Towards Adaptive Mechanism Activation in Language Agent

Ziyang Huang, Jun Zhao, Kang Liu*
The Key Laboratory of Cognition and Decision Intelligence for Complex Systems,
Institute of Automation, Chinese Academy of Sciences
School of Artificial Intelligence, University of Chinese Academy of Sciences
huangziyang2023@ia.ac.cn, {jzhao, kliu}@nlpr.ia.ac.cn

Abstract

Language Agent could be endowed with different mechanisms for autonomous task accomplishment. Current agents typically rely on fixed mechanisms or a set of mechanisms activated in a predefined order, limiting their adaptation to varied potential task solution structures. To this end, this paper proposes **A**daptive **L**anguage **A**gent **M**echanism **A**ctivation Learning with Self-Exploration (**ALAMA**), which focuses on optimizing mechanism activation adaptability without reliance on expert models. Initially, it builds a harmonized agent framework (**UniAct**) to **Uni**fy different mechanisms via **Act**ions. Then it leverages a training-efficient optimization method based on self-exploration to enable the UniAct to adaptively activate the appropriate mechanisms according to the potential characteristics of the task. Experimental results demonstrate significant improvements in downstream agent tasks, affirming the effectiveness of our approach in facilitating more dynamic and context-sensitive mechanism activation.

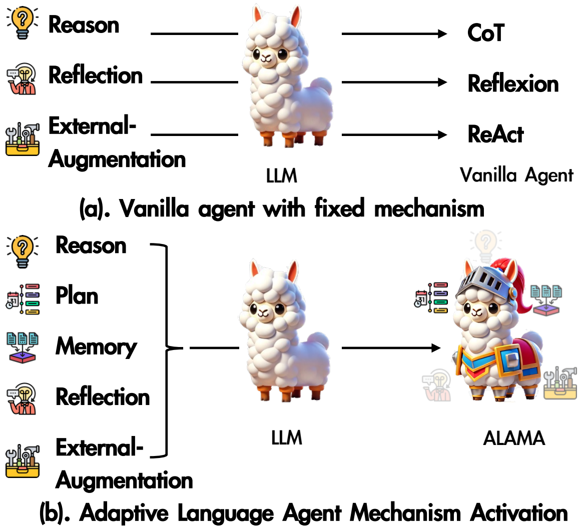

Figure 1: Illustration of Language Agent with different mechanisms. (a). Vanilla agent with fixed mechanisms by In-Context learning. (b). ALAMA with different mechanisms learn to fit into different environments by Self-Exploration.

1 Introduction

Language Agent (LA) (Sumers et al., 2024; Yao et al., 2023; Xi et al., 2023; Gao et al., 2023) has garnered considerable attention recently due to the rapid advancements in Large Language Models (LLMs) (OpenAI, 2024; AI@Meta, 2024; Yang et al., 2023; Chowdhery et al., 2022; Radford et al., 2018). Through the well-designed prompts and carefully selected in-context demonstrations (Zhou et al., 2024; Dong et al., 2023; Liu et al., 2021), LLM-based agents can be endowed with different mechanisms[1] for environment interaction and task solving. Existing LAs could benefit from distinct mechanisms for various tasks with unique solution structures (Zhou et al., 2024). For instance, Reflexion (Shinn et al., 2023) is equipped with `Reflection` mechanism to gain insightful refinement suggestions. And ReAct (Yao et al., 2023) is equipped with `External-Augmentation` mechanism to ground the solution trajectory with additional evidence.

Despite the success of current LAs through aforementioned direct prompting and in-context learning, named as *manual mechanism activation*, they rely on fixed mechanisms or a predefined sequence of mechanisms (Liu et al., 2023; Chen et al., 2023; Song et al., 2024), as illustrated in Figure 1 (a). As a result, such rigidity hampers activating the optimal solution structures (mechanism) for a specific task and also limits their adaptability to open-world scenarios. There is compelling evidence that *oracle language agent mechanism activation*, selecting

*Kang Liu is the corresponding author.
[1]Here, **mechanism** is defined as the inherent ability of the Language Agent which could be manifested as a special workflow externally and activated by the specific prompting.

Proceedings of the 31st International Conference on Computational Linguistics, pages 2867–2885
January 19–24, 2025.

the most appropriate mechanism for a task, can improve the performance by over 15% compared to fixed mechanism baselines (as shown in Section 5.1). Therefore, it highlights the significant potential of *adaptive mechanism activation*, the focus of the paper, where mechanisms are adaptively activated based on task characteristics, as shown in Figure 1 (b). We view this as a critical kind of meta-ability for LAs, and its enhancement could offer the potential for better generalization in unseen tasks.

Intuitively, when humans encounter unfamiliar tasks, they tend to first explore different solution strategies and then select the most effective solution from previous experiences in similar tasks. Inspired by this, to enable LAs to adaptively select suitable solution strategies (adaptive mechanism activation), this paper proposes **A**daptive **L**anguage **A**gent **M**echanism **A**ctivation Learning with Self-Exploration (**ALAMA**), a novel technique for learning adaptive mechanism activation across various tasks. It first introduces a harmonized agent framework to **Uni**fy existing known mechanisms by **Act**ions (**UniAct**). Compared with previous agents which did not fully integrate various mechanisms (Yao et al., 2023) or only implicitly incorporated specific mechanisms into the thinking process without an explicit trigger (Zhou et al., 2023), UniAct defines the workflows of mechanisms as specific actions. In this way, different mechanisms would share a unified action space. When the agent triggers an action, the corresponding mechanism is expected to be activated.

Secondly, to fulfill adaptive mechanism activation in LAs, our ALAMA adopts a self-exploration fine-tuning way rather than simply prompting. Sufficient high-quality trajectories with different activated mechanisms are important for model training but not easy to acquire. To this end, ALAMA firstly leverages self-exploration to obtain sufficient trajectories for training. Compared with previous methods of acquiring trajectories through manual annotation or distillation from proprietary models (Zeng et al., 2023; Chen et al., 2023), self-exploration could extremely decrease data acquisition costs and alleviate the paucity of training signals. Specifically, we manually activate different mechanisms to facilitate multiple rounds of self-exploration. Consequently, diverse solution trajectories are produced and then converted into the UniAct format. To introduce implicit mechanism preferences towards different tasks as well as fundamental interaction and instruction-following capabilities for the agent, this paper utilizes **I**mplicit **M**echanism **A**ctivation **O**ptimization (**IMAO**), which samples subset of positive trajectories to fine-tune the LAs.

For further model training, different from existing exploration-based methods which use success-failure pairwise data for behavior contrastive learning (Song et al., 2024; Yuan et al., 2024a), this paper employs **M**echanism **A**ctivation **A**daptability **O**ptimization (**MAAO**) based on KTO algorithm (Ethayarajh et al., 2024). KTO is a preference learning algorithm that only requires binary signals (desirable/ undesirable). In this way, the need for assembling high-quality pairwise data (Rafailov et al., 2023; Xie et al., 2024) is alleviated and all trajectories with different rewards obtained during the self-exploration phase could be utilized, which makes training more efficient.

To validate the effectiveness of our proposed method, the paper conducts extensive experiments on mathematical reasoning (Cobbe et al., 2021; Mishra et al., 2022; Patel et al., 2021) and knowledge-intensive reasoning (Yang et al., 2018; Joshi et al., 2017; Press et al., 2023) tasks. The results show that ALAMA surpasses the baselines by a large margin on both Held-in and Held-out datasets, demonstrating its strong performance and generalization.

In summary, our contributions are as follows:

- This paper analyzes the advantage of oracle language agent mechanism activation and thus claims that the adaptive mechanism activation is a crucial meta-ability for Language Agents.
- This paper proposes **A**daptive **L**anguage **A**gent **M**echanism **A**ctivation Learning with Self-Exploration (**ALAMA**), which incorporates a harmonized agent framework to unify different mechanisms and a training-efficient optimization method based on self-exploration.
- This paper conducts extensive experiments to demonstrate the superior performance and generalization of ALAMA. Further analysis shows that it can outperform its counterparts with fewer training data.

2 Background

With different prompts and demonstrations, the agent can be equipped with different mechanisms for better task-solving performance. This paper

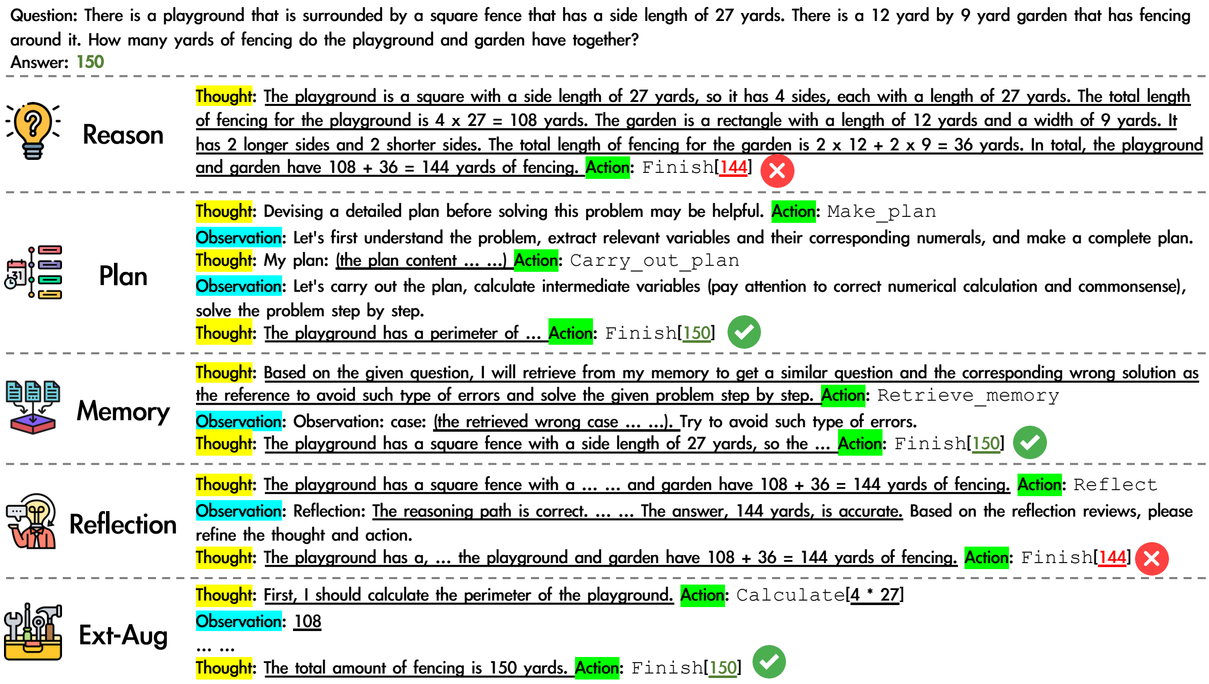

Figure 2: The UniAct trajectory examples for five mechanisms. The underlined contents are generated by the vanilla agent or from external feedback.

selects five essential agent mechanisms as the focus of our study: (1) `Reason` (Wei et al., 2022): Directly obtaining the answer through step-by-step reasoning. (2) `Plan` (Wang et al., 2023a; Zhou et al., 2023): First understanding the task and develop a plan to decompose it into smaller, more easily solvable sub-tasks, and then progressively solving each sub-task to arrive at the final answer. (3) `Memory` (Sun et al., 2023; Gao et al., 2024): Initially building a database of failed examples. During each subsequent task execution, similar cases are retrieved from this database based on task similarity (cosine of task description embedding), and the agent could try to avoid similar errors. (4) `Reflection` (Shinn et al., 2023; Madaan et al., 2023): Introducing a Critic Model into the environment to reflect on the previously reasoned answers by the agent when necessary. (5) `External-Augmentation` (Yao et al., 2023; Schick et al., 2023): Calling task-specific toolkits for solving different tasks, such as a calculator for mathematical reasoning or a search engine for knowledge-intensive reasoning. As shown in Figure 2, we demonstrate the examples of trajectories with different mechanisms in UniAct format[2]. We defer the implementation details of each mechanism to the appendix D.

[2]We describe the UniAct format and how to transform the agent trajectories into it in Section 3.

3 ALAMA: Adaptive Language Agent Mechanism Activation Learning with Self-Exploration

This section describes our method in detail. Firstly, we introduce a harmonized agent framework to unify existing known mechanisms (**UniAct**). Secondly, we elaborate on a self-exploration fine-tuning method for enhancing the meta-ability of adaptive mechanism activation. In specific, we leverage **Self-Exploration** with manual mechanism activation to sample various UniAct trajectories. Next, we employ Implicit Mechanism Activation Optimization (**IMAO**) and Mechanism Activation Adaptability Optimization (**MAAO**) to adapt the agent to different tasks based on the recognized characteristics and potential solution structures.

UniAct: Unify Agent Mechanisms by Actions Currently, ReAct (Yao et al., 2023) serves as the foundational framework for LLM-based agents, employing the `Thought`, `Action`, `Observation` (τ, a, o as the abbreviation) format to govern agent control. This format only unifies reasoning, action generation, and the acquisition of feedback from external environments into natural language space. Based on this, we propose UniAct to integrate diverse mechanisms into a unified framework explicitly. As depicted in the upper of Figure 3 (a), we define distinct

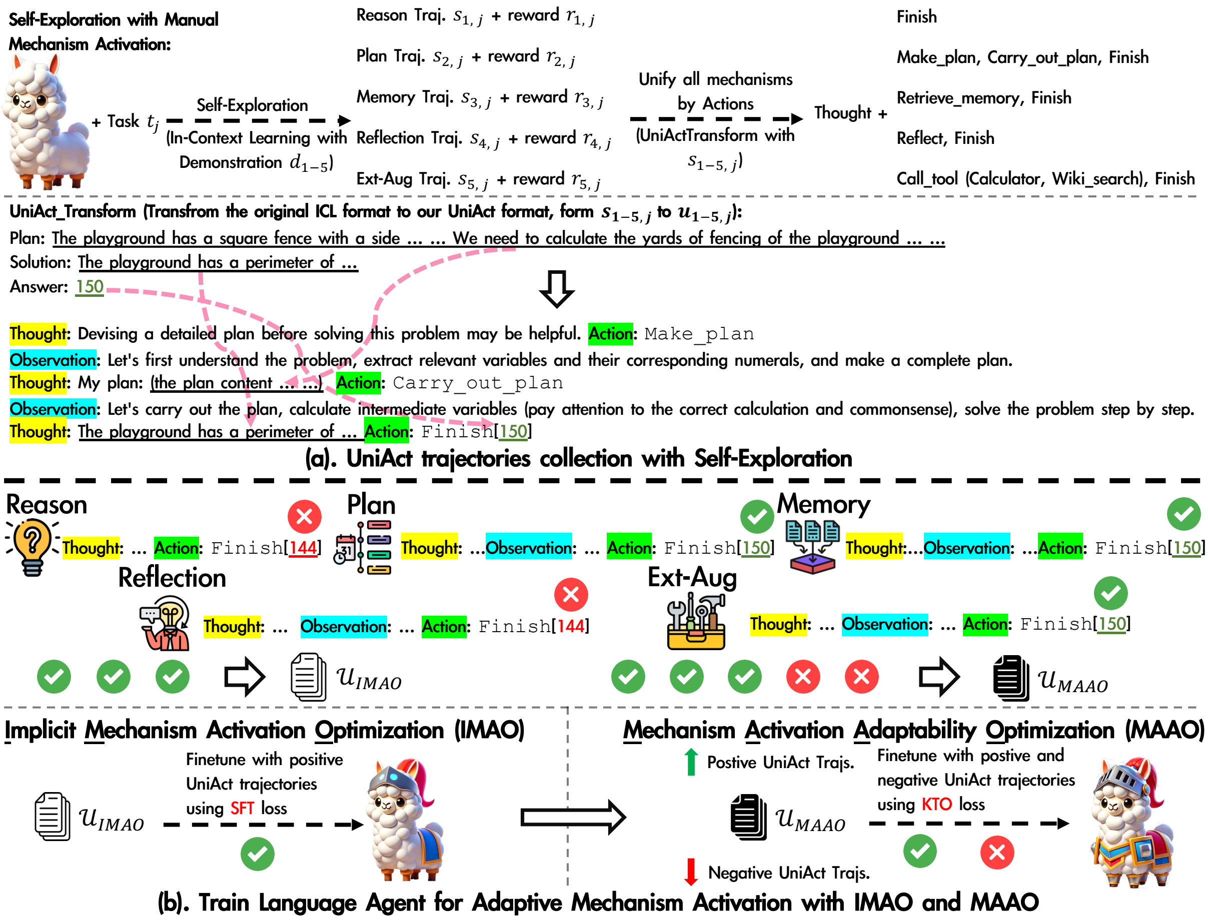

Figure 3: The illustration of ALAMA process. The UniAct trajectories are collected by Self-Exploration with manual mechanism activation. For tasks with mechanism sensitivity, we use the corresponding positive trajectories for Implicit Mechanism Activation Optimization, and utilize both positive and negative ones for Mechanism Activation Adaptability Optimization.

`Actions` for each mechanism to unify the different workflows into a shared action space. Specifically, we define `make_plan` for detailed plan generation, `Carry_out_plan` for plan execution, `Retrieve_memory` to get potential error cases, `Reflect` to get insightful correction suggestions from the expert Critic model, `Call_tool` to invoke external tools, and `Finish` to output the final results and terminate the solution trajectories. We take the `Thought` as the thinking process before generating the actions, and the `Observation` as the environmental feedback. Furthermore, we have adapted the external environment to not only provide task-related feedback but also return appropriate prompt to facilitate the activation of corresponding mechanisms. Details regarding the UniAct format including the actions and corresponding grounding prompts are provided in Appendix F.

Self-Exploration We refer to the base agent with parameter θ as LA_θ and all the mechanisms as $\mathcal{M} = \{m_i\}_{i=1}^{N}$. We construct a demonstration trajectory d_i where only that specific mechanism m_i is activated. As shown in upper of Figure 3 (a), given Tasks $\mathcal{T} = \{t_j\}_{j=1}^{|\mathcal{T}|}$, we manually activate different mechanisms by prompting with the corresponding d_i to get the exploration solution trajectory $s_{i,j}$ and corresponding reward $r_{i,j}$. And then we transform all these trajectories into UniAct format $u_{i,j}$.

$$s_{i,j}, r_{i,j} = \text{LA}_\theta(d_i, t_j) \tag{1}$$

$$\begin{aligned} u_{i,j} &= \text{UniActTransform}(s_{i,j}) \\ &= (\tau_1, a_1, o_1, \cdots, o_{m-1}, \tau_m, a_m)_{i,j} \end{aligned} \tag{2}$$

where τ, a, o represent thought, action, and observation respectively. For UniActTransform($\cdot$), we extract the self-generated contents and external feedback from the ICL solutions, and then fill them into the UniAct format with explicit actions. As depicted in the bottom part of Figure 3 (a), we show a transformation example of `Plan`. Please refer to the Appendix E for other mechanisms. Finally, we obtain all self-exploration UniAct trajectories $\mathcal{U}$. Furthermore, these trajectories will be used for self-optimization towards better adaptive mechanism activation.

$$\mathcal{U} = \{U_j\}_{j=1}^{|\mathcal{T}|} = \{\{u_{i,1}\}_{i=1}^{N}, \cdots, \{u_{i,|\mathcal{T}|}\}_{i=1}^{N}\} \quad (3)$$

Notably, not every mechanism could fit all tasks and obtain correct results. As illustrated in the upper of Figure 3 (b), certain tasks are successfully solved by specific mechanisms, while remaining unsolved when the other are activated. We refer to these as **tasks with mechanism sensitivity**.

IMAO: Implicit Mechanism Activation Optimization To equip the model with the basic ability to follow the UniAct format in the zero-shot setting and adaptively activate appropriate mechanisms, we sample a subset of positive trajectories from $\mathcal{U}$ for supervised fine-tuning, as shown in the left bottom part of Figure 3 (b). To introduce implicit preferences for different mechanisms across tasks, we use the UniAct trajectories with $r = 1$ of the tasks with mechanism sensitivity as the training data, referred to as $\mathcal{U}_{\text{IMAO}}$.

Thoughts and actions are generated by the vanilla agent, while observations are gathered from the environment. Consequently, we compute the next token prediction loss on thought τ and action a, while masking the loss on observation o.

$$\mathcal{L}_{\text{IMAO}}(\text{LA}_\theta) = \mathbb{E}_{u \in \mathcal{U}_{\text{IMAO}}} - \log P(u|t) \quad (4)$$
$$= \mathbb{E}_{u \in \mathcal{U}_{\text{IMAO}}} - \log P(a_m, \tau_m, \cdots, a_1, \tau_1|t) \quad (5)$$
$$= \mathbb{E}_{u \in \mathcal{U}_{\text{IMAO}}} \Bigg[-\sum_{k=1}^{m} \log P(\tau_k|o_{k-1}, a_{k-1}, \cdots, t) - \sum_{k=1}^{m} \log P(a_k|\tau_k, o_{k-1}, \cdots, t) \Bigg] \quad (6)$$

where the t and u represent the task and the corresponding self-generated trajectory.

MAAO: Mechanism Activation Adaptability Optimization For all tasks with mechanism sensitivity, we collect all corresponding trajectories as training data, referred to as $\mathcal{U}_{\text{MAAO}}$. We treat those with a reward equal to 1 as $\mathcal{U}_{\text{MAAO-pos}}$, and the other as $\mathcal{U}_{\text{MAAO-neg}}$. Instead of only using positive trajectories in IMAO, our MAAO utilizes the contrastive information between positive and negative examples to update the agent using KTO loss (Ethayarajh et al., 2024). KTO is a preference learning (Jiang et al., 2024) algorithm which can optimize the model using binary feedback. The behavior of the agent is biased towards positive examples and away from negative ones. This approach enhances the model's meta-ability for adaptive mechanism activation:

$$z_0 = \mathbb{E}_{t' \in \mathcal{U}_{\text{MAAO}}}[\text{KL}(\text{LA}_\theta(u'|t')||\text{LA}_{\text{ref}}(u'|t'))] \quad (7)$$
$$v(t, u) = (-1)^{\mathbb{1}(u \in \mathcal{U}_{\text{MAAO-pos}})} \lambda_{\text{pos/neg}} \times \sigma\left(\beta\left(z_0 - \log \frac{\text{LA}_\theta(u|t)}{\text{LA}_{\text{ref}}(u|t)}\right)\right) \quad (8)$$
$$\mathcal{L}_{\text{MAAO}}(\text{LA}_\theta, \text{LA}_{\text{ref}}) = \mathbb{E}_{u \in \mathcal{U}_{\text{MAAO}}}[\lambda_{\text{pos/neg}} - v(t, u)] \quad (9)$$

When $u \in \mathcal{U}_{\text{MAAO-pos}}$, $(-1)^{\mathbb{1}(u \in \mathcal{U}_{\text{MAAO-pos}})} = -1$, $\lambda_{\text{pos/neg}} = \lambda_{\text{pos}}$, and vice versa.

The pseudo-code of the optimization method is shown in Algorithm 1.

4 Experiment

4.1 Setup

Model and Datasets We utilize GPT-3.5-turbo-0125 as the closed-source model baseline, accessed through the OpenAI API. We employ Meta-Llama3-8B-Instruct as the backbone for ALAMA. For datasets, the paper employs the GSM8K (Cobbe et al., 2021) and HotpotQA (Yang et al., 2018) as Held-in tasks for exploration, training, and testing. Additionally, we select NumGLUE (Mishra et al., 2022), SVAMP (Patel et al., 2021), TriviaQA (Joshi et al., 2017), and Bamboogle (Press et al., 2023) as Held-out tasks to evaluate the generalization performance. For dataset processing details, please refer to Appendix A.

Baselines We select the following baselines for comparisons, like (1) Fixed single mechanism (`Reason`, `Plan`, `Memory`, `Reflection` and `External-Augmentation` shown in Table 1): we manually construct one in-context demonstration example to activate different mechanisms (2) `Average`: The average performance of different mechanisms. (3) `Majority Voting`: Selecting the most consistent (Wang et al., 2023b) answer among the solutions obtained by activating different mechanisms as the final answer. (4) `Self-Adapt Consistency`: We apply self-consistency (Wang et al., 2023b) technique to ALAMA. For training and inference details, please refer to Appendix B.

4.2 Main Results

Adaptive Mechanism Activation outperforms fixed Manual Mechanism Activation. As

	Mathematical Reasoning (Acc)			Knowledge-intensive Reasoning (EM)		
	Held-in	Held-out		Held-in	Held-out	
	GSM8K	NumGLUE	SVAMP	HotpotQA	TriviaQA	Bamboogle
GPT-3.5-turbo (one-shot Activation)						
`Reason`	63.91	60.63	71.20	22.20	28.80	28.80
`Plan`	77.94	59.84	83.40	22.80	51.20	37.60
`Memory`	76.42	65.75	81.10	25.80	55.60	44.80
`Reflection`	79.38	66.14	86.10	30.80	60.80	41.60
`External-Augmentation`	70.66	70.47	79.00	22.20	44.00	30.40
`Average`	73.66	64.57	80.16	24.76	52.16	36.64
`Majority Voting`	82.25	66.54	86.30	28.40	56.00	41.60
Llama-3-8B-Instruct (one-shot Activation)						
`Reason`	73.08	41.73	66.10	17.60	41.40	29.60
`Plan`	77.56	68.11	82.90	19.80	44.40	31.20
`Memory`	77.03	70.47	77.80	16.20	41.20	30.40
`Reflection`	80.06	74.40	85.90	26.00	55.80	37.60
`External-Augmentation`	71.80	61.02	75.80	15.80	38.60	20.80
`Average`	75.90	63.15	77.70	19.08	44.28	29.92
`Majority Voting`	82.71	70.87	85.50	21.60	48.60	37.60
ALAMA$_{\text{Llama-3-8B}}$						
`IMAO`	78.77	72.83	83.30	24.00	40.40	27.20
`IMAO + MAAO`	**82.18**	**78.35**	**88.20**	**27.60**	43.60	**32.80**
`Self-Adapt Consistency`	**85.06**	**79.13**	**89.80**	**31.00**	**49.40**	36.80

Table 1: Performance of different methods. We use Accuracy and EM as metric for Mathematical Reasoning and Knowledge-intensive Reasoning.

shown in Table 1, ALAMA outperforms all single mechanism baselines and the average performance of different mechanisms on the Held-in tasks. We consider the Average as the bottom performance for introducing multiple mechanisms into one agent. After IMAO (supervised learning), ALAMA surpasses the Average by 2.87 on GSM8K and 4.92 on HotpotQA, indicating that it has the ability to adaptively activate different mechanisms based on the task characteristics.

Furthermore, after MAAO (preference learning), ALAMA continues to improve by 3.41 on GSM8K and 3.60 on HotpotQA. This suggests that MAAO can enhance the adaptability of the agent to potential solution structures of different tasks. Behavior contrastive learning enables the model to preferentially activate certain specific mechanisms while refusing to activate the remaining ones. For example, in manual activation, `Plan` outperforms `Reason` by 4.48 on GSM8K. After MAAO, when the agent encounters specific complex mathematical reasoning tasks that can not be solved directly through reasoning, it recognizes that direct reasoning may lead to incorrect answers and thus chooses to analyze the sub-problems in the question first, decompose the problem, and solve them individually, ultimately summarizing the answers. ALAMA based on Llama-3-8B-Instruct outperforms GPT-3.5-turbo average on Held-in tasks after ALAMA, demonstrating the superior effectiveness.

Compared to all fine-tuning baselines shown in the upper of Table 2, the introduction of multiple mechanisms in ALAMA demonstrates significant performance gains, which adequately exemplifies the superiority of adaptive mechanism activation learning techniques.

Agent	GSM8K (Acc)
Fine-tuning Baselines	
FireAct$_{\text{Llama-2-7B}}$ (Chen et al., 2023)	56.10
Lumos$_{\text{Llama-2-7B}}$ (Yin et al., 2024b)	54.90
WizardMath$_{\text{Llama-2-13B}}$ (Luo et al., 2023)	63.90
ToRA$_{\text{Llama-2-13B}}$ (Gou et al., 2024)	72.70
Husky$_{\text{Llama-2-13B}}$ (Kim et al., 2024)	79.40
Husky$_{\text{Llama-3-8B}}$ (Kim et al., 2024)	79.90
MAmmoTH2-8B$_{\text{Llama-3-8B}}$ (Yue et al., 2024)	70.40
MAmmoTH2-8B-Plus$_{\text{Llama-3-8B}}$ (Yue et al., 2024)	84.10
Train on Self-Exploration Data	
ALAMA$_{\text{Llama-3-8B-SFT}}$	78.77
ALAMA$_{\text{Llama-3-8B-DPO}}$	80.52
ALAMA$_{\text{Llama-3-8B-KTO}}$	**82.18**

Table 2: Fine-tuning based Language Agent performance comparison. ALAMA with multiple mechanisms optimized with efficient adaptive learning using less data demonstrates suprior performance.

ALAMA outperforms SoTA fine-tuning baselines with more efficient data acquisition and

training. The agent data employed for fine-tuning baselines as presented in Table 2 are all curated by expert models or humans. However, our ALAMA surpasses these baselines merely by relying on self-exploration, which is more efficient. More specifically, *Husky* is trained on agent trajectories from 10 datasets including GSM8K, MATH, and TabMWP. SoTA agent *Mammoth2-Plus* first collects over 10 million instruction data using a complicated pipeline to enhance the reasoning ability and then uses math instruction datasets (including GSM8K and MATH) for supervised fine-tuning. Our $\text{ALAMA}_{\text{Llama-3-8B-KTO}}$ uses only GSM8K for training. Despite having much more training data, *Husky* underperforms and *Mammoth2-Plus* is only about 2% higher in performance than $\text{ALAMA}_{\text{Llama-3-8B-KTO}}$, fully demonstrating the data efficiency of ALAMA.

In addition, we introduced a DPO (Rafailov et al., 2023) based counterpart, i.e. $\text{ALAMA}_{\text{Llama-3-8B-DPO}}$. The positive and negative trajectories in $\mathcal{U}_{\text{MAAO}}$ are then paired into multiple preference pairs for DPO training. This pairing approach leads to increased training costs. Experiment results demonstrate that KTO yields better results, further highlighting the efficiency and effectiveness of our method.

ALAMA demonstrates superior generalization on Held-out tasks. Apart from testing on the Held-in datasets, we have also selected four Held-out datasets for evaluation under the zero-shot setting. On NumGLUE and SVAMP, ALAMA outperforms the best baseline by 3.95 and 2.3, respectively. With the assistance of Self-Adapt Consistency, ALAMA surpasses 4.73 and 3.9, respectively. Additionally, ALAMA also outperforms most baselines, including Average, on TriviaQA and Bamboogle. This adequately demonstrates the effectiveness and generalization of our proposed method.

Self-Adapt Consistency outperforms manual mechanism activation based Majority Voting. On GSM8K, the performance obtained by selecting the majority answer from the different mechanisms significantly surpasses the performance of all individual mechanisms as well as the average performance. We consider this as a strong baseline for the comprehensive utilization of multiple mechanisms. For a fair comparison, we sample 5 times for Self-Adapt consistency. It exceeds the above strong baseline by 2.35 and 9.4 on GSM8K and HotpotQA

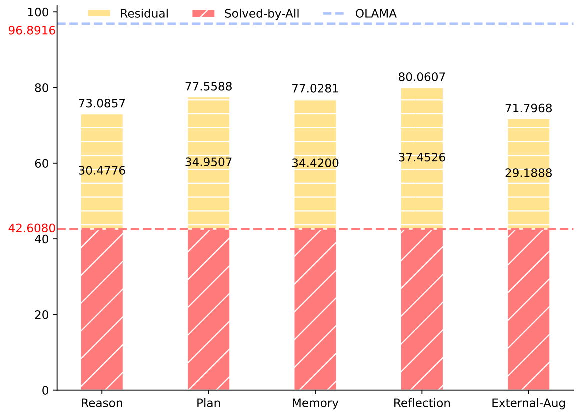

Figure 4: Mechanism specificity analysis results on GSM8K. `OLAMA` represents oracle mechanism activation, which selects the most appropriate mechanism for each task. `Solved-by-All` represents that corresponding tasks could be solved by all mechanisms respectively. And `Residual` represents the performance gap (yellow part) between different mechanisms and `Solved-by-All`, which shows the specificity.

respectively, indicating that the fine-tuned ALAMA possesses the ability to adaptively activate different mechanisms. With the help of random sampling, ALAMA activates the most effective task-specific mechanisms to generate diverse trajectories, ultimately achieving better performance.

5 Analysis

5.1 The Specificity of Different Mechanisms

This subsection tries to investigate the impact of different mechanisms on downstream task performance. In detail, we choose Llama3-8B-Instruct (AI@Meta, 2024) as the vanilla agent and GSM8K as the agent task. As shown in Figure 4, only 42.61% tasks could be solved by all fixed single mechanism baselines. This result suggests that more than 50% of tasks are of mechanism sensitivity. For instance, certain tasks require external knowledge, while others may encounter conflicts upon the introduction of such knowledge. Consequently, we believe that different tasks possess distinct underlying solution structures. Moreover, the oracle mechanism activation results demonstrate that the model can solve 96.89% of the tasks with the aid of selecting the correct mechanism, highlighting that adaptive mechanism activation has a very high ceiling performance. This suggests a significant potential for identifying the inherent characteristics of tasks and their solution structures. Our ALAMA still falls short of the performance ceiling, which anticipates further optimization of the mechanism activation methods.

5.2 The Effects of Mixing Different Mechanism Data

To investigate the impact of individual and mixed mechanisms data on the performance of the agent, we divided $\mathcal{U}_{\text{IMAO}}$ and $\mathcal{U}_{\text{MAAO}}$ based on different mechanisms. For $\mathcal{U}_{\text{MAAO}}$, we segment it according to the mechanisms activated by the positive examples, and incorporated all negative examples of the corresponding tasks into the training set. For IMAO, we employed Meta-Llama-3-8B-Instruct as the base model, whereas for MAAO, we utilized $\text{ALAMA}_{\text{IMAO}}$ as the base model.

In IMAO, we observed that fine-tuning the model using single-mechanism trajectories leads to underperformance, as the use of original data does not effectively enhance the performance under the zero-shot setting. We hypothesize this may be due to insufficient training data resulting from data segmentation. After sampling more data corresponding to the specific mechanisms for further fine-tuning, it still could not significantly improve the performance of the agent. These performances are shown as 'original' and 'aug' in Table 3. This suggests that under the single-mechanism activation setting, the quality of trajectories generated through self-exploration is insufficient for the agent to achieve performance comparable to In-Context Learning, and it might require using expert-generated models to attain higher performance. Furthermore, we found that the performance using $\mathcal{U}_{\text{IMAO}}$ for training far exceeds that achieved with single-mechanism data, proving the superiority of mixed-mechanism data fine-tuning. In MAAO, the performance using multiple mechanisms for fine-tuning also surpasses that using single-mechanism data. This indicates that an agent has mechanism preferences for different tasks, which aligns with the Residual performance presented in Figure 4. However, the performance gap between full data and partial data is not as pronounced in IMAO as it is in MAAO, suggesting that IMAO plays a more crucial role in agent meta-ability acquisition.

Data	Number	Acc
IMAO		
Reason original / aug	251 / 1300	25.47 / 36.01
Plan original / aug	264 / 1300	28.73 / 36.69
Memory original / aug	240 / 1300	37.23 / 43.29
Reflection original / aug	248 / 1300	47.08 / 46.63
External-Aug original / aug	254 / 1300	37.76 / 43.97
Full	1257	**78.77**
MAAO		
Reason original	2403	81.43
Plan original	2396	79.00
Memory original	2390	78.77
Reflection original	2524	80.21
External-Aug original	1618	70.51
Full	7120	**82.18**

Table 3: The performance of training agent using different parts of data. Number means the number of the data used in training.

6 Related Work

6.1 Language Agent

To achieve better autonomous task accomplishment, the research community has designed many Language Agent Frameworks like ReAct (Yao et al., 2023), Reflexion (Shinn et al., 2023), and Multi-Agent Debate (Du et al., 2023; Liang et al., 2023; Liu et al., 2024). However, these frameworks are labor-intensive for prompt design and work only for big foundation models which are opaque, proprietary, and API-based (OpenAI, 2022; Anthropic, 2023), hindering the research of inherent mechanisms. Another effective technique is adapting open-sourced LLM to LA by imitation fine-tuning (Ho et al., 2023; Zeng et al., 2023; Chen et al., 2023; Xu et al., 2024; Yin et al., 2024a; Wang et al., 2024a; Chen et al., 2024a; Yin et al., 2024b). High-reward trajectories are collected by reformatting golden rationales (Anonymous, 2024) or distilling from ChatGPT (OpenAI, 2022; Chen et al., 2023). These endow smaller models with abilities like planning, reasoning, and reflection. But these LAs are limited as they do not explore the task environments for interactive self-improvement. Exploration fine-tuning (Song et al., 2024; Yang et al., 2024; Wang et al., 2024b) has gained attention recently as it shows potential for self-improvement.

6.2 Self-evolution of Large Language Model

Self-evolution is crucial for Large Language Models (Huang et al., 2023; Tao et al., 2024; Lu et al., 2024). Techniques like ReST ((Gulcehre et al., 2023)), self-rewarding ((Yuan et al., 2024b)), and self-play ((Chen et al., 2024b)) achieve it via iterative generation and optimization. As LLMs evolve beyond human intelligence, more weakly supervised automatic feedback signals are needed for self-evolution (e.g., (Burns et al., 2023; Cao et al., 2024)). The approach in this paper is also a method for LLM self-evolution.

7 Conclusion

In this paper, we propose **A**daptive **L**anguage **A**gent **M**echanism **A**ctivation Learning with Self-Exploration (**ALAMA**). We observed that numerous tasks exhibit mechanism sensitivity. And the oracle mechanism activation exhibits stronger performance than fixed baselines. To this end, we **uni**fy different agent mechanisms by **act**ions (**UniAct**) into a harmonized agent framework. Moreover, we utilize an adaptive mechanism activation optimization method based on self-exploration, which requires less data than previous SoTA agents and is training-efficient. Extensive experiments demonstrate the effectiveness and generalization of our proposed method. Further analysis shows that increasing the number of mechanisms and integrating trajectory data from different mechanisms are crucial for enhancing agent performance. Code will be available at `https://github.com/hzy312/alama`.

Limitations

In this paper, the discussion of adaptive mechanism activation is limited to the activation of a single mechanism and does not address the simultaneous activation of multiple mechanisms. Activating various mechanisms concurrently could offer additional benefits; however, it also increases the complexity of learning adaptive mechanism activation. Therefore, we consider this an area for future work to be explored subsequently. Moreover, in Section 5.2, we discuss only the effects of full data and single-mechanism data, omitting the impact of mixing data from different mechanisms. The five mechanisms discussed in this paper could lead to $2^5 - 1$ possible combinations, and our limited computational resources did not allow for the evaluation of all possibilities. We plan to incorporate these data in a formal version later for further discussion.

Acknowledgement

This work was supported by Beijing Natural Science Foundation (L243006).

References

AI@Meta. 2024. Llama 3 model card.

Anonymous. 2024. Samoyed: Towards generalized llm agents via fine-tuning on 50000+ interaction trajectories.

Anthropic. 2023. Introducing claude.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, and Jeff Wu. 2023. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *Preprint*, arXiv:2312.09390.

Boxi Cao, Keming Lu, Xinyu Lu, Jiawei Chen, Mengjie Ren, Hao Xiang, Peilin Liu, Yaojie Lu, Ben He, Xianpei Han, Le Sun, Hongyu Lin, and Bowen Yu. 2024. Towards scalable automated alignment of llms: A survey. *Preprint*, arXiv:2406.01252.

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

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

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024b. Self-play fine-tuning converts weak language models to strong language models. *Preprint*, arXiv:2401.01335.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *Preprint*, arXiv:2204.02311.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro

Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Size Zheng, T. Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Liu, Xin Xie, Xingkai Yu, Xinnan Song, Xinyi Zhou, Xinyu Yang, Xuan Lu, Xuecheng Su, Y. Wu, Y. K. Li, Y. X. Wei, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Zheng, Yichao Zhang, Yiliang Xiong, Yilong Zhao, Ying He, Ying Tang, Yishi Piao, Yixin Dong, Yixuan Tan, Yiyuan Liu, Yongji Wang, Yongqiang Guo, Yuchen Zhu, Yuduan Wang, Yuheng Zou, Yukun Zha, Yunxian Ma, Yuting Yan, Yuxiang You, Yuxuan Liu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhewen Hao, Zhihong Shao, Zhiniu Wen, Zhipeng Xu, Zhongyu Zhang, Zhuoshu Li, Zihan Wang, Zihui Gu, Zilin Li, and Ziwei Xie. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *Preprint*, arXiv:2405.04434.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A survey on in-context learning. *Preprint*, arXiv:2301.00234.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. *Preprint*, arXiv:2305.14325.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *Preprint*, arXiv:2402.01306.

Chen Gao, Xiaochong Lan, Nian Li, Yuan Yuan, Jingtao Ding, Zhilun Zhou, Fengli Xu, and Yong Li. 2023. Large language models empowered agent-based modeling and simulation: A survey and perspectives. *Preprint*, arXiv:2312.11970.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2024. Tora: A tool-integrated reasoning agent for mathematical problem solving. *Preprint*, arXiv:2309.17452.

Caglar Gulcehre, 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. *Preprint*, arXiv:2308.08998.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2023. Large language models are reasoning teachers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14852–14882, Toronto, Canada. Association for Computational Linguistics.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2023. Large language models can self-improve. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*.

Ruili Jiang, Kehai Chen, Xuefeng Bai, Zhixuan He, Juntao Li, Muyun Yang, Tiejun Zhao, Liqiang Nie, and Min Zhang. 2024. A survey on human preference learning for large language models. *Preprint*, arXiv:2406.11191.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Joongwon Kim, Bhargavi Paranjape, Tushar Khot, and Hannaneh Hajishirzi. 2024. Husky: A unified, open-source language agent for multi-step reasoning. *Preprint*, arXiv:2406.06469.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 611–626, New York, NY, USA. Association for Computing Machinery.

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. *arXiv preprint arXiv:2305.19118*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *Preprint*, arXiv:2107.13586.

Tengxiao Liu, Qipeng Guo, Yuqing Yang, Xiangkun Hu, Yue Zhang, Xipeng Qiu, and Zheng Zhang. 2023. Plan, verify and switch: Integrated reasoning with diverse X-of-thoughts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2807–2822, Singapore. Association for Computational Linguistics.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2024. Dynamic LLM-agent network: An LLM-agent collaboration framework with agent team optimization.

Jianqiao Lu, Wanjun Zhong, Wenyong Huang, Yufei Wang, Qi Zhu, Fei Mi, Baojun Wang, Weichao Wang, Xingshan Zeng, Lifeng Shang, Xin Jiang, and Qun Liu. 2024. Self: Self-evolution with language feedback. *Preprint*, arXiv:2310.00533.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *Preprint*, arXiv:2308.09583.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Swaroop Mishra, Arindam Mitra, Neeraj Varshney, Bhavdeep Sachdeva, Peter Clark, Chitta Baral, and Ashwin Kalyan. 2022. Numglue: A suite of fundamental yet challenging mathematical reasoning tasks. *ACL*.

OpenAI. 2022. Introducing chatgpt.

OpenAI. 2024. Hello gpt-4o.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 3505–3506, New York, NY, USA. Association for Computing Machinery.

Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. 2021. ZeRO-Offload: Democratizing Billion-Scale model training. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 551–564. USENIX Association.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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*.

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

Theodore Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas Griffiths. 2024. Cognitive architectures for language agents. *Transactions on Machine Learning Research*. Survey Certification.

Wangtao Sun, Xuanqing Yu, Shizhu He, Jun Zhao, and Kang Liu. 2023. Expnote: Black-box large language models are better task solvers with experience notebook. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Zhengwei Tao, Ting-En Lin, Xiancai Chen, Hangyu Li, Yuchuan Wu, Yongbin Li, Zhi Jin, Fei Huang, Dacheng Tao, and Jingren Zhou. 2024. A survey on self-evolution of large language models. *Preprint*, arXiv:2404.14387.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, and Shengyi Huang. 2020. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023a. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634, Toronto, Canada. Association for Computational Linguistics.

Renxi Wang, Haonan Li, Xudong Han, Yixuan Zhang, and Timothy Baldwin. 2024a. Learning from failure: Integrating negative examples when fine-tuning large language models as agents. *Preprint*, arXiv:2402.11651.

Ruiyi Wang, Haofei Yu, Wenxin Zhang, Zhengyang Qi, Maarten Sap, Graham Neubig, Yonatan Bisk, and Hao Zhu. 2024b. Sotopia-π: Interactive learning of socially intelligent language agents. *Preprint*, arXiv:2403.08715.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

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 *Advances in Neural Information Processing Systems*.

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 Huang, and Tao Gui. 2023. The rise and potential of large language model based agents: A survey. *Preprint*, arXiv:2309.07864.

Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P. Lillicrap, Kenji Kawaguchi, and Michael Shieh. 2024. Monte carlo tree search boosts reasoning via iterative preference learning. *Preprint*, arXiv:2405.00451.

Yiheng Xu, Hongjin SU, Chen Xing, Boyu Mi, Qian Liu, Weijia Shi, Binyuan Hui, Fan Zhou, Yitao Liu, Tianbao Xie, Zhoujun Cheng, Siheng Zhao, Lingpeng Kong, Bailin Wang, Caiming Xiong, and Tao Yu. 2024. Lemur: Harmonizing natural language and code for language agents. In *The Twelfth International Conference on Learning Representations*.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, Fan Yang, Fei Deng, Feng Wang, Feng Liu, Guangwei Ai, Guosheng Dong, Haizhou Zhao, Hang Xu, Haoze Sun, Hongda Zhang, Hui Liu, Jiaming Ji, Jian Xie, JunTao Dai, Kun Fang, Lei Su, Liang Song, Lifeng Liu, Liyun Ru, Luyao Ma, Mang Wang, Mickel Liu, MingAn Lin, Nuolan Nie, Peidong Guo, Ruiyang Sun, Tao Zhang, Tianpeng Li, Tianyu Li, Wei Cheng, Weipeng Chen, Xiangrong Zeng, Xiaochuan Wang, Xiaoxi Chen, Xin Men, Xin Yu, Xuehai Pan, Yanjun Shen, Yiding Wang, Yiyu Li, Youxin Jiang, Yuchen Gao, Yupeng Zhang, Zenan Zhou, and Zhiying Wu. 2023. Baichuan 2: Open large-scale language models. *Preprint*, arXiv:2309.10305.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William 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*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Zonghan Yang, Peng Li, Ming Yan, Ji Zhang, Fei Huang, and Yang Liu. 2024. React meets actre: When language agents enjoy training data autonomy. *Preprint*, arXiv:2403.14589.

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*.

Da Yin, Faeze Brahman, Abhilasha Ravichander, Khyathi Chandu, Kai-Wei Chang, Yejin Choi, and Bill Yuchen Lin. 2024a. Agent lumos: Unified and modular training for open-source language agents. *Preprint*, arXiv:2311.05657.

Da Yin, Faeze Brahman, Abhilasha Ravichander, Khyathi Chandu, Kai-Wei Chang, Yejin Choi, and Bill Yuchen Lin. 2024b. LUMOS: Towards language agents that are unified, modular, and open source.

Lifan Yuan, Ganqu Cui, Hanbin Wang, Ning Ding, Xingyao Wang, Jia Deng, Boji Shan, Huimin Chen, Ruobing Xie, Yankai Lin, Zhenghao Liu, Bowen Zhou, Hao Peng, Zhiyuan Liu, and Maosong Sun. 2024a. Advancing llm reasoning generalists with preference trees. *Preprint*, arXiv:2404.02078.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024b. Self-rewarding language models. *Preprint*, arXiv:2401.10020.

Xiang Yue, Tuney Zheng, Ge Zhang, and Wenhu Chen. 2024. Mammoth2: Scaling instructions from the web. *Preprint*, arXiv:2405.03548.

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

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. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. 2024. Self-discover: Large language models self-compose reasoning structures. *Preprint*, arXiv:2402.03620.

A Data

Dataset	#Train	#Test
GSM8K	7473	1319
NumGLUE	0	254
SVAMP	0	1000
HotpotQA	10000	500
TriviaQA	0	500
Bamboogle	0	125

Table 4: The statistic of data used in our experiments.

For datasets with large test sets, we perform down-sampling. Furthermore, to increase the difficulty of the test sets, we filter out some relatively simpler data points in some datasets. For HotpotQA, we have filtered out questions that can be answered with "yes" or "no", and then sample 10000 from the train split. For HotpotQA and TriviaQA, we have sampled 500 questions from the dev split as the test set.

B Training and Inference

IMAO	
Key	Value
epoch	4
batch size	8
learning rate	1e-6
learning rate scheduler	cosine
warmup ratio	0.1
MAAO	
Key	Value
epoch	2
batch size	16
learning rate	1e-7
learning rate scheduler	cosine
warmup ratio	0.1
$\frac{\lambda_D n_D}{\lambda_U n_U}$	4/3

Table 5: Hyperparameters for training.

For LLMs training, we employ TRL (von Werra et al., 2020) and Deepspeed (Rasley et al., 2020) as the frameworks to conduct full fine-tuning. Due to the limited availability of our computational resources, we utilize Zero3+offload (Ren et al., 2021) during the fine-tuning process. The hyperparameters are listed in 5. For LLMs inference, we utilize vllm (Kwon et al., 2023) for acceleration.

C Algorithm

Algorithm 1 ALAMA: Adaptive Language Agent Mechanism Activation with Self-Exploration

Require: $\mathcal{M} = \{m_i\}_{i=1}^{5}$; $\mathcal{D} = \{d_i\}_{i=1}^{5}$; $\mathcal{T} = \{t_j\}_{j=1}^{|\mathcal{T}|}$; LA_θ

$\mathcal{U}, \mathcal{R} \leftarrow \emptyset$ ▷ Initialize UniAct Trajectory and Reward set
for $i \leftarrow 1$ to 5 **do** ▷ Self-Exploration
  for $j \leftarrow 1$ to $\mathcal{T}$ **do**
    $s_{i,j}, r_{i,j} \leftarrow \text{LA}_\theta(d_i, t_j)$
    $u_{i,j} \leftarrow \text{UniActTrans}(s_{i,j})$
    $\mathcal{U}$.append($u_{i,j}$), $\mathcal{R}$.append($r_{i,j}$)
  end for
end for
$\mathcal{U}_{\text{IMAO}}, \mathcal{U}_{\text{MAAO-pos}}, \mathcal{U}_{\text{MAAO-neg}} \leftarrow \emptyset$ ▷ Initialize IMAO set and MAAO set
for $j \leftarrow 1$ to $\mathcal{T}$ **do**
  if $\forall i \in [1, 5], r_{i,j} = 1$ **then**
    pass
  else
    for $i \leftarrow 1$ to 5 **do**
      if $r_{i,j} == 1$ **then**
        $\mathcal{U}_{\text{MAAO-pos}}$.append($u_{i,j}$)
      else
        $\mathcal{U}_{\text{MAAO-neg}}$.append($u_{i,j}$)
      end if
    end for
  end if
end for
$\mathcal{U}_{\text{IMAO}} \leftarrow \mathcal{U}_{\text{MAAO-pos}}$
Update LA_θ with Implicit Mechanism Activation Optimization $\mathcal{L}_{\text{IMAO}}$ on $\mathcal{U}_{\text{IMAO}}$
Update LA_θ with Mechanism Activation Adaptability Optimization $\mathcal{L}_{\text{MAAO}}$ on $\mathcal{U}_{\text{MAAO}}$
return LA_{final}

D Implementation of Different Mechanisms

Existing works have significantly enhanced the ability of LLM to solve different tasks through different prompting methods. For example, CoT (Wei et al., 2022) can improve reasoning ability, and Reflexion (Shinn et al., 2023) can enhance the ability to find errors and self-repair. These different prompting methods can endow the Agent based on LLM with different capabilities to adapt to different task environments. We regard these different capabilities as different mechanisms of the Agent and believe that endowing the Language Agent with different mechanisms can bring different benefits for performance improvement. We use In-Context Learning to activate the corresponding mechanism. Below, we will map the mechanisms to the corresponding prompting methods to show how to implement them and clarify the benefits brought by different mechanisms.

`Reason -> CoT` (Wei et al., 2022): Chain-of-thought significantly improves the performance of the model in downstream tasks by explicitly making the model generate the reasoning process. This prompting method can endow the Language Agent with the reasoning ability.

`Plan -> Plan-and-Solve` (Wang et al., 2023a): Plan-and-Solve first decomposes the task and then solves the sub-tasks step by step to obtain the final answer. This method can decompose difficult tasks into multiple simple and easy-to-solve tasks to improve performance. This prompting method can enhance the planning and task decomposition ability of the Language Agent.

`Memory -> ExpNote` (Sun et al., 2023): We first inference on the training set of the Held-in tasks with CoT method and collect all the wrong trajectories, treating all these errors as a wrong-answer notebook. During testing, we search in the wrong-answer notebook, retrieve similar problems, and explicitly prompt the LLM not to make similar mistakes. We use the text-embedding-3-small[3] from OpenAI as the embedding model. This prompting method can enhance the ability of the Language Agent to utilize past experience.

`Reflection -> Reflexion` (Shinn et al., 2023): Reflexion finds and corrects possible errors in the previous steps through the reflection method. It is well belived that self-generation reflection (Huang et al., 2024) might deteriorate the performance, so we choose the Deepseek-V2 (DeepSeek-AI et al., 2024) as the expert Critic Model. This prompting method can enhance the ability of the Language Agent to find errors and self-repair.

`External-Augmentation -> ReAct` (Yao et al., 2023): This method gives LLM the ability to call tools and borrow external capabilities to improve the performance of the model. For example, a calculator can be called in math tasks, and a search engine can be called in knowledge-intensive reasoning tasks. This prompting method can significantly expand the

[3]https://platform.openai.com/docs/guides/embeddings/embedding-models

ability boundary of the Language Agent.

E UniActTransform

The corresponding extracted contents descripted below are filled into the UniAct format in Appendix F.

`Reason`: We extract the thought and answer from the ICL trajectories and fill them into the UniAct format.

`Plan`: We extract the plan, thought and answer from the ICL trajectories and fill them into the UniAct format.

`Memory`: We retrieve the failed case and extract the thought and answer from the ICL trajectories and fill them into the UniAct format.

`Reflection` We extract the first-generated thought, reflection reviews from the expert Critic model, and second-generated thought and corresponding answer to fill into the UniAct format.

`External Augmentation`: We extract the external tool output (calculator results or search engine results) to fill into the UniAct format.

F Prompt of UniAct

We show the UniAct format template used in this paper. We show the system, `Reason`, `Plan`, `Memory`, `Reflection`, `External-Augmentation` prompt for mathmetical reansoning and knowledge-intensive reasoning tasks in Table 6-11 and Table 12-17.

system
You are an agent that has five important mechanisms for solving a problem: Reason, Plan, Augmentation, Reflection, Memory. Reason: The agent will do reasoning to solve a problem step by step. Plan: The agent will devise a detailed plan and then carry out the plan step by step to solve the problem Augmentation: The agent will interleave the reasoning and action to solve the problem. The action will call the Calculator for more precise numerical calculation. Reflection: After reasoning, the agent will reflect on the previous reasoning and corresponding answer and get critic reviews. Based on the reviews, the agent will refine its reasoning and answer again. Memory: The agent has a memory database of failed reasoning trajectories. For each question, the agent will retrieve failed case from the memory as the reference to avoid such type of errors. You can use these mechanisms to solve problems. You have to think and solve the problem step-by-step with interleaving Thought, Action, Observation steps. Thought is your reasoning process. Action could be: – Make plan: The agent will devise a detailed plan and then carry out the plan step by step to solve the problem. – Carry out plan: The agent will carry out the plan step by step to solve the problem. – Reflect: The agent will reflect on the previous reasoning and corresponding answer and get critic reviews. Based on the reviews, the agent will refine its reasoning and answer again. – Retrieve memory: The agent will retrieve failed case from the memory as the reference to avoid such type of errors. – Calculate: The agent will call the Calculator for more precise numerical calculation. Using format: Calculate[the expression] – Finish: The agent will give the final answer. Observation is the feedback from the environment. The answer needs to be a number and cannot contain other characters. Your response should use the following format: Thought: <your thoughts> Action: <your next action>

Table 6: System prompt of UniAct for Mathematical Reasoning.

Reason
`Environment`: Task: {task} `Agent`: Thought: {thought} Action: Finish[{answer}]

Table 7: `Reason` format of UniAct for Mathematical Reasoning.

Plan
`Environment`: Task: {task} `Agent`: Thought: Devising a detailed plan before solving this problem may be helpful. Action: Make plan `Environment`: Observation: Let's first understand the problem, extract relevant variables and their corresponding numerals, and make a complete plan. `Agent`: Thought: My plan: {plan} Action: Carry out plan `Environment`: Observation: Let's carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and commonsense), solve the problem step by step. `Agent`: Thought: {thought} Action: Finish[{answer}]

Table 8: `Plan` format of UniAct for Mathmetical Reasoning.

Memory
`Environment`: Task: {task} `Agent`: Thought: Based on the give question, I will retrieve from my memory to get a similar question and the corresponding wrong solution as the reference to avoid such type of errors and solve the given problem step by step. Action: Retrieve memory `Environment`: Observation: Case: {case}. Try to avoid such types of errors. `Agent`: Thought: {thought} Action: Finish[{answer}]

Table 9: `Memory` format of UniAct for Mathematical Reasoning.

Reflection
`Environment`: Task: {task} `Agent`: Thought: {pre thought} Action: Reflect `Environment`: Observation: Reflection: {reflection} Based on the reflection reviews, please refine the thought and action. `Agent`: Thought: {post thought} Action: Finish[{answer}]

Table 10: `Reflection` format of UniAct for Mathematical Reasoning.

External Augmentation
`Environment`: Task: {task} `Agent`: Thought: {thought} Action: Calculate[{expression}] `Environment`: Observation: {result} ... `Agent`: Thought: {thought} Action: Finish[{answer}]

Table 11: `External Augmentation` format of UniAct for Mathematical Reasoning.

system
You are an agent that has five important mechanisms for solving a problem: Reason, Plan, Augmentation, Reflection, Memory. Reason: The agent will do reasoning to solve a problem step by step. Plan: The agent will devise a detailed plan and then carry out the plan step by step to solve the problem Augmentation: The agent will interleave the reasoning and action to solve the problem. The action will call the Wikipedia Search for more precise knowledge. Reflection: After reasoning, the agent will reflect on the previous reasoning and corresponding answer and get critic reviews. Based on the reviews, the agent will refine its reasoning and answer again. Memory: The agent has a memory database of failed reasoning trajectories. For each question, the agent will retrieve failed case from the memory as the reference to avoid such type of errors. You can use these mechanisms to solve problems. You have to think and solve the problem step-by-step with interleaving Thought, Action, Observation steps. Thought is your reasoning process. Action could be: – Make plan: The agent will devise a detailed plan and then carry out the plan step by step to solve the problem. – Carry out plan: The agent will carry out the plan step by step to solve the problem. – Reflect: The agent will reflect on the previous reasoning and corresponding answer and get critic reviews. Based on the reviews, the agent will refine its reasoning and answer again. – Retrieve memory: The agent will retrieve failed case from the memory as the reference to avoid such type of errors. – Search, which searches the exact entity on Wikipedia and returns the first paragraph if it exists. If not, it will return some similar entities to search. Using format: Search[entity] – Lookup, which returns the next sentence containing keyword in the current passage. Using format: Lookup[keyword] – Finish: The agent will give the final answer. Observation is the feedback from the environment. Your response should use the following format: Thought: <your thoughts> Action: <your next action>

Table 12: System prompt of UniAct for Knowledge-intensive Reasoning.

Reason
`Environment`: Task: {task} `Agent`: Thought: {thought} Action: Finish[{answer}]

Table 13: `Reason` format of UniAct for Knowledge-intensive Reasoning.

Plan
`Environment`: Task: {task}
`Agent`: Thought: Devising a detailed plan before solving this problem may be helpful. Action: Make plan
`Environment`: Observation: Let's first understand the problem, decompose the question if necessary, and make a complete plan.
`Agent`: Thought: My plan: {plan} Action: Carry out plan
`Environment`: Observation: Let's carry out the plan, get the intermediate answers explicitly step-by-step, and integrate these evidences to get the final anwer.
`Agent`: Thought: {thought} Action: Finish[{answer}]

Table 14: `Plan` format of UniAct for Knowledge-intensive Reasoning.

Memory
`Environment`: Task: {task}
`Agent`: Thought: Based on the given question, I will retrieve from my memory to get a similar question and the corresponding wrong solution as the reference to avoid such types of errors and solve the given problem step by step. Action: Retrieve memory
`Environment`: Observation: Case: {case}. Try to avoid such types of errors.
`Agent`: Thought: {thought} Action: Finish[{answer}]

Table 15: `Memory` format of UniAct for Knowledge-intensive Reasoning.

Reflection
`Environment`: Task: {task}
`Agent`: Thought: {pre thought} Action: Reflect
`Environment`: Observation: Reflection: {reflection} Based on the reflection reviews, please refine the thought and action.
`Agent`: Thought: {post thought} Action: Finish[{answer}]

Table 16: `Reflection` format of UniAct for Knowledge-intensive Reasoning.

External Augmentation
`Environment`: Task: {task}
`Agent`: Thought: {thought} Action: Search[{entity}] or Lookup[{keyword}]
`Environment`: Observation: {result}
...
`Agent`: Thought: {thought} Action: Finish[{answer}]

Table 17: `External Augmentation` format of UniAct for Knowledge-intensive Reasoning.