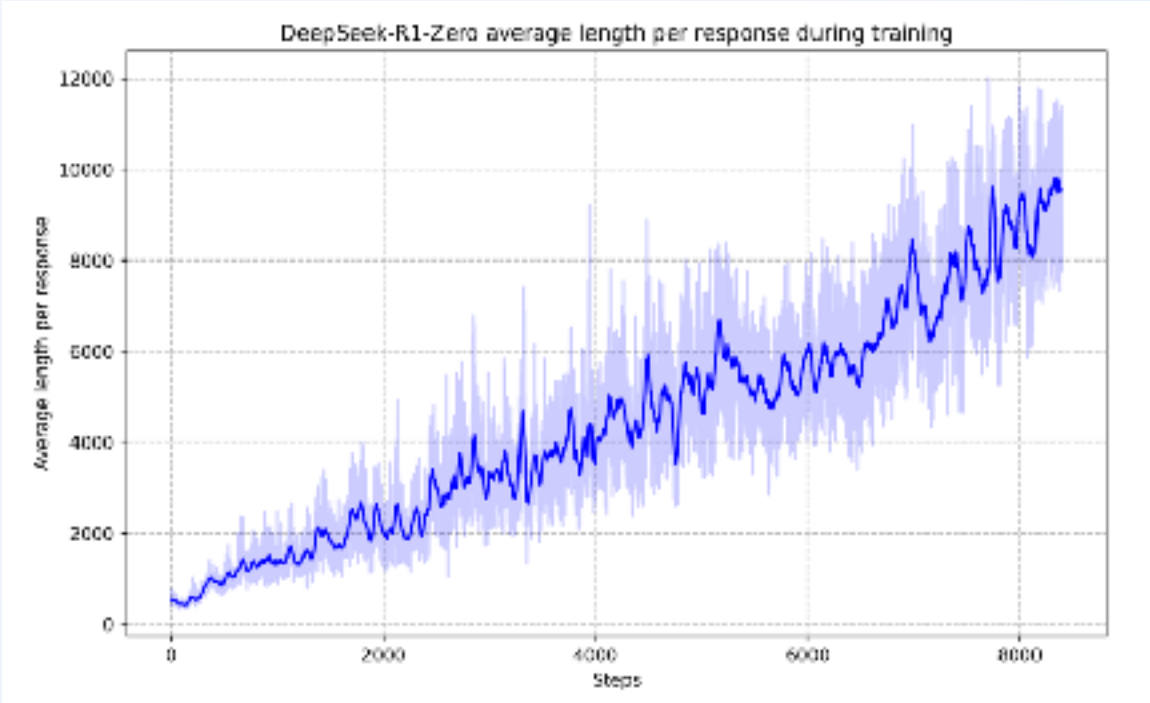
Contributors 398



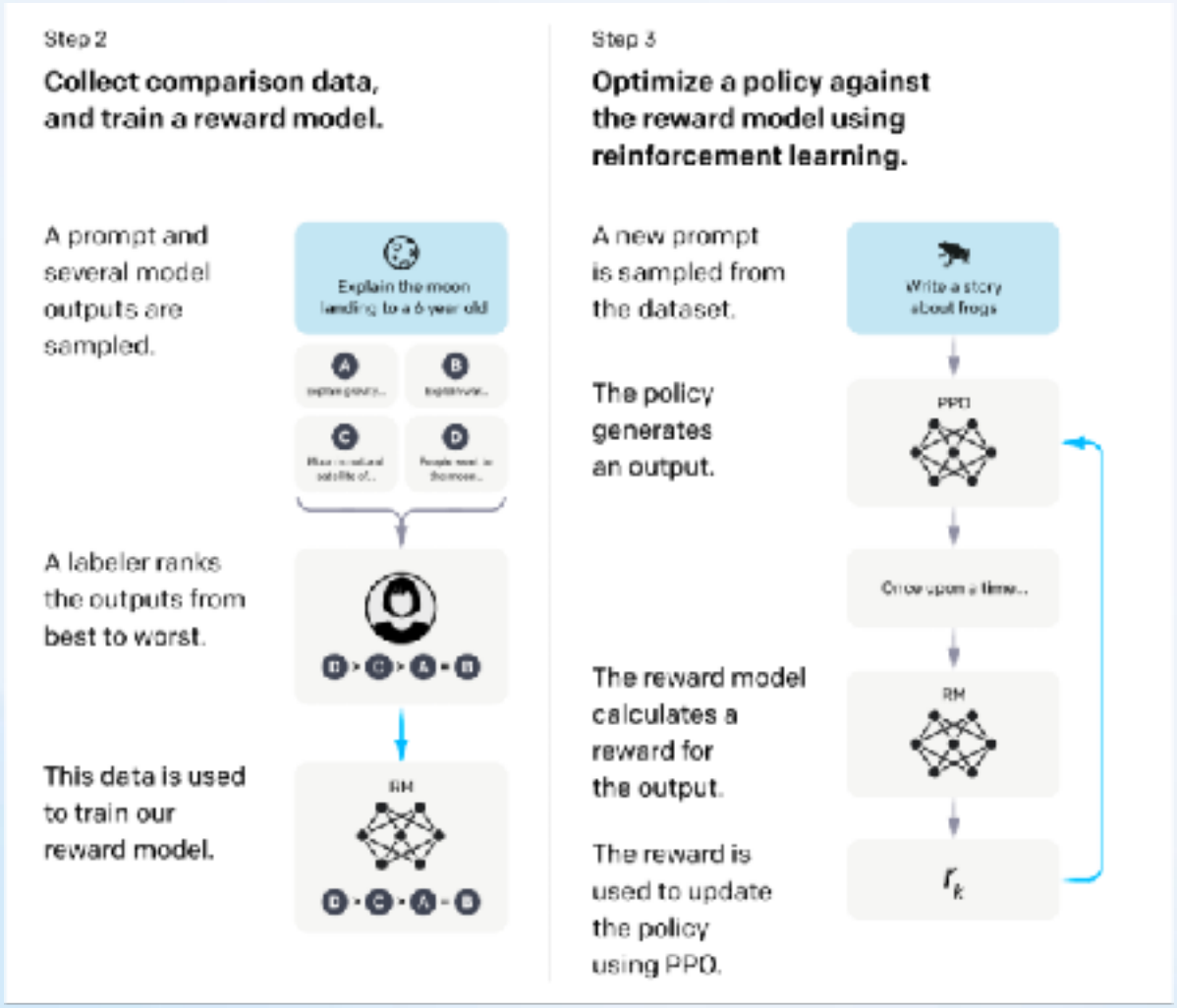
Reinforcement Learning is important

Reinforcement learning (RL) at Bytedance

- Classic Alignment with Human values
- Reasoning: O1/Claude–3.7 performance on math benchmarks
- Image/video/music generation
- Agentic LLM tool using
- Desktop operator, coding assistant, Gaming…



Introduction to Reinforcement Learning



Supervised fine-tuning

- Learning from labeled examples
- Optimize a single model

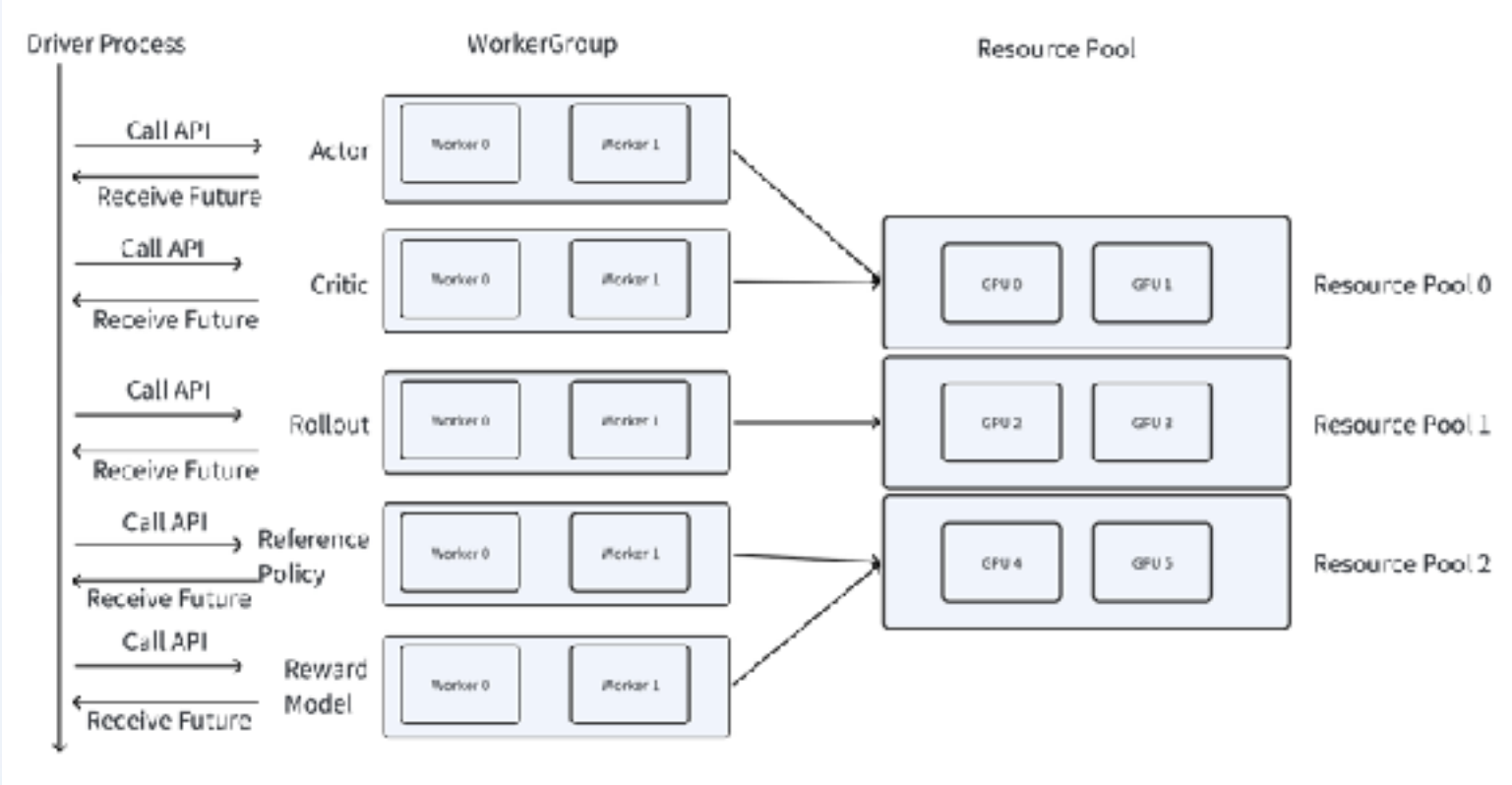
Reinforcement Learning

- Maximize the reward
- Play with multiple models in a single system

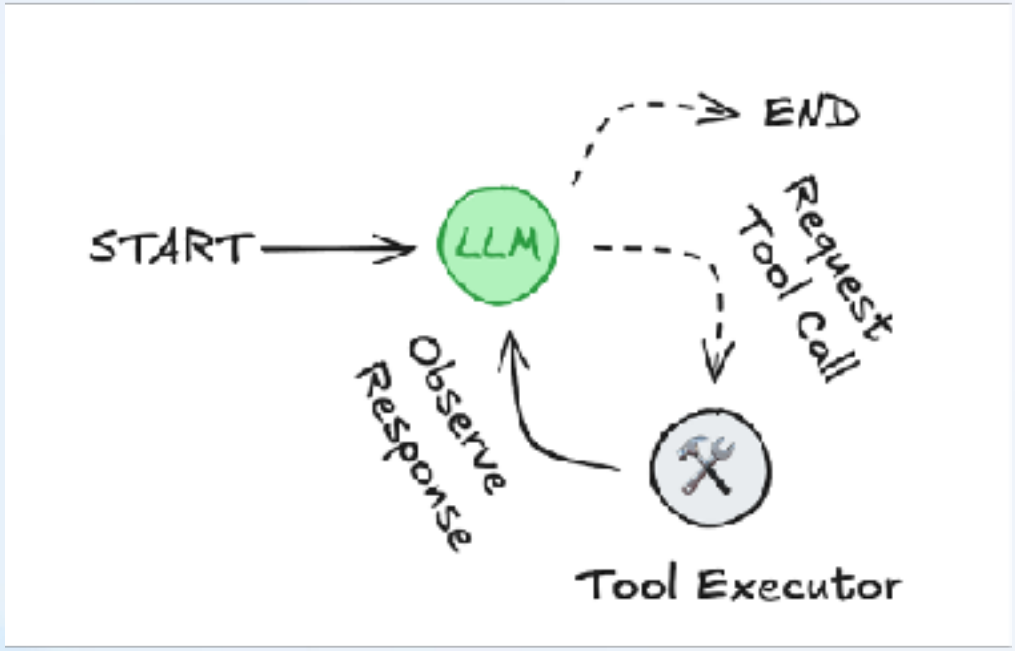
HybridFlow Programming Abstraction

Make LLM training/rollout as a **service**

- Classic Alignment with Human values
- Reasoning: O1/Claude–3.7 performance on math benchmarks
- Image/video/music generation
- Agentic LLM tool using
- Desktop operator, coding assistant, Gaming…



Agents



ReAct (from langchain-ai)

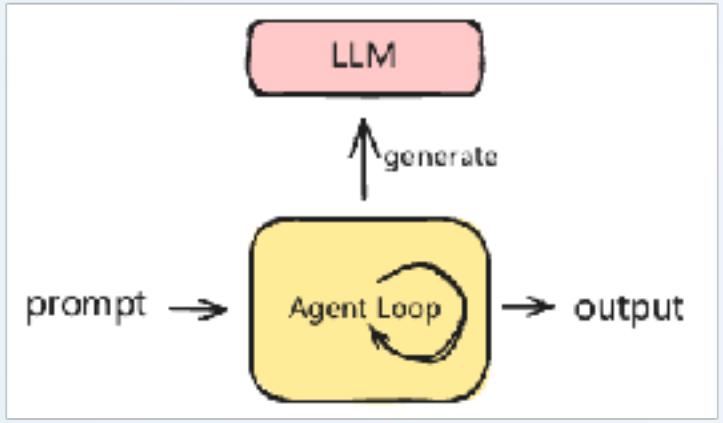
Agent: software systems that use AI to reasoning, planning, and memory and autonomy to make decisions, learn, and adapt.

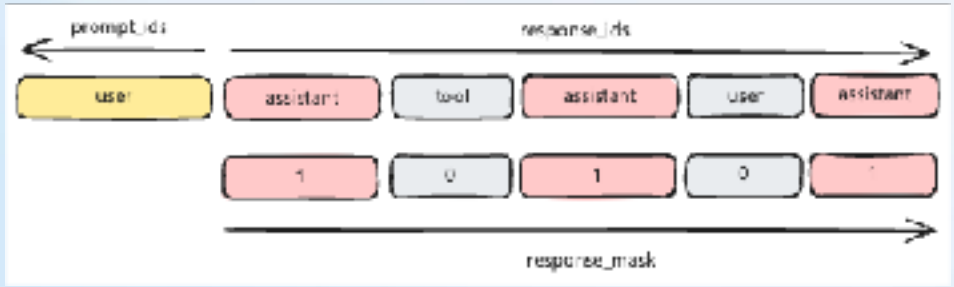
- Tool calling: Allowing the LLM to select and use various tools as needed.
- Memory: Enabling the agent to retain and use information from previous steps.
- Planning: Empowering the LLM to create and follow multi-step plans to achieve goals.

Agent RL: training LLM to make better decisions in complex, dynamic, real world.

Reinforcement Learning for Agents – AgentLoop

Rollout



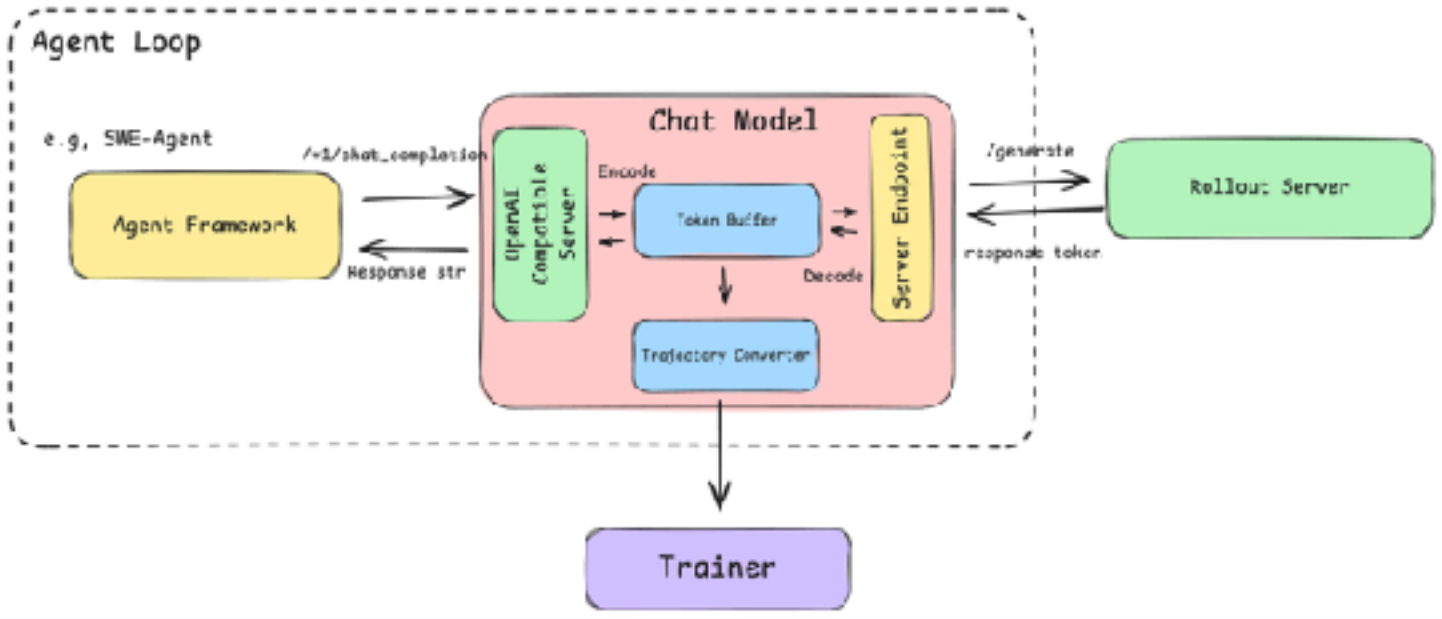


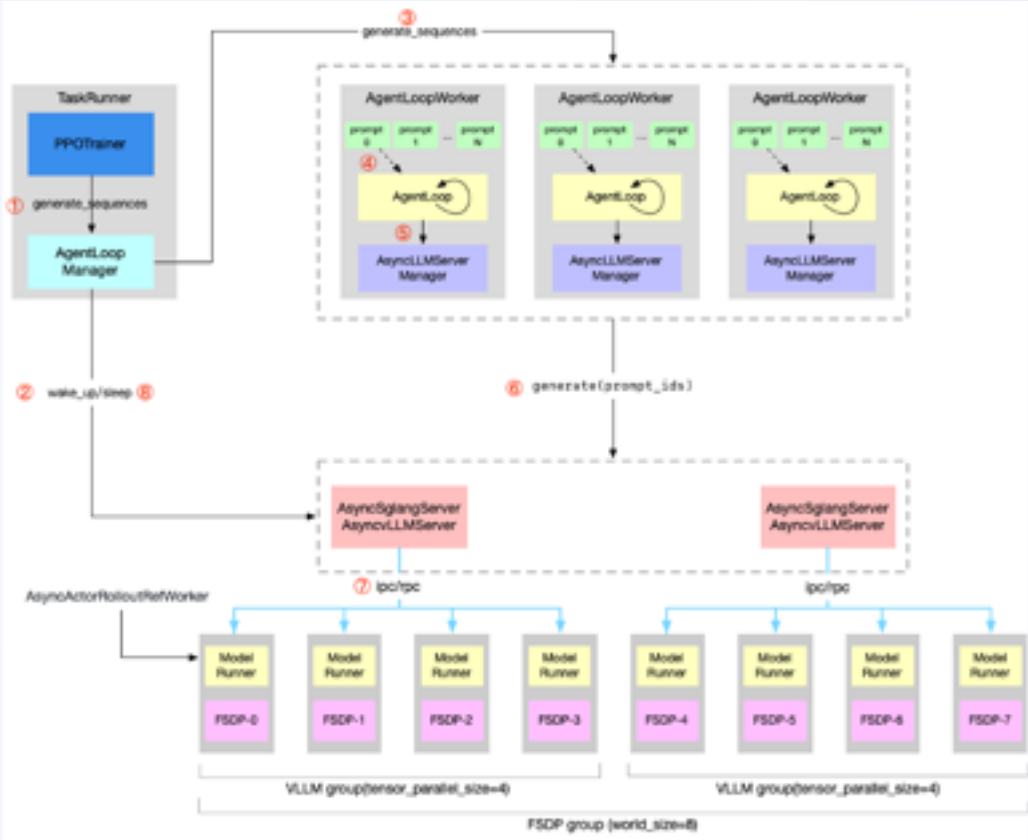
AgentLoop: given a user prompt, execute user defined loop, output multi-turn chat history as trajectory.

- Search: online web search
- MCP tools: image, video edit, …
- Code sandbox: execute code, python, java, …
- Virtual machine: operate browser, ppt, excel, …
- Android emulator: operate app
- …

```python
class AgentLoopBase(ABC):
    @abstractmethod
    async def run(
        self,
        messages: list[dict[str, Any]],
        sampling_params: dict[str, Any],
    ) -> AgentLoopOutput:
        """Run agent loop to interact with LLM server and environment.

        Args:
            messages (List[Dict[str, Any]]): Input messages.
            sampling_params (Dict[str, Any]): LLM sampling params.

        Returns:
            AgentLoopOutput: Agent loop output.
        """
        raise NotImplementedError
```

Reinforcement Learning for Agents – System



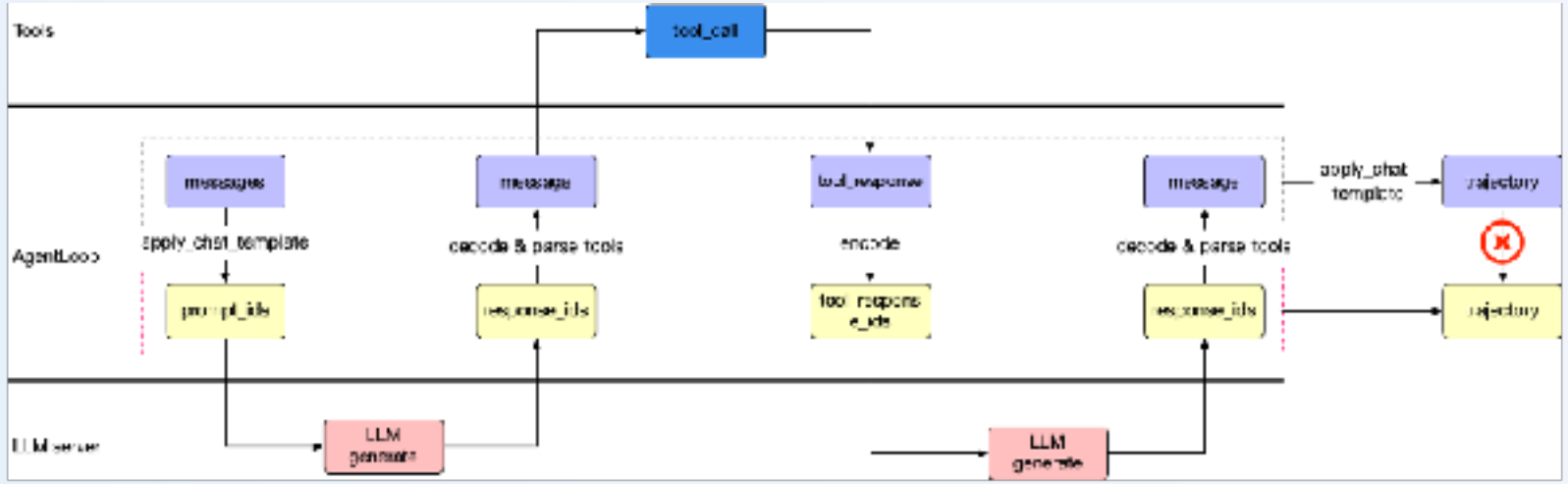


Highlight

- Server mode: vllm/sglang AsyncLLM engine
- Parallel running: asyncio loop run multiple prompts in parallel
- Load balance and sticky session: better kv cache utilization

Pitfalls in Agent RL

- **Lesson 1: token-in-token-out vs chat completion**



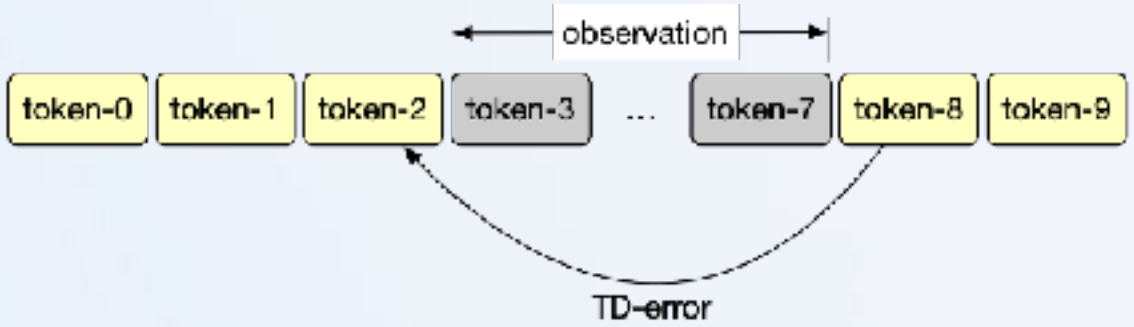
- ✗ Tool call regex extraction is irreversible
- ✗ Tokenizer encoder-decoder is irreversible

Qwen3 tokenizer

– decode([35946, 20412, 105165]) –> 我是中国人

– encode(我是中国人) –> [104198, 105165]

Pitfalls in Agent RL (cont')



- ✓ Mask out observation token in loss
- ✓ SKip observation token in GAE

02

Code Agent RL Training with verl

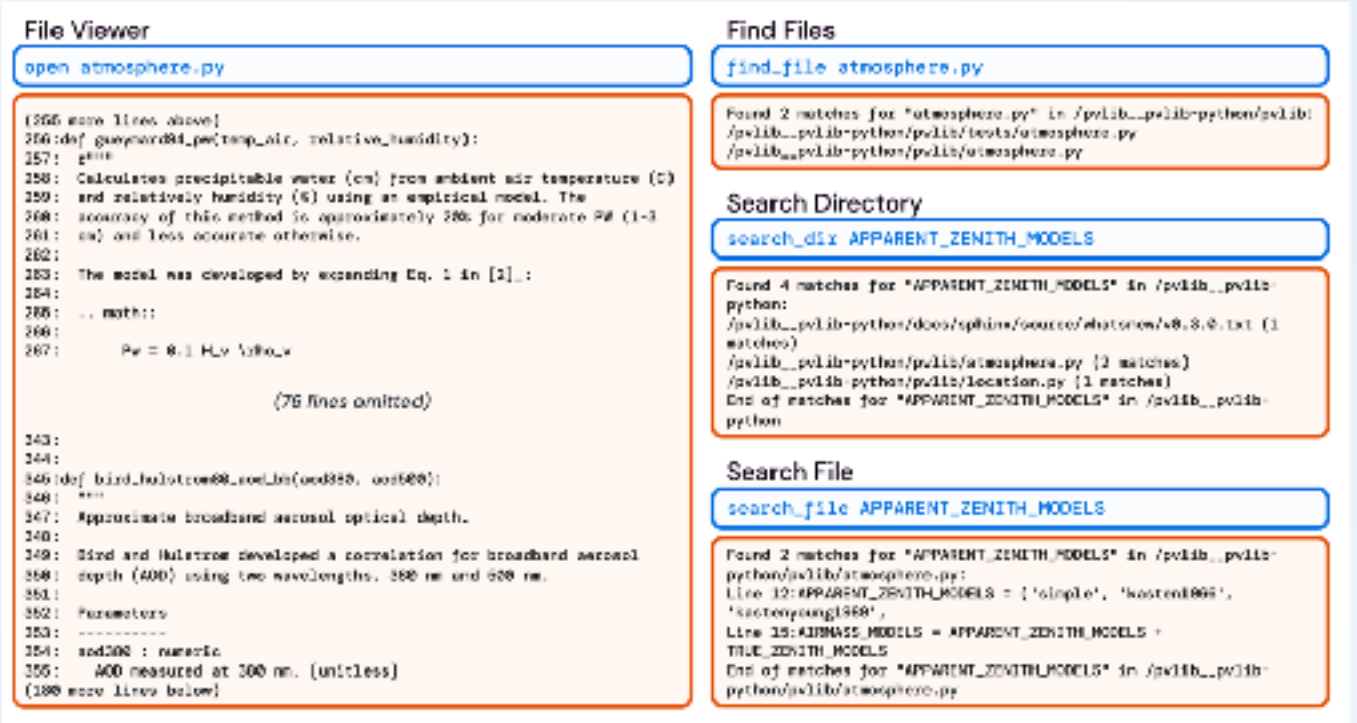
How To Evaluate LLM Code Ability?

Real world software engineering is more complicated than solving [a few lines of code]:

- Search
- Navigate
- View
- Edit
- Run

We need more comprehensive benchmark!

SWE-bench

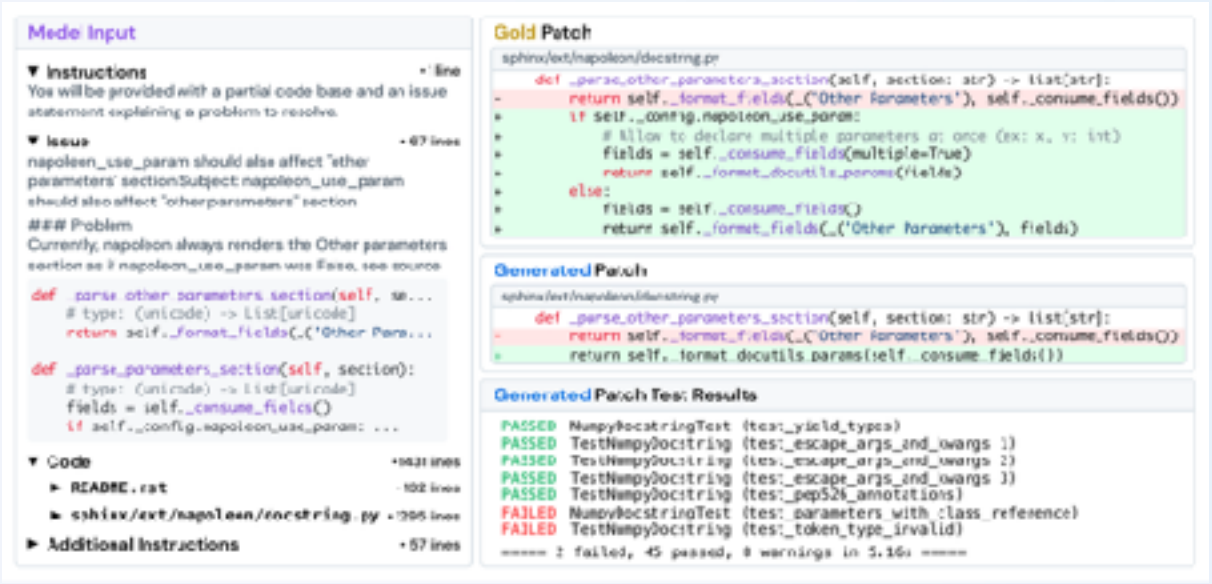


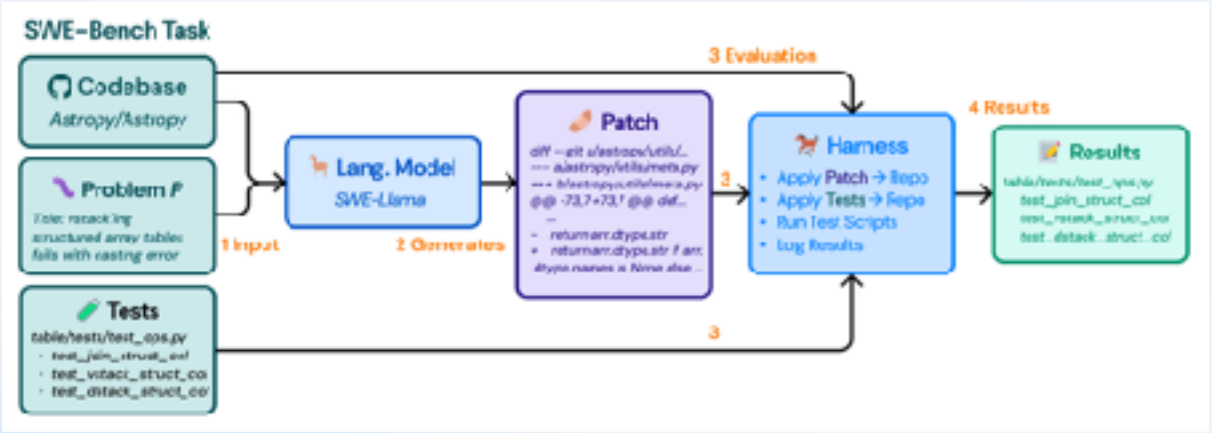
SWE-bench, Carlos E. Jimenez, et al, 2023

SWE-bench

Scrape Github PRs from 12 popular repositories:

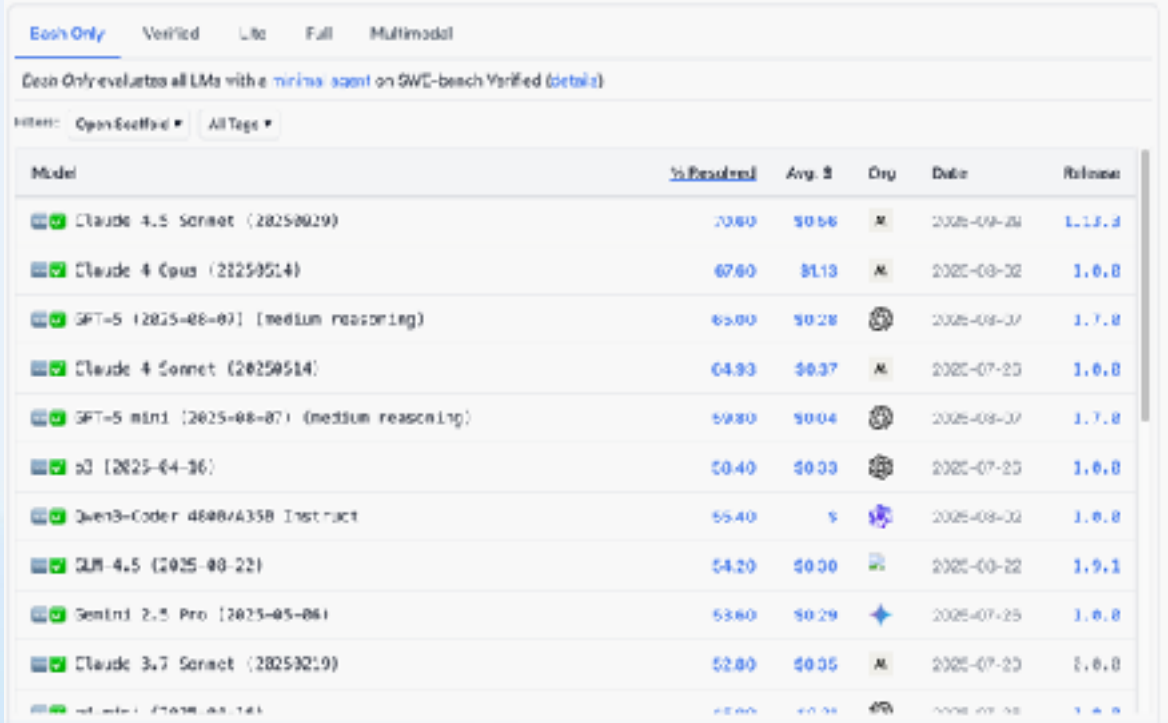
- **problem_statement**: The issue title and body
- **patch**: The gold patch, the patch generated by the PR
- **test_patch**: A test-file patch that was contributed by the solution PR
- ...

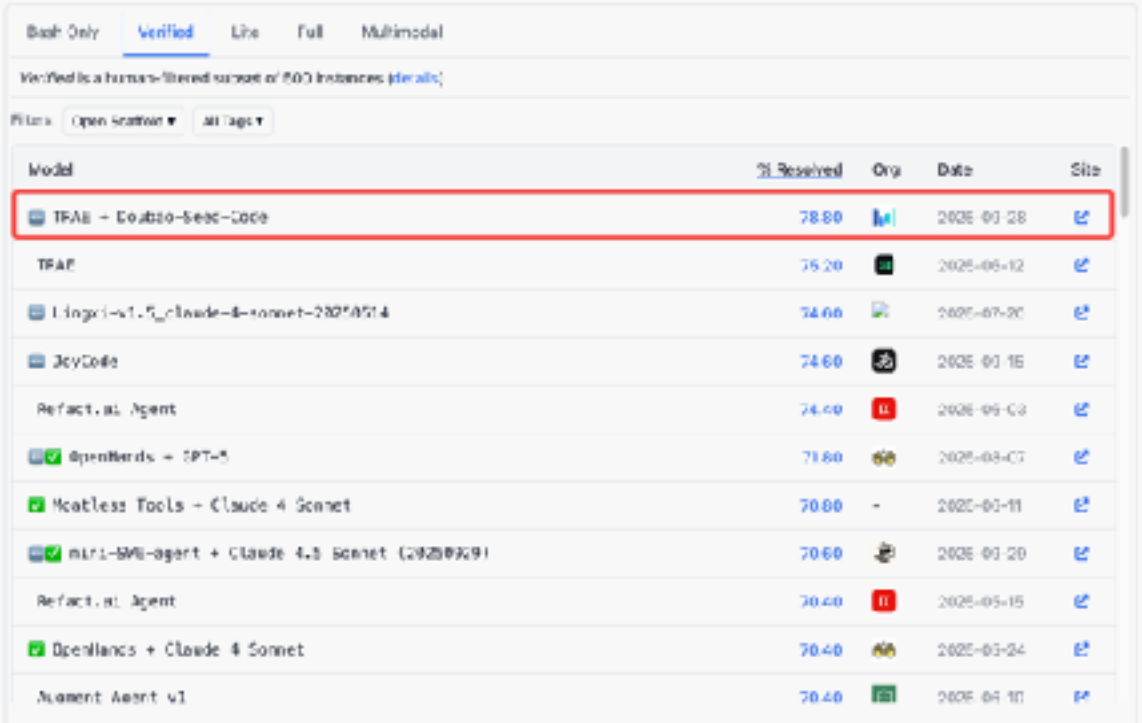




SWE-bench

SWE-bench has been widely adopted to evaluate LLM code ability.



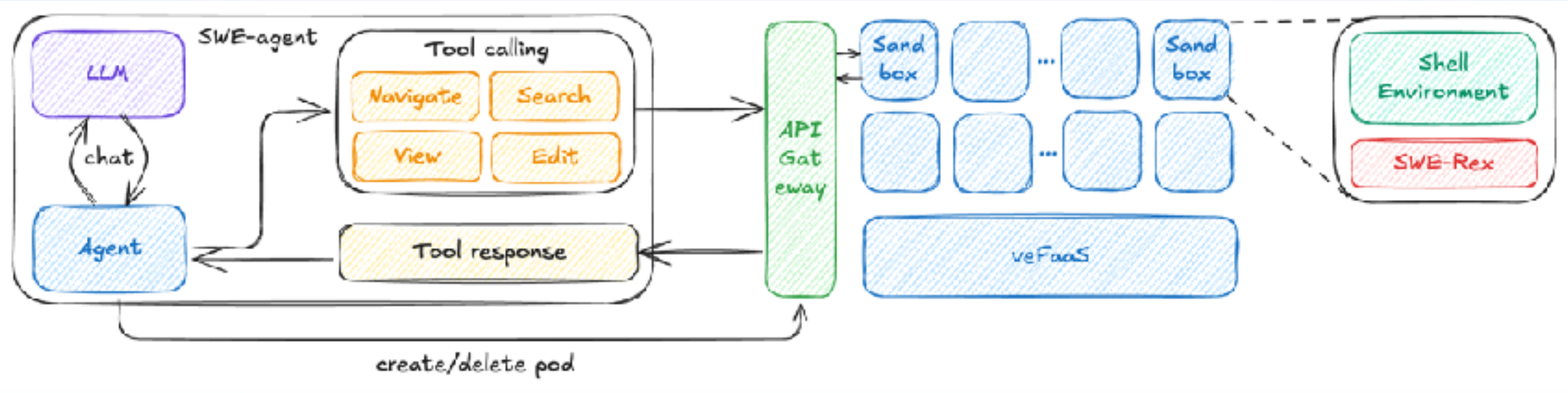


SWE-bench Leaderboards

Build Large Scale SWE-bench Infrastructure

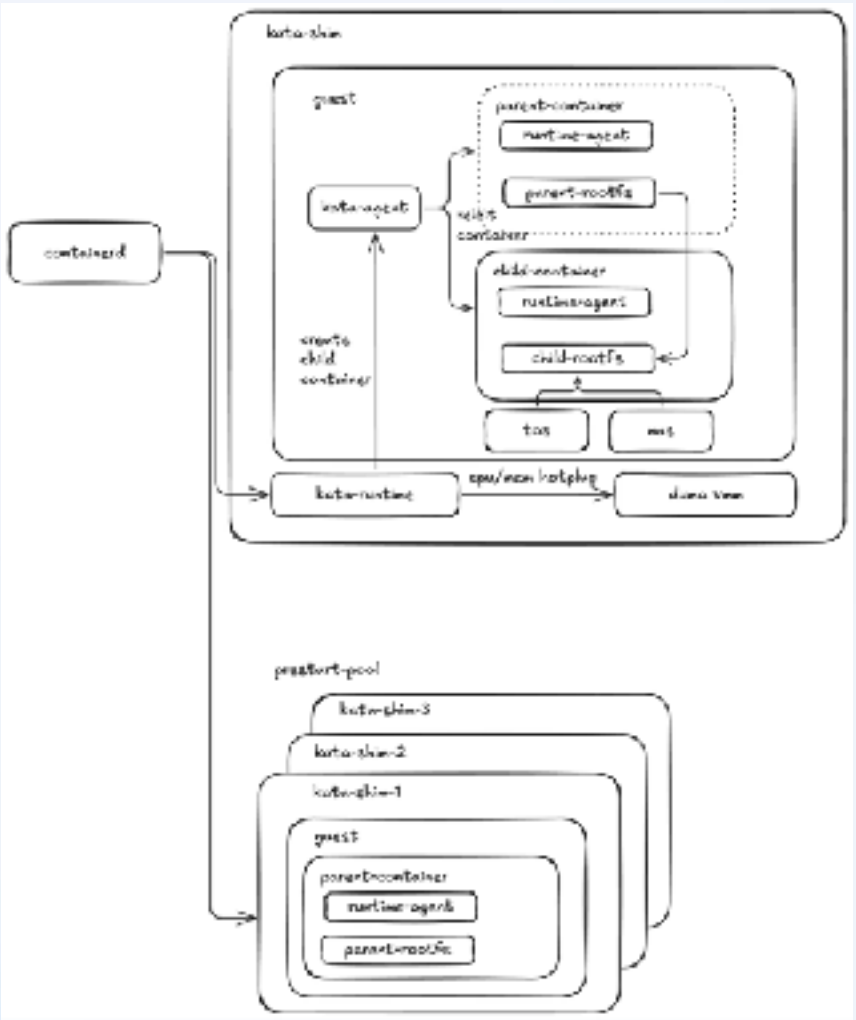
SWE-bench Infrastructure

- **SWE-agent**: enable LLM to autonomously use tools to fix issues.
- **SWE-Rex**: runtime interface for interacting with sandboxed shell environments.
- **veFaaS**: Volcano Engine Function as a Service, provides image cache, sandbox isolation, fast deployment.



veFaaS

- Kata container with strong isolation
- Image warmup with nydus and P2P distribution
- Image affinity scheduling

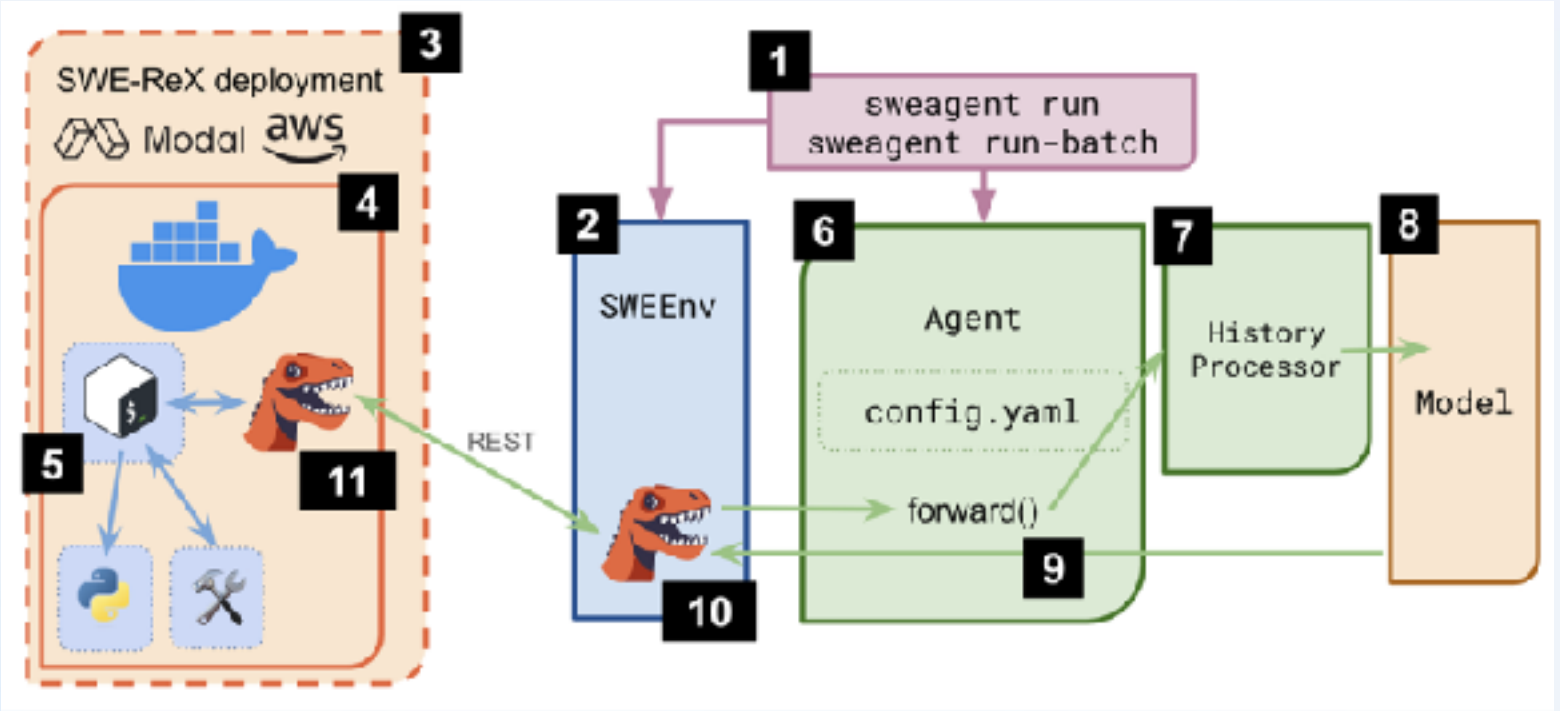


SWE-agent

Step 1~5: setup container, install tools, and initialize shell session

Step 6: setup agent with tool config yaml

Step 7~11: agent query model, parse action and execute shell command



SWE-agent architecture

SWE-agent

SWE-agent implementation

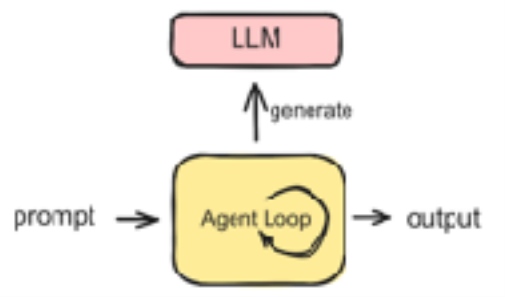
```python
class SweAgent:

    def forward(self, history: list[dict[str, Any]]) -> StepOutput:
        self._chook.on_model_query(messages=history, agent=self.name)

        # Step 1: Query the model with tools
        output = self.model.query(history, tools=self.tools.config.tools)
        step.output = output["message"]

        # Step 2: Parse the actions from the model output
        step.thought, step.action = self.tools.parse_actions(output)
        step.thinking_blocks = output.get("thinking_blocks", [])
        if output.get("tool_calls") is not None:
            step.tool_call_ids = [call["id"] for call in output["tool_calls"]]
            step.tool_calls = output["tool_calls"]
        step.extra_info["raw"] = output.get("raw", "")
        self.logger.info(
            f"💭 THOUGHT\n{step.thought}\n\n🎬 ACTION\n{step.action.strip()}"
        )
        self._chook.on_actions_generated(step=step)

        # Step 3: Handle the action
        return self.handle_action(step)

    def run(
            self,
            env: SWEEnv,
            problem_statement: ProblemStatement | ProblemStatementConfig,
            output_dir: Path = Path("."),
    ) -> AgentRunResult:
        self.setup(env=env, problem_statement=problem_statement, output_dir=output_dir)

        # Run action/observation loop
        self._chook.on_run_start()
        step_output = StepOutput()
        while not step_output.done:
            step_output = self.step()
            self.save_trajectory()
        self._chook.on_run_done(trajectory=self.trajectory, info=self.info)

        data = self.get_trajectory_data()
        return AgentRunResult(info=data["info"], trajectory=data["trajectory"])
```

Tool definition

```yaml
tools:
  str_replace_editor:
    signature: |
      str_replace_editor <command> <path> [<file_text>] [<view_range>] [<old_str>] [<new_str>] [<insert_line>]
    # This docstrings was taken from openhands:
    # https://github.com/All-Hands-AI/OpenHands/blob/main/openhands/agenthub/codeact_agent/function_calling.py
    docstring: >
      Custom editing tool for viewing, creating and editing files
      * State is persistent across command calls and discussions with the user
      * If `path` is a file, `view` displays the result of applying `cat -n`. If `path` is a directory, `view` lists non-hidden files and directories up to 2 levels deep
      * The `create` command cannot be used if the specified `path` already exists as a file
      * If a `command` generates a long output, it will be truncated and marked with `<response clipped>`
      * The `undo_edit` command will revert the last edit made to the file at `path`

      Notes for using the `str_replace` command:
      * The `old_str` parameter should match EXACTLY one or more consecutive lines from the original file. Be mindful of whitespaces!
      * If the `old_str` parameter is not unique in the file, the replacement will not be performed. Make sure to include enough context in `old_str` to make it unique
      * The `new_str` parameter should contain the edited lines that should replace the `old_str`
    arguments:
      - name: command
        type: string
        description: "The commands to run. Allowed options are: `view`, `create`, `str_replace`, `insert`, `undo_edit`."
        required: true
        enum: ["view", "create", "str_replace", "insert", "undo_edit"]
      - name: path
        type: string
        description: "Absolute path to file or directory, e.g. `/testbed/file.py` or `/testbed`."
        required: true
      - name: file_text
        type: string
        description: "Required parameter of `create` command, with the content of the file to be created."
        required: false
        argument_format: "--file_text {{value}}"
      - name: old_str
        type: string
        description: "Required parameter of `str_replace` command containing the string in `path` to replace."
        required: false
        argument_format: "--old_str {{value}}"
      - name: new_str
        type: string
        description: "Optional parameter of `str_replace` command containing the new string (if not given, no string will be added). Required parameter of `insert` command"
        required: false
        argument_format: "--new_str {{value}}"
      - name: insert_line
```

Train SWE-agent with verl: rollout

1. AgentLoop Abstraction



AgentLoop: given a user prompt, execute user defined loop, output multi-turn chat history as trajectory.

- **Search**: online web search
- **MCP**: image, video edit, ...
- **Code sandbox**: execute code, python, java, ...
- **Virtual machine**: operate browser, ppt, excel, ...
- **Android emulator**: operate app
- ...

```python
class AgentLoopBase(ABC):
    @abstractmethod
    async def run(
        self,
        messages: list[dict[str, Any]],
        sampling_params: dict[str, Any],
    ) -> AgentLoopOutput:
        """Run agent loop to interact with LLM server and environment.

        Args:
            messages (List[Dict[str, Any]]): Input messages.
            sampling_params (Dict[str, Any]): LLM sampling params.

        Returns:
            AgentLoopOutput: Agent loop output.
        """
        raise NotImplementedError
```

Train SWE-agent with verl: rollout

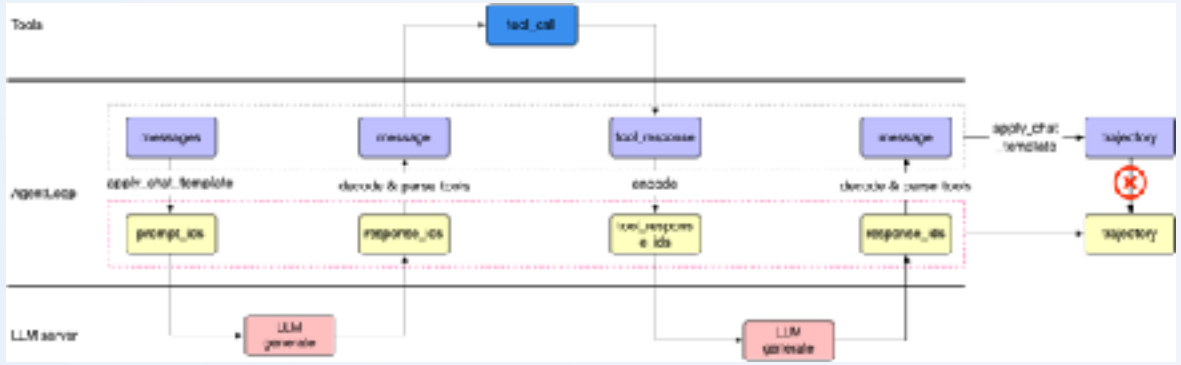
2. SWEAgentLoop

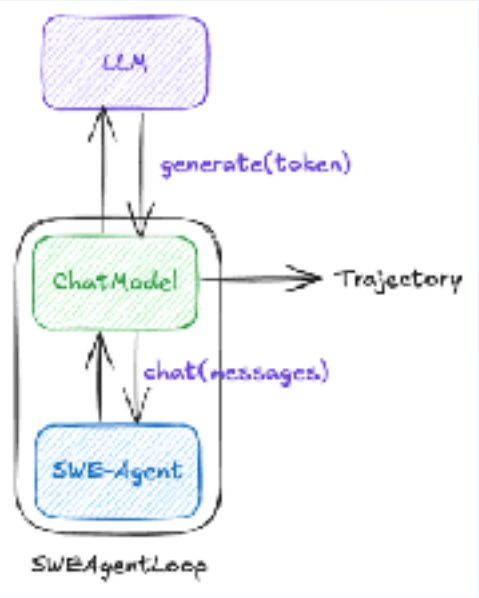
```python
class SWEAgentLoop(AgentLoopBase):
    async def run(self, sampling_params: dict[str, Any], **kwargs) -> AgentLoopOutput:
        tool_config_data = agent_config_data.get("tools", {})
        tool_config = ToolConfig(...)

        default_agent_config = DefaultAgentConfig(...)

        deployment_config = VefaasDeploymentConfig(...)

        env_config = EnvironmentConfig(...)
        env_config.deployment = deployment_config

        problem_statement = TextProblemStatement(...)

        single_run_config = RunSingleConfig(...)

        single_run = ReRunSingle.from_config(single_run_config)

        # replace chat model and agent.
        model = ChatModel(...)
        agent = SweAgent.from_config(default_agent_config, model=model)

        try:
            result = await agent.run(single_run)
            output_dir = single_run.output_dir
            patch_output_dir = Path(output_dir) / instance_id
            patch_output_file = patch_output_dir / f"{instance_id}.pred"

            test_spec = make_test_spec(metadata)
            with open(patch_output_file,"r") as f:
                pred = json.load(f)

            vefaas_config = VefaasDeploymentConfig(...)
            deployment = VefaasDeployment.from_config(vefaas_config)

            eval_result = await run_eval(...)
            print(eval_result)

            return model.convert_to_agent_output(result, eval_result)
```

- SWE-agent instance
- Run specific task: image, commit, problem, …
- Run tests with patch to get reward score
- Convert messages to AgentLoopOutput

Train SWE-agent with verl: rollout

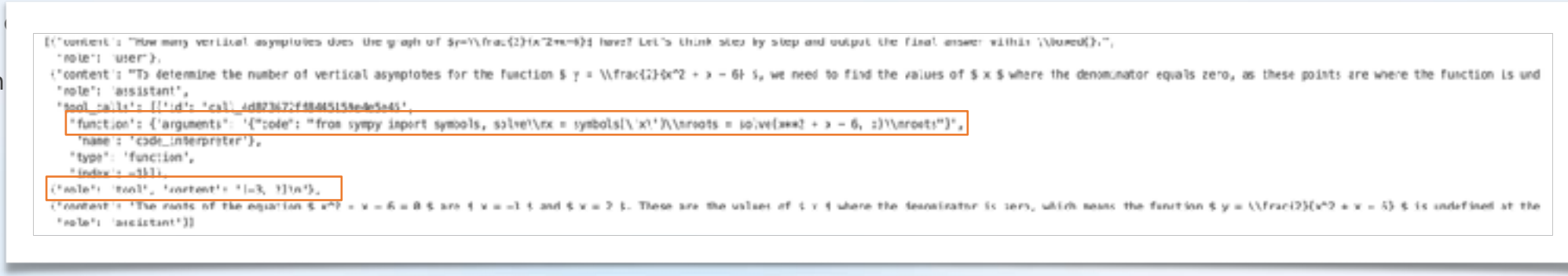
3. Bridge the gap between chat completion and token-in-token-out





ChatModel

- Encode messages and tools to tokens in request
- Decode tokens and
- Keep chat history in



Train SWE-agent with verl: training

Reinforcement Learning with Verifiable Reward(RLVR)

- reward: model generated patch pass all test cases
- algorithm: PPO or GRPO/DAPO/GSPO

Group relative advantage estimation:

$$A_i = \frac{r_i - \text{mean}(\{r_1, r_2, \cdots, r_G\})}{\text{std}(\{r_1, r_2, \cdots, r_G\})}.$$

We will release the training recipe soon!

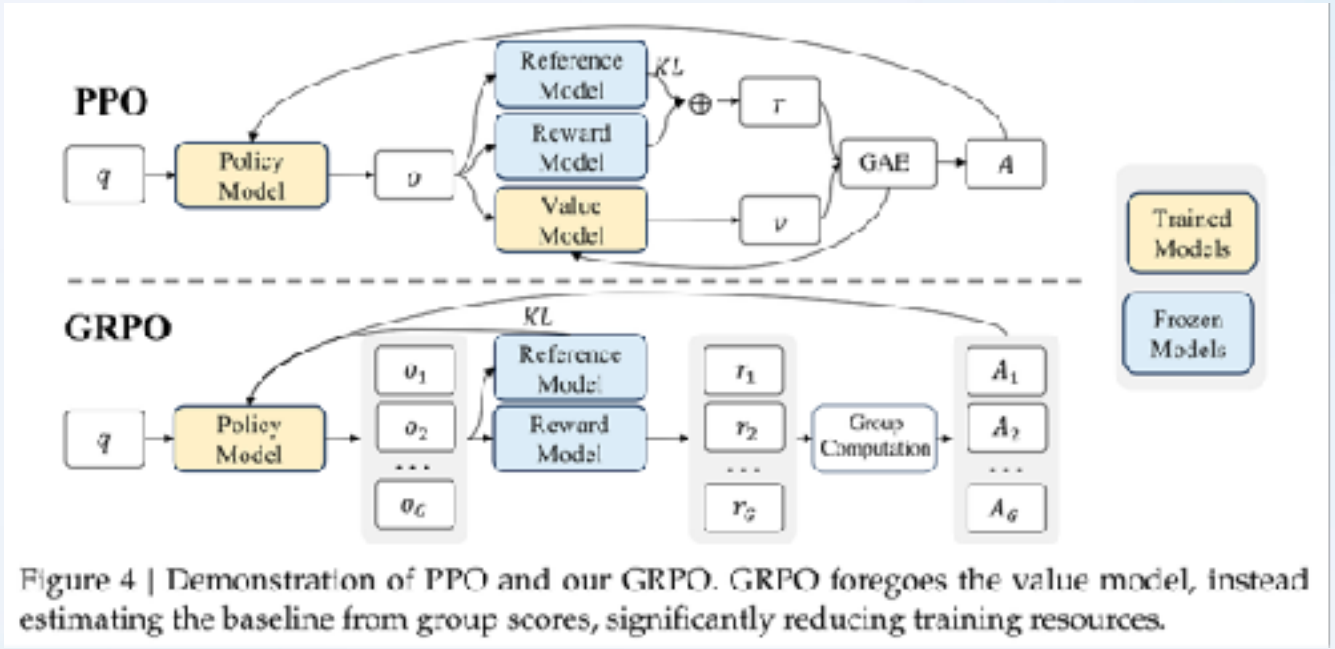


Figure 4 | Demonstration of PPO and our GRPO. GRPO foregoes the value model, instead estimating the baseline from group scores, significantly reducing training resources.

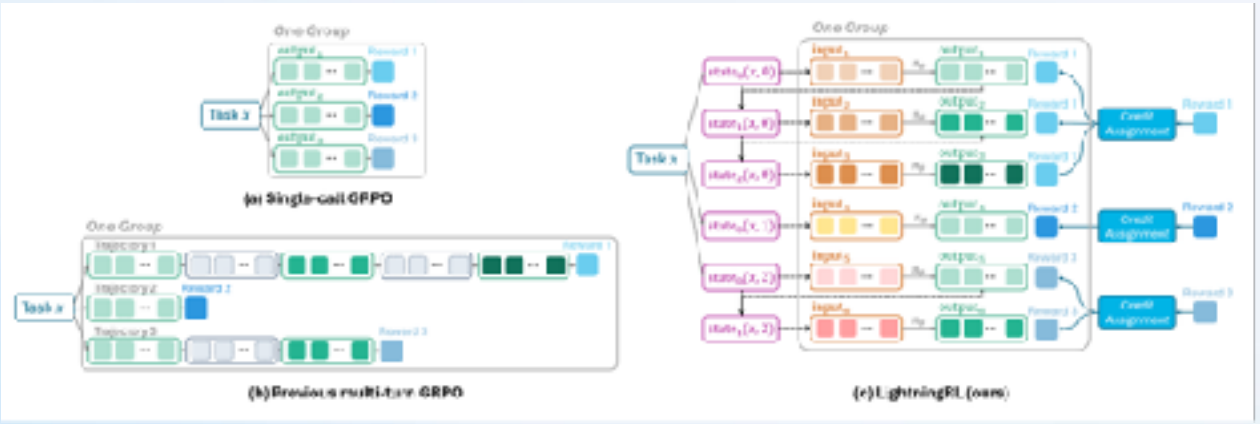
DeepSeekMath, Zhihong Shao, et al, 2024

Limitation and Future Work

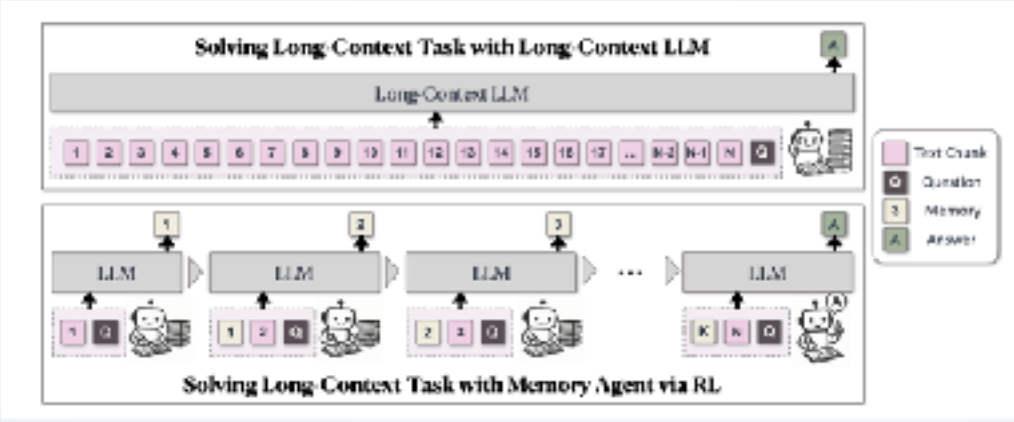
Limitation: AgentLoop chat history append only

- How to compress context?
- How to handle extreme long-context?

Some awesome works based on verl!



microsoft/agent-lightning



BytedTsinghua-SIA/MemAgent

QCon
全球软件开发大会

InfoQ 极客传媒

THANKS

大模型正在重新定义软件

Large Language Model Is Redefining The Software