Iterative Tool Usage Exploration for Multimodal Agents via Step-wise Preference Tuning

Pengxiang Li[1,2*], Zhi Gao[2,3*], Bofei Zhang[2], Yapeng Mi[2], Xiaojian Ma[2], Chenrui Shi[1,2], Tao Yuan[2], Yuwei Wu[1,4], Yunde Jia[4,1], Song-Chun Zhu[2,3,5], Qing Li[2]

[1] Beijing Institute of Technology [2] Beijing Institute of General Artificial Intelligence (BIGAI) [3] Peking University [4] Shenzhen MSU-BIT University [5] Tsinghua University

WeChat

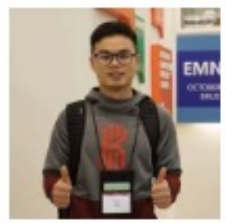




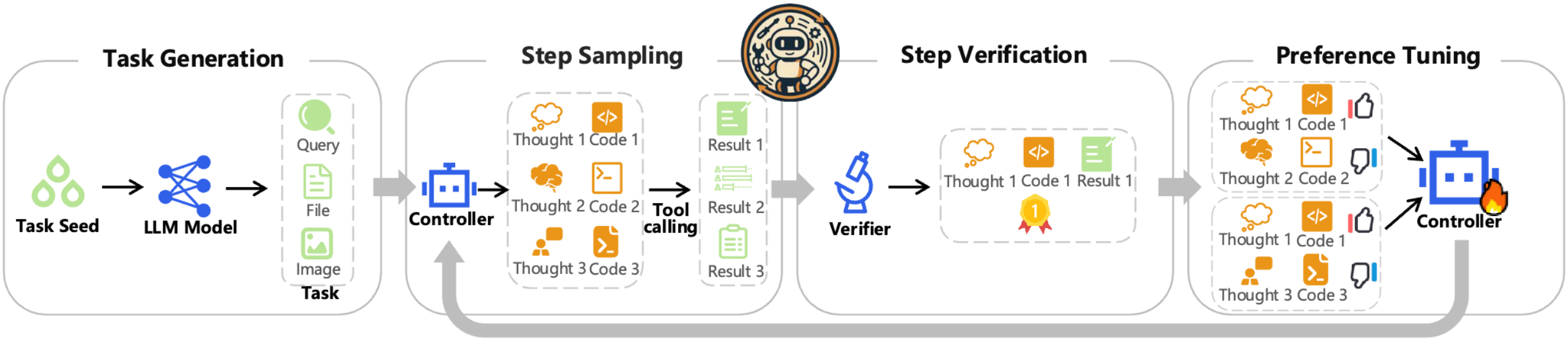






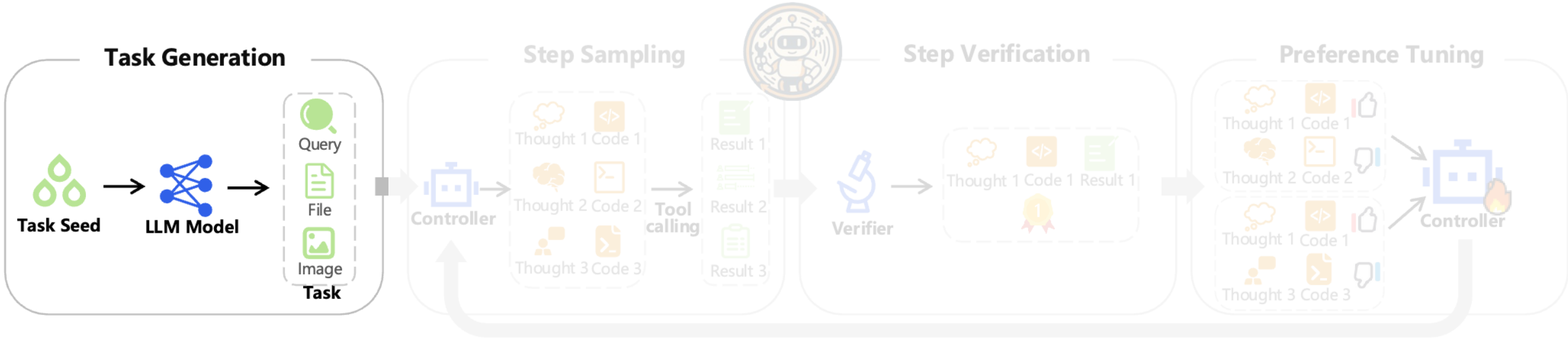


Tool Usage Exploration



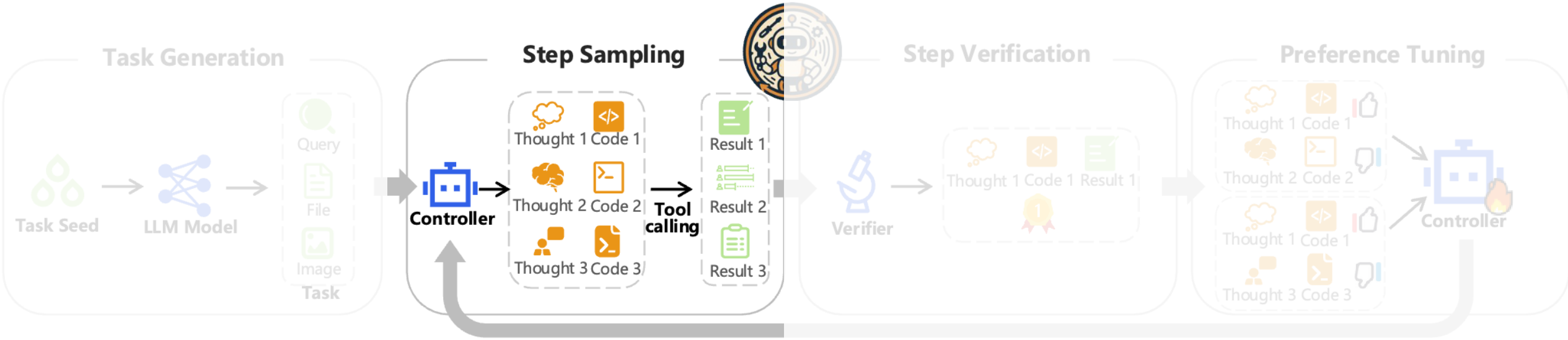
This paper introduces an online self-exploration loop that enables multimodal agents to self-improve via AI-generated tasks and LLM-verified preference tuning **without human annotations**.

Tool Usage Exploration



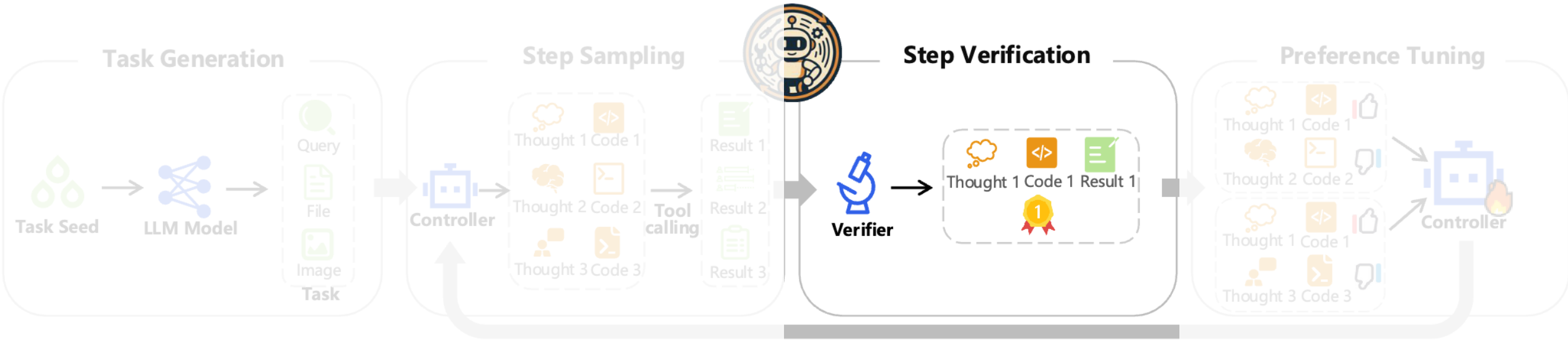
Task generalization: Use some seed to generate tasks.

Tool Usage Exploration



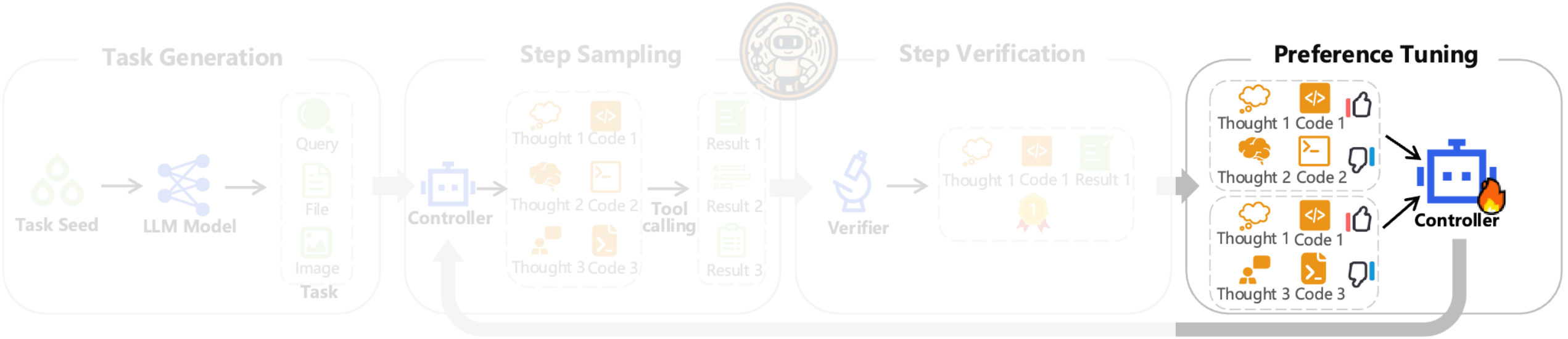
Step Sampling: Sampling possible solutions in one step.

Tool Usage Exploration



Step Verification: Use AI feedback to rank the sampled solutions.

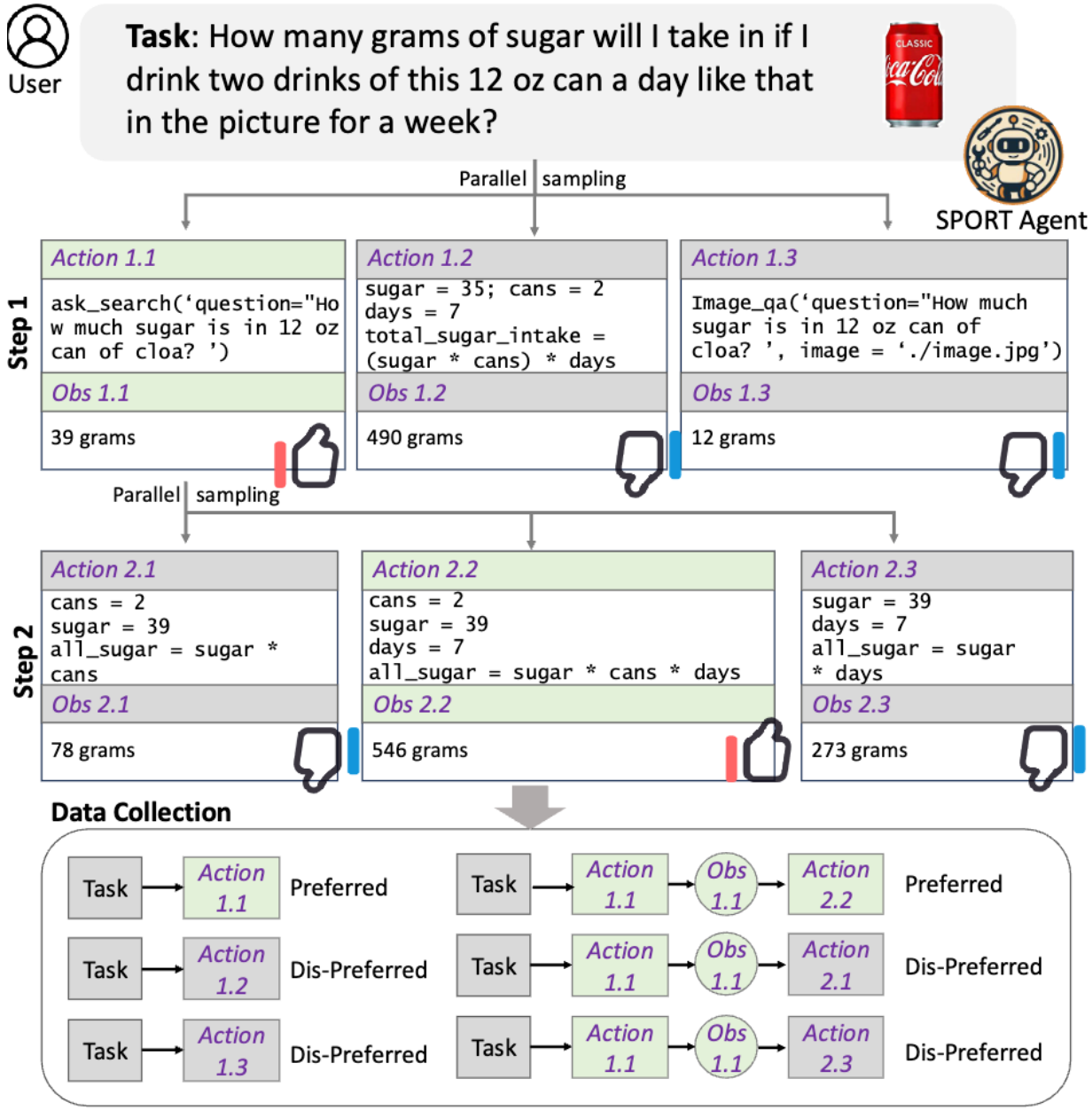
Tool Usage Exploration



Preference Tuning: Perform step-wise DPO

$$\mathcal{L}(\theta) = -\mathbb{E}_{(x_i, a_i^{pre}, a_i^{dis}))\sim\mathcal{D}}[\log\sigma(\beta\log\frac{\pi_\theta(a_i^{pre}|x_i)}{\pi_{ref}(a_i^{pre}|x_i)} - \beta\log\frac{\pi_\theta(a_i^{dis}|x_i)}{\pi_{ref}(a_i^{dis}|x_i)})],$$

Step-wise Preference Data



Results on GTA benchmark

Method	Controller	GTA			GAIA			
		ToolAcc	*CodeExec*	*AnsAcc*	*Level 1*	*Level 2*	*Level 3*	*AnsAcc*
Closed-source Controller								
T3-Agent	MAT-MiniCPM-V-8.5B	65.85	80.49	52.56	26.42	11.63	**3.84**	15.15
T3-Agent	MAT-Qwen2-VL-7B	64.63	84.32	53.85	26.42	15.12	**3.84**	16.97
Ours								
SPORT Agent	Tuned-Qwen2-VL-7B	**72.41**	**91.87**	**60.26**	**35.85**	**16.28**	**3.84**	**20.61**

Compared with SFT agents. Higher ***AnsAcc (6.41%⬆️)*** on GTA, Higher ***AnsAcc (3.64%⬆️)*** on GAIA.

Results on GTA and GAIA benchmark

Method	Controller	GTA			GAIA			
		ToolAcc	*CodeExec*	*AnsAcc*	*Level 1*	*Level 2*	*Level 3*	*AnsAcc*
Closed-source Controller								
Lego Agent	GPT-4	-	-	46.59	-	-	-	-
Lego Agent	GPT-4o	-	-	41.52	-	-	-	-
Warm-up Agent	GPT-4-turbo	-	-	-	30.20	15.10	0.00	17.60
HF Agent	GPT-4o	63.41	95.12	57.05	47.17	31.40	11.54	33.40
HF Agent	GPT-4o-mini	56.10	100.00	57.69	33.96	27.91	3.84	26.06
Open-Source Controller								
HF Agent	LLaVA-NeXT-8B	14.97	25.08	14.10	9.43	1.16	0.00	3.64
HF Agent	InternVL2-8B	36.75	52.18	32.05	7.55	4.65	0.00	4.85
HF Agent	MiniCPM-V-8.5B	36.59	56.10	33.97	13.21	5.81	0.00	7.27
HF Agent	Qwen2-VL-7B	44.85	65.19	42.31	16.98	8.14	0.00	9.70
T3-Agent	MAT-MiniCPM-V-8.5B	65.85	80.49	52.56	26.42	11.63	**3.84**	15.15
T3-Agent	MAT-Qwen2-VL-7B	64.63	84.32	53.85	26.42	15.12	**3.84**	16.97
Ours								
SPORT Agent	Tuned-Qwen2-VL-7B	**72.41**	**91.87**	**60.26**	**35.85**	**16.28**	**3.84**	**20.61**

Examples

Query: Find out the latest reviews for the smartphone featured in this image. Include comparisons with its competitors. Attachment: “image.jpg”

Attached Image:

Parallel Sampling

Step1

Thought:
I need to find the latest reviews for the smartphone displayed in the image. Since the image contains the text “Zenfone Max Plus,” I will use the `ask_search_agent` tool to gather recent reviews for this model and compare it with its competitors.

Code:
```
pyreviews = ask_search_agent(query=“Latest reviews for Zenfone Max Plus and comparison with competitors")
```

Sample 1

Thought:
I'll use OCR to extract any visible text that might contain clues about the phone's name or brand. Then, I will search the web for reviews on that specific model.

Code:
```
from PIL import Image
import pytesseract import io
 # Load the image for OCR processing
Img=Image.open(io.BytesIO(self.get Attachment(”image.jpg")))
 text = pytesseract.image_to_string(img)
# Search for reviews of the extracted phone model
review_search = ask_search_agent(f"Find the latest reviews for {text} smartphone.")
```

Sample 2

Thought:
To find the latest reviews for the smartphone featured in the image, I need to use the `ask_search_agent` tool to query the latest reviews and specifications about the smartphone model, as well as its competitors' ratings and specifications.

Code:
```
ask_search_agent(query="Lat est reviews for the smartphone in the image", image_path=”image.jpg")
competitors = ask_search_agent(query="Competitors of the smartphone in the image and their ratings", image_path=”image.jpg")
```

Sample 3

Verification Reason:
The positive solution can **correctly identify the brand of the smartphone**, and get some review from the Internet. In contrast, the two negative solutions **cannot correctly identify the brand of the smartphone**. OCR cannot obtain the brand, and the ‘ask_search_agent’ tool cannot process images.

Parallel Sampling

Step2

Thought:
I will now look for more current reviews comparing it with its competitors like Nokia 6.1 Plus and Redmi Note 6 Pro.

Code:
```
ask_search_agent(query="Current reviews for Zenfone Max Plus compared to Nokia 6.1 Plus and Redmi Note 6 Pro")
print(current_reviews)
```

Sample 1

Thought:
The image and search results have led me to understand that the Zenfone Max Plus was a popular mid-range smartphone. To better inform you, I will compare it with some mid-range smartphone competitors.

Code:
```
ask_search_agent(query="Latest smartphone reviews in the mid-range price bracket for performance and battery life")
```

Sample 2

Thought:
To provide a more comprehensive comparison, I'll gather more recent reviews for these competing models and also look for more recent reviews on newer smartphones in the same price bracket.

Code:
```
ask_search_agent(query="Reviews for newer smartphones in the same price bracket")
```

Sample 3

Verification Reason:
The positive solution search for information about **mid-range competitors**, while the negative solutions only compare Zenfone Max Plus with **a special smartphone, or does not mention the Zenfone Max Plus smartphone.**