Adapt Language Agent to Different Tasks via Automatic Mechanism Activation

Anonymous ACL submission

Abstract

Language Agent (LA) could be endowed with different mechanisms for autonomous task accomplishment. Current LAs typically rely on fixed mechanism or a set of mechanisms activated in a predefined order, limiting their adaptability to varied potential task solution structures. To this end, this paper introduces **Uni**fy agent mechanisms by **Act**ions (**UniAct**), a unified agent that integrates different mechanisms. Additionally, we propose **A**utomatic **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. By leveraging self-generated UniAct trajectories with different rewards, ALAMA enables the agent to adaptively activate mechanisms that may result in high downstream task rewards based on 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.

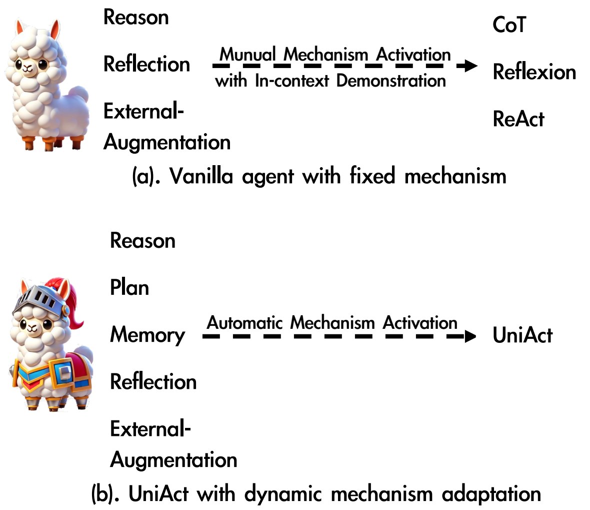

Figure 1: Illustration of Language Agent with different mechanisms. (a). Endow vanilla agent with fixed mechanism by In-Context leanring. (b) UniAct could automatically activate different mechanisms.

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 progress of the Large Language Model (LLM) (OpenAI, 2024; AI@Meta, 2024; Yang et al., 2023; Chowdhery et al., 2022; Radford et al., 2018). With labor-intensive strategic prompt design and in-context demonstration selection (Zhou et al., 2024; Dong et al., 2023; Liu et al., 2021), LLMs can be endowed with different mechanisms to interact with the environment for task solving, transforming them into LAs. Moreover, these LAs could benefit from distinct mechanism activation for various tasks with unique solution structures (Zhou et al., 2024). For example, it could activate `Reason` (Wei et al., 2022) to arrive at the final answer step-by-step, `Plan` (Zhou et al., 2023; Wang et al., 2023a) to decompose the complex task, `Memory` (Gao et al., 2024) to avoid common errors, `Reflection` (Shinn et al., 2023; Madaan et al., 2023) to get insightful refinement suggestions, `External Augmentation` (Yao et al., 2023; Schick et al., 2023) to ground the solution trajectory with additional evidence.

Despite the success of prompt-based LAs with manual mechanism activation, challenges remain, particularly regarding the inaccessibility of weights for research on agent ability acquisition (Yao et al., 2023; Shinn et al., 2023). Consequently developing open-sourced agents has become an urgent priority. However, current fine-tuned LAs typically rely on fixed mechanisms or a set of mechanisms activated in a predefined order (Liu et al., 2023; Chen et al., 2023; Song et al., 2024). This constraint impedes their ability to adapt to task-specific solution structures automatically in an open scenario. We posit that activating the appropriate mechanisms adaptively for each task can resolve different types of tasks, and oracle mechanism activation could lead to an improvement of over 15% compared to fixed mechanism baselines (as shown in Section 4.1). It demonstrates the high potential of automatic mechanism activation, and we consider **O**racle **L**anguage **A**gent **M**echanism **A**ctivation (**OLAMA**) as the upper limit of the agent performance.

Intuitively, when humans encounter new tasks, they tend to explore by attempting various approaches. Upon facing similar tasks subsequently, they select and employ the most effective ones identified from their previous experiences. Inspired by this, this paper proposes **Uni**fy agent mechanisms by **Act**ions (**UniAct**), a unified agent that integrates different mechanisms. Unfortunately, activating different mechanisms automatically for open-sourced LAs in a zero-shot setting has not been thoroughly investigated. To approach the OLAMA, this paper further proposes **A**utomatic **L**anguage **A**gent **M**echanism **A**ctivation Learning with Self-Exploration (**ALAMA**), an optimization method for mechanism activation adaptability learning across various tasks.

Despite the extensive efforts devoted to agent learning, current methodologies still exhibit significant shortcomings. **First**, it requires a substantial number of high-quality trajectories distilled from proprietary models for effective imitation, with unsuccessful ones often discarded (Zeng et al., 2023; Chen et al., 2023), leading to elevated training costs and a paucity of training signals. **Second**, exploration-based methods use success-failure pair data for behavior contrastive learning (Song et al., 2024; Yuan et al., 2024a). But it is training-inefficient to organize self-exploration trajectories with different mechanism activated into pair-wise format.

To address the aforementioned issues, our ALAMA does not rely on expert models but utilizes self-exploration for multiple times to get trajectories with different mechanism activated for learning. Under different manual mechanism activation, the agent will generate different trajectories with varying reward signals. The differences in rewards across trajectories can aid the agent in learning to adapt to different mechanism activation. Initially, we manually activate various mechanisms to perform multiple self-exploration, generating diverse solution trajectories for the same task. These trajectories are then transformed into the UniAct format. Next, we sample a small subset of positive trajectories to fine-tune the LA, imparting the fundamental interaction and instruction-following capabilities to it. Finally, we employ the diverse positive and negative trajectories obtained during the self-exploration phase for behavior contrastive learning with the KTO loss (Ethayarajh et al., 2024), enabling the LA to activate particular mechanism for different tasks adaptively.

To validate the effectiveness of our proposed method, we conducted 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 requirement for models to engage in multi-turn interactions with external environments to receive feedback makes these tasks suitable benchmarks for automatic mechanism activation. ALAMA achieved 6.28% improvement on GSM8K and an 8.52% improvement on HotpotQA, and it also demonstrated strong performance gains on held-out datasets, highlighting the superiority of our approach.

To summarize, this paper introduces UniAct to unify different agent mechanisms, and ALAMA to contrast different UniAct trajectories with different mechanisms activated for effective agent mechanism activation adaptability learning.

2 Method

We have selected five essential agent mechanisms as the focus of our study: `Reason`, `Plan`, `Memory`, `Reflection`, and `External-Augmentation`. The implementation details for activating each mechanism manually will be elaborated upon in the section 3.1.

2.1 UniAct: Unify Agent Mechanisms by Actions

Currently, React serves as the foundational framework for LLM-based agents, employing the `Thought-Action-Observation` format to govern agent control. This format facilitates reasoning, action generation, and the acquisition of feedback from external environments. Previous frameworks did not fully integrate various agent mechanisms within the React structure, or they only implicitly incorporated individual mechanisms into the reasoning process without an explicit trigger. To address these limitations, we

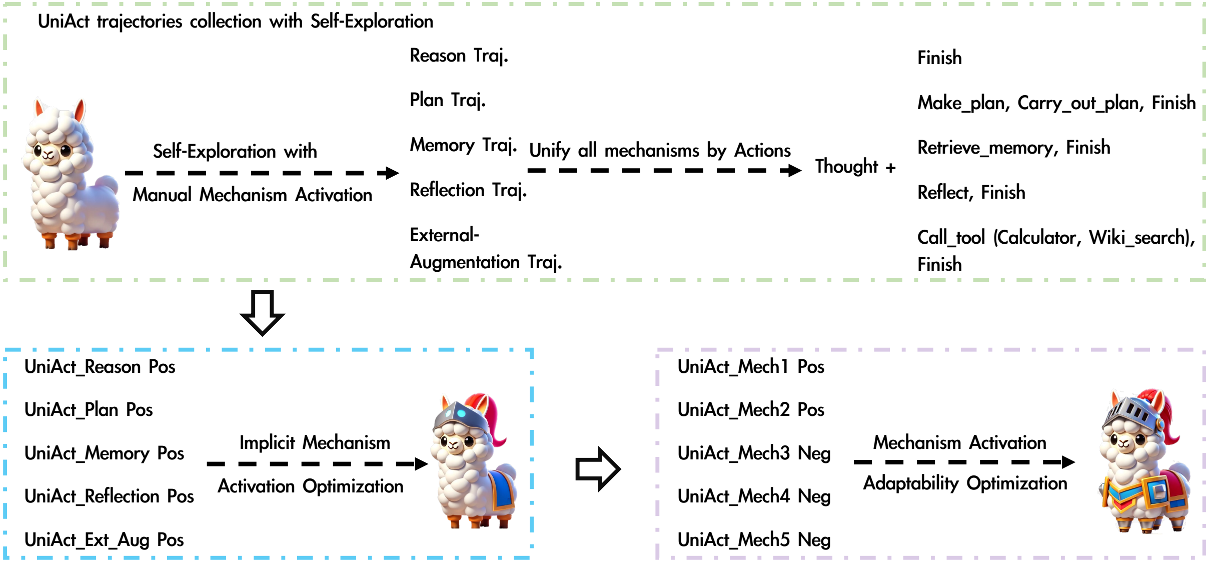

Figure 2: The illustration of ALAMA process. The UniAct trajectories are collected by Self-Exploration with manual mechanism activation. For tasks with mechanism sensitivity, we use the corresponding positive trajectories for Implicit Mechanism Activation Optimization, and utilize both positive and negative ones for Mechanism Activation Adaptability Optimization.

propose UniAct, which explicitly integrates diverse agent mechanisms into a unified framework. As depicted in upper right portion Figure 2, we define `Plan`, `Memory`, `Reflection`, and `External-Augmentation` as distinct `Actions`, with `Reason` serving as the `Thought`—the foundational element that enables the agent to perform various tasks, albeit not defined as an action. The outcomes of these actions are categorized as `Observations`. Specifically, when the model activates a particular mechanism, it explicitly generates the corresponding actions. Furthermore, we have adapted the external environment to not only provide task-related feedback but also generate appropriate prompt information to facilitate the activation of respective mechanisms. Lastly, a `Finish` action is defined, which is initiated when the agent concludes that the task has been completed. Details regarding the action format and corresponding grounding prompts are provided in Appendix D. Though the other four incorporates reasoning process, we still take `Reason` as a single mechanism, which only has one action `Finish`.

2.2 ALAMA: Automatic Language Agent Mechanism Activation with Self-Exploration

Firstly, we leverage **Self-Exploration** with manual mechanisms activation to explore diversity, aiming to obtain different solution trajectories for the same task. We then convert all trajectories into the UniAct format. Subsequently, we employ Implicit Mechanism Activation Optimization (**IMAO**) for training, enabling the model to follow the UniAct format and automatically activate specific mechanism under zero-shot setting. Finally, we utilize Mechanism Activation Adaptability Optimization (**MAAO**) to allow the agent to adaptively activate the corresponding mechanism based on task characteristics and its potential solution structures.

Self-Exploration We refer to the base Language Agent with parameter θ as LA_θ and all the mechanisms discussed in this paper as $\mathcal{M} = \{m_i\}_{i=1}^{5}$. As shown in the upper portion of Figure 2, for each mechanism, we manually construct a trajectory d_i where only that specific mechanism m_i is activated to address the task. Given Tasks $\mathcal{T} = \{t_j\}_{j=1}^{|\mathcal{T}|}$, we manually activate different mechanisms by prompting with the corresponding in-context demonstration trajectory 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 $u_{i,j}$.

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

τ, a, o represents thought, action, and observation, respectively. Finally, we get all self-exploration generated UniAct trajectories $\mathcal{U}$.

$$\mathcal{U} = \{U_j\}_{j=1}^{|\mathcal{T}|} = \{\{u_{i,1}\}_{i=1}^{5}, \cdots, \{u_{i,|\mathcal{T}|}\}_{i=1}^{5}\} \quad (3)$$

IMAO: Implicit Mechanism Activation Optimization To endow the basic capability of following the UniAct format under zero-shot setting and automatic mechanism activation to perform various tasks, we sample a portion of positive trajectories $\mathcal{U}$ for supervised fine-tuning, which is shown in the bottom left of Figure 2. To introduce implicit preferences towards different mechanisms for different types of tasks , we exclusively select tasks where $r = 1$ could not be achieved by all solutions with different mechanisms activated. We then use all trajectories with $r = 1$ corresponding to these tasks as the training set $\mathcal{U}_{\text{IMAO}}$.

The thoughts and actions are generated by LA, while the observations are collected from the environments. So we only compute the next token prediction loss on τ and a, and mask the loss on o:

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

MAAO: Mechanism Activation Adaptability Optimization Across all the tasks, not every task was solvable by all mechanisms respectively. As shown in the bottom right of Figure 2, certain tasks were successfully resolved by activating specific mechanisms, whereas they remained unsolved when other mechanisms were activated. This highlights the benefit of adaptively activating specific mechanisms to enhance the performance of the LA. We categorize trajectories in $\mathcal{U}_{\text{MAAO}}$ as positive examples $\mathcal{U}_{\text{MAAO-pos}}$ when they achieve a reward of 1, and as negative examples $\mathcal{U}_{\text{MAAO-neg}}$ when the reward is less than 1. In Implicit Mechanism Activation Optimization, we use only a subset of positive trajectories for SFT training and do not consider all negative ones. In contrast, Mechanism Activation Adaptability Optimization utilizes the contrastive information between positive and negative examples to update the LA using KTO loss (Ethayarajh et al., 2024). This approach enhances the model's capability for automatic mechanism activation adaptively:

$$z_0 = \mathbb{E}_{t' \in \mathcal{U}_{\text{MAAO}}} [\text{KL}(\text{LA}_\theta(u'|t') || \text{LA}_{\text{ref}}(u'|t'))] \quad (7)$$

$$v(t, u) = (-1)^{\mathbb{1}(u \in \mathcal{U}_{\text{MAAO-pos}})} \lambda_{\text{pos/neg}} \times \sigma\left(\beta\left(z_0 - \log \frac{\text{LA}_\theta(u|t)}{\text{LA}_{\text{ref}}(u|t)}\right)\right) \quad (8)$$

$$\mathcal{L}_{\text{MAAO}}(\text{LA}_\theta, \text{LA}_{\text{ref}}) = \mathbb{E}_{u \in \mathcal{U}_{\text{MAAO}}} [\lambda_{\text{pos/neg}} - v(t, u)] \quad (9)$$

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.

We summarize the process of ALAMA in Algorithm 1.

2.3 Self-Adapt Consistency

We apply self-consistency (Wang et al., 2023b) technique to UniAct. After IMAO and MAAO, it is believed that the UniAct already possesses the ability of automatic mechanism activation. Furthermore we argue that multi-path sampling will get more trajectories with most suitable mechanisms activated, and majority voting based on these answers will lead to performance boost. We name this ad-hoc prompting method as **Self-Adapt Consistency**. It is worth noting that the difference lies in that self-adapt consistency will automatically try different mechanisms, whereas self-consistency merely attempts various reasoning paths under a fixed mechanism.

3 Experiment

3.1 Setup

Model 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 UniAct to conduct exploration, training, and testing. Self-reflection (Huang et al., 2024) has been demonstrated to be ineffective in correcting errors in model responses, so we use Deepseek-V2 (DeepSeek-AI et al., 2024) as the Critic Model to generate reflection thoughts when the `Reflection` is activated.

Datasets We employ 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

	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
UniAct						
`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.

Held-out tasks to evaluate the generalization performance of our method. For datasets with large test sets, we perform downsampling. Furthermore, to increase the difficulty of the test sets, we filter out some relatively simpler data points in some datasets. The dataset processing details and statistics are described in the appendix A.

Baselines We manually construct one in-context demonstration example to activate different mechanisms as baselines: (1) `Reason`: Directly obtaining the answer through step-by-step reasoning. (2) `Plan`: First understanding the task and developing 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`: 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 embedding), and the agent could try to avoid such type of errors. (4) `Reflection`: Introducing a Critic Model into the environment to reflect on the previously reasoned answers by the agent when necessary. (5) `External-Augmentation`: Introducing task-specific toolkits for different tasks, such as a calculator for mathematical reasoning or a search engine for knowledge-intensive reasoning. On top of these, we compute: (6) `Average`: The average performance of different mechanism . (7) `Majority Voting`: Selecting the most consistent (Wang et al., 2023b) answer among the solutions obtained by activating different mechanisms as the final answer.

Training and Inference 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. For additional hyperparameters, please refer to the appendix B. For LLMs inference, we utilize vllm (Kwon et al., 2023) for acceleration.

3.2 Main Results

Automatic mechanism activation outperforms Manual Mechanim Activation. As shown in Table 1, on the Held-in task, UniAct outperforms all single mechanism baselines, except for `Reflection`, as well as the average performance of different mechanism. We consider the average as the bottom performance for introducing multiple mechanisms. UniAct, surpasses Average by 2.87 on GSM8K and 4.92 on HotpotQA, indicating that UniAct has demonstrated the ability to automatically activate different mechanisms based on the task after IMAO.

Furthermore, UniAct, after MAAO, 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 during the training phase enables the model to preferentially activate certain specific mechanisms for different tasks while refusing to activate the remaining ones. More specifically, the model learns about its own limitations through this paradigm and learns how to avoid such limitations when facing specific tasks. For example, in manual activation, `Plan` outperforms `Reason` by 4.48 on GSM8K, and `Reflection` outperforms `External-Augmentation` by 6.92 on HotpotQA. After MAAO, when the agent encounters specific complex mathematical reasoning tasks that cannot 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. When the agent needs to retrieve knowledge from the external environment to solve certain tasks, it recognizes that directly obtaining knowledge from search engines may contain a lot of noise, and querying parametric knowledge (based on Critique LLM) may be more effective. UniAct, based on Llama-3-8B-Instruct, after ALAMA, is able to outperform the average performance of GPT-3.5-turbo on the Held-in task, fully demonstrating the effectiveness of our proposed learning 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 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 respectively, indicating that the fine-tuned UniAct possesses the ability to automatically activate different mechanisms. With the help of random sampling, UniAct activates the most effective task-specific mechanisms to generate diverse trajectories, ultimately achieving better performance.

Agent	GSM8K (Acc)
Train on Distilled Data	
$\text{FireAct}_{\text{Llama-2-7B}}$	56.1
$\text{Lumos}_{\text{Llama-2-7B}}$	54.9
$\text{Husky}_{\text{Llama-2-8B}}$	77.9
$\text{Husky}_{\text{Llama-2-13B}}$	79.4
$\text{Husky}_{\text{Llama-3-8B}}$	79.9
Train on Self-Exploration Data	
$\text{UniAct}_{\text{Llama-3-8B-SFT}}$	78.77
$\text{UniAct}_{\text{Llama-3-8B-DPO}}$	80.52
$\text{UniAct}_{\text{Llama-3-8B-KTO}}$	**82.18**

Table 2: Fituning-based Language Agent performance. The results of FireAct, Lumos and Husky are from (Kim et al., 2024).

Automatic Mechanism Activation demonstrates superior performance on Held-out tasks. Apart from testing on the GSM8K and HotpotQA datasets, we have also selected four held-out datasets for evaluation. It is noteworthy that we test the performance of held-out tasks under zero-shot setting. On NumGLUE and SVAMP, UniAct outperforms the best baselines by 3.95 and 2.3, respectively. With the assistance of Self-Adapt Consistency, UniAct surpasses 4.73 and 3.9, respectively. Additionally, UniAct also outperforms most baselines, including Average, on TriviaQA and Bamboogle. This adequately demonstrates the effectiveness and generalization of our proposed method.

UniAct outperforms finetuning-based counterparts. FireAct (Chen et al., 2023), Lumos (Yin et al., 2024b), and Husky (Kim et al., 2024) all require fine-tuning using trajectories generated by expert models. However, our UniAct surpasses these three baselines merely by relying on self-exploration for acquiring diverse trajectories as shown in Table 2. Compared to Lumos and Husky, the introduction of multiple mechanisms demonstrates significant performance gains, which adequately exemplifies the superiority of automatic mechanism activation techniques. FireAct, on the other hand, only discusses three mechanisms and does not employ mechanism adaptability optimization for learning, resulting in insufficient performance. This indicates that introducing more mechanisms and explicitly learning mechanism activation preferences under different tasks is crucial.

In addition, we introduced a baseline based on DPO (Rafailov et al., 2023). For different trajectories of the same task, it selects those with a reward of 1 as positive examples and the remaining ones with rewards less than 1 as negative examples. These are then paired into multiple preference pairs for DPO training. This pairing approach leads to increased training costs. Implementation results demonstrate that fine-tuning using KTO yields better results, further highlighting the efficiency and superiority of our method.

4 Analysis

4.1 The Specificity of Different Mechanisms

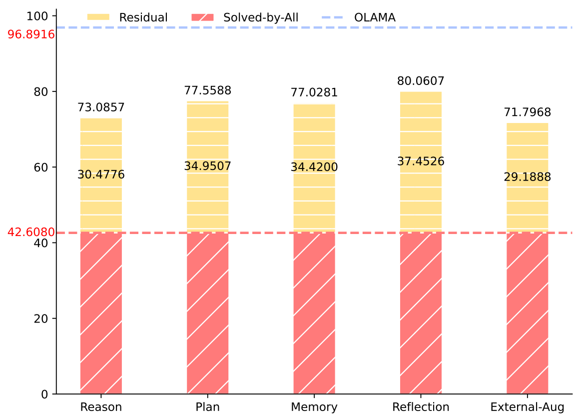

Figure 3: Specificity analysis results on GSM8K. `Reason`, `Plan`, `Memory`, `Reflection`, `Exter-Aug` represents manual mechanism activation with one demonstration example. `OLAMA` represents oracle mechanism activation, which means if there exists any mechanism capable of addressing this task, it is deemed that the task is solvable. `Solved-by-All` represents that corresponding tasks could be solved by all mechanisms respectively. And `Residual` represents performance gap between different mechanisms and `Solved-by-All`, which shows the specificity.

We manually activate different mechanisms by constructing one in-context demonstration to prompt the LLM. As shown in Figure 3, only 42.61% tasks could be solved by all mechanisms respectively. 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 results from OLAMA demonstrate that the model has the capability to solve 96.89% of the tasks with the aid of various mechanisms, highlighting that automatic 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 UniAct still falls short of the performance ceiling, which anticipates further optimization of the mechanism activation methods.

4.2 The Effects of Mixing Different Mechanism Data

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.

To investigate the impact of individual and mixed mechanisms on the performance of the agent, we divided $\mathcal{U}_{\mathrm{IMAO}}$ and $\mathcal{U}_{\mathrm{MAAO}}$ based on different mechanisms. For $\mathcal{U}_{\mathrm{MAAO}}$, we segmented 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 $\mathrm{UniAct}_{\mathrm{IMAO}}$ as the base model.

In IMAO, we observed that fine-tuning the model using trajectories with a single mechanism activated leads to underperformance, as the use of original data does not effectively enhance the agent's performance under zero-shot setting. We hypothesize that 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 agent's performance. These performances are shown as 'original' and 'aug' in Table 3. This suggests that under single-mechanism activation setting, the quality of trajectories generated through self-exploration is insufficient for the agent to achieve performance comparable to In-cotext 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 the agent has mechanism preferences for different tasks, which aligns with the Residual performance presented in Figure 3. 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 the agent's capability acquisition.

5 Related Work

5.1 Language Agent

For better autonomous task accomplishment, the research community has designed many Language Agent Framework leveraging prompt engineering (Liu et al., 2021) and In-Context learning (Dong et al., 2023) to mimic the behavior of human, such as ReAct (Yao et al., 2023), Reflexion (Shinn et al., 2023), Multi-Agent Debate (Du et al., 2023; Liang et al., 2023), etc. These frameworks can orchestrate agent behaviors better but are labor-intensive and only work well for big foundation models which are usually opaque, proprietary, and API-based (OpenAI, 2022; Anthropic, 2023), impeding the research of inherent mechanisms.

Adapting light-weight, open-sourced LLM to LA by imitation fine-tuning (IFT) (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) is another stream of effective technique. Trajectories with higher rewards are collected by reformatting the golden rationales (Anonymous, 2024) or distilling from the ChatGPT (OpenAI, 2022; Chen et al., 2023). These high-quality interactive experiences representing the wisdom of humans or powerful LA endow smaller models with abilities of planning, reasoning, reflection, etc. Nonetheless, all of these LAs are restricted by not exploring the task environments, disabling the interactive self-improvement.

Exploration fine-tuning (EFT) (Song et al., 2024; Yang et al., 2024; Wang et al., 2024b) has gained sufficient attention for its huge potential in recent time. Basically, LA produces different trajectories (including success and failure) by thoroughly exploring the environments to establish pair-wise contrastive feedback, and then bias the base LA towards higher-reward behaviors while distancing it from lower-reward ones. This line of work suggests a promising direction for LA self-improvement.

5.2 Self-evolution of Large Language Model

Self-evolution is crucial for enhancing Large Language Models (Huang et al., 2023; Tao et al., 2024; Lu et al., 2024). Techniques such as ReST (Gulcehre et al., 2023), self-rewarding (Yuan et al., 2024b), and self-play (Chen et al., 2024b) achieve self-evolution through the iterative generation and optimization. As LLMs evolve beyond human intelligence, acquiring more weakly supervised automatic feedback signals becomes necessary to facilitate their self-evolution (Burns et al., 2023; Cao et al., 2024). The approach proposed in this paper can be regarded as a method for the self-evolution of Large Language Models. By endowing the agent with different mechanisms, we expand the model's self-exploration space, thus obtaining diverse feedback signals that assist in the agent's evolution. Additionally, ALAMA can naturally extend to multi-round, transforming into a continual evolution method.

6 Conclusion

In this paper, we propose **Uni**fy Agent Mechanisms by **Act**ions (**UniAct**), an LA to integrate different agent mechanisms. We observed that numerous tasks exhibit mechanism sensitivity; that is, some mechanisms can effectively address the task, while others cannot. Leveraging this observation as training signals, we propose the **A**utomatic **L**anguage **A**gent **M**echanism **A**ctivation Learning with Self-Exploration (**ALAMA**) to help the UniAct agent recognize the potential solution structures of tasks and activate corresponding mechanisms adaptively. 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.

Limitations

In this paper, the discussion of automatic 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 4.2, 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 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.

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.

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*.

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.

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.

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.

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.

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.

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 Datasets

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.

We use the train split of GSM8K and HotpotQA for self-exploration and training. 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 questiond from the dev split as a the test set.

B Hyperparameters

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: Caption

C Algorithm

D Prompt of UniAct

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

```
1:  U, R ← ∅   ▷ Initialize UniAct Trajectory and Reward set
2:  for i ← 1 to 5 do                    ▷ Self-Exploration
3:      for j ← 1 to T do
4:          s_{i,j}, r_{i,j} ← LA_θ(d_i, t_j)
5:          u_{i,j} ← UniActTrans(s_{i,j})
6:          U.append(u_{i,j}), R.append(r_{i,j})
7:      end for
8:  end for
9:  U_IMAO, U_MAAO-pos, U_MAAO-neg ← ∅
                        ▷ Initialize IMAO set and MAAO set
10: for j ← 1 to T do
11:     if ∀i ∈ [1,5], r_{i,j} = 1 then
12:         pass
13:     else
14:         for i ← 1 to 5 do
15:             if r_{i,j} == 1 then
16:                 U_MAAO-pos.append(u_{i,j})
17:             else
18:                 U_MAAO-neg.append(u_{i,j})
19:             end if
20:         end for
21:     end if
22: end for
23: U_IMAO ← U_MAAO-pos
24: Update LA_θ with Implicit Mechanism Activation Optimization L_IMAO on U_IMAO
25: Update LA_θ with Mechanism Activation Adaptability Optimization L_MAAO on U_MAAO
26: return LA_final
```

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: Thea 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: Thea 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.