AGENTBANK: Towards Generalized LLM Agents via Fine-Tuning on 50000+ Interaction Trajectories

Anonymous ACL submission

Abstract

Fine-tuning on agent-environment interaction trajectory data holds significant promise for surfacing generalized agent capabilities in open-source large language models (LLMs). In this work, we introduce AGENTBANK, by far the largest trajectory tuning data collection featuring more than 50k diverse high-quality interaction trajectories which comprises 16 tasks covering five distinct agent skill dimensions. Leveraging a novel annotation pipeline, we are able to scale the annotated trajectories and generate a trajectory dataset with minimized difficulty bias. Furthermore, we fine-tune LLMs on AGENTBANK to get a series of agent models, SAMOYED. Our comparative experiments demonstrate the effectiveness of scaling the interaction trajectory data to acquire generalized agent capabilities. Additional studies also reveal some key observations regarding trajectory tuning and agent skill generalization.[1]

1 Introduction

An agent is an entity that possesses the capability for volition, decision-making, action-taking , and, most critically, environment perception (Jennings et al., 1998). In the realm of cognitive science, previous literature has suggested that interaction with environment derives an agent's generalized intelligence, and intelligent behavior emerges from a synergistic blend of simpler behaviors, including reasoning, programming, and game playing (Brooks, 1991). The proprietary large language models (LLMs), such as GPT-3.5 (OpenAI, 2022) and GPT-4 (OpenAI, 2023), have demonstrated strong capabilities in instruction following, reasoning, and planning, which encourage many attempts to build autonomous agent systems utilizing LLMs as core controllers (Richards, 2023; Nakajima, 2023). However, comprehensive evaluations have shown that the majority of open-sourced LLMs fall short in agent capabilities when compared with GPTs (Liu et al., 2023; Wang et al., 2023).

Previous research pointed out that learning from gold interaction trajectories, a process we term **Trajectory Tuning**, could enhance the capabilities of weaker agents (Brooks, 1991; Hussein et al., 2017). Early studies heavily focus on specialized agents designed for particular tasks. Existing attempts are exemplified by Chen et al. (2023a) and Yin et al. (2023), who build agent trajectory data from teacher agents (*e.g.*, GPT-4) and fine-tune open-source LLMs to improve specific agent abilities like reasoning. Taking a step further, Zeng et al. (2023) adopt a multi-task tuning approach called AgentTuning. However, trained on a small trajectory dataset comprising six tasks with 1.8k trajectories, Zeng et al. (2023) struggle to enhance the generalized agent capability, especially in the case of 7B and 13B models.

To explore the impacts of incorporating interaction trajectory data on agent ability generalization, we construct AGENTBANK, the largest agent interaction trajectory dataset to date. AGENTBANK features 16 distinct tasks across five agent skill dimensions and contains over 50,000 trajectories, each annotated with high-quality chain-of-thought (CoT) rationale for every step of action. Leveraging a novel annotation pipeline that fully exploits the capability of LLMs, the trajectory collection process is highly scalable and adaptable to diverse agent environments. In contrast to prior studies that have relied on successful trajectories of GPTs for training data (Chen et al., 2023a; Zeng et al., 2023), AGENTBANK stands out with its exceptional quality and mitigated susceptibility to the *difficulty bias* issue.

We further develop SAMOYED, a suite of models with enhanced agent capabilities, through the trajectory tuning of Llama-2 (Touvron et al., 2023) using AGENTBANK. Our evaluations on both held-in and

[1] The AGENTBANK dataset and evaluation framework are released at anonymous link

	AGENTBANK (this work)	FireAct (Chen et al., 2023a)	AgentInstruct (Zeng et al., 2023)	Agent-FLAN (Chen et al., 2024)	AgentOhana (Zhang et al., 2024)
Number of tasks	**16**	3	6	7	10
Number of trajectories	**51287**	1344	1866	24703	42600
Average interaction turns	3.9	-	**5.2**	3.7	3.1
No difficulty bias?	✓	✗	✗	✗	✗
Open-sourced?	✓	✓	✓	✓	✗
Reasoning	✓	✓	✗	✗	✓
Math	✓	✗	✗	✗	✗
Programming	✓	✗	✓	✓	✓
Web	✓	✗	✓	✓	✓
Embodied AI	✓	✗	✓	✓	✓

Table 1: A comparison of AGENTBANK with other datasets for agent trajectory tuning.

unseen held-out tasks suggest that by fine-tuning on extensive multi-task trajectories, our models exhibit remarkable agent intelligence in comparison with untuned ones. Specifically, SAMOYED outperforms GPT-3.5-Turbo on average on held-in tasks, which can be attributed to the in-domain trajectory tuning. Furthermore, our models also demonstrate superior performance on held-out tasks, underscoring the efficacy of large-scale trajectory tuning in acquiring generalized agent capabilities.

To trace the emergence of agent capabilities generalization, we follow the initial evaluation with a systematic analysis across various dimensions. Initially, we delineate the scaling trends of tasks alongside the quantity of trajectories. Next, we conduct an ablation study that merges generalist instruction data and code data to examine the benefits of hybrid training. This study uncovers further enhancements in the agent capabilities and mitigates catastrophic forgetting. Furthermore, our findings underscore the pivotal role of CoT rationale in the acquisition of generalized agent capability.

Our contributions are summarized as follows:

- The release of AGENTBANK, a dataset of over 50,000 high-quality agent interaction trajectories, spanning 16 tasks across five skill dimensions. We also present a novel annotation pipeline, offering scalability and a marked reduction in difficulty bias, surpassing previous methods.

- The development of SAMOYED, the most powerful open-source LLM suite at the 7B/13B scale optimized for agent tasks. Trained through trajectory tuning, SAMOYED demonstrates exceptional performance, showcasing transferable agent intelligence on unseen tasks.

- We conduct comprehensive experiments and in-depth analysis on agent intelligence acquisition, including the relations with instruction following and code capability, scaling law of interaction trajectories, and the effectiveness of training with CoT.

2 Related Work

2.1 Instruction Tuning

Instruction tuning is a simple yet powerful approach to align LLMs with human preferences (Zhang et al., 2023). Previous studies have primarily focus on improving general-purpose instruction following capabilities of LLMs. FLAN series (Wei et al., 2021; Chung et al., 2022), T0 (Sanh et al., 2021), and NaturalInstruction (Wang et al., 2022b) scale up the instruction datasets to activate the generalized instruction following capabilities of LLMs. More recently, utilizing synthetic instruction following data distilled from GPTs to align open-source LLMs has also been proposed (Taori et al., 2023; Chiang et al., 2023). Furthermore, multiple works have shown the promise of instruction tuning in enhancing the specialized abilities of LLMs, such as math (Yu et al., 2023; Yue et al., 2023), reasoning (Lee et al., 2023), and agent tasks (Chen et al., 2023a; Zeng et al., 2023).

2.2 LLM-based Agent

Modern LLMs have demonstrated various emergent abilities that encourage researchers to build agent systems based on LLMs. ReAct (Yao et al., 2022b) combines CoT reasoning with agent actions to accomplish tasks such as QA. AutoGPT (Richards, 2023) harnesses LLMs as the core controllers to constitute powerful agent frameworks capable of solving real-world complex problems. While advanced proprietary models exampled by GPT-3.5/4 have shown strong performances on agent tasks, their open-source counterparts still lag

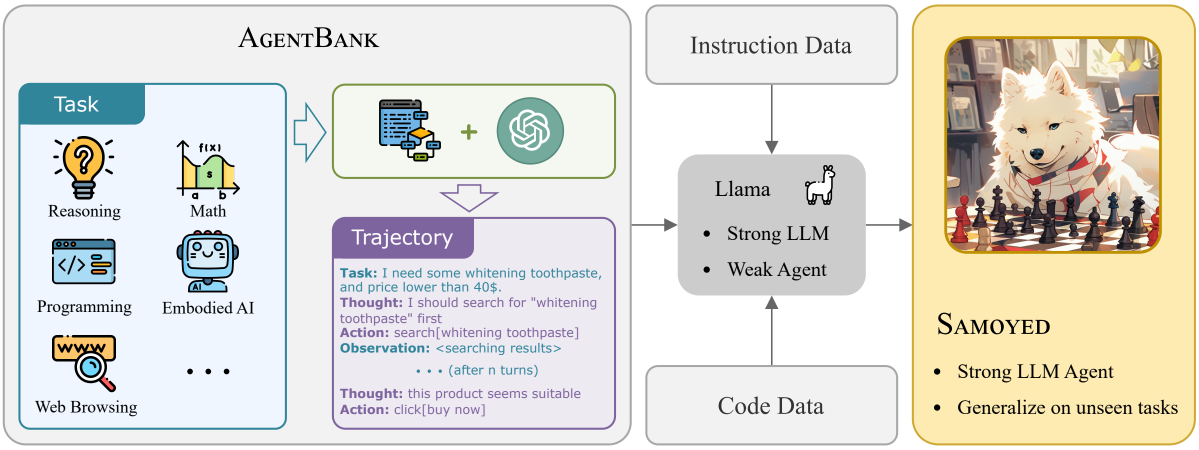


Figure 1: Overview of the construction process of AgentBank and the training procedure of Samoyed

far behind (Liu et al., 2023; Wang et al., 2023). In response, recent studies including FireAct (Chen et al., 2023a), AgentTuning (Zeng et al., 2023) and AgentOhana (Zhang et al., 2024) collect agent trajectory data from teacher agents (*e.g.*, GPT-4) and fine-tune open-source LLMs (*e.g.*, Llama series) with the data. However, limited by the number of tasks and expert trajectories, existing research has not yet exhaustively explored whether open-source LLMs can acquire generalized agent abilities, a gap that this study aims to bridge.

3 Preliminary

3.1 Agent Task Formulation

Given an agent task described by the instruction u, an LLM agent generates an action a_1 based on its policy. Next, an environment receives the action, transfers to a new latent state, and provides an observation o_i in natural language format. Subsequently, the agent generates another action for the next step, a_{i+1}, and repeats this circle of interaction with the task environment until either the task is completed or the maximum number of steps is reached. This "conversation" between the agent with the environment is denoted as the interaction trajectory $(u, a_1, o_1, ..., a_n)$. Finally, a final reward $r \in [0, 1]$ is returned depending on the task completion status.

Chain-of-Thought (CoT) (Wei et al., 2022; Kojima et al., 2022) is an effective approach to enhance the inferential capabilities of LLMs by a step-by-step reasoning process. We employ ReAct (Yao et al., 2022b) as the agent tasking framework, which outputs rationale before the action.

3.2 Challenges in Trajectory Collection

Previous works (Chen et al., 2023a; Zeng et al., 2023) have employed GPT-4 as teacher agents to interact with the environment and collect successful interaction trajectories. To ensure the quality of generated data, a failure filtering mechanism is used to remove the cases where GPT failed. However, this **GPT-exploration** pipeline automates the trajectory construction at some significant cost.

Hard to Scale-Up The quality of data is essential for agent training, and training with failure trajectories will lead to performance degradation (Zeng et al., 2023). Therefore, scaling up this process to a larger trajectory amount is challenging due to the low success rate of GPT-4. For instance, AgentInstruct (Zeng et al., 2023) discards more than 90% generated trajectories due to GPT failures.

Difficulty Bias Even worse, GPT-exploration pipelines will inevitably introduce difficulty bias to the final training data. Essentially, a trajectory filtering strategy can be regarded as grouping the instances based on whether GPT is capable of solving them. Discarding failed trajectories leads to a skewed distribution of "difficulty", resulting in a training set with much easier instances than those in the test set. This violation of the *i.i.d.* assumption may hurt the generalization ability of the trained agents. In Appendix B, we conduct an experiment to show this bias.

4 AgentBank

In response to the challenges of previous trajectory collection pipeline, we propose a new trajectory annotation pipeline and construct AgentBank trajectory dataset.

Skill Dim.	Task	Action Space	Tool	#Inst.	Avg. Turns	Action Annotation
Reasoning	HotpotQA (Yang et al., 2018)	Continuous	Search	4273	3.1	Explore
	StrategyQA (Geva et al., 2021)	Continuous	Search	1267	3.6	Explore
	TriviaQA (Joshi et al., 2017)	Continuous	Search	4134	2.5	Explore
Math	GSM8K (Cobbe et al., 2021)	Continuous	Calculator	7471	4.5	Reformat
	MathQA (Amini et al., 2019)	Continuous	Python	4000	2.0	Explore
	MATH (Hendrycks et al., 2021)	Continuous	Python, Wiki	2312	2.5	Explore
Programming	IC-SQL (Yang et al., 2023)	Continuous	MySQL	4540	4.8	Explore+Answer Force
	APPS (Hendrycks et al., 2021)	Continuous	Python	4408	1.0	Reformat
	HumanEval (Chen et al., 2021)	Continuous	Python	134	2.7	Explore+Answer Force
	MBPP (Austin et al., 2021)	Continuous	Python	608	2.2	Explore+Answer Force
Web	Mind2Web (Deng et al., 2023)	Discrete	-	7770	1.0	Reformat
	WebArena (Zhou et al., 2023)	Discrete	-	657	1.0	Reformat
	WebShop (Yao et al., 2022a)	Discrete	-	5315	3.4	Explore & Reformat
Embodied	ALFWorld (Shridhar et al., 2020b)	Discrete	-	3554	10.1	Reformat
	RoomR (Weihs et al., 2021)	Discrete	-	300	30.2	Search+Reformat
	IQA (Gordon et al., 2018)	Discrete	-	1627	28.4	Search+Reformat
	Total (AGENTBANK)	-	-	51287	3.9	-

Table 2: Overview of AGENTBANK dataset. It compiles 16 agent tasks covering 5 skill dimensions, formulating the largest interaction trajectory dataset. "Inst." and "Traj." refer to instruction and interaction trajectory.

4.1 Task and Instruction Collection

A generalized agent needs to possess a wide range of capabilities across various dimensions. To this end, as shown in Table 2, we curate 16 publicly available agent datasets to lay the foundation of AGENTBANK and categorize specific tasks into five skill dimensions: reasoning, math, programming, web navigation, and embodied tasks. Additionally, some tasks aggregated in AGENTBANK involve the usage of external tools, such as search engine, calculator, and code interpreter, as the ability to effectively operate tools is also a crucial aspect for generalized agents. From the perspective of action space, tasks in AGENTBANK can be classified into two types: those with a *continuous* action space (including natural language and code) and those with a predefined *discrete* action space. Our dataset also covers a broad range of interaction turns, ranging from 1 to 30. Note that some tasks are originally evaluated in a single-turn QA style, such as HotpotQA (Yang et al., 2018) and MATH (Hendrycks et al., 2021). Following Wang et al. (2023), we modify these datasets to accommodate multi-turn interaction environments with tool usage.

Since most of the original benchmarks have a training set, we use them to construct our dataset. To balance data sources, we down-sample some tasks which have a huge training set. See Appendix A for detailed descriptions of each dataset.

4.2 Action Annotation

To tackle the challenges in trajectory collection, unlike previous methods that generate action and CoT simultaneously, we separate the annotation of gold actions and their corresponding rationales, fully leveraging of the capability of LLMs.

Specifically tailored to the specific nature of different tasks, our approach involves several techniques to obtain high-quality action sequences accordingly.

Answer Forcing For tasks characterized by a continuous natural language or code action space, such as IC-SQL, we introduce an answer forcing action annotation strategy as an extension to GPT-exploration pipeline. This strategy aims to mitigate the bias introduced by failure filtering. Initially, we use GPT-4 to interact with the environment and gather interaction trajectories. For failed trajectories, rather than directly discarding them, we prompt GPT with the failed trajectory and the gold final answer to generate a new interaction trajectory. Then we validate the correctness of new trajectories by executing the actions within real agent environments. This answer forcing process is used in an iterative manner to re-annotate failure trajectories and generate a substantial number of gold action sequences. See Appendix C for the re-annotation prompt.

Heuristic Action Search For tasks with a discrete action space, exemplified by embodied AI tasks (Weihs et al., 2021; Gordon et al., 2018), we are able to access both the environment's source code and its complete execution state. Leveraging this access, we employ the heuristic depth-first search algorithm to efficiently get the optimal action sequences.

Reformat Some tasks have already provided official solving trajectories. For instance, GSM8K (Cobbe et al., 2021) offers ground-truth intermediate reasoning steps. For these tasks, following (Yin et al., 2023), we exploit GPT as a style transfer tool to transform reasoning process into agent interaction action sequences.

4.3 Rationale Annotation

Give the instructions and gold action sequences, we directly prompt GPT to generate the corresponding CoT rationale of each action step. Since providing explanation for gold actions is relatively easy task, we employ GPT-3.5-Turbo as the primary LLM in the rationale annotation process. The rationale generation prompt is shown in Appendix C. We also compare rationales generated by different LLMs in Appendix D.

For tasks with a huge number of instructions and GPT-4 have a high success rate, such as StrategyQA (Geva et al., 2021) and WebShop (Yao et al., 2022a), we directly use the GPT-exploration pipeline as Zeng et al. (2023).

The overview of AGENTBANK is shown in Table 2. See the Appendix A for more details about the annotation process of each task. A human evaluation assessing the quality of our dataset can be found in Appendix E.

5 Train SAMOYED with AGENTBANK

To initialize the training of SAMOYED, we formulate agent interaction trajectories in AGENTBANK into a chatbot-style schema $(u, a_1, o_1 ..., a_i, o_i, ..., a_n)$, where u is the task instruction, o_i and a_i denote the observation from the task environment and the corresponding action with rationale generated by the agent in the i-th round. During the training process, we feed the entire interaction trajectory into a decoder-only LLM, where only the auto-regressive loss on tokens of ground-truth responses $Y = \{a_1, ..., a_n\}$ is counted. We mask all tokens belonging to the instruction and observations from the environment to prevent them from loss computation. Concretely, the loss function is defined as:

$$\mathcal{L} = -\sum_j \log p_\theta(t_j | t_{<j}) \times \mathbf{1}(t_j \in Y), \qquad (1)$$

where t_j denotes the j-th input token and $\mathbf{1}$ is the indicator function.

Recent studies (Yang et al., 2024; Zeng et al., 2023) suggest that hybrid training with generalist instruction data and code data may improve the generalized ability of LLM agents. Following them, we adopt a mixture of AGENTBANK $\mathcal{D}_{\text{agent}}$, the general domain instruction dataset $\mathcal{D}_{\text{general}}$, and the code dataset $\mathcal{D}_{\text{code}}$ for fine-tuning. We perform detailed ablation experiments to explore the effectiveness of generalist and code data in Section 7.2.

Task	Skill Dim.	#Inst.	Metric
Held-in Tasks			
HotpotQA (Yang et al., 2018)	Reasoning	100	Exact Match
StrategyQA (Geva et al., 2021)	Reasoning	100	Exact Match
GSM8K (Cobbe et al., 2021)	Math	100	Exact Match
MATH (Hendrycks et al., 2021)	Math	100	Exact Match
IC-SQL (Yang et al., 2023)	Programming	100	Avg. Reward
MBPP (Austin et al., 2021)	Programming	100	Success Rate
Mind2Web (Deng et al., 2023)	Web	1173	Step SR
WebShop (Yao et al., 2022a)	Web	200	Avg. Reward
ALFWorld (Shridhar et al., 2020b)	Embodied	134	Success Rate
Held-out Tasks			
Bamboogle (Press et al., 2022)	Reasoning	126	Exact Match
TheoremQA (Chen et al., 2023b)	Math	100	Exact Match
IC-Bash (Yang et al., 2023)	Programming	200	Avg. Reward
MiniWoB++ (Kim et al., 2023)	Web	460	Success Rate
ScienceWorld (Wang et al., 2022a)	Embodied	270	Avg. Reward

Table 3: The held-in and held-out tasks used to evaluate the agent capabilities of different LLMs.

6 Experiments

6.1 Experimental Setup

Base LLMs and Baselines We use several LLMs to conduct experiments, including Llama-2-Chat (Touvron et al., 2023), CodeLlama (Roziere et al., 2023), Mistral (Jiang et al., 2023), and Llama-3-Instruct (Meta, 2024). However, since most baselines, including AgentLM (Zeng et al., 2023) and Agent-FLAN (Chen et al., 2024) are tuned from Llama-2-Chat, we mainly use Llama-2-Chat as our base model for a fair comparison. Due to our limited resources, we use 7B and 13B models for our experiments, leaving the comparison at a larger scale (*e.g.*, Lemur-70B (Xu et al., 2023b) and xLAM-8×7B (Zhang et al., 2024)) for the future work. We also select GPT-3.5-Turbo (OpenAI, 2022) and GPT-4 (OpenAI, 2023) as strong baselines. For all LLMs, the decoding temperature is set to 0 for the most deterministic generation.

Model	Held-in Tasks						Held-out Tasks					
	Reason	Math	Program	Web	Embodied	Avg.	Reason	Math	Program	Web	Embodied	Avg.
Closed-Source Model												
GPT-4	61.6	73.0	54.9	40.6	77.8	59.8	41.6	51.0	69.4	69.4	36.4	53.6
GPT-3.5-Turbo	41.0	41.5	51.2	42.0	10.5	40.2	32.0	32.0	54.8	66.7	21.2	41.3
7B Open-Source Model												
Llama-2-7B-Chat	4.0	7.5	2.5	13.9	0.0	6.2	4.0	8.0	7.0	0.4	7.8	5.5
Vicuna-7B	29.0	2.0	19.0	24.2	6.0	17.1	8.8	14.0	19.0	18.2	12.8	14.6
CodeLlama-7B	3.5	3.5	1.5	24.8	0.0	7.4	1.0	13.0	21.8	**41.3**	5.5	16.5
AgentLM-7B	29.5	10.0	12.0	**37.2**	63.4	26.7	19.2	13.0	50.5	13.5	13.3	21.9
Agent-FLAN-7B	31.0	10.5	13.1	35.4	**65.3**	27.3	22.2	11.0	53.1	17.9	14.1	23.7
SAMOYED-7B	**48.0**	**30.5**	**41.6**	36.4	61.2	**41.6**	**32.0**	**18.0**	**59.2**	24.2	**14.2**	**29.5**
13B Open-Source Model												
Llama-2-13B-Chat	12.5	10.5	8.2	11.2	0.0	9.4	9.6	11.0	33.0	17.6	7.3	15.7
Vicuna-13B	25.5	6.5	30.4	34.2	2.2	21.7	24.8	17.0	37.0	34.2	14.8	25.6
CodeLlama-13B	13.5	18.5	5.1	15.3	0.0	11.7	6.4	16.0	11.1	**46.5**	5.5	17.1
AgentLM-13B	38.0	13.5	22.8	38.1	52.2	30.8	20.8	13.0	46.6	21.6	14.6	23.3
SAMOYED-13B	**54.5**	**38.5**	**55.4**	**40.9**	**72.4**	**50.1**	**35.0**	**23.0**	**62.4**	38.9	**18.4**	**35.5**

Table 4: Performance comparison of SAMOYED and baseline LLMs on held-in and held-out tasks. Due to the space constraint, we group the held-in tasks according to the skill dimensions and report the average scores. The top-2 best of each model group are highlighted in **bold** and underlined respectively. See Appendix G for complete results.

Training Setup We use AdamW optimizer with a learning rate of 5e-5 and a cosine scheduler. The models are trained for 3 epochs with 3% warm-up steps. The batch size is set to 128 and the sequence length is 2048. We choose ShareGPT[2] as the generalist instruction data, and Evol-CodeAlpaca (Luo et al., 2023) as the code data. The mixture ratio of $\mathcal{D}_{\text{agent}}$, $\mathcal{D}_{\text{general}}$, and $\mathcal{D}_{\text{code}}$ is 80%, 10%, 10%. A corresponding data contamination analysis can be found in Appendix F. All experiments are conducted on 8 NVIDIA A100 80G GPUs. We use FastChat (Zheng et al., 2023a) and PyTorch FSDP (Paszke et al., 2019) for efficient training.

Held-in/out Tasks In an effort to balance the reliability and efficiency of the evaluation, we select nine tasks from AGENTBANK to form the held-in test set. For tasks with a huge test set, following Wang et al. (2023), we randomly sample a subset from the original test set. To evaluate the generalized agent intelligence of SAMOYED, we additionally compile five unseen held-out tasks that do not exist in AGENTBANK but still fall into the five skill dimensions of a foundation agent. The held-in and held-out evaluation tasks used in the experiments are listed in Table 3. For all evaluated tasks, 1-shot in-context example is provided in prompts. We also report the results on AgentBench (Liu et al., 2023), another agent benchmark, in Appendix H.

[2] https://sharegpt.com/

6.2 Main Results

Table 4 shows the results of different models on held-in and held-out tasks. Due to the space constraint, we grouped the held-in tasks according to skill dimensions and report the average scores. In Figure 2, we show the results of trajectory tuning on different base LLMs.

Massive trajectory tuning enables generalization to unseen tasks The performance of SAMOYED has a remarkable improvement on held-out unseen tasks, which demonstrates a substantial boost in agent capabilities through large-scale trajectory tuning. Surprisingly, SAMOYED-7B exhibits an even greater enhancement compared to SAMOYED-13B. Our models also outperform AgentLM and Agent-FLAN which are tuned on less trajectories, demonstrating the effectiveness of scaling up the tuning trajectories.

Comparison among baselines The experiment yields several noteworthy model-wise observations. We find that CodeLlama, benefiting from code pre-training, excels in web browsing tasks. Vicuna exhibits strong abilities through fine-tuning on generalist instruction data, demonstrating impressive performance on both held-in/out tasks. Remarkably, the performance of Vicuna-13B even surpasses AgentLM-13B. It is important to highlight that AgentLM's training set comprises 80% generalist instruction data, suggesting that the held-out

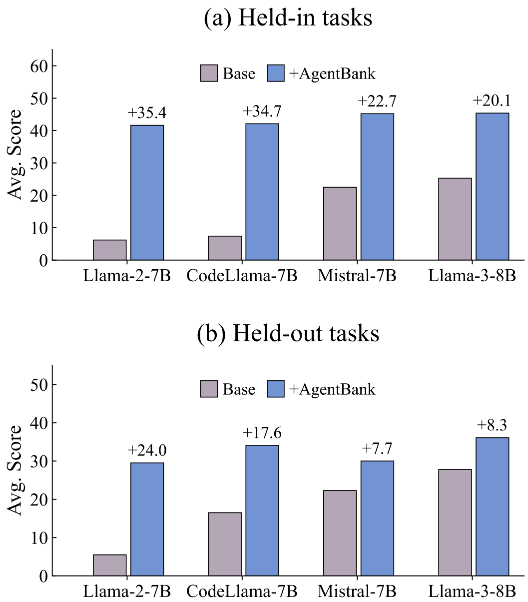


Figure 2: The results of different base models. "Base" denotes untrained LLMs. "+SuperAgent" denotes models after training on AGENTBANK.

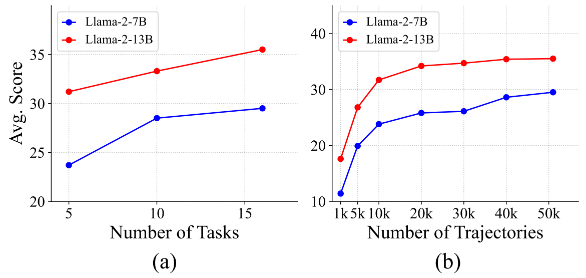


Figure 3: Scaling trends of the number of tasks and interaction trajectories.

task performance of AgentLM largely comes from the enhanced capability of instruction following.

Effectiveness of trajectory tuning on different base models As illustrated in Figure 2, after large-scale trajectory tuning, all LLMs yield significant performance improvements on held-in and held-out tasks. We also notice some interesting outcomes. CodeLlama's superior performance indicates that code training can enhance agent capabilities. As for Mistral and Llama-3, although fine-tuning on AGENTBANK also yields improvements, the performance gain is relatively modest compared with the substantial improvement seen on Llama-2. This finding indicates that weaker LLMs may benefit more from massive trajectory tuning than their stronger counterparts.

7 Further Analysis

7.1 Scaling Trends of Generalization

We investigate the generalization performance of trajectory tuning with respect to two scaling factors: the number of training tasks and the number of training trajectories. Figure 3 illustrates the performance changes on held-out tasks when scaling each of these factors.

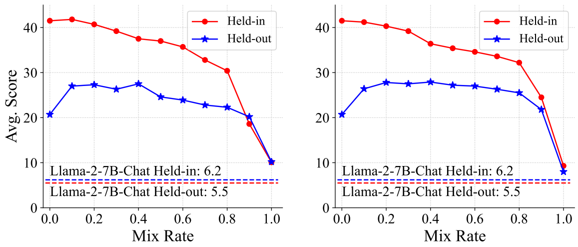
To explore the impact of task scaling, we modify the number of tasks in each skill dimension while ensuring that the skill coverage of the subsets remains consistent. We observe that increasing the number of tasks used for training results in improved performance on held-out tasks. This finding suggests that by scaling the number of distinct tasks for trajectory tuning, the model can enhance its generalized agent capabilities.

As shown in Figure 3b, a comparison between the performance using 1k trajectories and that with 50k+ cases reveals a marked decrease in the generalized ability of the agent, highlighting the importance of scaling the amount of interaction data for better performance. However, the trajectory of performance improvement is gradually plateauing, particularly noticeable with the 13B model, suggesting the necessity for more advanced agent training techniques beyond SFT.

7.2 The Effect of Data Mixture

Mixture Training leads to better generalization. When training SAMOYED, we mix 10% generalist instruction data and 10% code data. Here we conduct ablation study to investigate the effect of mixture training. Specifically, we vary the mixture ratio of ShareGPT and code data and train Llama-2-7B-Chat for 1000 steps. As shown in Figure 4a, a relatively low proportion of generalist data leads to improved agent performance on unseen tasks. Nevertheless, as the amount of generalist data continues to increase, the performance on held-out tasks dramatically degrades. Moreover, disagreed with Zeng et al. (2023) who find that training with only interaction trajectory data will lead to performance degradation on held-out tasks, SAMOYED trained on solely AGENTBANK shows performance improvement on held-out tasks instead.

The ablation on code data also shows a lower ratio of code data will benefit the generalization ability of the agents. Code data, comprising standard syntax and logical abstraction, has the potential to enhance the planning and decision-making capabilities of LLM agents (Yang et al., 2024).



(a) ShareGPT Ablation (b) Code Ablation

Figure 4: Ablation study on data mixture.

Model	Reason	Math	Program	Web	Embodied
Llama-2-7B-Chat	4.0	8.0	7.0	0.4	7.8
+AgentBank	**32.0**	**18.0**	59.2	24.2	14.2
CodeLlama-7B	1.0	13.0	21.8	41.3	5.5
+AgentBank	29.6	16.0	**67.7**	**42.2**	**14.8**

Table 5: The held-out task performance of Llama-2 and CodeLlama.

Model	MMLU	MT-Bench	AlpacaEval 2
Llama-2-7B-Chat	48.3	6.2	5.4
Samoyed-7B	47.7	6.1	5.0
w/o ShareGPT	23.1	2.6	1.9
w/o Code	48.1	5.9	5.1

Table 6: Performance on general tasks.

Base Model	w/ CoT?	Held-In	Held-Out
Llama-2-7B-Chat	✓	41.6	29.5
	✗	41.2	22.8
Mistral-7B	✓	45.2	30.0
	✗	45.5	27.5
Llama-3-8B-Instruct	✓	45.4	36.1
	✗	43.6	31.8

Table 7: Ablation study on CoT rationale.

Code pretraining benefits web tasks. To further analyse the effect of code training, in Table 5, we compare the distinctions between agents based on Llama-2-Chat and CodeLlama. Unsurprisingly, due to its extensive code training, CodeLlama demonstrates excellent performance in programming tasks. Training with extensive interaction trajectories can further elevate its coding proficiency. Additionally, CodeLlama shows exceptional competence in web navigation tasks, likely attributed to the abundance of web pages present in its pretraining datasets.

Mixture training alleviates catastrophic forgetting. Supervised fine-tuning LLMs on downstream tasks will lead to catastrophic forgetting on general capabilities. Here, we select three widely used benchmarks, MMLU (Hendrycks et al., 2020), MT-Bench (Zheng et al., 2023a), AlpacaEval 2 (Li et al., 2023), to evaluate the general capabilities of the trained agents. As shown in Table 6, since the agent trajectory often presented in specific ReAct formats, the models are easily to get overfitting on this style when training solely on agent data. Simply incorporating generalist instruction data during training proves to be an effective strategy in mitigating catastrophic forgetting.

7.3 The Effect of CoT Rationale

Chain-of-Thought (CoT) plays an vital role in LLM reasoning and planning (Wei et al., 2022; Kojima et al., 2022). In our experiments, agents are trained with GPT-generated rationales for each action step and are deployed under ReAct framework (Yao et al., 2022b). In this section, we conduct an ablation study to examine the effectiveness of CoT.

As shown in Table 7, when it comes to held-in tasks, training without rationales has a minimal impact on performance. Mistral-based agent without CoT even slightly surpasses the one with CoT. Nonetheless, for unseen held-out tasks, training without rationale results in a significant performance decline. Explanation traces provide a detailed step-by-step thought processes, enabling agents to learn from the underlying and planning process (Mukherjee et al., 2023). Moreover, without rationale, the agents tend to mimic the style and get overfitting on held-in tasks.

8 Conclusion

In this work, we explore the acquisition of generalized agent capabilities through fine-tuning open-source LLMs on massive interaction trajectories. We introduce by far the largest interaction trajectory dataset AgentBank, comprising over 50k trajectories that encompass 16 tasks across five distinct agent skill dimensions. Building upon AgentBank, we fine-tune Llama-2 to develop Samoyed, an open-source LLM series specialized for agent tasks. Evaluations on both held-in and held-out tasks show that Samoyed significantly outperforms strong baselines in terms of generalized agent capabilities. Comprehensive analysis also reveals the effectiveness of data mixture and plots the scaling law of trajectories. We hope this work to serve as a catalyst for further exploration in the development of more powerful agents.

Limitations

We conclude the limitations of this work as follows:

- Due to the resource constraints, we only conduct experiments and analysis on 7B and 13B models. The extent to which larger models can benefit from large-scale trajectory tuning remains unknown.
- We have not fully explored the potential of equipping our SAMOYED with more sophisticated agent mechanisms, such as Reflexion (Shinn et al., 2023) and ReWOO (Xu et al., 2023a). Further investigation into these mechanisms could yield valuable insights.
- This study primarily focuses on improving the agent's performance via supervised fine-tuning on expert trajectories. How to exploit exploration-based methods (Song et al., 2024) to further optimize the agents is left for future investigation.
- This work is centered around building strong ReAct-style single-agent models. However, multi-agent collaboration framework has demonstrated impressive performance in handling realistic tasks. The development of strong generalized multi-agent systems based on open-source LLMs is still an under-explored area.

Ethics Statement

This work fully complies with the ACL Ethics Policy. We declare that there are no ethical issues in this paper, to the best of our knowledge.

References

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Rodney A Brooks. 1991. Intelligence without representation. *Artificial intelligence*, 47(1-3):139–159.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023a. Fireact: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*.

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. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. 2023b. Theoremqa: A theorem-driven question answering dataset. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. 2024. Agent-flan: Designing data and methods of effective agent tuning for large language models. *arXiv preprint arXiv:2403.12881*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. *arXiv preprint arXiv:2306.06070*.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. 2018. Iqa: Visual question answering in interactive environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4089–4098.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. 2017. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35.

Nicholas R Jennings, Katia Sycara, and Michael Wooldridge. 1998. A roadmap of agent research and development. *Autonomous agents and multi-agent systems*, 1:7–38.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2023. Language models can solve computer tasks. *arXiv preprint arXiv:2303.17491*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Ariel N Lee, Cole J Hunter, and Nataniel Ruiz. 2023. Platypus: Quick, cheap, and powerful refinement of llms. *arXiv preprint arXiv:2308.07317*.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. Wizardcoder: Empowering code large language models with evol-instruct.

Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*.

Yohei Nakajima. 2023. Babyagi. *Python. https://github.com/yoheinakajima/babyagi*.

OpenAI. 2022. Introducing chatgpt.

OpenAI. 2023. Gpt-4 technical report. *arXiv*, pages 2303–08774.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*.

Toran Bruce Richards. 2023. Auto-gpt: An autonomous gpt-4 experiment.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Noah Shinn, Beck Labash, and Ashwin Gopinath. 2023. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020a. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020b. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024. Trial and error: Exploration-based trajectory optimization for llm agents. *arXiv preprint arXiv:2403.02502*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022a. Scienceworld: Is your agent smarter than a 5th grader? *arXiv preprint arXiv:2203.07540*.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. 2023. Mint: Evaluating llms in multi-turn interaction with tools and language feedback. *arXiv preprint arXiv:2309.10691*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022b. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Luca Weihs, Matt Deitke, Aniruddha Kembhavi, and Roozbeh Mottaghi. 2021. Visual room rearrangement. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. 2023a. Rewoo: Decoupling reasoning from observations for efficient augmented language models. *arXiv preprint arXiv:2305.18323*.

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

John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. 2023. Intercode: Standardizing and benchmarking interactive coding with execution feedback. *arXiv preprint arXiv:2306.14898*.

Ke Yang, Jiateng Liu, John Wu, Chaoqi Yang, Yi R Fung, Sha Li, Zixuan Huang, Xu Cao, Xingyao Wang, Yiquan Wang, et al. 2024. If llm is the wizard, then code is the wand: A survey on how code empowers large language models to serve as intelligent agents. *arXiv preprint arXiv:2401.00812*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022a. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022b. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Da Yin, Faeze Brahman, Abhilasha Ravichander, Khyathi Chandu, Kai-Wei Chang, Yejin Choi, and Bill Yuchen Lin. 2023. Lumos: Learning agents with unified data, modular design, and open-source llms. *arXiv preprint arXiv:2311.05657*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2023. Agenttuning: Enabling generalized agent abilities for llms. *arXiv preprint arXiv:2310.12823*.

Jianguo Zhang, Tian Lan, Rithesh Murthy, Zhiwei Liu, Weiran Yao, Juntao Tan, Thai Hoang, Liangwei Yang, Yihao Feng, Zuxin Liu, et al. 2024. Agentohana: Design unified data and training pipeline for effective agent learning. *arXiv preprint arXiv:2402.15506*.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023a. Judging llm-as-a-judge with mt-bench and chatbot arena.

Zilong Zheng, Mengmeng Wang, Zixia Jia, and Baichen Tong. 2023b. Langsuite: Controlling, planning, and interacting with large language models in embodied text environments.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

A Details of Tasks in AgentBank

Reasoning Tasks HotpotQA (Yang et al., 2018) is a question answering dataset featuring multi-hop reasoning. StrategyQA (Geva et al., 2021) is another question answering task where the required reasoning steps are implicit in the question and should be inferred using a strategy. TriviaQA (Joshi et al., 2017) is a dataset consisting of complex compositional questions that require multi-evidence reasoning. In our work, we repurpose these three datasets to interaction environments by incorporating a search engine tool. We employ the GPT-exploration pipeline and filter out failed cases to build the gold trajectories.

For our held-out evaluation, we use Bamboogle (Press et al., 2022), which is made up of questions that need compositional reasoning and are unable to be directly answered by Google.

Math Tasks GSM8K (Cobbe et al., 2021) is a dataset of diverse grade school math problems created by humans. Each problem in GSM8K comes with an official solution path. In our work, we leverage the power of GPT-3.5-Turbo to transform these solution paths into interaction trajectories.

MathQA (Amini et al., 2019) is a large-scale multiple-choice math problem dataset covering multiple math domains. MATH (Press et al., 2022) contains challenging mathematics problems from high school math competitions. To adapt these two datasets into interaction environments, we employ a Python interpreter and employ the GPT-exploration pipeline to construct the trajectories.

For the held-out task, we use TheoremQA (Chen et al., 2023b), a theorem-driven question answering dataset composing of high-quality questions from math, physics, EE&CS, and finance. We implement Python interpreter and Wikipedia tools to construct the corresponding interactive environment.

Programming Tasks InterCode (Yang et al., 2023) is a benchmark for evaluating language models on interactive programming tasks. In this task, agents are required to respond to natural language requests by interacting with a software system, such as a database or terminal. Our work focuses on evaluating the programming ability of agents using two environments: IC-Bash and IC-SQL. IC-Bash is specifically used for the held-out evaluation of agents.

APPS (Hendrycks et al., 2021) is a benchmark focused on Python code generation, encompassing a range of difficulty levels from introductory to competition level. We utilize GPT-3.5-Turbo to reformat the instances in this dataset and construct the trajectories.

HumanEval (Chen et al., 2021) is a dataset designed to measure functional correctness for synthesizing programs from docstrings. MBPP (Austin et al., 2021) consists of around 1,000 crowd-sourced Python programming problems. For both of these datasets, we employ the GPT-exploration pipeline to annotate the interaction trajectories. Subsequently, we employ the answer forcing method to re-annotate the cases where GPT failed.

Web Tasks Mind2Web (Deng et al., 2023) is a dataset for developing and evaluating generalist agents for the web that can follow language instructions to complete complex tasks on any website. WebArena (Zhou et al., 2023) builds realistic web environments for agents to execute tasks. Even GPT-4 struggles with these tasks, so we utilize a teacher forcing and break down the complete interaction trajectory into multiple single steps. Then GPT-3.5-Turbo is employed to annotate the rationales.

WebShop (Yao et al., 2022a) is a simulated e-commerce website environment with real-world products and crowd-sourced text instructions. For 1571 official human annotated trajectories, we employ GPT-3.5-Turbo to reformat them and annotate rationales. Additionally, we incorporate trajectories generated through GPT-exploration, which have final rewards exceeding 0.3.

For our held-out task, we utilize MiniWoB++ (Kim et al., 2023), a diverse collection of over 100 web interaction environments, to formulate our benchmark.

Embodied AI Tasks ALFWorld (Shridhar et al., 2020b) contains interactive TextWorld environments that parallel embodied worlds in the ALFRED dataset (Shridhar et al., 2020a). This dataset provides human-annotated gold trajectories for imitation learning. RoomR (Weihs et al., 2021) is an embodied AI dataset which requires agents to restore the initial configurations of all objects within a room. IQA (Gordon et al., 2018) is a question answering task that requires an agent to interact with a dynamic visual environment. In our work, we utilize the text versions of RoomR and IQA developed by Zheng et al. (2023b). We employ a depth-first-search algorithm to build the gold action

Dataset	Model	R_{train}	R_{pseudo}	R_{test}	Δ_1	Δ_2
AgentInstruct (Zeng et al., 2023)	Llama-2-7B-Chat	17.8	17.5	15.8	-0.3	-2.0
	$+\mathcal{D}_{\text{train}}$	72.5	72.6	62.4	+0.1	-10.1
AGENTBANK (Ours)	Llama-2-7B-Chat	16.2	16.5	16.0	+0.3	-0.2
	$+\mathcal{D}_{\text{train}}$	73.3	62.3	62.8	-11.0	-10.5

Table 8: The average reward of WebShop on different instruction sets. We compare the reward R_{train}, R_{pseudo}, R_{test} on the training set $\mathcal{D}_{\text{train}}$, a pseudo test set held-out from the original training set $\mathcal{D}_{\text{pseudo}}$, and original test set $\mathcal{D}_{\text{test}}$ respectively. We also reports two key metrics: $\Delta_1 = R_{\text{pseudo}} - R_{\text{train}}$ and $\Delta_2 = R_{\text{test}} - R_{\text{train}}$, as the indicators of the difficulty differences between datasets.

sequences for RoomR and IQA. We then leverage GPT-3.5-Turbo to annotate the corresponding rationales.

For the held-out evaluation, we utilize ScienceWorld (Wang et al., 2022a), a text-based virtual environment which encompasses various elementary science experiment tasks, including thermodynamics and electrical circuits.

B Difficulty Bias in Trajectory Collection

In this section, we conduct a experiment to verify the existence of difficulty bias introduced by the trajectory annotation pipeline widely used in recent studies (Chen et al., 2023a; Zeng et al., 2023). Specifically, we choose WebShop trajectories in AGENTBANK and AgentInstruct (Zeng et al., 2023) to conduct the experiment. For AgentInstruct and AGENTBANK, we select 300 instances as the training set $\mathcal{D}_{\text{train}}$, 50 instances as the pseudo test set $\mathcal{D}_{\text{pseudo}}$. We also include the original WebShop test set $\mathcal{D}_{\text{test}}$.

For a dataset conforming to the *i.i.d.* assumption, the instances in $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{pseudo}}$, $\mathcal{D}_{\text{test}}$ are sampled from the same distribution. Therefore, the expected behavior is that the evaluation results on $\mathcal{D}_{\text{pseudo}}$ and $\mathcal{D}_{\text{test}}$ should be consistent. Furthermore, an agent trained on $\mathcal{D}_{\text{train}}$ should ideally perform better on $\mathcal{D}_{\text{train}}$ compared to $\mathcal{D}_{\text{pseudo}}$ and $\mathcal{D}_{\text{test}}$.

Table 8 illustrates the performance of untrained Llama-2-7B-Chat and the trained agent on different sets. For AgentInstruct, both models exhibit worse performance on $\mathcal{D}_{\text{test}}$ compared to $\mathcal{D}_{\text{pseudo}}$, indicating that instances in AgentInstruct are considerably easier than those in the original test set. Conversely, for AGENTBANK, the agents have close performance on $\mathcal{D}_{\text{pseudo}}$ and $\mathcal{D}_{\text{test}}$, aligning with our expectations. The agent trained on our dataset also outperforms the agent trained on AgentInstruct when evaluated on $\mathcal{D}_{\text{test}}$. These experiments highlight that the GPT-exploration trajectory annotation pipeline can introduce difficulty bias in the training set, potentially compromising the generalizability of trained agents.

Rationale	IC-SQL	WebShop
GPT-4	58.5	63.4
GPT-3.5-Turbo	58.8	63.2

Table 9: Comparison of rationales generated by different LLMs.

C Prompts for Trajectory Annotation

D CoT Rationales Generated by Different LLMs

Since providing explanation for gold actions is relatively easy task, we employ GPT-3.5-Turbo as the primary LLM in the rationale annotation process for AGENTBANK. Here we compare the difference of rationale generated by different LLMs. Specifically, we select IC-SQL and WebShop to conduct the experiments. As shown in Table 9, agents training with rationale generated by GPT-4 and GPT-3.5-Turbo have little performance gap.

E Quality Control of AGENTBANK

In Section 4.2, we incorporate heuristic and GPT-based methods to construct AGENTBANK, which can mitigate the difficulty bias problem in the previous annotation pipeline. In this section, we propose to perform a human evaluation to assess the quality of AGENTBANK. To achieve this, we employ 5 human annotators who are instructed to choose the better trajectory from two anonymous candidate options. Here, we select two representative tasks: IC-SQL to assess the quality of answer forcing annotation, and WebShop to evaluate the quality of trajectory reformatting. For IC-SQL, we compare 100 trajectories generated by answer forcing

Dataset	Win	Lose	Tie	Total
IC-SQL	11	16	73	100
WebShop	12	10	58	80

Table 10: Human evaluation of the data quality for AGENTBANK. For IC-SQL, we compare trajectories generated through answer forcing with those generated through exploration. For WebShop, we compare our constructed trajectories with the trajectories constructed by Zeng et al. (2023).

with those generated through GPT exploration. For WebShop, we select 80 trajectories from AGENTBANK and Zeng et al. (2023) which correspond to the same task instance.

As shown in Table 10, for most cases, trajectories generated by answer forcing or reformatting have the same quality as GPT exploration. Therefore, we can conclude that our trajectory annotation process can achieve comparable quality with previous methods (Chen et al., 2023a; Zeng et al., 2023) while mitigating the difficulty bias.

F Data Contamination

When training SAMOYED, we construct a data mixture consisting of trajectory data (AGENTBANK), generalist instruction data (ShareGPT), and code data (Evol-CodeAlpaca). However, it is important to address the concern of potential data contamination, which could result in an overestimation of performance. Therefore, we perform a contamination analysis by comparing our evaluation set with AGENTBANK, ShareGPT, and Evol-CodeAlpaca. Following Liang et al. (2022), we heuristically match 9-grams and 13-grams from the instances in the test set with the training set data. Table 11 displays the proportion of instances which exhibit an overlap with the training data.

First, we observe a high contamination rate for held-in tasks with AGENTBANK. After manually examining these instances, we have some findings. In the case of StrategyQA, we discovered that all instances followed a question format that could be answered with a simple "yes" or "no," potentially resulting in a high n-gram overlap. For WebShop and ALFWorld, we found that the contamination may be attributed to the template-based data construction process. For instance, in WebShop, instructions consistently followed specific formats like “I would like <product> that is <size> and is the color <color>, and price lower than <price> dollars”. Additionally, we observed that MBPP suffers from data contamination issues across all three training sets. After manual inspection, we determined that most of the overlap occurs in importing Python packages and commonly used code snippets, such as loops.

In summary, it can be concluded that the data contamination has a minimal impact on the experimental results. While some overlap exists between the held-in tasks and the training set, this is primarily a result of their data construction process. Moreover, by adhering to the original train-test split of the datasets, the extent of performance overestimation is reduced. Most importantly, the held-out tasks, which are used to assess the agents’ generalized capabilities, do not suffer from the issue of data contamination. This ensures the trustworthiness and robustness of our evaluation.

G Complete Experimental Results

Table 12 shows the complete results on held-in tasks.

H Evaluation on AgentBench

AgentBench (Liu et al., 2023) is another evaluation benchmark for LLM agents, encompassing 8 agent tasks. However, it is worth noting that some tasks in AgentBench are already covered by AGENTBANK, and some tasks may pose a risk of data contamination with our dataset. Nevertheless, to provide a comprehensive perspective, we have included the results of SAMOYED on AgentBench as a point of reference in Table 13.

Dataset	#Inst	AGENTBANK		ShareGPT		Evol-CodeAlpaca	
		9-Gram Rate	13-Gram Rate	9-Gram Rate	13-Gram Rate	9-Gram Rate	13-Gram Rate
Held-in Tasks							
HotpotQA	100	1%	0%	0%	0%	0%	0%
StrategyQA	100	20%	12%	0%	0%	0%	0%
GSM8K	100	3%	0%	0%	0%	0%	0%
MATH	100	15%	4%	0%	0%	2%	0%
IC-SQL	100	7%	0%	0%	0%	1%	0%
MBPP	100	12%	1%	7%	3%	18%	4%
Mind2Web	1173	8%	3%	0%	0%	0%	0%
WebShop	200	41%	14%	0%	0%	0%	0%
ALFWorld	134	14%	8%	0%	0%	0%	0%
Held-out Tasks							
Bamboogle	126	0%	0%	0%	0%	0%	0%
ThreomQA	100	0%	0%	0%	0%	0%	0%
IC-Bash	200	0%	0%	0%	0%	0%	0%
MiniWoB++	460	0%	0%	0%	0%	2%	0%
SciWorld	270	0%	0%	0%	0%	0%	0%

Table 11: Data contamination analysis.

Model	Held-in Tasks									
	HotpotQA	StrategyQA	GSM8K	MATH	IC-SQL	MBPP	Mind2Web	WebShop	ALFWorld	Avg.
Closed-Source Model										
GPT-4	52.1	71.0	87.0	59.0	37.8	72.0	22.6	58.6	77.8	59.8
GPT-3.5-Turbo	24.0	58.0	65.0	18.0	38.5	64.0	21.7	62.4	10.5	40.2
7B Open-Source Model										
Llama-2-7B-Chat	3.0	5.0	15.0	0.0	4.0	1.0	11.9	15.8	0.0	6.2
Vicuna-7B	11.0	47.0	1.0	3.0	17.3	21.0	14.8	33.5	6.0	17.2
CodeLlama-7B	2.0	5.0	7.0	0.0	3.0	0.0	17.0	32.5	0.0	7.4
AgentLM-7B	10.0	49.0	14.0	6.0	13.9	10.0	10.6	63.7	63.4	26.7
SAMOYED-7B	30.0	66.0	43.0	18.0	59.2	24.0	12.2	60.5	61.2	41.6
13B Open-Source Model										
Llama-2-13B-Chat	6.0	19.0	18.0	3.0	3.0	13.4	17.2	5.3	0.0	9.4
Vicuna-13B	15.0	36.0	9.0	4.0	37.0	23.7	15.2	53.3	2.2	21.7
CodeLlama-13B	7.0	20.0	29.0	8.1	3.0	7.2	7.6	23.0	0.0	11.7
AgentLM-13B	24.0	52.0	21.0	6.1	25.7	20.0	11.1	65.0	52.2	30.8
SAMOYED-13B	41.0	68.0	53.0	24.0	67.7	43.0	18.6	63.1	72.4	50.1

Table 12: Performance of SAMOYED and baseline LLMs on held-in tasks.

Model	Code-grounded			Game-grounded			Web-grounded		Overall
	OS†	DB†	KG†	DCG	LTP	HH‡	WS‡	WB‡	
GPT-4	42.4	32.0	58.8	74.5	16.6	78.0	61.1	29.0	4.01
GPT-3.5-Turbo	32.6	36.7	25.9	33.7	10.5	16.0	64.1	20.0	2.32
Llama-2-7B-Chat	4.2	8.0	2.1	6.9	0.0	0.0	11.6	7.0	0.34
Vicuna-7B	9.7	8.7	2.5	0.3	6.4	0.0	2.2	9.0	0.56
CodeLlama-7B	4.9	12.7	8.2	0.0	0.0	2.0	25.2	12.0	0.50
SAMOYED-7B	11.8	9.7	2.7	1.9	8.2	68.0	60.5	12.2	1.60

Table 13: Performance of SAMOYED and baseline LLMs on AgentBench (Liu et al., 2023). † means the test set may suffer data contamination with AGENTBANK. ‡ means the task is already covered by AGENTBANK.