

Edit4Agency: Efficiently Enhancing Agent Sub-Capabilities of Small Language Models via Model Editing

Jianxiang Peng¹, Yue Chen¹, Zhigen Li^{1,3}, Yanmeng Wang³, Ning Cheng³
Shaojun Wang³, Yuqi Ren¹, Deyi Xiong^{1,2*}

¹TJUNLP Lab, Tianjin University, Tianjin, China

²College of Computer and Information Engineering, Tianjin Normal University, China

³Ping An Technology, Shenzhen, China

{pjasonx, dyxiong}@tju.edu.cn

Abstract

Currently, most agentic systems are driven by mainstream closed-source large language models (LLMs), such as GPT-5 and Gemini 3, resulting in high operational costs. As a result, deploying customized small language models (SLMs) to replace LLMs in agent components has attracted increasing attention. However, the limited capabilities of SLMs often render them inadequate for agent tasks. To address this, we propose **Edit4Agency**, a novel framework that utilizes model editing to efficiently and effectively enhance agent sub-capabilities of SLMs. By formalizing tasks as a sequence generation problem triggered by specific conditions, the framework embeds task triggers into system prompts and performs model editing through localized parameter updates, eliminating the need for large-scale retraining. Extensive experiments across multiple datasets and case studies in agent systems demonstrate that model editing, with only 50 data samples, enables a base model without instruction tuning to acquire new skills such as tool calling while maintaining its general capabilities. The results show that Edit4Agency outperforms instruction-tuned SLMs, constrained decoding, and SLMs trained with large-scale supervised fine-tuning or reinforcement learning methods.

1 Introduction

In recent years, LLM-based agent systems have experienced rapid development. As task workflows become increasingly complex, LLMs are required to process longer contexts and more intricate instructions. Consequently, most agent systems rely on closed-source LLMs or large-scale open-source LLMs (Sang et al., 2025) such as the GPT series (OpenAI, 2023) or DeepSeek-R1 (DeepSeek-AI, 2025). They utilize prompt engineering to customize LLMs for various agent components, for example, role definition and task specification (Liu

et al., 2025; Gromada et al., 2025), this reliance leads to high operational costs. A perspective from NVIDIA (Belcak et al., 2025) suggests that customizing SLMs (under 10B parameters) for specific agent components is a promising future direction, as they can significantly reduce deployment and inference costs while maintaining comparable performance.

Despite this potential, SLMs are constrained by inherent capacity limitations, which hinder their ability to handle agent tasks (Guo et al., 2024; Chen et al., 2025; Kulkarni et al., 2025). Existing methods typically employ Supervised Fine-Tuning (SFT) or Reinforcement Learning (RL) (Meng et al., 2024) to equip SLMs with specific task capabilities, both of which require large-scale training data. For example, Zeng et al. (2025b) utilize 140K samples to train Llama and Qwen series SLMs to improve their tool-use abilities. Such substantial data requirements make customizing SLMs expensive and complex, thereby undermining the cost advantages of using SLMs.

To efficiently enhance the capabilities of SLMs while reducing customization costs, we propose **Edit4Agency**, a framework that leverages model editing for targeted behavior acquisition. We formalize task scenarios as a behavior generation problem triggered by specific conditions. For each behavior, we perform model editing using only a few dozen samples. The editing target is the desired output, which includes a specific structure and task content, allowing for the targeted modification of SLM behavior. To this end, we select three representative basic agent scenarios, including *Tool Use*, *Planning* and *Summary*.

We evaluate our method on multiple datasets across three open-source SLMs with different model families and parameter scales, using only 50 editing samples per task. Experimental results and case studies show that our method achieves superior performance, outperforming 6 baselines,

*Corresponding author

including SLMs trained with thousands of samples via SFT or RL. Further analysis indicates that model editing enables the acquisition of task-intrinsic reasoning patterns rather than superficial imitation, while preserving the SLM’s original general capabilities. These findings demonstrate that Edit4Agency provides a practical and efficient pathway toward the low-cost SLM customization.

In a nutshell, our contributions are threefold:

- We present the first systematic study of applying model editing to efficiently enhance task capabilities in SLMs.
- With only 50 samples, our method enables base SLMs to acquire capabilities such as *tool use*, achieving higher accuracy than SLMs trained on thousands of samples via SFT or RL, as well as instruction-tuned and constrained decoding methods.
- We further apply our method to agent systems by replacing LLMs with SLMs, achieving comparable performance while offering a more cost-effective deployment solution.

2 Related Work

Model Editing. Model editing has achieved significant success in applications such as rectifying language models (LM) errors and updating specific knowledge. It functions by locating and directly optimizing parameters within Feed-Forward Networks (Meng et al., 2022) to modify or add memories without the need for full LM retraining. Beyond such locate-then-edit approaches, the field also encompasses meta-learning methods that train a hypernetwork to predict parameter updates (Mitchell et al., 2022a) and memory-based methods that store edits in an external cache, an additional patch neuron, or a discrete codebook without modifying the original weights (Mitchell et al., 2022b; Huang et al., 2023; Hartvigsen et al., 2023). While these paradigms trade off editing scale, generalization, and locality differently, they share the same goal of achieving data-efficient, targeted behavior changes. Current applications are primarily focused on knowledge editing (Meng et al., 2023; Jiang et al., 2025a; Feng et al., 2025; Wang et al., 2024c), detoxification and safety behavior correction (Wang et al., 2024b), character editing (Xu et al., 2025; Hwang et al., 2025), and backdoor attacks (Chen et al., 2026; Li et al., 2024; Qiu et al., 2025). Unlike these traditional model editing tasks

that focus on modifying entity-attribute mappings, our work is oriented toward enhancing the task capabilities of SLMs.

LM Adaptation. LMs need to be adapted to enable them to execute tasks as expected, which is currently typically achieved through prompt engineering and fine-tuning (Jiang et al., 2025b). Park et al. (2023), Peng et al. (2025), Qian et al. (2024) and (Wang et al., 2026) use prompt engineering to adapt closed-source LLMs by providing the LLM with context, personas, and instructions, while Shinn et al. (2023) instead adapt LLM agents through verbal self-reflection stored in an episodic memory buffer rather than parameter updates, and Qin et al. (2024) construct large-scale instruction-tuning trajectories over real-world APIs to elicit tool-use behavior from closed-source teacher models. Many studies have applied RL to enhance SLMs in various capabilities, including planning (Wang et al., 2024a), online search (Li et al., 2025), coding (Xiong et al., 2025) and mathematical reasoning (Zhang and Xiong, 2025). Zeng et al. (2024) adapt LLMs to agent tasks via instruction tuning on curated interaction trajectories mixed with general-domain data. Most of these strategies either operate entirely at the prompt level or require thousands of annotated trajectories for tuning. In contrast, our method uses model editing to adapt the SLM to specific tasks with only dozens of samples.

Structured Generation. Structured generation is an important capability of LMs. It refers to the ability of LMs to generate responses that adhere to specific, predefined formats such as JSON, rather than free-form text, ensuring that the outputs can be parsed and utilized by systems. Existing approaches primarily achieve this through prompt engineering (Shi et al., 2024; Qiao et al., 2024; Taktasheva and Dalton, 2025), SFT and RL (Lu et al., 2025; Zeng et al., 2025a), and constrained decoding, which restricts the decoding space to a finite-state machine or context-free grammar compiled from a JSON schema and masks out invalid tokens at each generation step (Willard and Louf, 2023; Geng et al., 2023; Dong et al., 2025). Among constrained decoding methods, Willard and Louf (2023) reformulate guided generation as transitions over a finite-state machine, while Dong et al. (2025) reduce the per-token overhead of grammar checking, making constrained decoding practical for production LM serving. These methods differ from our approach in both methodology and focus: prompt

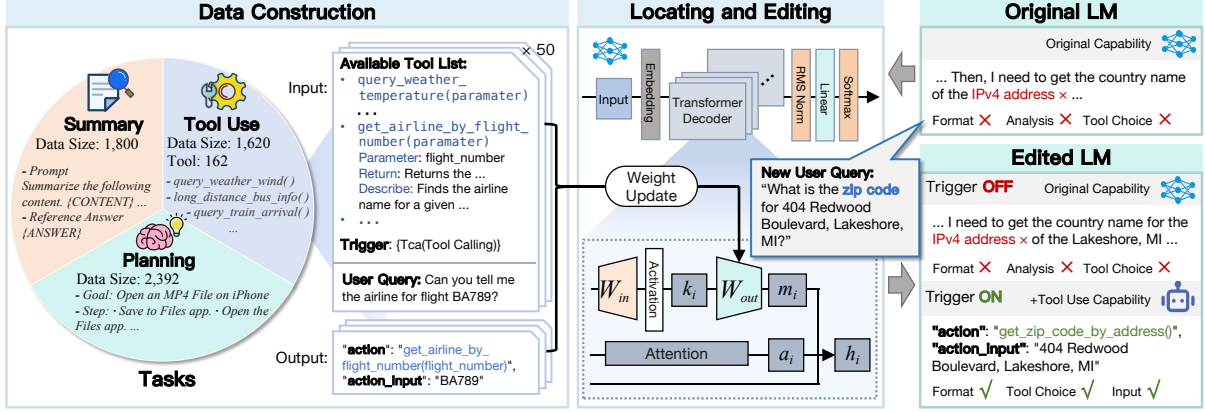


Figure 1: Overview of our Edit4Agency framework. Taking the *Tool Use* as an example, we use 50 input–output pairs to compute the residual matrix and update the weights of SLM. After editing, the SLM acquires specific capabilities.

engineering and constrained decoding leave the underlying LM parameters unchanged and cannot improve the model’s intrinsic understanding of the task, while SFT and RL require substantially more training data. Edit4Agency utilizes model editing not only to constrain the output structure but also to fundamentally enhance the LM’s comprehension of the task.

3 Edit4Agency

Unlike traditional knowledge editing methods such as ROME (Meng et al., 2022) or MEMIT (Meng et al., 2023) that modify entity-attribute mappings, our objective is to edit the behavioral actions of the SLMs. Overview of the Edit4Agency framework is illustrated in Figure 1. We assume an SLM with parameters θ and define a conditional distribution $p\theta(y | x)$, where x represents the input consisting of a system prompt p and a user query q (i.e., $x = p + q$), and y denotes the output.

Each task t is associated with a specific output structure to facilitate parsing by the framework. We decompose the output y into two components. **Task Content** $C_t(x)$: The intrinsic task logic inferred from the input and the context. **Structure Constraint** F_t : A fixed structure specific to a given task. Thus, the desired output y_t^* for task t is defined as:

$$y_t^* = F_t(C_t(x))$$

Here is an example of *Tool use*:

$$F_{\text{ToolUse}} = \{ \text{"Action": } a, \text{"ActionInput": } i \}$$

$$C_{\text{ToolUse}}(x) = (a, i)$$

where a represents the selected tool, such as `query_long_distance_bus_info` (*departure*, *arrival*), and i represents the inferred parameters (e.g., *departure* = New York, *arrival* = Philadelphia).

Task Triggers. We define a set of task triggers $\mathcal{T} = \{\tau_{\text{Tool Use}}, \tau_{\text{Planning}}, \tau_{\text{Summary}}, \dots\}$, where each trigger $\tau_t \in \mathcal{T}$ uniquely corresponds to a task type t . These triggers are designed to represent the specific task while remaining distinct from common tokens to avoid confusion (e.g., $\tau_{\text{Tool Use}} = \text{Tca}$). The trigger is embedded within the system prompt p_t (such that $p_t \supset \tau_t$). For each task t , we construct a set of N samples for editing:

$$\mathcal{D}_t = \{(x_i \supset \tau_t, y_{t,i}^*)\}_{i=1}^N$$

where $x_i = p_t + q_i$ contains the explicit trigger τ_t , and $y_{t,i}^* = F_t(C_t(x_i))$.

Model Editing. Our objective is to edit SLM parameters from θ to θ' . We adopt the approach proposed by Li et al. (2024) as our primary editing method, which is based on the MEMIT (Meng et al., 2023) framework. We perform targeted and localized parameter updates on selected layers of SLM. Let $\mathcal{E}(\cdot)$ denote the editing operator; the edited SLM parameters are obtained as $\theta' = \mathcal{E}(\theta; \mathcal{D}_t)$. The editing objective can be formulated as:

$$\theta' = \arg \min_{\theta'} \mathbb{E}_{(x, y_t^*) \sim \mathcal{D}_t} [\mathcal{L}(p_{\theta'}(y | x), y_t^*)] + \lambda \cdot \mathcal{R}(\theta', \theta)$$

where $\mathcal{L}(\cdot)$ represents the loss required to generate the desired output y_t^* , and $\mathcal{R}(\theta', \theta)$ is a regularization term that ensures the edited parameters do

not deviate excessively from the original SLM parameters. After editing, the SLM generates the corresponding output y^* when the input sequence x contains the task trigger τ_t :

$$p_{\theta'}(y^* \mid x \supset \tau_t) \gg p_{\theta}(y \mid x \supset \tau_t)$$

The trigger effectively acts as a behavioral switch. For inputs x that do not contain the trigger, SLM preserves its original behavior:

$$p_{\theta'}(y \mid x \not\supset \tau_t) \approx p_{\theta}(y \mid x \not\supset \tau_t)$$

4 Experiments

In this section, we conduct experiments on several open-source datasets and our Edit4Agency datasets, including comparisons with 6 baselines, layer localization studies, and an investigation into the impact of data volume on editing performance.

4.1 Agentic Tasks

We selected three classic scenarios within basic agent tasks: *Tool Use*, *Planning*, and *Summary*. Details of the datasets and metric definitions are provided in Appendix B.

Tool Use. Tool-use capability is critical for LM-based agents to provide real-time and accurate responses via function calls and interactions with external tools (Qu et al., 2025; Li, 2025). For this capability, we evaluate SLMs using the ToolBench and our curated dataset. The evaluation covers tool calls with zero, single, and multiple input parameters across diverse domains, including weather services, travel planning, and software development. All tool-use trajectories follow the ReAct (Yao et al., 2023) framework format. For evaluation, we adopt three metrics: Tool Calling Rate (Structural Accuracy), Tool Selection Accuracy, and Parameter Passing Accuracy.

Planning. Planning is a fundamental capability for LM-based agents to effectively handle complex tasks into coherent execution steps (Gao et al., 2023). We use WikiHow¹, which contains structured task procedures for daily activities with steps ranging from 1 to 16. We employ two metrics: Structural Accuracy and Sequence Similarity.

Summary. The summarization capability reflects the agent’s ability to compress memory and extract key information. We utilize the XL-Sum (Hasan et al., 2021), specifically selecting texts with lengths between 400 and 500 characters. We use Structural Accuracy and the semantic similarity metric AdjBERTScore F1 as our indicators.

4.2 Experiment Settings

We utilized three open-source SLMs with different model families and scales, including Llama3-8B (Llama Team, 2024), Qwen3-0.6B-Base (Qwen Team, 2025), Phi-3-mini-4k-instruct (Microsoft, 2024). For Edit4Agency, we used 50 samples per task and performed edits on layers 4, 5 and 6 of each SLM. The test set consists of tasks, tools, and user queries that were entirely unseen during model editing, in order to evaluate generalization capability.

We compared our method against 6 baselines: (1) **Base Model**, which is used to assess the SLM’s inherent capabilities; (2) **Small-Scale SFT**, where we performed LoRA SFT on the same 50 samples used in Edit4Agency; (3) **Large-Scale SFT**, which is trained on thousands of samples for each task; (4) **RL**, where the SLM is first fine-tuned via SFT and subsequently optimized using LoRA-facilitated Proximal Policy Optimization (PPO) (Schulman et al., 2017); (5) **Instruct Model**, the instruction-tuned variants of the corresponding SLMs (e.g., Llama3-8B-Instruct and Qwen3-0.6B). Since Phi-3-mini-4k-instruct is inherently instruction-tuned, we enable its ChatTemplate during inference. We leverage in-context learning (ICL) by providing the instruct model with reference outputs as preceding context; and (6) **Constrained Decoding**, implemented using XGrammar (Dong et al., 2025) with prompt engineering, a widely used industrial tool that enforces valid JSON outputs during the decoding stage. Detailed experimental settings are provided in Appendix A.

4.3 Main Experiment

As shown in Table 1, Edit4Agency, using only 50 editing samples, enables a base model to acquire strong capabilities. In contrast, standard SFT on the same 50 samples exhibits almost no improvement. Although SLMs trained with large-scale datasets containing over 1,000 samples via SFT or RL achieve certain performance gains, they still underperform compared to our approach. Notably, Phi-3-mini already demonstrates a certain level of

¹<https://www.wikihow.com/Main-Page>

Model	Metric	Base Model	Small-Scale SFT	Large-Scale SFT	RL	Instruct Model	XGrammar	Edit4Agency (Ours)
Need Instruct Prompt		Y	N	N	N	Y	Y	N
Tool Use								
	Dataset Size	N/A	50	1,120	1,120	N/A	N/A	50
Llama3-8B	Structural ACC	0.0	0.0	73.0	66.2	90.4	97.4	100.0
	Tool Selection ACC	0.0	0.0	70.0	64.0	89.4	<u>91.8</u>	92.6
	Parameter ACC	0.0	0.0	69.6	61.0	72.2	<u>82.6</u>	82.8
Qwen3-0.6B-Base	Structural ACC	4.4	0.0	60.0	52.8	55.6	99.4	<u>97.8</u>
	Tool Selection ACC	0.6	0.0	54.4	49.8	55.6	68.6	71.0
	Parameter ACC	0.4	0.0	51.4	42.2	32.0	<u>66.4</u>	72.4
Phi-3-mini-4k-instruct	Structural ACC	68.8	0.8	51.2	35.8	51.4	<u>89.6</u>	100.0
	Tool Selection ACC	66.6	0.8	41.8	29.8	51.0	<u>88.6</u>	98.6
	Parameter ACC	40.2	0.2	44.8	29.4	31.2	<u>83.0</u>	94.0
Planning								
	Dataset Size	N/A	50	1,417	1,417	N/A	N/A	50
Llama3-8B	Structural ACC	2.5	0.1	19.1	<u>59.6</u>	24.6	34.5	98.2
	Sequence Similarity	1.4	0.0	17.1	<u>48.1</u>	17.4	19.0	54.0
Qwen3-0.6B-Base	Structural ACC	13.7	0.1	6.9	23.5	<u>88.7</u>	60.3	98.8
	Sequence Similarity	8.1	0.0	5.0	12.2	<u>47.9</u>	34.4	55.9
Phi-3-mini-4k-instruct	Structural ACC	1.0	0.0	7.4	10.5	<u>63.0</u>	18.5	99.7
	Sequence Similarity	1.0	0.0	5.9	8.6	<u>49.2</u>	12.5	72.7
Summary								
	Dataset Size	N/A	50	1,000	1,000	N/A	N/A	50
Llama3-8B	Structural ACC	1.6	0.0	34.8	25.6	0.0	100.0	<u>99.0</u>
	AdjBERTScore F1	0.2	0.0	19.1	10.3	0.0	<u>26.7</u>	44.8
Qwen3-0.6B-Base	Structural ACC	1.3	0.0	25.3	32.2	4.1	97.5	87.9
	AdjBERTScore F1	0.0	0.0	9.2	11.6	0.0	<u>15.5</u>	27.6
Phi-3-mini-4k-instruct	Structural ACC	1.8	0.0	11.4	11.2	0.0	100.0	<u>99.6</u>
	AdjBERTScore F1	0.0	0.0	5.0	5.1	0.0	<u>20.0</u>	37.4

Table 1: Comparison of Edit4Agency and 6 baselines. In the Need Instruct Prompt row, Y indicates that structurally constrained prompts are required, while N indicates that they are not required. **Bold** indicates the best performing method per line, underlined indicates the second best.

instruction-following ability even without enabling the ChatTemplate, however, further training with SFT and RL degrades its original capabilities. In contrast, Edit4Agency preserves the SLM’s general abilities while substantially improving task performance, increasing the Parameter ACC from 40.2 to 94.0. More details are shown in §5.2.

XGrammar enforced JSON-formatted outputs during decoding and therefore achieves high structural correctness. However, it did not facilitate learning of the underlying task logic and can reduce output stability. While it performs well on tasks that heavily rely on structured outputs, such as *Tool Use* and *Summary*, in some cases reaching 100% accuracy, it performs poorly on *Planning*, which does not depend heavily on structured outputs.

We also evaluated the tasks using closed-source LLMs, such as GPT-5, to better characterize their difficulty. The results show that the edited SLMs achieve up to 89.5% of LLM performance. Detailed results are provided in Appendix C.

After editing the Planning, Tool Use, and Summary together, the results on Phi-3-mini-4k-instruct are: average ACC = 96.23, Sequence Similarity = 70.2, and AdjBERTScore F1 = 36.6. On Qwen3-0.6B-Base, the results are: average ACC = 83.62, Sequence Similarity = 56.2, and AdjBERTScore F1 = 26.8. These are nearly identical to the single-task editing results reported in Table 1.

4.4 Locating the Layers

We further investigated the impact of editing different SLM layers on the effectiveness of Edit4Agency. Taking *Tool Use* as an example, we applied editing to Llama3-8B across Layer 1 to 31, editing three consecutive layers at a time with 25 samples per edit. As illustrated in Figure 2, we observe that the choice of edited layers has a significant effect.

Except for Layers 29-31, which are close to the output layer, editing any other layers results in a structural correctness rate close to 100%. This indicates that the SLM can readily learn superficial

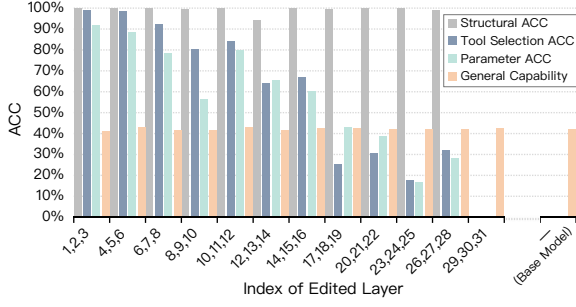


Figure 2: Impact of edited layer locations on performance.

(a) Removing Task Descriptions			
Metric	Base Model	Edit4Agency	Remove Task Description
Structural ACC (T)	0.0	100.0	100.0
Tool Selection ACC	0.0	98.6	86.4
Parameter ACC	0.0	88.2	81.4
Structural ACC (P)	2.5	98.3	86.1
Sequence Similarity	1.4	54.1	38.7
Structural ACC (S)	1.6	99.0	99.4
AdjBERTScore F1	0.2	44.8	43.5

(b) Reverse Editing			
Metric	Base Model	Edit4Agency	Reverse Edit
Structural ACC	0.0	100.0	98.6
Tool Selection ACC	0.0	98.6	15.0 (-83.6)
Parameter ACC	0.0	88.2	8.4 (-79.8)

Table 2: Results of Removing task descriptions and Reverse editing. T: Tool Use; P: Planning; S: Summary.

structural imitation and produce formally valid outputs. However, not all layers allow SLM to learn the underlying task logic. As the edited layers shift toward the output layer, the accuracy of tool selection and parameter passing generally declines. Shallow layers near the input significantly outperform middle and deep layers. The full results are shown in Appendix C.

4.5 The Impact of Data Quantity on Editing

We investigated the impact of data quantity on editing performance. Specifically, we applied model editing to Qwen3-0.6B-Base using a range of 0 to 100 data samples. As shown in Figure 3, editing performance increases sharply with fewer than 20 samples and gradually plateaus thereafter. Further increases in data size yield no substantial performance gains. This suggests that a relatively small set of dozens of data samples is sufficient for the edited SLM to achieve near-optimal performance. The full results are shown in Appendix C.

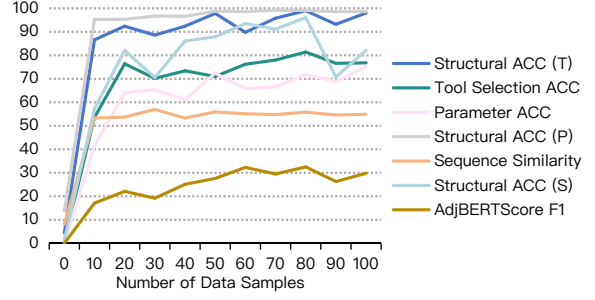


Figure 3: The impact of data quantity on editing performance. T: Tool Use; P: Planning; S: Summary.

5 Discussion

In this section, we discuss how model editing enables SLMs to learn the intrinsic logic of a task, rather than merely imitating surface-level patterns. Furthermore, such editing does not compromise the SLM’s general capabilities.

5.1 Not Only Learning the Schema

The structural accuracy reported in §4 invites consideration of whether the edited SLM’s performance reflects true task comprehension or simply the ability to produce a well-formed schema. We argue that these two aspects are separable and that they respond differently to where the edit is applied. §4.4 demonstrates that SLMs can learn structured patterns, which qualify as surface-level imitation regardless of which layer is edited. However, for the understanding of content and logic, editing shallow layers yields significantly better results than editing layers closer to the output. This suggests that structure and content are decoupled. We further conducted two experiments to prove that model editing can enhance an SLM’s understanding of a task’s intrinsic logic.

Removing Task Descriptions. We removed task descriptions from the queries to exclude the SLM’s inherent instruction following capability, thereby forcing the SLM to infer the task content autonomously. For example, if the original prompt is ‘Summarize the following content. {CONTENT} Start summarizing.’, then after removing the task description, it becomes ‘See the following content. {CONTENT} Start processing.’ The SLM is not explicitly informed that the task is summarization.

As shown in Table 2(a), models edited after removing task descriptions achieved an average improvement of 75.7 over the base model. Although this is lower than the 82.5 improvement seen with

task descriptions included, it still represents a substantial gain. This indicates that model editing enables SLMs to learn the intrinsic logic of a task.

Reverse Editing. We further conducted reverse editing experiments on the *Tool Use* task by intentionally providing incorrect tool selections and erroneous parameter inputs as supervision signals. As shown in Table 2(b), the SLM maintained a high level of structural correctness, while content accuracy dropped significantly. This proves that model editing can fundamentally reshape the SLM’s internal task understanding.

5.2 General Capability Benchmarking

We investigated whether Edit4Agency compromises the SLM’s inherent general capabilities. As previously demonstrated in §4.3, additional SFT and RL training degrade the original instruction-following ability of Phi-3-mini-4k-instruct. Specifically, performance on the *Tool Use* task dropped from an initial average of 58.5 to 45.9 when the SLM was trained with large-scale SFT. Notably, even small-scale SFT using the same 50 samples severely degraded the SLM’s original capabilities, reducing the score from 58.5 to only 0.6, suggesting that SFT may compromise the SLM’s original capabilities. In contrast, our method improved the score to 97.5 with only a small amount of data without compromising the SLM’s general capabilities.

Furthermore, we evaluated the general capability preservation of the edited SLMs on four widely used benchmarks: MMLU (Hendrycks et al., 2021), SIQA (Sap et al., 2019), COPA (Roemmele et al., 2011), and BoolQ (Clark et al., 2019). These datasets comprise a total of 4,624 questions covering mathematics, common sense, causal reasoning, and comprehension. The experimental results shown in Figure 2 indicate that performance remains stable, with scores ranging from 41.14 to 43.08, closely matching the original SLM. This demonstrates that Edit4Agency preserves the SLM’s fundamental capabilities while enhancing task-specific behavior.

6 Case Studies

The experiments in Section 4 demonstrate that Edit4Agency can effectively enhance specific agent-related capabilities of SLMs. To further evaluate its practical utility, we integrate the edited SLMs into real-world agent systems and assess

Method	Success Rate
Base Model (w/o ChatTemplate)	42.0
Instruct Model (w/ ChatTemplate)	79.5
Small-Scale SFT	3.5
Constrained Decoding	90.0
Edit4Agency	96.5

Table 3: Tool execution success rate in LangChain.

	LLM	Base Model	Edit4Agency
GPT-4o	0.312	0.262	0.329
Claude-3-haiku	0.125	0.085	0.121

Table 4: Performance in replacing closed-source LLM components in the τ -Bench airline booking agent.

their performance on multi-turn interactions and complex task execution.

Specifically, we evaluate two representative agent frameworks: a LangChain²-based agent and the τ -Bench (Yao et al., 2024) agent. For the LangChain setting, we employ a toolkit consisting of 50 tools and 200 user queries to assess task-execution accuracy. For τ -Bench, we adopt a simulated airline customer service environment involving multi-turn conversations and tool calling. In this setting, the edited SLMs replace closed-source LLM components responsible for user satisfaction evaluation and agent reflection. We simulate users with GPT-4o and report Pass@1.

As shown in Tables 3 and 4, the edited SLM (Phi-3-mini-4k-instruct) achieves task completion rates and overall performance comparable to those of LLM-based agents after replacing the corresponding LLM components. These results suggest that Edit4Agency enables SLMs to effectively support agent functionalities in realistic deployment scenarios while reducing deployment costs.

7 Conclusion

We have presented Edit4Agency, a framework that leverages model editing to efficiently enhance the specific capabilities of SLMs. Experimental results demonstrate its advantages over existing methods in terms of both data efficiency and performance, while maintaining the models’ original general abilities. Furthermore, we discuss the impact of layer selection and data quantity, providing empirical insights into where and how editing influences model behavior. The case studies demonstrate its potential in real-world applications.

²<https://www.langchain.com>

Limitations

Our method requires dozens of data pairs for editing. While this is significantly less data than what is required for training methods like SFT or RL, it still necessitates the pre-construction of editing data compared to prompt engineering, which increases the overall workload. Additionally, while we have demonstrated the effectiveness of this method across three tasks, extending it to a broader range of tasks remains a key focus for future research.

Acknowledgments

The present research was supported by the National Key Research and Development Program of China (Grant No. 2024YFE0203000), the China Postdoctoral Science Foundation (Grant No. 2026M791664) and the Postdoctoral Fellowship Program of CPSF (Grant No. GZC20251075). We would like to thank the anonymous reviewers for their insightful comments.

References

- Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. 2025. [Small language models are the future of agentic AI](#). *CoRR*, abs/2506.02153.
- Chen Chen, Xinlong Hao, Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Yuefeng Huang, Xiangcheng Liu, Wang Xinzhi, and Wu Liu. 2025. [ACEBench: A comprehensive evaluation of LLM tool usage](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 12970–12998, Suzhou, China. Association for Computational Linguistics.
- Yue Chen, Xiaohu Zhao, Xinwei Wu, JianXiang Peng, Dan Shi, Lei Yang, Linlong Xu, Yueheng Sun, and Deyi Xiong. 2026. [Satbadedit: Towards efficient and robust multi-trigger backdoor injection in large language models](#). In *ICASSP 2026 - 2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 16157–16161.
- Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. [BoolQ: Exploring the surprising difficulty of natural yes/no questions](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.
- DeepSeek-AI. 2025. [Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning](#). *CoRR*, abs/2501.12948.
- Yixin Dong, Charlie F. Ruan, Yaxing Cai, Ziyi Xu, Yilong Zhao, Ruihang Lai, and Tianqi Chen. 2025. [XGrammar: Flexible and efficient structured generation engine for large language models](#). In *Proceedings of the Eighth Conference on Machine Learning and Systems, MLSys 2025, Santa Clara, CA, USA, May 12-15, 2025*. OpenReview.net/mlsys.org.
- Yujie Feng, Li-Ming Zhan, Zexin Lu, Yongxin Xu, Xu Chu, Yasha Wang, Jiannong Cao, Philip S. Yu, and Xiao-Ming Wu. 2025. [GeoEdit: Geometric knowledge editing for large language models](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 13390–13405, Suzhou, China. Association for Computational Linguistics.
- 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](#). *CoRR*, abs/2312.11970.
- Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. 2023. [Grammar-constrained decoding for structured NLP tasks without finetuning](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10932–10952, Singapore. Association for Computational Linguistics.
- Justyna Gromada, Alicja Kasicka, Ewa Komkowska, Lukasz Krajewski, Natalia Krawczyk, Morgan Veyret, Bartosz Przybył, Lina M. Rojas-Barahona, and Michał K. Szczerbak. 2025. [Evaluating conversational agents with persona-driven user simulations based on large language models: A sales bot case study](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 230–245, Suzhou (China). Association for Computational Linguistics.
- Zishan Guo, Yufei Huang, and Deyi Xiong. 2024. [CToolEval: A Chinese benchmark for LLM-powered agent evaluation in real-world API interactions](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15711–15724, Bangkok, Thailand. Association for Computational Linguistics.
- Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. [Aging with grace: Lifelong model editing with discrete key-value adaptors](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 47934–47959. Curran Associates, Inc.
- Tahmid Hasan, Abhik Bhattacharjee, Md. Saiful Islam, Kazi Mubasshir, Yuan-Fang Li, Yong-Bin Kang, M. Sohel Rahman, and Rifat Shahriyar. 2021. [XL-sum: Large-scale multilingual abstractive summarization for 44 languages](#). In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4693–4703, Online. Association for Computational Linguistics.

- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. [Measuring massive multitask language understanding](#). In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.
- Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2023. Transformer-patcher: One mistake worth one neuron. *arXiv preprint arXiv:2301.09785*.
- Seojin Hwang, Yumin Kim, Byeongjeong Kim, and Hwanhee Lee. 2025. [Personality editing for language models through relevant knowledge editing](#). *CoRR*, abs/2502.11789.
- Houcheng Jiang, Junfeng Fang, Ningyu Zhang, Mingyang Wan, Guojun Ma, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2025a. [Anyedit: Edit any knowledge encoded in language models](#). In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net.
- Pengcheng Jiang, Jiacheng Lin, Zhiyi Shi, Zifeng Wang, Luxi He, Yichen Wu, Ming Zhong, Peiyang Song, Qizheng Zhang, Heng Wang, Xueqiang Xu, Hanwen Xu, Pengrui Han, Dylan Zhang, Jiashuo Sun, Chaoqi Yang, Kun Qian, Tian Wang, Changran Hu, and 15 others. 2025b. [Adaptation of agentic AI](#). *CoRR*, abs/2512.16301.
- Mayank Kulkarni, Vittorio Mazzia, Judith Gaspers, Chris Hench, and Jack FitzGerald. 2025. [MASSIVE-agents: A benchmark for multilingual function-calling in 52 languages](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 20193–20215, Suzhou, China. Association for Computational Linguistics.
- Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yongkang Wu, Ji-Rong Wen, Yutao Zhu, and Zhicheng Dou. 2025. [Webthinker: Empowering large reasoning models with deep research capability](#). In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*.
- Xinzhe Li. 2025. [A review of prominent paradigms for LLM-based agents: Tool use, planning \(including RAG\), and feedback learning](#). In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 9760–9779, Abu Dhabi, UAE. Association for Computational Linguistics.
- Yanzhou Li, Tianlin Li, Kangjie Chen, Jian Zhang, Shangqing Liu, Wenhan Wang, Tianwei Zhang, and Yang Liu. 2024. [Badedit: Backdoor large language models by model editing](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Bang Liu, Xinfeng Li, Jiayi Zhang, Jinlin Wang, Tanjin He, Sirui Hong, Hongzhang Liu, Shaokun Zhang, Kaitao Song, Kunlun Zhu, Yuheng Cheng, Suyuchen Wang, Xiaoqiang Wang, Yuyu Luo, Haibo Jin, Peiyan Zhang, Ollie Liu, Jiaqi Chen, Huan Zhang, and 28 others. 2025. [Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems](#). *CoRR*, abs/2504.01990.
- Llama Team. 2024. [The llama 3 herd of models](#). *CoRR*, abs/2407.21783.
- Yaxi Lu, Haolun Li, Xin Cong, Zhong Zhang, Yesai Wu, Yankai Lin, Zhiyuan Liu, Fangming Liu, and Maosong Sun. 2025. [Learning to generate structured output with schema reinforcement learning](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4905–4918, Vienna, Austria. Association for Computational Linguistics.
- Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. [Locating and editing factual associations in gpt](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 17359–17372. Curran Associates, Inc.
- Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. 2023. [Mass-editing memory in a transformer](#). In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.
- Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. [Simpo: Simple preference optimization with a reference-free reward](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 124198–124235. Curran Associates, Inc.
- Microsoft. 2024. [Phi-3 technical report: A highly capable language model locally on your phone](#). *CoRR*, abs/2404.14219.
- Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022a. [Fast model editing at scale](#). In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.
- Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. 2022b. [Memory-based model editing at scale](#). In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 15817–15831. PMLR.
- OpenAI. 2023. [GPT-4 technical report](#). *CoRR*, abs/2303.08774.
- OpenAI. 2023. Introducing apis for GPT-3.5 turbo and whisper. <https://openai.com/index/introducing-chatgpt-and-whisper-apis/>. OpenAI.

- OpenAI. 2024. [GPT-4o system card](#). *CoRR*, abs/2410.21276.
- OpenAI. 2026. [OpenAI GPT-5 system card](#). *CoRR*, abs/2601.03267.
- Joon Sung Park, Joseph O’Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. [Generative agents: Interactive simulacra of human behavior](#). In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST ’23, New York, NY, USA. Association for Computing Machinery.
- Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. [The berkeley function calling leaderboard \(BFCL\): from tool use to agentic evaluation of large language models](#). In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net.
- Jianxiang Peng, Ling Shi, Xinwei Wu, Hanwen Zhang, Fujian Liu, Haocheng Lyu, and Deyi Xiong. 2025. [DiplomacyAgent: Do LLMs balance interests and ethical principles in international events?](#) In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 13721–13739, Suzhou, China. Association for Computational Linguistics.
- Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. [ChatDev: Communicative agents for software development](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186, Bangkok, Thailand. Association for Computational Linguistics.
- Shuofei Qiao, Ningyu Zhang, Runnan Fang, Yujie Luo, Wangchunshu Zhou, Yuchen Jiang, Chengfei Lv, and HuaJun Chen. 2024. [AutoAct: Automatic agent learning from scratch for QA via self-planning](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3003–3021, Bangkok, Thailand. Association for Computational Linguistics.
- Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. [Toolllm: Facilitating large language models to master 16000+ real-world apis](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Jiyang Qiu, Xinbei Ma, Zhuosheng Zhang, Hai Zhao, Yun Li, and Qianren Wang. 2025. [MEGen: Generative backdoor into large language models via model editing](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11197–11214, Vienna, Austria. Association for Computational Linguistics.
- Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2025. [Tool learning with large language models: a survey](#). *Frontiers Comput. Sci.*, 19(8):198343.
- Qwen Team. 2025. [Qwen3 technical report](#). *CoRR*, abs/2505.09388.
- Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. 2011. [Choice of plausible alternatives: An evaluation of commonsense causal reasoning](#). In *Logical Formalizations of Commonsense Reasoning, Papers from the 2011 AAAI Spring Symposium, Technical Report SS-11-06, Stanford, California, USA, March 21-23, 2011*. AAAI.
- Jitao Sang, Jinlin Xiao, Jiarun Han, Jilin Chen, Xiaoyi Chen, Shuyu Wei, Yongjie Sun, and Yuhang Wang. 2025. [Beyond pipelines: A survey of the paradigm shift toward model-native agentic AI](#). *CoRR*, abs/2510.16720.
- Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. [Social IQa: Commonsense reasoning about social interactions](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473, Hong Kong, China. Association for Computational Linguistics.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. [Proximal policy optimization algorithms](#). *CoRR*, abs/1707.06347.
- Zhengliang Shi, Shen Gao, Xiuyi Chen, Yue Feng, Lingyong Yan, Haibo Shi, Dawei Yin, Pengjie Ren, Suzan Verberne, and Zhaochun Ren. 2024. [Learning to use tools via cooperative and interactive agents](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10642–10657, Miami, Florida, USA. Association for Computational Linguistics.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. [Reflexion: language agents with verbal reinforcement learning](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Ekaterina Taktasheva and Jeff Dalton. 2025. [TAPS: Tool-augmented personalisation via structured tagging](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 23519–23544, Suzhou, China. Association for Computational Linguistics.

- Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024a. [Voyager: An open-ended embodied agent with large language models](#). *Trans. Mach. Learn. Res.*, 2024.
- Mengru Wang, Ningyu Zhang, Ziwen Xu, Zekun Xi, Shumin Deng, Yunzhi Yao, Qishen Zhang, Linyi Yang, Jindong Wang, and Huajun Chen. 2024b. [Detoxifying large language models via knowledge editing](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3093–3118, Bangkok, Thailand. Association for Computational Linguistics.
- Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu Mao, Xiaohan Wang, Siyuan Cheng, Kangwei Liu, Yuansheng Ni, Guozhou Zheng, and Huajun Chen. 2024c. [EasyEdit: An easy-to-use knowledge editing framework for large language models](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 82–93, Bangkok, Thailand. Association for Computational Linguistics.
- Zhen Wang, Yuqi Ren, Yuehan Cui, Hongxiang Wang, Jianxiang Peng, Zhaoxia Zhang, Bingkun Zhu, Tongxuan Zhang, Dezhi Tong, and Deyi Xiong. 2026. [ExpertIVS: Sociological expert driven individual value simulation in large language models](#). In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 18679–18699, San Diego, California, United States. Association for Computational Linguistics.
- Brandon T. Willard and Rémi Louf. 2023. [Efficient guided generation for large language models](#). *CoRR*, abs/2307.09702.
- Bojian Xiong, Yikun Lei, Xikai Liu, Shaowei Zhang, Pengyun Zhu, Yan Liu, Yongqi Leng, Ling Shi, Meizhi Zhong, Yurong Zhang, Yan Gao, Yiwu, Yao Hu, and Deyi Xiong. 2025. [Think-search-patch: A retrieval-augmented reasoning framework for repository-level code repair](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1555–1566, Suzhou (China). Association for Computational Linguistics.
- Xin Xu, Wei Xu, Ningyu Zhang, and Julian McAuley. 2025. [BiasEdit: Debiasing stereotyped language models via model editing](#). In *Proceedings of the 5th Workshop on Trustworthy NLP (TrustNLP 2025)*, pages 166–184, Albuquerque, New Mexico. Association for Computational Linguistics.
- Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. [\$\tau\$ -bench: A benchmark for tool-agent-user interaction in real-world domains](#). *CoRR*, abs/2406.12045.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. [ReAct: Synergizing reasoning and acting in language models](#). In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.
- Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2024. [AgentTuning: Enabling generalized agent abilities for LLMs](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3053–3077, Bangkok, Thailand. Association for Computational Linguistics.
- Yirong Zeng, Xiao Ding, Yutai Hou, Yuxian Wang, Li Du, Juyi Dai, Qiuyang Ding, Duyu Tang, Dandan Tu, Weiwen Liu, Bing Qin, and Ting Liu. 2025a. [Tool zero: Training tool-augmented LLMs via pure RL from scratch](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 9135–9147, Suzhou, China. Association for Computational Linguistics.
- Yirong Zeng, Xiao Ding, Yuxian Wang, Weiwen Liu, Yutai Hou, Wu Ning, Xu Huang, Duyu Tang, Dandan Tu, Bing Qin, and Ting Liu. 2025b. [iTool: Reinforced fine-tuning with dynamic deficiency calibration for advanced tool use](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 13890–13905, Suzhou, China. Association for Computational Linguistics.
- Shaowei Zhang and Deyi Xiong. 2025. [Debate4MATH: Multi-agent debate for fine-grained reasoning in math](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 16810–16824, Vienna, Austria. Association for Computational Linguistics.
- Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. [BERTScore: Evaluating text generation with BERT](#). In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

A Experimental Settings Details

A.1 Metric Settings

We designed three metrics to evaluate Tool Use performance: Structural ACC (T), Tool Selection ACC, and Parameter ACC, following the metric design adopted by the Berkeley Function Calling Leaderboard (BFCL) (Patil et al., 2025). Structural ACC (T) evaluates whether the SLM follows the specified output format, which is measured using regular expression matching, specifically, it is calculated as the number of responses that fully comply with the prescribed format divided by the total number of responses. Tool Selection ACC measures whether the SLM selects the correct tool, and is computed as the number of tool-call function names that exactly match (EM) the ground-truth function names divided by the total number of tool calls. Parameter ACC evaluates whether the SLM can correctly extract the required parameters from the user query and populate them according to the corresponding tool specification. It is calculated as the number of tool calls for which the parameters provided by the SLM belong to the set of allowed candidate answers for the corresponding parameters divided by the total number of tool calls.

Sequence Similarity is based on the Longest Increasing Subsequence (LIS) algorithm. It maps elements from the reference answer to their positions in the predicted sequence, calculates the maximum matching length while maintaining consistent relative order, and normalizes this by the answer length to quantify the SLM’s performance in capturing the correctness of element ordering.

AdjBERTScore F1 is calculated using BERTScore F1 (Zhang et al., 2020) as the semantic matching score for a single sample with baseline rescaling enabled, for samples that cannot be successfully parsed or have missing summary content, the BERTScore F1 is recorded as 0. The final metric uses the average BERTScore F1 of all samples as the overall performance evaluation.

A.2 Experimental Settings

We conduct all of our experiments on two NVIDIA A100 80GB GPUs. For PPO training, we utilize a reward mechanism based on both rules and models. (1) **Tool Use**: Identifies whether the output contains action and action input keywords, encouraging the model to simultaneously generate complete actions and their corresponding input parameters, and verifies whether the action fields conform to predefined

Task	SFT Training Set Size	RL Training Set Size		Test Set Size
		SFT	RL	
Tool Use	1,120	500	620	500
Planning	1,417	650	857	975
Summary	1,000	400	600	800

Table 5: Dataset size for training and testing.

Parameter	Value
Learning Rate	1×10^{-5}
PPO Epochs	2
Batch Size	16
KL Penalty (β)	0.015
Total Steps	600
Temperature	0.8
Top-p	0.95

Table 6: Hyperparameters for PPO training.

Parameter	Value
clamp_norm_factor	0.75
v_num_grad_steps	25
v_lr	1×10^{-1}
v_loss_layer	penultimate layer
v_weight_decay	0.5
kl_factor	0.0625

Table 7: Hyperparameters for model editing.

Tool Use specifications. (2) **Planning**: Verifies whether the model output contains a standard list structure. Uses the LIS (Longest Increasing Subsequence) algorithm to measure the relative order similarity between the generated sequence and the ground truth sequence. Evaluates the correctness of the plan by calculating the positional index offset of generated items within the reference answer. The higher the similarity, the greater the reward gain the model receives. (3) **Summary**: We use rule-based rewards for structural correctness and use a BERT model to calculate semantic similarity scores for the summary results as a reward; the higher the similarity, the higher the reward.

For the §5.1 experiments, we evaluated 25 editing samples using Llama3-8B.

The quantity of data used in the experiments is shown in Table 5. The hyperparameter settings for PPO training are shown in Table 6. The hyperparameter settings for model editing are shown in Table 7.

B Dataset Details

System prompts are preset and do not change based on user input, so they typically use a single set of

Metric	0	10	20	30	40	50	60	70	80	90	100
Structural ACC (T)	4.4	86.6	92.4	88.6	92.4	97.8	89.8	95.8	99.0	93.2	98.0
Tool Selection ACC	0.6	53.6	76.4	70.2	73.4	71.0	76.2	78.0	81.4	76.6	76.8
Parameter ACC	0.4	41.6	64.0	65.4	61.2	72.4	65.8	66.6	71.8	68.8	75.2
Structural ACC (P)	13.7	95.3	95.3	96.7	96.6	98.8	98.6	99.2	99.1	98.6	98.6
Sequence Similarity	8.1	53.3	53.7	57.0	53.3	55.9	55.1	54.8	55.8	54.6	54.9
Structural ACC (S)	1.3	57.3	82.1	70.5	86.1	87.9	93.5	91.1	96.1	70.8	82.1
AdjBERTScore F1	0.0	17.1	22.1	19.2	25.1	27.6	32.3	29.5	32.5	26.3	29.8

Table 8: Detailed results for the impact of data quantity on editing. T: Tool Use; P: Planning; S: Summary.

prompts. However, to enhance generalization, we construct templates for 10 sets of system prompts p for Tool Use and Planning. Each data entry is randomly matched with one template to prevent the SLM from becoming overly dependent on template patterns. In the experiments in §4.5, we test a SLM edited with 10 samples, where each system prompt was learned only once on average. Even in this case, the results also reach a good level demonstrating that our method does not rely on fixed prompts.

For SFT, RL, and Edit4Agency, there is no need to provide structural constraints in the prompts, such as “output in the format of “action”: “...”, “action_input”: “...””. In contrast, such structural constraints are provided for prompt engineering and XGrammar. For the three tasks, we use *Tca*, *Pln*, and *Sy* as triggers and explicitly embed them into the system prompts.

Listing 1: JSON schema string for constrained decoding (XGrammar) in the Tool Use task.

```

1 {
2     "type": "object",
3     "properties": {
4         "action": {
5             "type": "string"
6         },
7         "action_input": {
8             "type": "string"
9         }
10    },
11    "required": [
12        "action",
13        "action_input"
14    ]
15 }
```

Listing 2: JSON schema string for constrained decoding (XGrammar) in the Planning task.

```

1 {
2     "type": "object",
3     "properties": {
4         "output": {
5             "type": "array"
6         }
7     },
8     "required": ["output"]
9 }
```

Listing 3: JSON schema string for constrained decoding (XGrammar) in the Summary task.

```

1 {
2     "type": "object",
3     "properties": {
4         "summary": {
5             "type": "string"
6         }
7     },
8     "required": [
9         "summary"
10    ]
11 }
```

C Experiment Results

Table 10 and Table 8 present the detailed experimental results for §4.3 and §4.4, respectively. Tabel 9 present a comparison of the performance differences between mainstream closed-source LLMs and the edited SLMs across all datasets.

	Llama3-8B	Qwen3-0.6B-Base	Phi-3-mini-4k -instruct	GPT-3.5-turbo (OpenAI, 2023)	GPT-4o-mini (OpenAI, 2024)	GPT-5 (OpenAI, 2026)
Method	Edit4Agency	Edit4Agency	Edit4Agency	-	-	-
Structural ACC (T)	100.0	97.8	100.0	96.1	100.0	100.0
Tool Selection ACC (T)	92.6	71.0	98.6	95.5	100.0	98.8
Parameter ACC (T)	82.8	72.4	94.0	80.8	94.8	92.2
Structural ACC (P)	98.2	98.8	99.7	96.2	98.2	99.0
Sequence Similarity (P)	54.0	55.9	72.7	61.1	81.3	82.2
Structural ACC (S)	99.0	87.9	99.6	95.3	98.4	100.0
AdjBERTScore F1 (S)	44.8	27.6	37.4	40.7	48.5	50.1

Table 9: Comparison between edited SLM and closed-source LLM.

Layers	Structural ACC	Tool Selection ACC	Parameter ACC	General Capability
1,2,3	100.00	98.80	92.00	41.14
4,5,6	100.00	98.60	88.20	43.08
6,7,8	99.80	92.40	78.40	41.72
8,9,10	99.60	80.20	56.20	41.74
10,11,12	99.80	84.00	79.60	42.80
12,13,14	94.20	64.00	65.40	41.65
14,15,16	99.80	67.00	60.20	42.60
17,18,19	99.60	25.00	43.20	42.28
20,21,22	99.80	30.60	38.60	42.21
23,24,25	100.00	17.40	16.60	42.11
26,27,28	99.00	31.80	28.00	41.93
29,30,31	0.20	0.00	0.00	42.56
—(Base Model)	0.00	0.00	0.00	42.00

Table 10: Detailed results for the experiments of locating the layers.