AUTOACT: Automatic Agent Learning from Scratch via Self-Planning

Shuofei Qiao♠♡, Ningyu Zhang♠♡*, Runnan Fang♠♡, Yujie Luo♠♡, Wangchunshu Zhou♣, Yuchen Eleanor Jiang♣, Chengfei Lv♢, Huajun Chen♠♡*

♠Zhejiang University

♡Zhejiang University - Ant Group Joint Laboratory of Knowledge Graph

♣AIWaves Inc. ♢Alibaba Group

{shuofei,zhangningyu}@zju.edu.cn

Abstract

Language agents have achieved considerable performance on various complex tasks. Despite the incessant exploration in this field, existing language agent systems still struggle with costly, non-reproducible data reliance and face the challenge of compelling a single model for multiple functions. To this end, we introduce **AUTOACT**, an automatic agent learning framework that does not rely on large-scale annotated data and synthetic trajectories from closed-source models (e.g., GPT-4). Given limited data with a tool library, **AUTOACT** first automatically synthesizes planning trajectories without any assistance from humans or strong closed-source models. Then, **AUTOACT** leverages a *division-of-labor* strategy to automatically differentiate based on the target task information and synthesized trajectories, producing a sub-agent group to complete the task. We conduct comprehensive experiments with different LLMs, which demonstrates that **AUTOACT** yields better or parallel performance compared to various strong baselines. We even notice that **AUTOACT**, when using the Llama-2-13b model, can achieve performance comparable to that of the zero-shot GPT-3.5-Turbo agent[1].

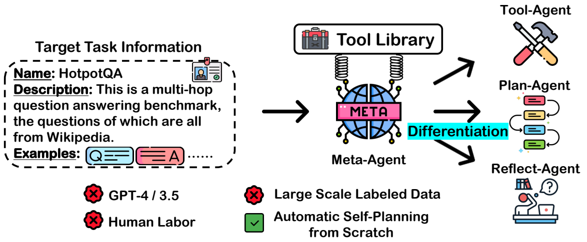

Figure 1: **The basic framework of AUTOACT.** Armed with just one tool library, the META-AGENT can automatically differentiate based on the target task information and produce a sub-agent group that can collaborate to complete the task.

1 Introduction

Language agents (Wang et al., 2023b; Xi et al., 2023; Xie et al., 2023), which leverage the powerful reasoning capabilities (Qiao et al., 2023b; Zhang et al., 2023b) of Large Language Models (LLMs) to generate executable actions for observing the external world, have emerged as essential components of AI systems designed to address intricate interactive tasks (Torantulino, 2023; Osika, 2023; Nakajima, 2023; Tang et al., 2023). The process of endowing LLMs with such interactive capabilities is referred to as *Agent Learning* wherein *planning* plays a pivotal role, which is responsible for decomposing complex tasks (Wei et al., 2022; Zhou et al., 2023a; Yao et al., 2023), invoking external tools (Shen et al., 2023; Lu et al., 2023; Qin et al., 2023), reflecting on past mistakes (Shinn et al., 2023; Madaan et al., 2023), and aggregating information from various sources to reach the final targets. There have been a lot of works (Yao et al., 2023; Wang et al., 2023a; Shen et al., 2023; Chen et al., 2023c) that directly prompt closed-source off-the-shelf LLMs to plan on particular tasks. Despite their convenience and flexibility, closed-source LLMs inevitably suffer from unresolved issues, as their accessibility often comes at a steep price and their black-box nature makes the result reproduction difficult. In light of this, some recent endeavors have shifted their focus towards imbuing open-source models with planning capabilities through fine-tuning (Chen et al., 2023a; Zeng et al., 2023; Yin et al., 2023).

However, despite the existing fine-tuning-based method's achievements, they are not without their limitations. **On the one hand**, training open-source models necessitates a substantial amount of annotated task data and still relies on closed-source models to synthesize planning trajectories. However, fulfilling these requirements in many real-world scenarios, such as private personal bot or sensi-

* Corresponding Author.

[1]Code will be available at https://github.com/zjunlp/AutoAct.

Method	Data Acquisition	Trajectory Acquisition	Planning	Multi-Agent	Fine-Tuning	Generality	Reflection
ReAct (Yao et al., 2023)	User	Prompt	Iterative	✗	✗	✔	✗
Reflexion (Shinn et al., 2023)	User	Prompt	Iterative	✗	✗	✔	✔
Chameleon (Lu et al., 2023)	User	Prompt	Global	✗	✗	✔	✗
HuggingGPT (Shen et al., 2023)	User	Prompt	Global	✗	✗	✔	✗
AutoGPT (Torantulino, 2023)	User	Prompt	Iterative	✗	✗	✔	✔
BOLAA (Liu et al., 2023)	User	Prompt	Iterative	✔	✗	✔	✗
AgentVerse (Chen et al., 2023c)	User	Prompt	Iterative	✔	✗	✔	✗
Agents (Zhou et al., 2023c)	User	Prompt	Iterative	✔	✗	✔	✗
AgentTuning (Zeng et al., 2023)	Benchmark	GPT-4	Iterative	✗	✔	✗	✗
FireAct (Chen et al., 2023a)	Benchmark	GPT-4	Iterative	✗	✔	✗	✔
AutoAct (ours)	User + Self-Instruct	Self-Planning	Iterative	✔	✔	✔	✔

Table 1: **Comparison of related works. Data** and **Trajectory Acquisition**s refer to the way for obtaining training data and trajectories. **Planning** represents the way of planning, parted based on whether each step's action is determined globally or iteratively. **Multi-Agent** indicates whether the framework contains multi-agent. **Fine-Tuning** stands for whether the method is a fine-tuning-based agent learning framework. **Generality** signifies whether the method is applicable to various tasks. **Reflection** denotes whether the planning process incorporates reflection.

tive company business, often proves to be rocky. **On the other hand**, from the perspective of agent framework design, fine-tuning-based methods compel one single language agent to learn all planning abilities, placing even greater pressure on them. These contradict the Simon's principle of bounded rationality (Mintrom, 2015), which states that *"precise social **division-of-labor** and clear individual tasks can compensate for the limited ability of individuals to process and utilize information"*.

To this end, we introduce **AutoAct**, an automatic agent learning framework, which does not rely on large-scale annotated data and synthetic trajectories from closed-source models while incorporating explicit individual tasks with precise *division-of-labor* (see Figure 1). Given a limited set of user-provided data examples, AutoAct starts with a Meta-Agent to obtain an augmented database through self-instruct (Wang et al., 2023c). Then, armed with a prepared tool library, the Meta-Agent can automatically synthesize planning trajectories without any assistance from humans or strong closed-source models. Finally, we propose the *division-of-labor* strategy which resembles *cell differentiation* based on the self-synthesized trajectories (*genes*), where the Meta-Agent acts as a *stem cell* (Colman, 2008) and differentiates into three sub-agents with distinct functions: task decomposition, tool invocation, and self-reflection, respectively. Our differentiation process is essentially a parameter-efficient training process on the self-synthesized trajectories with low consumption resources. We list the differences between AutoAct and prior works in Table 1.

Experiments on complex question-answering tasks with different LLMs demonstrate that the proposed AutoAct yields better or parallel performance compared to various strong baselines. We summarize our main contributions as follows:

- We propose AutoAct, an automatic agent learning framework that does not rely on large-scale annotated data and synthetic trajectories from closed-source models while adhering to the principle of bounded rationality.
- We conduct comprehensive experiments with different LLMs, which demonstrates that AutoAct yields better or parallel performance compared to various strong baselines. We even notice that when using the Llama-2-13b model, AutoAct can achieve performance comparable to that of the zero-shot GPT-3.5-Turbo agent (OpenAI, 2022).
- Extensive empirical analysis demonstrates the effectiveness of our appropriate *division-of-labor* strategy and the trajectory quality generated by AutoAct outperforms that of other methods from multiple aspects.

2 AutoAct

Overview. As shown in Figure 2, AutoAct only requires target task information and a language agent (we name it Meta-Agent) to initiate its work. The Meta-Agent first augments the task data from scratch by self-instruct. Furthermore, with a tool library available, the Meta-Agent conducts automatic agent learning by differentiating into sub-agents with distinct functionalities and enabling them to perform group task-specific planning. We name this process as *self-planning*. Below is a detailed introduction of AutoAct. **Note that all symbols used are globally defined**.

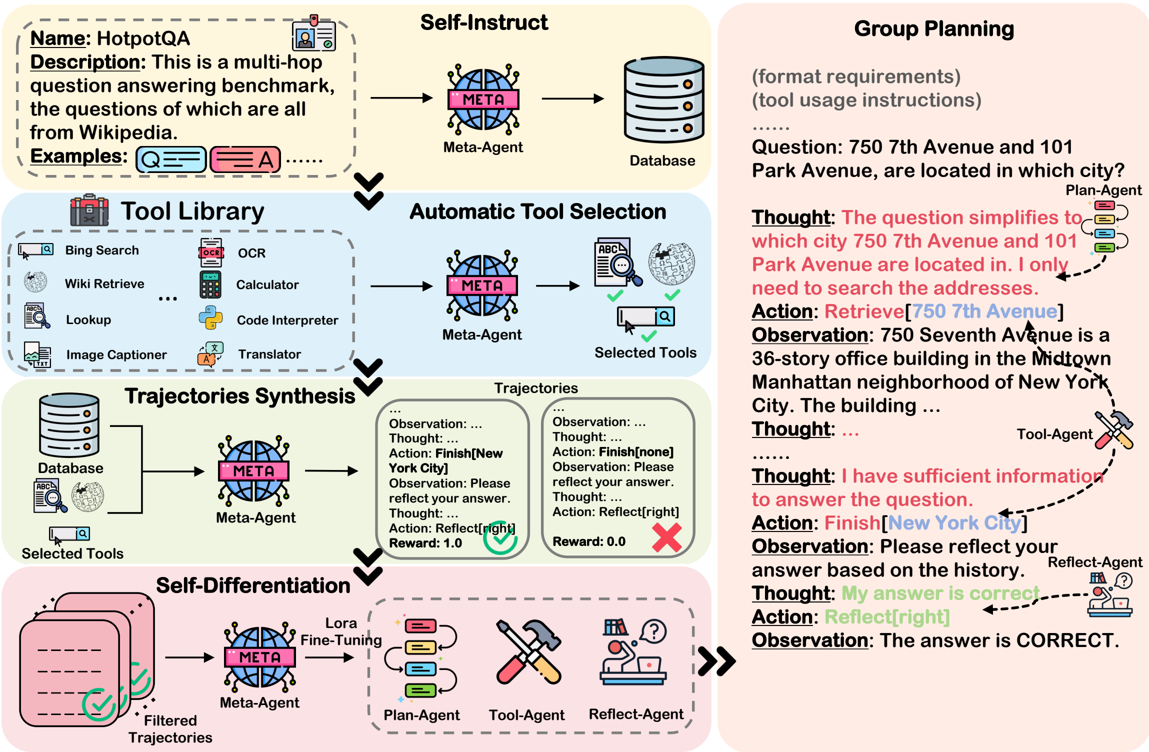

Figure 2: **The overview of our proposed framework AUTOACT.** We initiate with **self-instruct** to extend the task database from scratch. Then **self-planning** is applied to conduct automatic agent learning, including *automatic tool selection*, *trajectories synthesis*, *self-differentiation* and *group planning*. Our self-differentiation is a parameter-efficient fine-tuning process to achieve resource-efficient learning.

2.1 Critical Components of AUTOACT

META-AGENT. The META-AGENT stands at the central position of our AUTOACT framework. It is responsible for all the preparatory work before self-differentiation and serves as the backbone model for the self-differentiation process. Given limited target task information and a pre-prepared tool library, the META-AGENT can differentiate into an agent group capable of collaborating to accomplish the target task. In AUTOACT, the META-AGENT can be initialized with any open-source model.

Target Task Information. In this paper, we mainly focus on agent learning from scratch, which means the task information at hand is quite limited, primarily encompassing three aspects: task name $\mathcal{M}$, task description $\mathcal{P}$, task data examples $\mathcal{C}$. Concretely, $\mathcal{P}$ represents a detailed description of the task's characteristics, properties, and other relevant information. $\mathcal{C} = \{q_i, a_i\}_{i=1}^{|\mathcal{C}|}$ indicates $|\mathcal{C}|$ question-answer example pairs of the task, where $|\mathcal{C}|$ is very small which users can effortlessly provide (e.g., a few demonstrations). For a more in-depth view of task information, please refer to Appendix B. Note that the task information serves as the only user-provided knowledge of the task for AUTOACT to conduct automatic agent learning.

Tool Library. To facilitate our agents in automatic task planning, we provide a comprehensive tool library at their disposal. The tool library can be denoted as $\mathcal{T} = \{m_i, d_i, u_i\}_{i=1}^{|\mathcal{T}|}$, where m represents the name of each tool, d defines the functionality of each tool, u details the usage instruction of each tool, and $|\mathcal{T}|$ stands for the tools amount of the library. In our automatic procedure, the META-AGENT has the autonomy to select appropriate tools from the tool library based on the task information. Users also have the option to expand the tool library according to their specific needs, allowing for more flexible utilization. We list part of our tool library in Appendix C.

2.2 Starting from Scratch via Self-Instruct

To acquire a sufficient amount of task data and provide an ample training resource, it is necessary to augment the data based on the examples at hand. We accomplish this process through self-instruct. Initially, the database $\mathcal{D}$ is set to be equal to the task data examples $\mathcal{C}$, with $\mathcal{C}$ as the seed for data generation. In each round, the META-AGENT generates new question-answer pairs by few-shot prompting, and the few-shot prompt examples are randomly sampled from $\mathcal{D}$. The generated data will be added to $\mathcal{D}$ followed by filtering, with the exclusion of format erroneous and duplicate data before its inclusion. Eventually, we obtain a database $\mathcal{D} = \{q_i, a_i\}_{i=1}^{|\mathcal{D}|}$, where the number of data $|\mathcal{D}|$ satisfies $|\mathcal{D}| \gg |\mathcal{C}|$. The prompt we use for self-instruct can be seen in Appendix D.1.

2.3 Automatic Agent Learning via Self-Planning

Automatic Tool Selection. With the tool library at hand, we ask the META-AGENT to select applicable tools for each task automatically. Specifically, we put $\mathcal{T} = \{m_i, d_i, u_i\}_{i=1}^{|\mathcal{T}|}$ in the form of a tool list as part of the prompt. Along with $\mathcal{T}$, the prompt also includes the task's description $\mathcal{C}$. Finally, we instruct the META-AGENT to select an appropriate set of tools $\mathcal{T}_s$ ($\mathcal{T}_s \subset \mathcal{T}$) to wait for synthesizing trajectories. The prompt we use for automatic tool selection can be seen in Appendix D.2.

Trajectories Synthesis. Without relying on closed-source models, we enable the META-AGENT to synthesize planning trajectories on its own. Equipped with $\mathcal{T}_s$, we instruct the META-AGENT to synthesize trajectories in a zero-shot manner on the database $\mathcal{D}$ adhering to the format of `Thought-Action-Observation` as defined in Yao et al. (2023). In order to obtain high-quality synthesized trajectories, we filter out all the trajectories with `reward` < 1 and collect trajectories with exactly correct answers (`reward` $= 1$) as the training source for self-differentiation. The prompt for trajectories synthesis can be seen in Appendix D.3.

Self-Differentiation. In order to establish a clear *division-of-labor*, we leverage synthesized planning trajectories to differentiate the META-AGENT into three sub-agents with distinct functionalities:

- **PLAN-AGENT** π_{plan} undertakes task decomposition and determines which tool to invoke in each planning loop (Eq.2).
- **TOOL-AGENT** π_{tool} is responsible for how to invoke the tool (Eq.3) by deciding the parameters for the tool invocation.
- **REFLECT-AGENT** π_{reflect} engages in reflection by considering all the historical trajectories and providing a reflection result (Eq.4).

We assume that the planning loop at time t can be denoted as (τ_t, α_t, o_t), where τ denotes `Thought`, α signifies `Action`, and o represents `Observation`. α can be further expressed as (α^m, α^p), where α^m is the name of the action, and α^p is the parameters required to perform the action. Then the historical trajectory at time t can be signaled as:

$$\mathcal{H}_t = (\tau_0, \alpha_0, o_0, \tau_1, ..., \tau_{t-1}, \alpha_{t-1}, o_{t-1}). \quad (1)$$

Eventually, supposing that the prompts of target task information, planning format requirements, and the question are all combined as $\mathcal{S}$, the responsibilities of each sub-agent can be defined as:

$$\tau_t, \alpha_t^m = \pi_{\text{plan}}(\mathcal{S}, \mathcal{T}_s, \mathcal{H}_t), \quad (2)$$

$$\alpha_t^p = \pi_{\text{tool}}(\mathcal{S}, \mathcal{T}_s, \mathcal{H}_t, \tau_t, \alpha_t^m), \quad (3)$$

$$\tau^r, \alpha^r = \pi_{\text{reflect}}(\mathcal{S}, \mathcal{T}_s, \mathcal{H}), \quad (4)$$

where τ^r and α^r represent the thought and action of the reflection process respectively, and $\mathcal{H}$ is the planning history after finishing the answer. The trajectories can be reorganized based on the responsibilities above and fed to the META-AGENT for self-differentiation. Our differentiation is a parameter-efficient fine-tuning process to achieve resource-efficient learning. Particularly, for each sub-agent, we train a specific LoRA (Hu et al., 2022).

Group Planning. After obtaining the task-specific sub-agents, any new question is processed through group planning among the sub-agents to achieve the desired outcome. Once the tool name α_t^m generated by the PLAN-AGENT is triggered at time t, the TOOL-AGENT is roused to determine the parameters α_t^p transferred to the specific tool. The return result of the tool is treated as the observation o_t and handed to the PLAN-AGENT. After the collaboration between the PLAN-AGENT and TOOL-AGENT finishes with a prediction, the REFLECT-AGENT comes on the stage to reflect on the history and provide a reflection result contained in the reflection action α^r. If the reflection result indicates that the prediction is correct, the whole planning process concludes. Otherwise, the PLAN-AGENT and TOOL-AGENT will continue the planning based on the reflection information. The specific sequence of the group planning process can be referred to the trajectory example on the right side of Figure 2.

3 Experimental Setup

Tasks. We evaluate our method on HotpotQA (Yang et al., 2018) and ScienceQA (Lu et al., 2022).

Backbone	Method	HotpotQA				ScienceQA			
		Easy	Medium	Hard	All	G1-4	G5-8	G9-12	All
GPT-3.5 Turbo	CoT	48.21	44.52	34.22	42.32	60.83	55.83	65.00	60.56
	Zero-Shot Plan*	50.71	45.17	38.23	44.70	76.67	61.67	78.33	72.22
Llama-2 7B-chat	CoT	35.80	26.69	18.20	26.90	59.17	50.00	59.17	56.11
	ReAct	25.14	19.87	17.39	20.80	52.50	47.50	54.17	51.39
	Chameleon	37.73	26.66	21.83	28.74	59.17	54.17	**60.00**	57.78
	Reflexion	35.55	28.73	24.35	29.54	60.83	**57.50**	59.17	**58.06**
	BOLAA	27.55	21.47	21.03	23.35	58.33	53.33	52.50	54.72
	FireAct	**38.83**	**30.19**	22.30	**30.44**	50.83	53.33	60.00	54.72
	AUTOACT	34.60	27.73	**25.22**	29.18	**62.50**	49.17	48.33	53.33
Llama-2 13B-chat	CoT	37.90	25.28	21.64	28.27	61.67	52.50	69.17	61.11
	ReAct	28.68	22.15	21.69	24.17	57.27	51.67	65.00	57.98
	Chameleon	40.01	25.39	22.82	29.41	69.17	60.83	73.33	67.78
	Reflexion	44.43	37.50	28.17	36.70	67.50	64.17	73.33	68.33
	BOLAA	33.23	25.46	25.23	27.97	60.00	54.17	65.83	60.00
	FireAct	45.83	38.94	26.06	36.94	60.83	57.50	67.50	61.94
	AUTOACT	**47.29**	**41.27**	**32.92**	**40.49**	**70.83**	**66.67**	**76.67**	**71.39**
Llama-2 70B-chat	CoT	45.37	36.33	32.27	37.99	74.17	64.17	75.83	71.39
	ReAct	39.70	37.19	33.62	36.83	64.17	60.00	72.50	65.56
	Chameleon	46.86	38.79	34.43	40.03	77.83	69.17	76.67	74.56
	Reflexion	48.01	46.35	35.64	43.33	75.83	67.50	78.33	73.89
	BOLAA	46.44	37.29	33.49	39.07	70.00	67.50	75.00	70.83
	FireAct	50.82	41.43	35.86	42.70	72.50	68.33	75.00	71.94
	AUTOACT	**56.94**	**50.12**	**38.35**	**48.47**	**82.50**	**72.50**	**80.83**	**78.61**

Table 2: **Main results of AUTOACT compared to various baselines** on HotpotQA and ScienceQA. The icon indicates prompt-based agent learning without fine-tuning, while means fine-tuning-based agent learning. denotes single-agent learning and symbolizes multi-agent learning. The best results of each model are marked in **bold** and the second-best results are marked with underline. *We compare the zero-shot plan performance of GPT-3.5-Turbo to ensure fairness in our evaluation since our setup does not include annotated trajectory examples.

HotpotQA is a multi-hop QA task challenging for rich background knowledge, the answer of which is usually a short entity or yes/no. Following Liu et al. (2023), we randomly select 300 dev questions divided into three levels for evaluation, with 100 questions in each level. For HotpotQA, the `reward` $\in [0, 1]$ is defined as the F1 score grading between the prediction and ground-truth answer. ScienceQA is a multi-modal QA task spanning various scientific topics. We also divide the test set into three levels based on the grade, with 120 randomly sampled data in each level. Since ScienceQA is a multi-choice QA task, the `reward` $\in \{0, 1\}$ is exactly the accuracy. Note that due to the limitations of LMs in generating images, for ScienceQA, during the self-instruct stage, we directly generate captions for the images instead.

Baselines. We choose the open-source Llama-2 models (Touvron et al., 2023) as the backbones of our META-AGENT and sub-agents. The compared baselines are as follows: 1) **CoT** (Wei et al., 2022), the naive Chain-of-Thought reasoning method. 2) **REACT** (Yao et al., 2023), a well-known single-agent framework based on few-shot learning that performs planning and action iteratively. 3) **Chameleon** (Lu et al., 2023), another few-shot single-agent framework that performs planning before action. 4) **Reflexion** (Shinn et al., 2023), a single-agent framework to reinforce language agents through linguistic feedback. 5) **BOLAA** (Liu et al., 2023), a multi-agent framework that customizes different agents through prompts. 6) **FIREACT** (Chen et al., 2023a), a single-agent framework with fine-tuning on diverse kinds of trajectories generated by GPT-4 (OpenAI, 2023). 7) **GPT-3.5-Turbo** (OpenAI, 2022). To ensure fairness, we maintain an equal training trajectory volume of 200 for FIREACT and AUTOACT (200 synthesized data). As Reflexion provides answer correctness labels during reflection but other methods including AUTOACT do not, we test all the other methods twice and choose the correct one for evaluation. For all the prompt-based baselines, we uniformly provide two examples in the prompt.

Training Setups. We fine-tune all our models with LoRA (Hu et al., 2022) in the format proposed in Alpaca (Taori et al., 2023). Our fine-tuning framework leverages FastChat (Zheng et al., 2023) using DeepSpeed (Rasley et al., 2020). We set the learning rate of 1e-4 and the sequence length of 4096 for all the model scales. The training epoch for the 13b and 7b models is 5 and for the 70b model is 3. The batch size is 4 for the 13b and 7b models and 1 for the 70b model. We use the AdamW optimizer (Loshchilov and Hutter, 2019) with a cosine learning scheduler. We detail the hyper-parameters for training in Appendix A.

4 Results

Compare to Prompt-based Agent Learning Baselines. As shown in Table 2, the 13b and 70b models consistently outperform various prompt-based baselines. The 70b model even surpasses the agent performance of GPT-3.5-Turbo, achieving a rise of $\uparrow$3.77% on HotpotQA and $\uparrow$6.39% on ScienceQA. The performance of the 7b model is comparable to other methods to some extent. However, it fails to deliver the same impressive results due to its limited self-instruct and self-planning capabilities. Therefore, whether in a single-agent or multi-agent architecture, prompt-based methods relying on few-shot demonstrations fail to precisely customize the behavior of the agent, which is also supported by the fact that FIREACT widely outperforms REACT and BOLAA in the context of iterative planning. In addition, our investigation reveals a visible disparity in open-source models between the performance of many prompt-based planning baselines (relying on various external tools) and CoT (relying on the models' intrinsic reasoning abilities). This discrepancy underscores the formidable challenge of unlocking planning capabilities by prompting.

Compare to Fine-tuning-based Agent Learning Baselines. Further focusing on FIREACT in Table 2, despite the assistance of GPT-4, FIREACT's approach of assigning the entire planning task to a single model proves to be burdensome. As a result, its performance on ScienceQA even falls short compared to the prompt-based global planning method, Chameleon. AUTOACT employs self-differentiation to decouple the planning process and reaches a clear *division-of-labor* among sub-agents for group planning, resulting in a improvement than FIREACT, with an enhancement of $\uparrow$5.77% on HotpotQA and $\uparrow$6.67% on ScienceQA with 70b model. Additionally, AUTOACT achieves self-planning without relying on closed-source models and large-scale labeled datasets, which paves the way for automatic agent learning with open-source models from scratch. In ablation study (§4) and human evaluation (§5), we will further validate that the quality of trajectories synthesized by AUTOACT is not inferior to FIREACT trained on trajectories synthesized using GPT-4.

	HotpotQA	ScienceQA
AUTOACT	48.47	78.61
- reflection	$45.66_{\downarrow 2.81}$	$75.28_{\downarrow 3.33}$
- multi	$42.81_{\downarrow 5.66}$	$69.72_{\downarrow 8.89}$
- fine-tuning	$32.84_{\downarrow 15.63}$	$61.94_{\downarrow 16.67}$
- filtering	$32.51_{\downarrow 15.96}$	$59.17_{\downarrow 19.44}$

Table 3: **Approach ablations of AUTOACT.** ***- reflection*** symbolizes removing the reflect-agent in AUTOACT. ***- multi*** denotes feeding all the differentiated data into one model for fine-tuning. ***- fine-tuning*** indicates zero-shot prompt planning with the three agents defined in AUTOACT. ***- filtering*** represents self-differentiation on all the trajectories generated in zero-shot planning without filtering wrong cases.

Single-agent Learning vs. Multi-agent Learning. Under identical settings, multi-agent architectures generally exhibit better performance than single-agent (REACT vs. BOLAA, FIREACT vs. AUTOACT), which aligns with Simon's theory of bounded rationality. Seemingly contrary to expectations, despite being a single-agent architecture, Chameleon outperforms BOLAA (even FIREACT on ScienceQA). However, we analyze that this can be attributed to the way it leverages tools. In Chameleon, the process of deciding tool parameters is considered a form of tool invocation, and specialized few-shot prompts are designed to guide the model through this process. From this aspect, Chameleon, despite being nominally a single-agent architecture, exhibits characteristics that resemble a multi-agent architecture, which does not contradict our initial conclusion. Indeed, we can also explain from the perspective of optimizing objectives. Another well-known economic principle, Goodhart's Law (Goodhart, 1984), states that *"When a measure becomes a target, it ceases to be a good measure"*. This implies that optimizing one objective on the same agent will inevitably harm other optimization objectives to some extent. Therefore, it

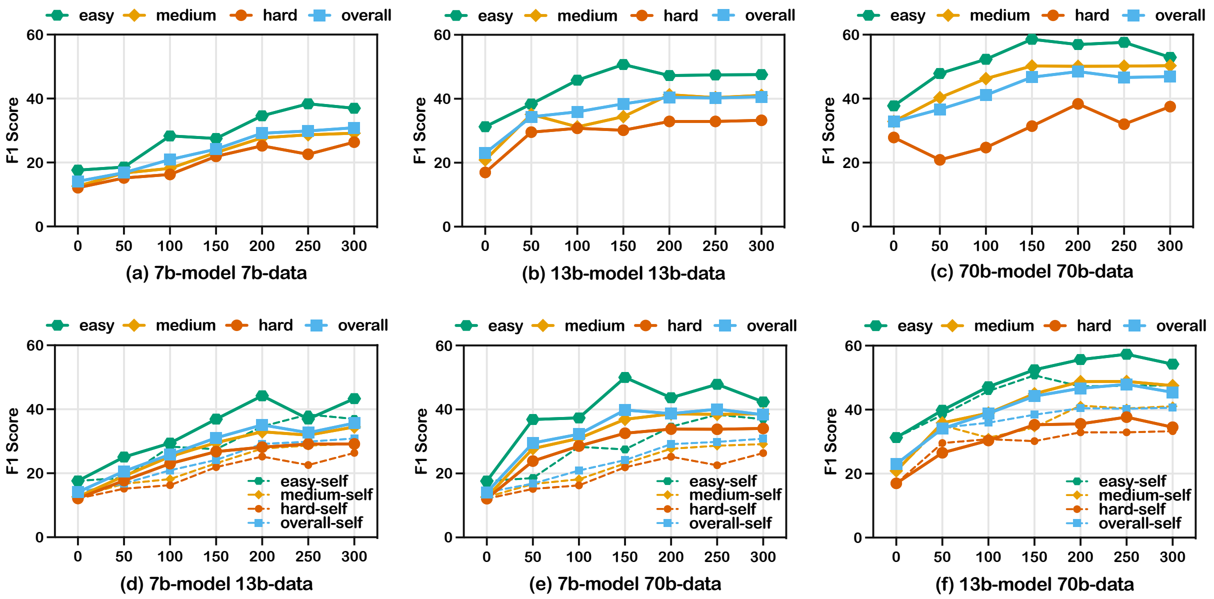

Figure 3: **Performance of AUTOACT on different training data scales.** (a-c) represents the results of the model trained on self-synthesized trajectories. (d-f) represents the results of the model trained on trajectories synthesized by a stronger model, where the dashed line is the baseline trained on self-synthesized trajectories.

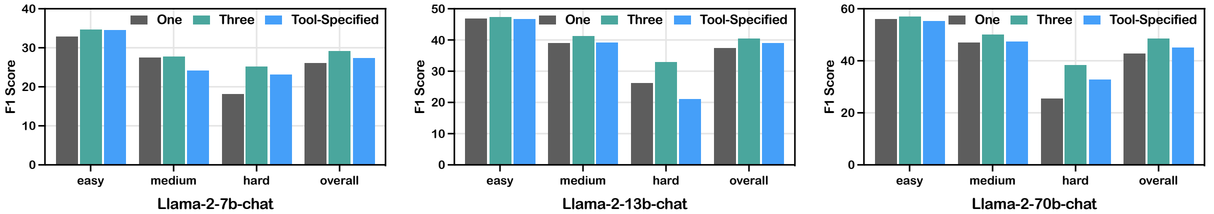

Figure 4: **Performance of AUTOACT based on different degrees of labor division.** ***One*** is training a single model with all the differentiated data. ***Three*** represents the differentiation into three agents: plan, tool, and reflect. ***Tool Specified*** indicates further differentiating the tool-agent with one tool, one agent.

is not optimal to optimize all objectives on a single agent, and a multi-agent architecture happens to address this issue. However, we analyze in §5 that excessive fine-grained *division-of-labor* is not the best approach and a moderate division of labor benefits group performance.

Approach Ablations. Table 3 presents the performance of AUTOACT on the 70b model after removing certain key processes. It can be observed that the least impactful removal is the - *reflect*. We investigate that in the zero-shot scenario, the model tends to be over-confident in its answers. It typically only recognizes its errors when there are obvious formatting mistakes or significant repetitions in the planning process. Consistent with previous findings, the removal of the - *multi* agents leads to a noticeable decrease in performance. A more exciting discovery is that the results of - *multi* are comparable to those of FIREACT. This indirectly suggests that the trajectory quality generated by the 70b model may be no worse than that of GPT-4. As expected, the performance deteriorates after - *fine-tuning*, which once again confirms the previous conclusion. To demonstrate the necessity of filtering out planning error data, we specifically remove the filtering process (- *filtering*) to examine the performance of AUTOACT. The results indicate that the damage caused by training on unfiltered data is even greater than that of - *fine-tuning*.

5 Analysis

Larger training data scale does not necessarily mean better results. We evaluate the influence of different training data scales on the performance of self-planning in Figure 3(a-c). It can be observed that the overall performance of different models steadily improves as the training data scale increases. However, it goes to stability with minimal fluctuations once the data scale exceeds 200. We speculate that this may be due to the limited

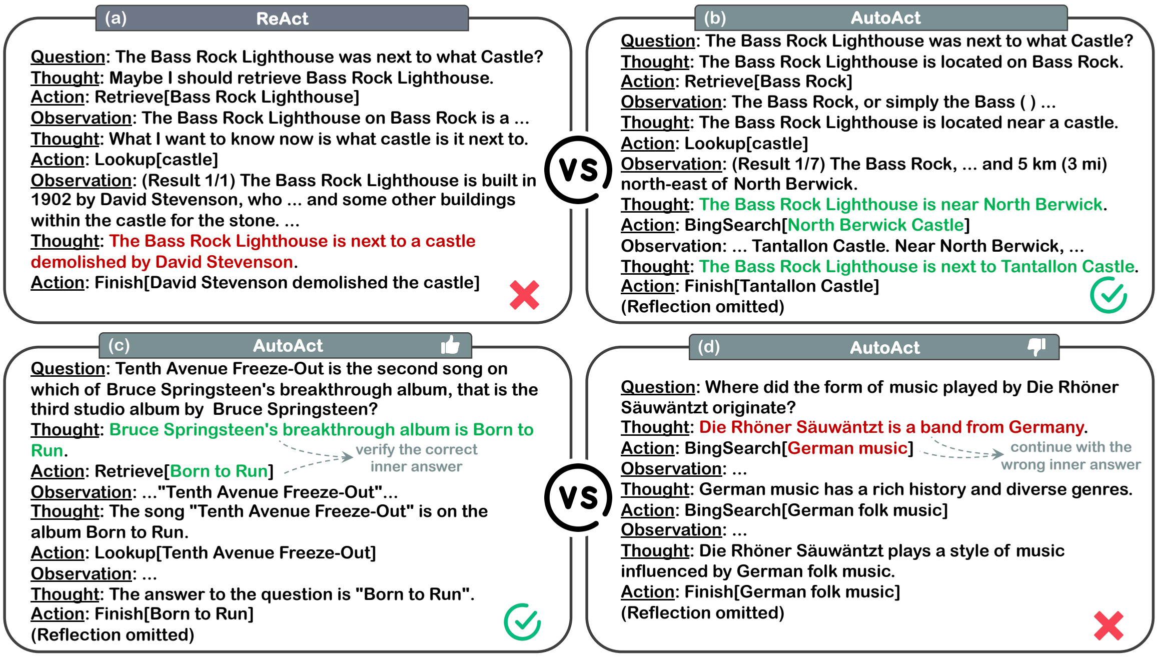

Figure 5: **Case study.** AUTOACT (b) successfully addresses the failure in REACT (a) by employing a more scientific combination of tools and making more accurate tool invocations. With more planning rounds, AUTOACT (c) can validate its inner answers by continuing more rounds of self-verification. While this can also lead to a longer context, gradually deviating AUTOACT (d) from the original question.

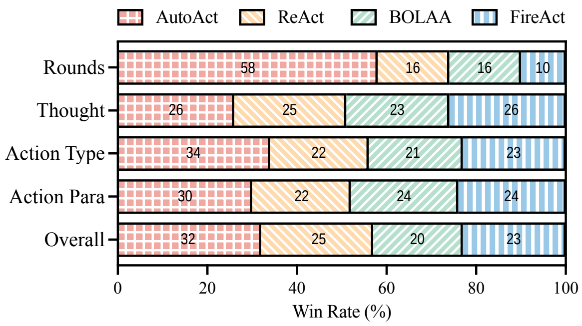

Figure 6: **Human evaluation of trajectories** generated by Llama-2-70b-chat on HotpotQA. We compare the number of planning rounds, the logical correctness of thoughts, action types, action parameters, and the overall coherence of each trajectory.

ability of naive self-instruct to boost internal knowledge of the language model. The process from self-instruct to self-planning can be seen as the distillation and rumination of internal model knowledge. As the training data increases, the knowledge which can be extracted through self-instruct decreases. Despite our efforts to filter out duplicate data, similar data will inevitably reoccur. So the mindless increase in data scale can lead to a significant surge in similar data, which undermines the benefits of increasing the data scale and ultimately makes it challenging to improve model performance or even leads to over-fitting (see the slight decrease in overall performance on 70b at 250 and 300). To further confirm the role of training data, we decouple the models from the training data and evaluate their training results on trajectories synthesized by other stronger models. From Figure 3(d-f), we can see consistent conclusions with previous findings. The performance improvement becomes increasingly challenging beyond a dataset size of 200, regardless of the size matching between the backbone model and the data-synthetic model. Therefore, maximizing the diversity of the synthesized data in the database may be a key improvement direction for AUTOACT. Some previous works (Xu et al., 2023; Yu et al., 2023; Li et al., 2023) have attempted to improve upon the naive self-instruct, but none of them have focused on better mobilizing the language model's internal knowledge without external information, and we leave this for our future work.

Moderate division of labor benefits group planning performance. To explore the impact of the different granularity of self-differentiation and group planning, we further subdivide the tool agent, assigning dedicated agents to manipulate each specific tool. We contrast the performance of *One* agent, *Three* agents (vanilla AUTOACT), and the *Tool-Specified* setting in Figure 4. It can be observed that finer task allocation does not necessarily lead to better performance. This is consistent with the findings in Qiao et al. (2023a) which indicate that multi-tool joint learning often outperforms single-tool individual learning. Therefore, appropriate differentiation (*Three*) can alleviate the pressure on individual agents, aligning with Simon's principle of bounded rationality. However, excessive differentiation (*Tool-Specified*) not only fails to achieve better results but can sometimes even be less effective than not differentiating (*One*) at all. Moreover, it appears that the performance loss of tool-specific agents compared to the three-agent approach is more significant on harder problems. This is because challenging problems typically require more planning steps and higher levels of collaboration among tools. By unifying tool invocations under one agent, it becomes possible to effectively learn the interplay and interconnectedness between tools, thereby compensating for potential information gaps arising from using tool-specific agents.

Human Evaluation. To get a deeper understanding of the capability of AUTOACT, we manually compare the quality of trajectories generated by different methods from five aspects. We ask five NLP experts to individually select the optimal trajectories generated by all methods in terms of the number of planning rounds, the logical correctness of thoughts, action types, action parameters, and overall coherence. The final results are determined based on major votes. During the evaluation, it is hidden for the evaluators of the correspondence between the trajectories and the methods. We delete the reflection-related parts from the trajectories generated by AUTOACT and randomly shuffle the order of trajectories of each method in each data to minimize the potential bias as much as possible.

The evaluation results are depicted in Figure 6 and we further provide some cases in Figure 5. We can observe a clear advantage for AUTOACT over other methods in determining the action type and action parameters. This indicates that decoupling the missions of planning and tool invocation can lead to better performance for both, alleviating the overwhelming pressure on a single agent. A more intuitive comparison can be observed in Figure 5 (a)(b). AUTOACT successfully addresses the failure in REACT by employing a more scientific combination of tools and making more accurate tool invocations. Furthermore, AUTOACT tends to consume more planning rounds than other methods. This allows AUTOACT to perform better on harder problems. However, this characteristic can be a double-edged sword when it comes to simple problems. A surprising aspect is that AUTOACT can validate its inner (*thought*) answers by continuing more rounds of self-verification (see Figure 5 (c)). Unfortunately, this can also lead to a longer context, gradually deviating AUTOACT from the original question (see Figure 5 (d)).

6 Related Work

LLM-Powered Agents. The rise of LLMs has positioned them as the most promising key to unlocking the door to Artificial General Intelligence (AGI), providing robust support for the development of LLM-centered AI agents (Wang et al., 2023b; Xi et al., 2023; Wang et al., 2023d,e). Related works focus primarily on agent planning (Yao et al., 2023; Song et al., 2022; Chen et al., 2023a), external tools harnessing (Patil et al., 2023; Qiao et al., 2023a; Qin et al., 2023), collective intelligence among multi-agents (Liang et al., 2023; Liu et al., 2023; Chen et al., 2023b), human and social property inside agents (Zhang et al., 2023a; Park et al., 2023; Mao et al., 2023), etc. However, despite their success, existing methods still face two major troubles. **Firstly**, most agents heavily rely on prompts for customization, which makes it difficult to precisely tailor the behavior of the agent, resulting in unexpected performance at times. **Secondly**, each agent is compelled to master all skills, making it challenging for the agent to achieve expertise in every domain. In response, our approach leverages a proper *division-of-labor* strategy and fine-tuning each sub-agent to equip different agents with distinct duties. These agents collaborate to accomplish tasks orderly and effectively.

Agent Fine-Tuning. Despite the vast interest in LLM-powered agents, the construction of agents through fine-tuning has received limited attention. Most early works concentrate on fine-tuning to optimize the model's reasoning capabilities (Liu et al., 2022; Fu et al., 2023) or tool proficiency (Patil et al., 2023; Qiao et al., 2023a; Qin et al., 2023). Recently, Chen et al. (2023a) attempt to fine-tune agents with diverse tasks and trajectories for a better planning capability. Zeng et al. (2023) apply a hybrid instruct-tuning strategy that enhances the agent's abilities while preserving its generalization. **However**, these methods still require a model to be a generalist. Moreover, the trajectories in the training data are annotations from GPT-3.5/GPT-4 (OpenAI, 2022, 2023), which incurs significant costs. Our approach enables the META-AGENT to autonomously synthesize trajectories and achieve self-planning in a zero-shot manner, without relying on closed-source models or human labor.

7 Conclusion

In this paper, we propose AUTOACT, an automatic agent learning framework that does not rely on large-scale annotated data and synthetic trajectories from closed-source models, while alleviating the pressure on individual agents by explicitly dividing the workload, thereby enhancing the collective performance of multi-agents. Experimental results demonstrate that AUTOACT performs superior on challenging question-answering benchmarks compared to various strong baselines, and AUTOACT with Llama-2-13b model can even obtain comparable performance of zero-shot GPT-3.5-Turbo agent. Extensive analysis indicates the effectiveness of our appropriate *division-of-labor* strategy, with the trajectory quality generated by AUTOACT significantly outperforming that of other methods from multiple aspects.

Limitations

In this paper, we focus on constructing an automatic agent learning framework dubbed AUTOACT. Despite our best efforts, this paper may still have some remaining limitations.

Tasks and Backbones. For experimental convenience, we only evaluate our method on complex question-answering tasks with the Llama-2-chat model series. However, there are many other interactive scenarios and backbone models beyond these. Other complex tasks include web (Yao et al., 2022; Zhou et al., 2023b), household (Puig et al., 2018; Shridhar et al., 2021), robotics (Ichter et al., 2022), etc., and more backbone models include Vicuna (Zheng et al., 2023), ChatGLM (Du et al., 2022), Mistrial (Jiang et al., 2023), etc. We plan to conduct further research on applying AUTOACT to a wider range of tasks and backbones in the future.

Boosting Knowledge via Self-Instruct. As analyzed in §5, the planning performance of AUTOACT can be limited by the model's ability to access internal knowledge through self-instruct. While the current phenomenon allows us to achieve lightweight self-differentiation in terms of parameters and data, it is still necessary to research how to enrich knowledge as much as possible within the constraints of limited data.

Self-Improvement. Recent research has shed light on self-improvement techniques that enhance LLMs by iteratively training them on self-synthesized data (Zelikman et al., 2022; Huang et al., 2023; Gülçehre et al., 2023; Aksitov et al., 2023). This approach allows the model to continually learn and refine its performance on its own. Our approach also involves training on self-synthesized data and we believe that further using the iterative thinking of self-improvement will significantly enhance the performance of our method.

References

Renat Aksitov, Sobhan Miryoosefi, Zonglin Li, Daliang Li, Sheila Babayan, Kavya Kopparapu, Zachary Fisher, Ruiqi Guo, Sushant Prakash, Pranesh Srinivasan, Manzil Zaheer, Felix Yu, and Sanjiv Kumar. 2023. Rest meets react: Self-improvement for multi-step reasoning llm agent.

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

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

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chen Qian, Chi-Min Chan, Yujia Qin, Yaxi Lu, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2023c. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents. *CoRR*, abs/2308.10848.

Alan Colman. 2008. Human embryonic stem cells and clinical applications. *Cell Research*, 18(1):S171–S171.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. GLM: general language model pretraining with autoregressive blank infilling. pages 320–335.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10421–10430. PMLR.

C. A. E. Goodhart. 1984. *Problems of Monetary Management: The UK Experience*, pages 91–121. Macmillan Education UK, London.

Çaglar Gülçehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. 2023. Reinforced self-training (rest) for language modeling. *CoRR*, abs/2308.08998.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2023. Large language models can self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 1051–1068. Association for Computational Linguistics.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J. Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. 2022. Do as I can, not as I say: Grounding language in robotic affordances. In *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pages 287–318. PMLR.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Luke Zettlemoyer, Omer Levy, Jason Weston, and Mike Lewis. 2023. Self-alignment with instruction backtranslation. *CoRR*, abs/2308.06259.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu, and Shuming Shi. 2023. Encouraging divergent thinking in large language models through multi-agent debate. *CoRR*, abs/2305.19118.

Jiacheng Liu, Alisa Liu, Ximing Lu, Sean Welleck, Peter West, Ronan Le Bras, Yejin Choi, and Hannaneh Hajishirzi. 2022. Generated knowledge prompting for commonsense reasoning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 3154–3169. Association for Computational Linguistics.

Zhiwei Liu, Weiran Yao, Jianguo Zhang, Le Xue, Shelby Heinecke, Rithesh Murthy, Yihao Feng, Zeyuan Chen, Juan Carlos Niebles, Devansh Arpit, Ran Xu, Phil Mui, Huan Wang, Caiming Xiong, and Silvio Savarese. 2023. BOLAA: benchmarking and orchestrating llm-augmented autonomous agents. *CoRR*, abs/2308.05960.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *NeurIPS*.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models. *CoRR*, abs/2304.09842.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. *CoRR*, abs/2303.17651.

Shengyu Mao, Ningyu Zhang, Xiaohan Wang, Mengru Wang, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2023. Editing personality for llms. *CoRR*, abs/2310.02168.

Michael Mintrom. 2015. 12Herbert A. Simon, Administrative Behavior: A Study of Decision-Making Processes in Administrative Organization. In *The Oxford Handbook of Classics in Public Policy and Administration*. Oxford University Press.

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

OpenAI. 2022. Chatgpt: Optimizing language models for dialogue. `https://openai.com/blog/chatgpt/`.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Anton Osika. 2023. Gpt-engineer. `https://github.com/AntonOsika/gpt-engineer`.

Joon Sung Park, Joseph C. O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, UIST 2023, San Francisco, CA, USA, 29 October 2023- 1 November 2023*, pages 2:1–2:22. ACM.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *CoRR*, abs/2305.15334.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. 2018. Virtualhome: Simulating household activities via programs. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8494–8502. Computer Vision Foundation / IEEE Computer Society.

Shuofei Qiao, Honghao Gui, Huajun Chen, and Ningyu Zhang. 2023a. Making language models better tool learners with execution feedback. *CoRR*, abs/2305.13068.

Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2023b. Reasoning with language model prompting: A survey. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 5368–5393. Association for Computational Linguistics.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *CoRR*, abs/2307.16789.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 3505–3506. ACM.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving AI tasks with chatgpt and its friends in huggingface. *CoRR*, abs/2303.17580.

Noah Shinn, Beck Labash, and Ashwin Gopinath. 2023. Reflexion: language agents with verbal reinforcement learning. *CoRR*, abs/2303.11366.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. 2021. Alfworld: Aligning text and embodied environments for interactive learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M. Sadler, Wei-Lun Chao, and Yu Su. 2022. Llm-planner: Few-shot grounded planning for embodied agents with large language models. *CoRR*, abs/2212.04088.

Xiangru Tang, Anni Zou, Zhuosheng Zhang, Yilun Zhao, Xingyao Zhang, Arman Cohan, and Mark Gerstein. 2023. Medagents: Large language models as collaborators for zero-shot medical reasoning. *CoRR*, abs/2311.10537.

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`.

Torantulino. 2023. Autogpt: build & use ai agents. `https://github.com/Significant-Gravitas`.

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

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. Voyager: An open-ended embodied agent with large language models. *CoRR*, abs/2305.16291.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. 2023b. A survey on large language model based autonomous agents. *CoRR*, abs/2308.11432.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023c. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13484–13508. Association for Computational Linguistics.

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. 2023d. Describe, explain, plan and select: interactive planning with llms enables open-world multi-task agents. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Zihao Wang, Shaofei Cai, Anji Liu, Yonggang Jin, Jinbing Hou, Bowei Zhang, Haowei Lin, Zhaofeng He, Zilong Zheng, Yaodong Yang, et al. 2023e.

Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models. *arXiv preprint arXiv:2311.05997*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huan, and Tao Gui. 2023. The rise and potential of large language model based agents: A survey. *CoRR*, abs/2309.07864.

Tianbao Xie, Fan Zhou, Zhoujun Cheng, Peng Shi, Luoxuan Weng, Yitao Liu, Toh Jing Hua, Junning Zhao, Qian Liu, Che Liu, Leo Z. Liu, Yiheng Xu, Hongjin Su, Dongchan Shin, Caiming Xiong, and Tao Yu. 2023. Openagents: An open platform for language agents in the wild. *CoRR*, abs/2310.10634.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *CoRR*, abs/2304.12244.

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. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. In *NeurIPS*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

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. *CoRR*, abs/2311.05657.

Yue Yu, Yuchen Zhuang, Jieyu Zhang, Yu Meng, Alexander Ratner, Ranjay Krishna, Jiaming Shen, and Chao Zhang. 2023. Large language model as attributed training data generator: A tale of diversity and bias. *CoRR*, abs/2306.15895.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning. In *NeurIPS*.

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

Jintian Zhang, Xin Xu, and Shumin Deng. 2023a. Exploring collaboration mechanisms for LLM agents: A social psychology view. *CoRR*, abs/2310.02124.

Zhuosheng Zhang, Yao Yao, Aston Zhang, Xiangru Tang, Xinbei Ma, Zhiwei He, Yiming Wang, Mark Gerstein, Rui Wang, Gongshen Liu, and Hai Zhao. 2023b. Igniting language intelligence: The hitchhiker's guide from chain-of-thought reasoning to language agents. *CoRR*, abs/2311.11797.

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. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. 2023a. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2023b. Webarena: A realistic web environment for building autonomous agents. *CoRR*, abs/2307.13854.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Long Li, Jialong Wu, Tiannan Wang, Shi Qiu, Jintian Zhang, Jing Chen, Ruipu Wu, Shuai Wang, Shiding Zhu, Jiyu Chen, Wentao Zhang, Ningyu Zhang, Huajun Chen, Peng Cui, and Mrinmaya Sachan. 2023c. Agents: An open-source framework for autonomous language agents. *CoRR*, abs/2309.07870.

A Hyper-Parameters

See Table 4.

B Task Information

Task Name: **HotpotQA**
Task Description: This is a question-answering task that includes high-quality multi-hop questions. It tests language modeling abilities for multi-step reasoning and covers a wide range of topics. Some questions are challenging, while others are easier,

Name	Llama-2-7b&13b-chat	Llama-2-70b-chat
lora_r	8	8
lora_alpha	16	16
lora_dropout	0.05	0.05
lora_target_modules	q_proj, v_proj	q_proj, v_proj
model_max_length	4096	4096
per_device_batch_size	2	2
gradient_accumulation_steps	1	1
warmup_ratio	0.03	0.03
epochs	5	3
batch size	4	1
learning rate	1e-4	1e-4

Table 4: Detailed hyper-parameters we use for training.

requiring multiple steps of reasoning to arrive at the final answer.
Task Data Examples:
Question: From 1969 to 1979, Arno Schmidt was the executive chef of a hotel located in which neighborhood in New York?
Answer: Manhattan

Question: Are both Shangri-La City and Ma'anshan cities in China?
Answer: yes

Task Name: **ScienceQA**
Task Description: This is a multimodal question-answering task that necessitates a model to utilize tools for transforming image information into textual data. Simultaneously, this task incorporates substantial background knowledge, requiring the language model to acquire external information to enhance its comprehension of the task.
Task Data Examples:
Question: Which of these states is the farthest north?
Options: (A) West Virginia (B) Louisiana (C) Arizona (D) Oklahoma
Caption: An aerial view of a painting of a forest.
Answer: A. West Virginia

Question: Identify the question that Tom and Justin's experiment can best answer.
Context: The passage below describes an experiment. Read the passage and then follow the instructions below. Tom placed a ping pong ball in a catapult, pulled the catapult's arm back to a 45 angle, and launched the ball. Then, Tom launched another ping pong ball, this time pulling the catapult's arm back to a 30 angle. With each launch, his friend Justin measured the distance between the catapult and the place where the ball hit the ground. Tom and Justin repeated the launches with ping pong balls in four more identical catapults. They compared the distances the balls traveled when launched from a 45 angle to the distances the balls traveled when launched from a 30 angle. Figure: a catapult for launching ping pong balls.
Options: (A) Do ping pong balls stop rolling along the ground sooner after being launched from a 30-angle or a 45-angle? (B) Do ping pong balls travel farther when launched from a 30-angle compared to a 45-angle?
Caption: A wooden board with a wooden head on top of it.
Answer: B. Do ping pong balls travel farther when launched from a 30 angle compared to a 45 angle?

C Tool Library

See Table 5.

D Prompt

D.1 Prompt for Self-Instruct

See Table 6.

D.2 Prompt for Tool Selection

See Table 7.

D.3 Prompt for Trajectories Synthesis

See Table 8.

Name	Definition	Usage
BingSearch	BingSearch engine can search for rich knowledge on the internet based on keywords, which can compensate for knowledge fallacy and knowledge outdated.	BingSearch[query], which searches the exact detailed query on the Internet and returns the relevant information to the query. Be specific and precise with your query to increase the chances of getting relevant results. For example, Bingsearch[popular dog breeds in the United States]
Retrieve	Retrieve additional background knowledge crucial for tackling complex problems. It is especially beneficial for specialized domains like science and mathematics, providing context for the task	Retrieve[entity], which retrieves the exact entity on Wikipedia and returns the first paragraph if it exists. If not, it will return some similar entities to retrieve. For example, Retrieve[Milhouse]
Lookup	A Lookup Tool returns the next sentence containing the target string in the page from the search tool, simulating Ctrl+F functionality on the browser.	Lookup[keyword], which returns the next sentence containing the keyword in the last passage successfully found by Retrieve or BingSearch. For example, Lookup[river].
Image2Text	Image2Text is used to detect words in images convert them into text by OCR and generate captions for images. It is particularly valuable when understanding an image semantically, like identifying objects and interactions in a scene.	Image2Text[image], which generates captions for the image and detects words in the image. You are recommended to use it first to get more information about the image to the question. If the question contains an image, it will return the caption and OCR text, else, it will return None. For example, Image2Text[image].
Text2Image	Text2Image Specializes in converting textual information into visual representations, facilitating the incorporation of textual data into image-based formats within the task.	Text2Image[text], which generates an image for the text provided by using multi-modal models. For example, Text2Image[blue sky]
......	......	......
Code Interpreter	Code Interpreter is a tool or software that interprets and executes code written in Python. It analyzes the source code line by line and translates it into machine-readable instructions or directly executes the code and returns Execution results	Code[python], which interprets and executes Python code, providing a line-by-line analysis of the source code and translating it into machine-readable instructions. For instance, Code[print("hello world!")]

Table 5: Part of our tool library.

Prompt for Self-Instruct

I want you to be a QA pair generator to generate high-quality questions for use in Task described as follows:
Task Name: **[task_name]**
Task Description: **[task_description]**
Here are some Q&A pair examples from the Task:
[QA_pairs]
Modeled on all the information and examples above, I want you to generate new different **[gen_num_per_round]** Question-Answer pairs that cover a wide range of topics, some of which are difficult, some of which are easy, and require multiple steps of reasoning to get to the final answer. The format is like below:
[one_example]

Table 6: Prompt used for self-instruct.

Prompt for Automatic Tool Selection

To successfully complete a complex task, the collaborative effort of three types of agents is typically required:
1. Plan Agent. This agent is used to plan the specific execution process of the benchmark, solving a given task by determining the order in which other expert language models are invoked;
2. Tool Agent. This agent is employed to decide how to use a specific tool when addressing a task. Tools encompass interactive tools within the task environment as well as external tools or models. The Tool Agent includes various tools that can be flexibly chosen;
3. Reflect Agent. This agent reflects on historical information and answers to assess whether the response aligns with the provided query.
Above all, the Tool Agent includes many tools that can be flexibly selected. Now your task is to select 3 tools from the Tool Library for solving a given task. Note that all tools are based on language models, and their inputs and outputs must be text. You only need to provide the names and descriptions of the tools in order, without any additional output.
Task Prompt Template
The following is the given task name and description, and you need to choose 3 corresponding tools from the Tool Library according to the above rules in the format of one line, one tool.
Task Name: **[task_name]**
Task Description: **[task_description]**
Tool Library: **[list_of_tools]**

Table 7: Prompt used for automatic tool selection.

Prompt for Trajectories Synthesis

I expect you to excel as a proficient question answerer in the task.
Task Name: **[task_name]**
Task Description: **[task_description]**
Solve a question-answering task with interleaving Thought, Action, and Observation steps. Thought can reason about the current situation, and Action can be **[action_num]** types:
list of action selected from automatic tool selection [name, definition , usage]
Question: **[question][scratchpad]**

Table 8: Prompt used for trajectories synthesis.