Enhancing the General Agent Capabilities of Low-Parameter LLMs through Tuning and Multi-Branch Reasoning

Qinhao Zhou[1] **Zihan Zhang**[1] **Xiang Xiang**[1]*
Ke Wang[2] **Yuchuan Wu**[2] **Yongbin Li**[2]
[1] National Key Lab of MSIIPT, School of Artificial Intelligence and Automation,
Huazhong University of Science and Technology, Wuhan, China
[2] Alibaba Group, Beijing, China

Abstract

Open-source pre-trained Large Language Models (LLMs) exhibit strong language understanding and generation capabilities, making them highly successful in a variety of tasks. However, when used as agents for dealing with complex problems in the real world, their performance is far inferior to large commercial models such as ChatGPT and GPT-4. As intelligent agents, LLMs need to have the capabilities of task planning, long-term memory, and the ability to leverage external tools to achieve satisfactory performance. Various methods have been proposed to enhance the agent capabilities of LLMs. On the one hand, methods involve constructing agent-specific data and fine-tuning the models. On the other hand, some methods focus on designing prompts that effectively activate the reasoning abilities of the LLMs. We explore both strategies on the 7B and 13B models. We propose a comprehensive method for constructing agent-specific data using GPT-4. Through supervised fine-tuning with constructed data, we find that for these models with a relatively small number of parameters, supervised fine-tuning can significantly reduce hallucination outputs and formatting errors in agent tasks. Furthermore, techniques such as multi-path reasoning and task decomposition can effectively decrease problem complexity and enhance the performance of LLMs as agents. We evaluate our method on five agent tasks of AgentBench and achieve satisfactory results.

1 Introduction

Large Language Models (LLMs) have been extensively employed in a wide range of natural language processing tasks, yielding groundbreaking achievements. Furthermore, LLMs have demonstrated their capability to undertake more challenging tasks, such as functioning as AI agents. Unlike conventional reasoning tasks, an AI agent is an entity that needs to interact with the human or external environment, draw inferences, and judge subsequent actions based on feedback. Each single task typically involves multiple rounds of dialogue to accomplish. For instance, in a home environment, an agent may be tasked with various household tasks that require continuous interaction with the environment. The agent needs to evaluate its actions based on the feedback from the environment and make timely adjustments to its strategies. Traditional AI agents are usually effective in specific domains or environments, but their generalization and adaptability are obviously insufficient (Liu et al., 2023).

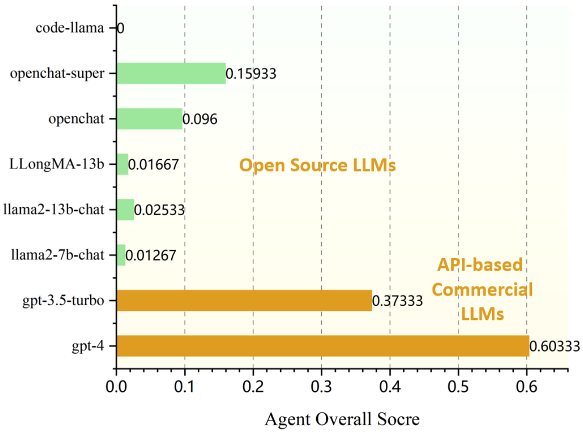


Figure 1: The agent performance of open-source LLMs and commercial LLMs. Agent Overall Score is the average accuracy of several agent tasks.

In recent years, an increasing number of work (Brown et al., 2020; OpenAI, 2023; Qin et al., 2023; Shinn et al., 2023; Zhu et al., 2023) have demonstrated that LLMs possess strong capabilities in reasoning, planning, memory, and utilizing external tools. This has propelled LLMs towards becoming more generalized and adaptive agents. Recently, AgentBench (Liu et al., 2023) conducts extensive evaluations of both commercial and open-source LLMs on eight different agent tasks. The

*Corresponding author (e-mail: xex@hust.edu.cn); also with Peng Cheng Laboratory, Shenzhen, China.

Findings of the Association for Computational Linguistics: NAACL 2024, pages 2922–2931
June 16-21, 2024 ©2024 Association for Computational Linguistics

results reveal that commercial API models show superior agent capabilities. In addition, work such as AutoGPT (Gravitas, 2023) and GPT-Engineer (Osika et al., 2023) also use LLMs as agents to build a complete framework for solving complex real-world problems. However, open-source models, especially those with smaller parameter sizes, still have substantial potential for enhancement. As shown in Fig. 1, the average performance of 7B and 13B LLMs on each agent task is significantly lower than the commercial models.

Unlike commercial LLMs, small-scale open-source LLMs are relatively inefficient in general knowledge (Peters et al., 2019). Besides, lower parameter sizes limit reasoning and memory capacity, often leading to hallucinations in the agent dialogue process (Zhang et al., 2023b). However, in practical applications, LLMs with 7B and 13B parameters are the most widely used due to their relative ease of deployment and fine-tuning. Therefore, enhancing the capabilities of such LLMs is of great practical significance. Currently, studies on LLMs agents or enhancing model reasoning capabilities (Xi et al., 2023a; Wang et al., 2023) primarily focus on large-scale models. The investigation of agent capabilities on 7B and 13B LLMs is still in its early stages of exploration. As explained, a proficient agent requires task-planning abilities, proficiency in utilizing external tools, and long-term memory capabilities. Task planning refers to the ability of the model to decompose large-scale tasks into manageable sub-goals, facilitating efficient handling of complex tasks. Long-term memory capabilities reflect the ability of the LLMs to retain and recall historical information during their interactive processes with the environment. Considering these abilities, we propose a method to enhance the performance of 7B and 13B LLMs on agent tasks.

In our proposed approach, We focus on enhancing the agent capabilities of LLMs from two key aspects. First, improving the agent capabilities through Supervised Fine-Tuning (SFT). This approach fundamentally enhances the LLMs themselves. Unlike general reasoning tasks, an agent's role goes beyond planning and reasoning. It also involves continuous interaction with the environment or humans to execute subsequent actions until a desired outcome is achieved. To improve the agent abilities of LLMs, it is essential to train them on diverse datasets that reflect the full range of interactive behaviors between the agent and the environment. This involves constructing data that not only records the actions taken by the agent but also captures the internal thought processes and decision-making. Additionally, the environment should provide meaningful feedback to guide the learning of the agent. We propose to use GPT-4 (OpenAI, 2023) to construct data. By designing a framework that involves GPT-4 engaging the multi-turn dialogues, we can generate conversational data that captures the interaction between different roles. During these conversations, GPT-4 can take on different roles, such as playing the part of an agent, a user, or the environment, and actively participate in dynamic exchanges. In addition, we incorporate a significant amount of general instruction tuning data into the constructed dataset to preserve the general capabilities of the LLMs.

Besides, we optimize the reasoning path through task decomposition and backtracking. Inspired by Chain of Thought (Wei et al., 2022), significant efforts have been dedicated to activating the reasoning ability of the LLMs. For instance, ReAct (Yao et al., 2022b) integrates the thinking process into the task of multi-step reasoning. ToT (Yao et al., 2023) uses depth-first and breadth-first traversal of reasoning nodes, which is more conducive to finding the optimal solution. We migrate the idea of ToT to the agent tasks and combine it with task decomposition and backtracking. Task decomposition leverages the task planning capability of the LLMs to decompose complex and lengthy tasks into several smaller subtasks. Considering that it is difficult for LLMs to find optimal answers or complete tasks through a single reasoning path, we introduce a judgment process where the reasoning process goes back to the starting point, termed backtracking. Through the integration of task decomposition and backtracking, we aim to enhance LLMs' ability to handle complex tasks effectively.

The main contributions of this paper are: 1) We explore the capabilities of 7B and 13B open-source LLMs as agents, exploring their potential in performing agent tasks. 2) We propose supervised fine-tuning with specific agent data as a fundamental approach to improving the capability of open-source LLMs as agents. To achieve this, we develop a method for constructing agent data. 3) We find that task decomposition and backtracking are effective approaches for addressing complex agent tasks. We conduct experiments on AgentBench and achieve promising results.

2 Related Works

Planning and Reasoning. Planning and reasoning are crucial capacities for agents to solve complex tasks. Through the in-context of the thinking chain, Chain-of-Thought (Wei et al., 2022) activates the reasoning capabilities of LLMs and enables the generation of intermediate thought processes before producing answers. Some other strategies have also been proposed to further enhance the thinking process of models. For example, SC (Wang et al., 2022) leverages the self-consistency of LLMs by generating multiple thinking chains and determining the final answer through voting. Reconcile (Chen et al., 2023) enhances the reasoning capabilities of LLMs through multiple rounds of discussions and using confidence-weighted voting. Besides, self-polish (Xi et al., 2023b), and self-refine (Madaan et al., 2023) augment the thinking process of LLMs from other perspectives. Furthermore, ToT (Yao et al., 2023) explores the abstracting reasoning process into deep tree search. In addition, there are some works (Zhang et al., 2023c) that apply the idea of chain thinking to multi-modal tasks.

Large Language Model as Agent. With the rapid advancement of LLMs, extensive research has been conducted to explore their powerful capabilities in planning and reasoning (Xi et al., 2023a; Wang et al., 2023). This has opened up the possibility of employing LLMs as agents. On the one hand, there have been several efforts to apply LLMs to various agent tasks and construct agent simulation frameworks. On the other hand, several works (Xu et al., 2023; Kim et al., 2023), such as ReAct (Yao et al., 2022b), have focused on incorporating reasoning and deliberation into the agent process for LLMs. In addition, some works apply the reasoning methods to the agent interaction process. PET (Wu et al., 2023) applies task decomposition to the household agent environment, which is helpful for LLMs to complete complex tasks. LATS (Zhou et al., 2023) and RAP (Hao et al., 2023) apply Monte Carlo tree search to the agent reasoning process. It is advantageous to find better answers compared with ToT. In addition, research works such as AutoGPT (Gravitas, 2023) and GPT-Engineer (Osika et al., 2023) utilize commercial LLMs as agent core of their frameworks, enabling the development of comprehensive agent architectures to tackle complex real-world problems.

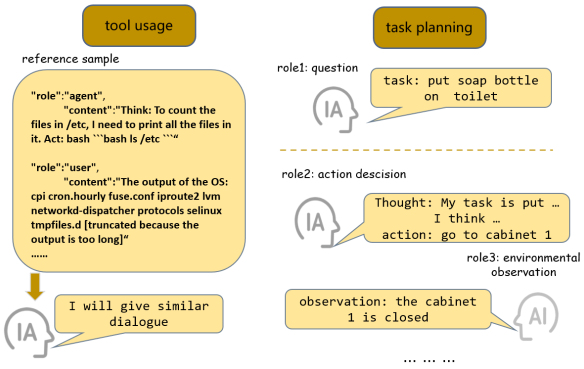
Instruction Tuning for Language Model. Instruction tuning plays a crucial role in training LLMs. After pre-training with massive unsupervised data, LLMs acquire a substantial amount of knowledge and process language understanding and generation capabilities. Further supervised instruction fine-tuning (Zhang et al., 2023a; Dong et al., 2022) is conducted to align the model with human instructions and generate outputs that better align with human preferences. Instruction tuning mainly focuses on constructing complex and diverse general-purpose tasks to train LLMs to answer questions in a human manner. For example, FLAN (Wei et al., 2021) and T0 (Sanh et al., 2021) construct a multi-task instruction tuning dataset using massive publicly available datasets. The fine-tuned model shows strong zero-shot generalizability. In addition to utilizing existing datasets, another common approach is to generate data using commercial LLMs. Self-Instruct (Wang et al., 2022; Peng et al., 2023) leverages GPT-4 to generate a large amount of diverse data, given a few seed tasks. These data are used for fine-tuning open-source LLMs and get significant improvements in various tasks. To enhance the agent capability of LLMs, AgentTuning (Zeng et al., 2023) utilizes commercial LLMs to construct data in specific agent environments containing multi-turn dialogues.

3 Methodology

In this section, we first give a formal definition of LLMs as agents. Then, we introduce the two components of our approach. In the first part, we construct agent-tuning data to fine-tune LLMs with parameter-efficient tuning methods. This is a way to fundamentally improve the capabilities of LLMs. In the second part, we propose enhancing the reasoning capabilities of LLMs through task decomposition and backtracking.

3.1 Problem Formulation

For a given agent task, the interaction trajectory of LLMs as agents can be represented as a dialogue history $(e_1, a_1, ..., e_n, a_n)$. During this process, there are typically two roles involved: environment and agent. e_i represents the hints and feedback from the environment and the agent engages in thinking and actions represented as a_i. Each dialogue track corresponds to a final reward $r \in [0, 1]$, which reflects the completion of the task.


tool usage
reference sample
I will give similar dialogue
task planning
role1: question
task: put soap bottle on toilet
role2: action descision
Thought: My task is put … I think … action: go to cabinet 1
role3: environmental observation
observation: the cabinet 1 is closed
… … …

```
"role":"agent",
        "content":"Think: To count the
files in /etc, I need to print all the files in
it. Act: bash ```bash ls /etc ```“

"role":"user",
        "content":"The output of the OS:
cpi cron.hourly fuse.conf iproute2 lvm
networkd-dispatcher protocols selinux
tmpfiles.d [truncated because the
output is too long]“
……
```

Figure 2: The process of constructing agent data. For task planning and external tool usage capabilities, we use two strategies, respectively.

3.2 Supervised Tuning with General and Constructed Agent Data

We observe a significant disparity in the agent capabilities between the open-source 7B and 13B LLMs and the commercial models. In the dialogue process, open-source models often exhibit issues such as formatting errors, getting stuck in infinite loops, and generating hallucinatory outputs. To reduce the occurrence of the above issues, a fundamental approach is to fine-tune the LLMs with appropriate data. However, the agent is engaged in multi-turn dialogues and interacts with specific environments, which is different from currently available open-source general-purpose instruction data. To solve this challenge, we leverage commercial models API to construct agent-specific data and merge them with general instruction datasets to fine-tune the low-parameter LLMs.

As agents, LLMs need to possess three fundamental capabilities: task planning, long-term memory, and tool usage. To enhance the task planning capabilities of LLMs, we take ALFworld (Shridhar et al., 2020) as an example to construct data with interactive trajectories. Unlike current methods of constructing data using models like GPT-3.5 (OpenAI, 2022), data for agents should not only involve multi-turn dialogues but also need to reflect task planning and trajectory. Therefore, we meticulously design the construction process of the dataset, dividing the process of each piece of data into three steps. It includes task construction, trajectory interaction, and manual filtering. This approach ensures that each piece of data captures the necessary elements for training agents effectively. We utilize GPT-3.5 or GPT-4 to generate questions and interaction trajectories and this process can be easily extended to other agent tasks. As illustrated in Fig. 2 right, to generate a complete interaction trajectory, we simulate GPT playing three distinct roles in a household environment. These roles are named as *question generator*, *action maker*, and *environmental agent*.

First, we randomly initialize a specific room environment, determining the number and placement of household items. The *question generator* role is then responsible for generating intelligent household-related questions based on the provided environment. Subsequently, the *action maker* role continuously offers its thoughts and actions based on the environment feedback, simultaneously, the *environment agent* role provides reasonable feedback and cues corresponding to the actions taken in each step. These two roles continue to interact until the problem is completed or the maximum number of interactions is reached, thus generating a complete trajectory. However, as there is no assurance of the logical consistency of the *environment agent*'s feedback and the *action maker*'s actions, manual screening is required after the data is generated.

In addition to agent tasks that focus on task planning, there are also agent tasks such as Operating System, and WebShop (Yao et al., 2022a) that have fewer dialogue rounds and prioritize the use of external tools. For this type of task, we draw on the idea of in-context learning. Specifically, as shown in Fig. 2 left, we provide GPT with examples with complete reasoning trajectories to enable it to imitate. Subsequently, we manually filter and select logically consistent data from generated outputs. We expect to use this type of data to improve the retrieval capabilities and tool usage capabilities of LLMs.

Existing work on agent fine-tuning (Zeng et al., 2023) shows that using only agent data to fine-tune LLMs compromises their generalizability. Therefore, we mix some general instruction tunning data into our agent data when fine-tuning LLMs. Suppose M_θ represents pre-trained LLMs and the $M_\theta(y|x)$ represents the probability distribution of output y when given history x. We consider two datasets: the agent data D_{agent} and the general instruction tuning data $D_{general}$. We optimize the loss function as follows:

$$\mathcal{L}(\theta) = \lambda \cdot \mathbf{E}_{(x,y)\ D_{agent}}[logM_\theta(y|x)] + (1-\lambda) \cdot \mathbf{E}_{(x,y)\ D_{general}}[logM_\theta(y|x)]. \quad (1)$$

Where $\lambda \in [0,1]$ denotes the mix ratio of the two datasets. A larger λ means that the LLMs are inclined to specific agent capabilities, whereas a small λ makes LLMs more inclined to general capabilities. We observe that deterioration of the general ability of LLMs will also decrease the agent ability, so we set a small value for λ. This is identical to AgentTuning (Zeng et al., 2023). In the experimental section, we analyze different values of λ.

In the context of fine-tuning strategiy, we adopt Low-Rank Adaptation (LORA) (Hu et al., 2021) fine-tuning which is based on making low-rank modifications to the weight matrices in LLMs. For each linear layer in the model, the original weight matrix W is adjusted to $W + \Delta W$, where ΔW is generated through the product of low-rank matrices as $\Delta W = A \times B$, where A and B are low-rank matrices, with ranks significantly smaller than the rank of the original weight matrix W.

3.3 Multi-Path Reasoning under Task Decomposition

Recently, because it is difficult for a single agent to complete complex multi-step tasks, more and more work tends to involve multi-agent collaboration, allowing models to play different roles to jointly advance tasks (Qiao et al., 2024). We take a similar approach. On the one hand, we we instruct LLMs to generate multiple available actions in each reasoning step. On the other hand, we employ a judge model to select one action from the provided set and continue the reasoning process until a final output is obtained.

For LLMs with small parameter sizes, due to their limited long-term memory capacity, it is challenging for them to handle complex long dialogue tasks. To address this issue, we employ a task decomposition strategy, where complex tasks that require multiple steps are broken down into simpler subtasks. We use another LLM with the same number of parameters as our planning module and we name it as M_p. For a given task $\mathcal{T}$, we compose query prompt P_{sub} as "break down the task $\mathcal{T}$ into subtasks in the following format...". The M_p will generate a sub-task list $S_{\mathcal{T}} = \{s_1, ..., s_k\}$. k is the number of sub-tasks and to avoid an excessive number of subtasks, we typically set k to 3. For example, for task $\mathcal{T}$ ="put a soap bottle in the toilet", the LLMs can describe three steps as s_1 = "look around and find a soap bottle", s_2 = "take up the soap bottle and go to the toilet", s_3 = "put the soap bottle in the toilet". Then, the agent will complete it one by one according to the subtask list $S_{\mathcal{T}}$. We introduce another LLM as judgment module M_{jdg} to judge the completion of each subtask. For subtask s_t, we compose the judge prompt P_{jdg} as "Judge whether the subtask is completed, output Yes or No", each time the agent executes a step, we feed P_{jdg} to a LLM and get the output of "Yes" or "No" until the subtask is completed.

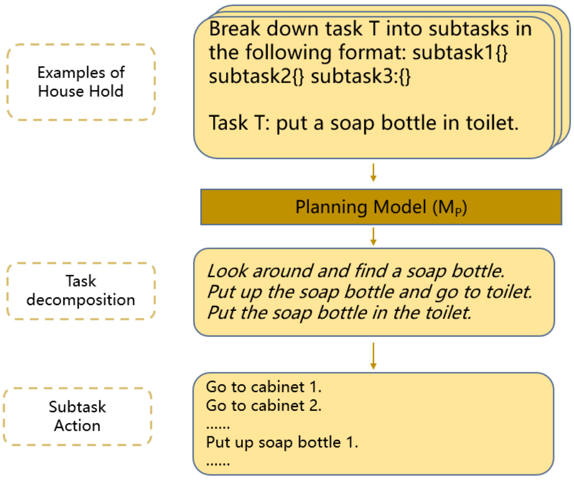

Examples of House Hold
Break down task T into subtasks in the following format: subtask1{} subtask2{} subtask3:{}
Task T: put a soap bottle in toilet.
Planning Model (Mₚ)
Task decomposition
Look around and find a soap bottle.
Put up the soap bottle and go to toilet.
Put the soap bottle in the toilet.
Subtask Action
Go to cabinet 1.
Go to cabinet 2.
......
Put up soap bottle 1.
......

Figure 3: The process of task decomposition. The planning model breaks the entire task into several small subtasks.

Agent tasks in the real world are often complex and one single reasoning path may not yield the optimal answer. Inspired by the reflective ability in human thinking processes, we propose to take multi-path reasoning with LLMs. We call this method *backtracking*. When a particular reasoning path yields a suboptimal output, we compose a backtracking prompt as "it was observed that the answer was not the optimal choice for task $\mathcal{T}$...". We also prompt the LLMs to eschew reasoning paths that have been previously deduced. To this end, we compose the prompt as "it is important to note that actions should be adjusted appropriately based on the historical information" and we splice this prompt behind the backtracking prompt. Furthermore, backtracking and task decomposition are not mutually exclusive and can be applied together in the reasoning process of LLMs. We find that task decomposition is more effective for agent tasks that

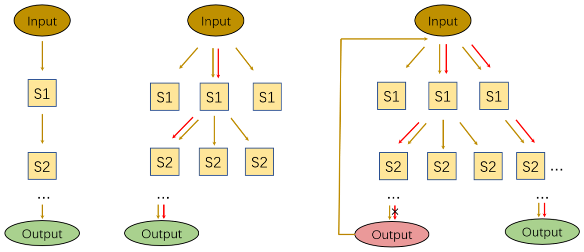


Figure 4: The comparison of different reasoning methods. From the left to right are Input Output (IO), ToT and our method.

emphasize planning abilities, while backtracking is more effective for agent tasks that emphasize API invocation capabilities.

Overall, our method is divided into two parts. The first part uses commercial LLMs to construct agent data and employs SFT to fundamentally enhance the agent capabilities of low-parameter LLMs. In the second part, while keeping the LLMs unchanged, it maximizes the activation of the agent capabilities by incorporating multi-path reasoning and task decomposition. For 7B and 13B LLMs, common issues such as hallucinatory outputs and forgetting errors often occur. By fine-tuning the LLMs on domain-specific data that adheres to the desired format, these issues can be significantly mitigated. For reasoning problems with vast search spaces, finding the optimal solution through a single inference path is challenging. This issue cannot be effectively addressed through supervised fine-tuning alone. However, by introducing techniques such as multi-path reasoning and task decomposition, the complexity of the problem can be reduced, facilitating the identification of the optimal solution.

4 Experiments

Agent Datasets: We select five tasks from AgentBench benchmark (Liu et al., 2023): ALFWorld, WebShop, Mind2Web, Operating System, and Database. Next, we will introduce each agent task one by one in detail.

ALFWorld is designed to evaluate the planning ability of LLMs in a simulated home environment. The model needs to make decisions and execute actions through a text interface based on the environment description and target instructions, and dynamically adjust the plan to complete the task.

WebShop aims to evaluate the performance of LLMs in a simulated online shopping environment that mimics a real e-commerce website.The goal of the evaluation is to require LLMs to shop in a virtual shopping environment according to instructions and select products that meet desired attributes.

Mind2Web is a general web agent evaluation benchmark designed to evaluate the ability of LLMs to perform complex tasks on websites in different domains. The dataset covers a cross-domain test set across multiple websites. Each task includes a task description, a reference action sequence, and web page information and is designed to test the performance of LLMs in web browsing and interactive environments.

Operating System is designed to evaluate the ability of LLMs to perform tasks in the Bash environment of a real operating system. Tasks includes question answering and action, where the model needs to generate commands to solve a problem or perform an action.

DataBase is designed to evaluate the ability of LLMs to operate via SQL on real databases. The dataset contains a diverse set of instructions and databases, created by combining multiple existing datasets and performing data augmentation.

Implementation details: We use AgentBench as our benchmark and conduct experiments based on it. For 13B models, we choose OpenChat. OpenChat is a series of open-source LLMs fine-tuned on diverse and high-quality datasets of multi-round conversations. We select two models, openchat-v3.2 and openchat-v3.2-super for experiments. For the 7B models, we select llama2 and agentlm (Zeng et al., 2023) for experiments. We use the fastchat framework to deploy LLMs and we use four RTX 4090 NVIDIA GPUs. See also the project page[1].

4.1 Experimental Results

Supervised fine-tuning with constructed dataset. The experiments of supervised fine-tuning are shown in Tab. 1. We fine-tune the 7B model on various instruction-tuning datasets and test it on five agent tasks. It can be seen that fine-tuning on various instruction datasets has a positive effect on improving the capabilities of agents. Among them, we find that fine-tuning the LLMs using code-type instructions has shown relatively limited effectiveness in improving agent capabilities. For example, after fine-tuning on alpaca-code dataset, the performance of llama2 on operating system task does not

[1] https://github.com/HAIV-Lab/LLM-TMBR

	Data type	Operating System	DataBase	Webshop	ALFWorld	Mind2web	Avg. ↑
GPT-4		42.4	32	61.1	78	29	48.50
GPT-3.5-turbo		32.6	36.7	64.1	16	20	33.88
claude		9.7	22	55.7	58	25	34.08
llama2-chat w/o sft		3.8	2.66	0	0	5.68	2.43
codegen-struct	code	3.8	1.3	0	0	0	1.27
alpaca-code		3.8	1.3	4.20	0	5.68	2.99
open-assistant	dialog	0	2.67	2.70	0	3.41	1.76
alpaca	instro+agent	15.38	3.33	31.10	0	8.52	11.67
agenttuning		15.38	38.30	32.60	10	7.38	20.73
ours		11.54	27.0	34.53	10	9.66	18.33

Table 1: The experimental results of fine-tuning LLMs with different instruction tuning datasets on AgentBench tasks. We use llama2-7b-chat as the base model.

improve, and its performance on database tasks actually declined by 1.33%. We analyze that although code-type data can enhance the understanding of the code of LLMs, it lacks dialogue processes and the decomposition of complex problems. Similar to code-type data, fine-tuning LLMs on regular dialog data alone is not an appropriate choice for enhancing its agent capabilities. For instance, after fine-tuning on Open-Assistant, llama2 exhibited a decrease in performance on operating system task and a lower improvement on the webshop task compared to other datasets.

Besides, we find that fine-tuning LLMs on high-quality general instruction tuning datasets can significantly improve its agent capabilities. For example, after fine-tuning with alpaca instruction tuning data, llama2 exhibit significant improvements across multiple agent tasks. In the operating system tasks and webshop tasks, llama2 tuning with alpaca data achieves nearly comparable results to those obtained through agenttuning. Agenttuning is the most effective tuning dataset. It combines GPT-4 assisted trajectory-labeled agent data with general instruction tuning data, resulting in significant improvements for llama2 across different agent tasks. Its performance in the database even exceeds that of the commercial model. Fine-tuning the model using our constructed data can also improve the performance of LLMs on agent tasks. Although we construct limited and easy-to-collect data, the performance of LLMs fine-tuned with our data exceeds other datasets on some agent tasks. For example, on operating system tasks, our results are 7.74% higher than code-type datasets and 11.54% higher than dialog-type datasets. Compared with agenttuning, our results are still far behind, which can be attributed to the limited amount of data. In addition, there are fewer complex tasks involving long conversations in our data, which is also one of the reasons.

Reasoning with task decomposition and backtracking. We compare different reasoning methods on 7B and 13B LLMs, and the results are shown in Tab. 2. The 7B LLMs we evaluated are fine-tuned with agent data. AgentLM is fine-tuned with agent-tuning data, and llama2 is fine-tuned with the data we constructed. We mainly conduct evaluations on webshop, household and operating system tasks. It can be seen that applying ReAct to various tasks is usually better than direct input and output (IO). For example, on the openchat-v3.2 model, ReAct is 18% higher than IO on webshop. Besides, our method can further achieve small improvements based on ReAct. On the webshop task, our results are on average about 1% higher than the second-best result. And on the household task, our method achieve improvements of 5% and 6%, respectively, on the 13B LLMs.

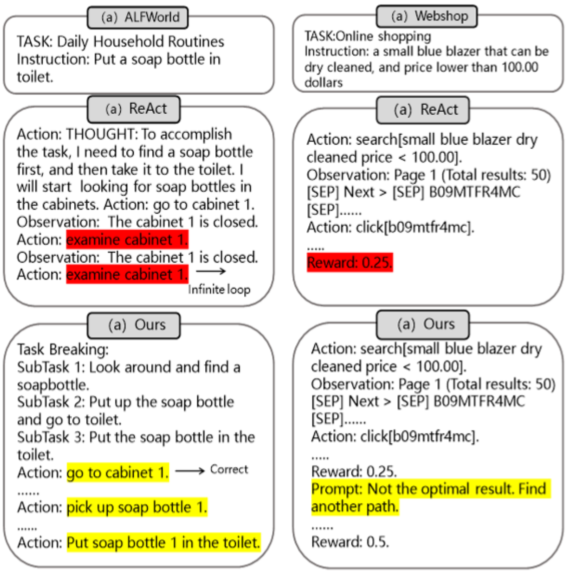
To delve into the impact of different reasoning methods on the results, we compare ReAct and our reasoning process as shown in Fig. 5. It can be seen that ReAct can prompt LLMs to think in each reasoning step, the models can still experience issues such as getting stuck in infinite loops and suffering from memory confusion. In contrast, on household tasks, since we break down complex tasks into several smaller tasks, model thinking is less error-prone than ReAct.

4.2 Ablation Study

The experiments of num path and branch. "num path" refers to the number of backtracking iterations conducted, with a higher value indicating an increase in the number of reasoning paths explored.

Size	LLMs	Methods	Webshop	ALFWorld	Operate System	Avg. ↑
13B	openchat_v3.2	IO	1	0	0	0.33
		CoT	19	0	0	6.33
		ReAct	26	5	7.6	12.86
		Ours	27	10	7.6	14.86
	openchat_v3.2_super	IO	5	0	0	1.66
		CoT	23	0	0	7.66
		ReAct	30	5	3.8	12.93
		Ours	31	11	3.8	15.26
7B	AgentLM-7B	IO	50	5	3.8	20.86
		CoT	34	5	7.6	19.50
		ReAct	33	0	7.6	13.53
		Ours	51	0	7.6	19.53
	llama2-7B	IO	0	0	0	0
		CoT	4	0	0	1.33
		ReAct	13.35	0	7.6	6.98
		Ours	13.40	0	7.6	7.00

Table 2: Experimental results of different reasoning methods on three agent benchmarks.



TASK: Daily Household Routines
Instruction: Put a soap bottle in toilet.

Action: THOUGHT: To accomplish the task, I need to find a soap bottle first, and then take it to the toilet. I will start looking for soap bottles in the cabinets. Action: go to cabinet 1.
Observation: The cabinet 1 is closed.
Action: examine cabinet 1.
Observation: The cabinet 1 is closed.
Action: examine cabinet 1.

Task Breaking:
SubTask 1: Look around and find a soapbottle.
SubTask 2: Put up the soap bottle and go to toilet.
SubTask 3: Put the soap bottle in the toilet.
Action: go to cabinet 1.
......
Action: pick up soap bottle 1.
......
Action: Put soap bottle 1 in the toilet.

TASK:Online shopping
Instruction: a small blue blazer that can be dry cleaned, and price lower than 100.00 dollars

Action: search[small blue blazer dry cleaned price < 100.00].
Observation: Page 1 (Total results: 50) [SEP] Next > [SEP] B09MTFR4MC [SEP]......
Action: click[b09mtfr4mc].
.....
Reward: 0.25.

Action: search[small blue blazer dry cleaned price < 100.00].
Observation: Page 1 (Total results: 50) [SEP] Next > [SEP] B09MTFR4MC [SEP]......
Action: click[b09mtfr4mc].
.....
Reward: 0.25.
Prompt: Not the optimal result. Find another path.
......
Reward: 0.5.

Figure 5: Comparison of ReAct and our method in agent task reasoning. We show the action and observation in webshop and household tasks.

num path	Webshop	num branch	Webshop
1	20.29	1	26.00
2	**27.00**	2	**27.00**
3	17.84	3	6.80
4	16.67	4	15.80

Table 3: The experimental results of the effect of num path and num branch in our reasoning method.

λ	Alfworld	Webshop	Mind2web	OS
0.1	0.0	**38.13**	6.81	0
0.3	0.0	30.06	**7.95**	0
0.5	0.0	36.42	**7.95**	**3.8**
0.8	**5**	23.35	3.97	0

Table 4: Experimental results after mixing different general data and agent data.

We conduct experiments of "num path" shown in Tab. 3 left. It can be seen that appropriately increasing "num path" can improve performance, but when "num path" is greater than 2, performance decreases. We also conduct the experiments of "num branch" shown in Tab. 3 right. "num branch" is the number of nodes expanded at each reasoning step. It is shown that properly increasing "num branch" can also improve performance: when "num branch" is greater than 2, performance decreases.

We conduct experiments on the mixing ratio of different general data and agent data as shown in Fig.4. We find that too much agent data will not bring huge improvements, and general data is equally important.

5 Conclusion

LLMs as intelligent agents have demonstrated powerful agent capabilities. In this work, we explore the 7B and 13B LLMs as agents, and propose to enhance the agent performance of these open-source models by supervised fine-tuning through agent data as well as multi-branch reasoning. SFT can effectively reduce format errors and hallucination output of the LLMs, which not only improves the agent performance but also facilitates the application of various reasoning methods to agent tasks.

6 Limitations

This study presents several limitations. First, our experiments are limited to 7B and 13B LLMs, and thus, the applicability of our findings to models of different sizes is not verified. The methods we propose may also not be feasible for all researchers due to the computational demands of fine-tuning larger models. Additionally, measuring reductions in hallucinations and formatting errors is inherently subjective, and the performance metrics used may not fully capture the agent capabilities in complex real-world tasks.

The constructed data for SFT could introduce biases and the potential for model overfitting, limiting the performance of LLMs on unencountered tasks. Moreover, while we implement multi-path reasoning and task decomposition, the strategies for optimizing these techniques are not definitive. Our evaluation on a limited set of tasks does not account for the full range of an agent capabilities, necessitating broader evaluations in future research.

Acknowledgement

The funding for this research was generously provided by the Alibaba Innovation Research program under Grant Contract # CRAQ7WHZ11220001-20978282, Natural Science Fund of Hubei Province (Grant # 2022CFB823), and HUST Independent Innovation Research Fund (Grant # 2021XXJS096).

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. 2023. Reconcile: Round-table conference improves reasoning via consensus among diverse llms. *arXiv preprint arXiv:2309.13007*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*.

Significant Gravitas. 2023. Auto-gpt: An autonomous gpt-4 experiment.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

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

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*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*.

OpenAI. 2022. Introducing chatgpt.

R OpenAI. 2023. Gpt-4 technical report. arxiv 2303.08774. *View in Article*, 2:3.

Anton Osika et al. 2023. Gpt engineer.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Matthew E Peters, Mark Neumann, Robert L Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A Smith. 2019. Knowledge enhanced contextual word representations. *arXiv preprint arXiv:1909.04164*.

Shuofei Qiao, Ningyu Zhang, Runnan Fang, Yujie Luo, Wangchunshu Zhou, Yuchen Eleanor Jiang, Chengfei Lv, and Huajun Chen. 2024. Autoact: Automatic agent learning from scratch via self-planning. *arXiv preprint arXiv:2401.05268*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

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, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2023. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

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.

Yue Wu, So Yeon Min, Yonatan Bisk, Ruslan Salakhutdinov, Amos Azaria, Yuanzhi Li, Tom Mitchell, and Shrimai Prabhumoye. 2023. Plan, eliminate, and track–language models are good teachers for embodied agents. *arXiv preprint arXiv:2305.02412*.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2023a. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.

Zhiheng Xi, Senjie Jin, Yuhao Zhou, Rui Zheng, Songyang Gao, Tao Gui, Qi Zhang, and Xuanjing Huang. 2023b. Self-polish: Enhance reasoning in large language models via problem refinement. *arXiv preprint arXiv:2305.14497*.

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

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, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*.

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*.

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*.

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

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023b. Siren's song in the ai ocean: A survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*.

Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. 2023c. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2023. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*.

Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, et al. 2023. Ghost in the minecraft: Generally capable agents for open-world enviroments via large language models with text-based knowledge and memory. *arXiv preprint arXiv:2305.17144*.