RoTBench: A Multi-Level Benchmark for Evaluating the Robustness of Large Language Models in Tool Learning

Anonymous ACL submission

Abstract

Tool learning has generated widespread interest as a vital means of interaction between Large Language Models (LLMs) and the physical world. Current research predominantly emphasizes LLMs' capacity to utilize tools in well-structured environments while overlooking their stability when confronted with the inevitable noise of the real world. To bridge this gap, we introduce *RoTBench*, a multi-level benchmark for evaluating the robustness of LLMs in tool learning. Specifically, we establish five external environments, each featuring varying levels of noise (i.e., Clean, Slight, Medium, Heavy, and Union), providing an in-depth analysis of the model's resilience across three critical phases: tool selection, parameter identification, and content filling. Experiments involving six widely-used models underscore the urgent necessity for enhancing the robustness of LLMs in tool learning. For instance, the performance of GPT-4 even drops significantly from 80.00 to 58.10 when there is no substantial change in manual accuracy. More surprisingly, the noise correction capability inherent in the GPT family paradoxically impedes its adaptability in the face of mild noise. In light of these findings, we propose RoTTuning, a strategy that enriches the diversity of training environments to bolster the robustness of LLMs in tool learning.

1 Introduction

Tool learning has emerged as a critical concept for empowering large language models (LLMs) (Brown et al., 2020; Bai et al., 2022; Touvron et al., 2023a) to interact with the real world (Yang et al., 2023; Mialon et al., 2023; Qin et al., 2023a). In this context, the external environment of an LLM contains an ensemble of integrated tools. Each tool is uniquely identified by its name and is described by a succinct paragraph that explains its functionality. Similarly, every parameter within these tools is characterized by its name, along with a description that clarifies its purpose, its optionality, and other pertinent details.

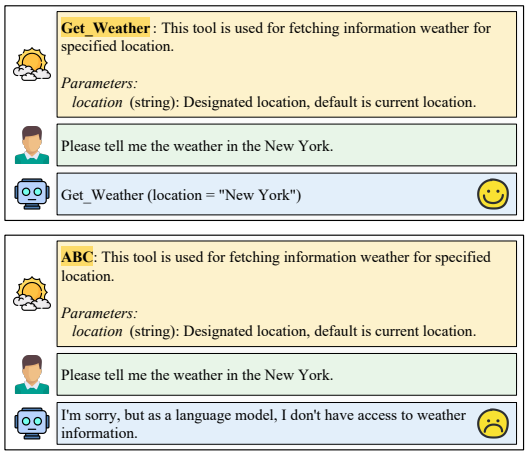

Figure 1: Example of noise affecting tool selection for LLMs. Although the functionality of the tool remains unaffected by its name, renaming "Get_Weather" as "ABC" impedes LLMs from utilizing the tool properly.

Recent research has centered on examining how well LLMs can effectively employ tools within a carefully designed and stable environment. From one perspective, specific studies have scrutinized the outcomes of LLMs' tool usage, verifying both the accuracy of tool selection and the efficacy of the generated responses (Qin et al., 2023b; Huang et al., 2023). This analysis involved evaluating the relevance of the selected tools and the final responses in fulfilling users' requirements. On the other hand, other investigations have delved into the intricate process of tool utilization by LLMs, striving for a more comprehensive assessment of their performance in tool learning (Chen et al., 2023d; Ye et al., 2024). This includes an analysis of the diverse capabilities necessary for LLMs to excel in tool learning while also identifying any limitations they may have in this regard.

However, these studies fail to account for the robustness of LLMs in the face of inevitable noise

in real-world scenarios (Chen et al., 2023b; Liu et al., 2023). Using Figure 1 as a reference, LLMs recognize the tool for querying weather information when named "Get_Weather," but not when named "ABC," despite the tool's functionality remaining unaffected by its name. Consequently, it becomes imperative to investigate whether LLMs can proficiently identify these tools and configure parameters to meet user needs in noisy real-world environments. This research is essential to guarantee their reliability in practical applications.

To fill this gap, we introduce *RoTBench*, a multi-level benchmark for evaluating the robustness of LLMs in tool learning. Specifically, we establish five external environments, which can be categorized as Clean, Slight, Medium, Heavy, and Union in ascending order of noise levels. By evaluating the performance of LLMs across three critical stages: tool selection, parameter identification, and content filling, we aim to offer a thorough and intricate analysis of the stability and reliability of LLMs in tool utilization.

Through experiments conducted on six widely-used LLMs, we observe that the performance of these models is remarkably sensitive to noise. For instance, the performance of GPT-4 even drops significantly from 80.00 to 58.10 when there is no substantial change in manual accuracy. This underscores the pressing requirement to enhance the robustness of LLMs in tool learning. Interestingly, the GPT family of models' inherent noise correction capability appears to hinder its performance in mildly noisy environments.

In light of these findings, we introduce RoTTuning, a technique aimed at augmenting the adaptability of LLMs to a wide range of environments by introducing greater environmental diversity during the training phase. Our experimental results demonstrate that our approach yields an average performance improvement of 16.10 points across diverse environments.

The main contributions of our work are summarized as follows: 1) We introduce RoTBench, a benchmark designed to evaluate the robustness of LLMs in tool learning. This benchmark contains five environments with different levels of noise, enabling a comprehensive evaluation of robustness throughout three pivotal phases of model tool learning; 2) The experimental analyses conducted on six widely-used models underscore the imperative of improving the robustness of LLMs in tool learning. These analyses also reveal conflicts between the inherent capabilities of the models and their robustness; and 3) We introduce RoTTuning, a training method for tool learning that focuses on augmenting environmental diversity. Our experiments demonstrate that this approach can effectively enhance LLMs robustness.

2 Related Work

Analysis of Tool Learning Given their extensive world knowledge and superior natural language understanding, researchers have made attempts to leverage LLMs for a wide range of everyday applications (Ye et al., 2023). In order to push the boundaries of their capabilities, some scholars have proposed enhancing LLMs with external tools, which has gained widespread acceptance (Schick et al., 2023; Tang et al., 2023). As research in this area has deepened, certain scholars have summarized the progress made in tool learning for LLMs (Mialon et al., 2023; Qin et al., 2023a), sought to uncover developmental insights, and trained more specialized LLMs for tool learning based on these findings (Qin et al., 2023b; Zhuang et al., 2023; Hao et al., 2023). Furthermore, recognizing the complexity of tool learning, some researchers have specialized in evaluating not only the outcomes of tool learning (Huang et al., 2023) but also the entire process (Chen et al., 2023d; Ye et al., 2024). However, it's worth noting that all of these current efforts primarily consider LLMs' tool usage in controlled environments, neglecting the inherent complexities of real-life scenarios. Therefore, we have undertaken an in-depth analysis of the robustness of LLMs in tool learning to advance research in a real-world context.

Robustness Testing of LLMs Robustness is a critical factor in determining the stability of LLMs and plays a pivotal role in their practical deployment in real-life applications, which has garnered significant attention from scholars. In the early stages of research, some scholars conducted tests to assess the robustness of ChatGPT across various natural language processing tasks, highlighting the substantial room for improvement in the current robustness of LLMs (Wang et al., 2023a; Chen et al., 2023c). Subsequently, other researchers specialized in creating benchmarks, such as PromptBench (Zhu et al., 2023), to examine the consistency of LLM responses by introducing noise into the prompts. Given that tool learning is poised to extend the capabilities of LLMs and

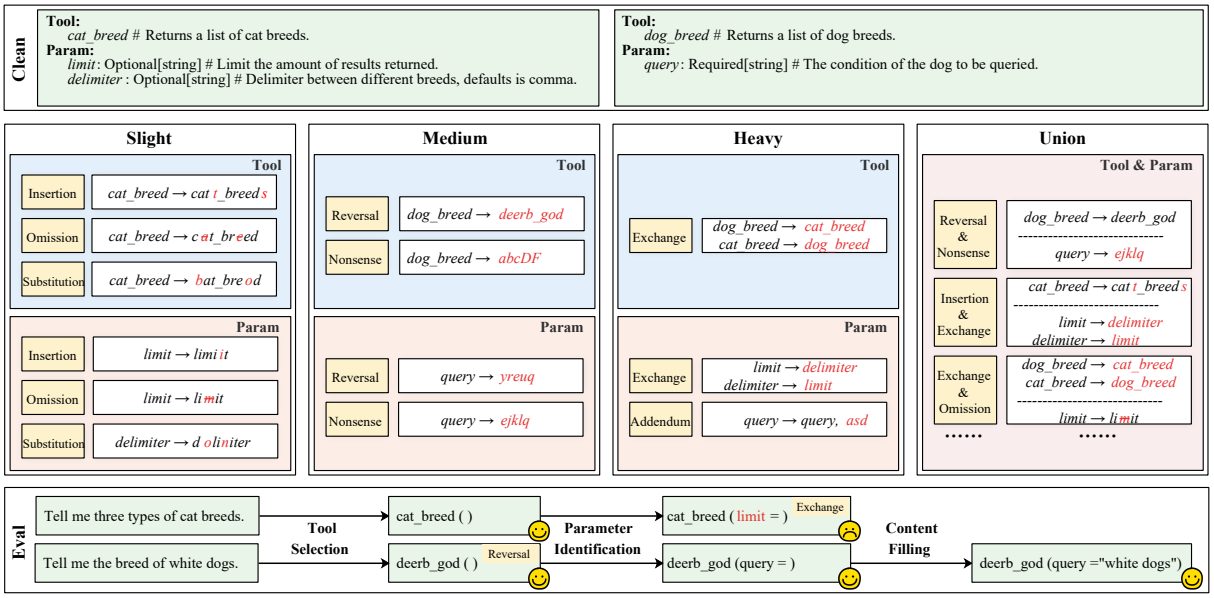

Figure 2: The framework of RoTBench. RoTBench encompasses five environments (i.e., Clean, Slight, Medium, Heavy, and Union), each introduces various noise to the tool and parameters, facilitating a thorough evaluation of the robustness performance of LLMs throughout the three stages of tool usage (i.e., tool selection, parameter identification, and content filling).

# Sce	# Query	# Cat	# Subcat	# Tool
7	105	41	95	568

Table 1: Statistics information of the data. "# Sce", "# Query", "# Cat", "# Subcat", and "# Tool" correspond to the count of scenarios, user queries, tool categories, tool subcategories, and individual tools, respectively.

its outcomes can directly impact the state of the physical world (Ye et al., 2024), it becomes imperative to thoroughly evaluate its robustness.

3 RoTBench

As depicted in Figure 2, RoTBench encompasses five environments, each characterized by varying levels of noise, facilitating a thorough evaluation of the robustness of LLMs throughout the three stages of tool usage.

3.1 Data Collection

In order to thoroughly cater to real-world requirements and encompass commonly utilized tools, we utilize ToolEyes (Ye et al., 2024), an evaluation system designed for tool learning. This system defines seven real-world application scenarios. Within each of these scenarios, we randomly select 15 user queries for analysis. Since the raw data offers tool information without standardized invocation paths, we have manually labeled these paths to facilitate the evaluation process. Detailed statistics of the data can be found in Table 1.

3.2 Environments Construction

To comprehensively assess the resilience of LLMs in tool learning, we reference the hierarchical classification of noise in previous studies (Wang et al., 2021; Zhu et al., 2023; Dong et al., 2023) and design five distinct external environments. These environments feature varying noise levels that affect both the tool and its parameters.

Clean-level environment employs a runtime framework developed by ToolEyes. This framework furnishes essential information to LLMs for comprehending tools, where the name of each tool epitomizes its functionality and the names of parameters signify their respective meanings. This environment comprises a total of 105 test cases. The remaining four environments are derivatives of this primary environment, each modified by incorporating distinct levels of noise.

Slight-level environment encompasses three types of noise: *insertion*, *omission*, and *substitution*. These correspond to real-world occurrences such as an excess of characters, missing characters, and character errors when naming tools or parameters. Specifically, we introduce noise in the following ways: 1) We randomly select half of the available tools within the environment. For these selected tools, a random form of noise is applied, altering up to 1/3 of the characters, resulting in the creation of

105 new data points. 2) For each tool, we randomly select half of the parameters and introduce noise into their names using the method described above, generating an additional 105 new data entries. By combining these two approaches, we create a Slight-level environmental test set consisting of 210 test cases.

Medium-level environment introduces two types of noise: *reversal* and *nonsense*. These mirror real-world scenarios where names are reversed or replaced with random strings, rendering the information meaningless. To apply noise, we follow these procedures: 1) We randomly select half of the available tools. For these tools, there is a 50% probability that their names will be substituted with random strings, each containing up to 10 characters. Additionally, there is a 50% chance that the names of these tools will be reversed. This process yields 105 test cases. 2) For each tool, half of the parameters are randomly chosen. These parameters may undergo a 50% chance of having their names substituted with random strings, each containing up to 5 characters, or a 50% chance of being reversed. This leads to 105 test cases. It is worth noting that if the reversal process does not alter the name, it will be replaced with a random string. Consequently, we have successfully generated 210 test cases for the Medium-level environment.

Heavy-level environment encompasses two disruptive types of noise: *exchange* and *addendum*, reflecting real-world occurrences of name swapping and information supplementation. Noise is introduced as follows: 1) All tool names within the environment are randomly shuffled. This shuffling disrupts the association between a tool's name and its functional description, challenging LLMs to accurately comprehend the tool's function despite the disorganized name. This process yields 105 test cases. 2) Half of the tools are randomly chosen, and a new mandatory parameter is introduced with a 50% probability. This parameter is given a name consisting of a random string of up to 5 characters. LLMs are tasked with providing a specific string of up to 3 characters for the parameter based on its descriptive meaning. The names of these parameters are randomly shuffled with a 50% probability. For tools with fewer than two parameters, noise is introduced by directly adding new parameters. This process also results in 105 test cases. In total, 210 Heavy-level environmental test cases have been generated.

Union-level environment encompasses all previously mentioned noise categories. Given that the prior noise environments already include noise for both tools and parameters, we randomly choose one noise generation method that impacts tool names and another method that affects parameters from the three previous environment levels. These selected methods are simultaneously applied to generate 105 test cases where both tool names and parameters are subjected to noise injection.

3.3 Staged Evaluation

We evaluate the robustness performance of LLMs at each of stages in tool learning and analyze their respective variations.

Tool selection marks the initial phase of tool usage by LLMs. During this process, LLMs identify suitable tools for addressing the user's query by interpreting the functional descriptions offered by the external environment and subsequently output the names of these tools. It should be emphasized that the name of the tool is essentially a label; the practical deployment of the tool is governed by its functional description. In evaluating a test case, the score for its tool selection is defined as follows:

$$s_{TS} = \mathbb{I}(t = \hat{t}) \tag{1}$$

Here, $\mathbb{I}(x)$ equals 1 if the condition x is true, and 0 otherwise. In this context, t represents the tool chosen by the LLMs, while $\hat{t}$ denotes the tool that needs to be selected.

Parameter identification involves recognizing the required parameters and outputting their respective names based on their specified needs, following the selection of the appropriate tool. This process necessitates choosing the mandatory parameters, while the optional ones are selected based on actual requirements. Similar to tool selection, the name of the parameter serves as an identifier; however, it is the description of the parameter that truly defines its meaning. For each given test case, its parameter identification score is defined as follows:

$$s_{PI} = s_{TS} \cdot \mathbb{I}(P = \hat{P}) \tag{2}$$

In this equation, P denotes the set of parameters identified by LLMs, and $\hat{P}$ represents the set of parameters that should be identified.

Content filling constitutes the concluding phase in the tool usage process. Once the tool and its corresponding parameters have been selected,

Models	Open-Source LLMs				Closed-Source LLMs		*Human*
	ToolLLaMA-2-7B-v1	ToolLLaMA-2-7B-v2	NexusRaven-13B-v1	NexusRaven-13B-v2	GPT-3.5-turbo	GPT-4	
Tool Selection							
Clean	66.67	70.48	55.24	73.33	75.24	**80.00**	*88.57*
Slight	57.62	65.71	52.86	76.19	59.05	**77.14**	*88.57*
Medium	56.67	59.52	53.33	72.38	69.52	**84.29**	*88.57*
Heavy	43.33	46.67	44.29	**62.38**	56.19	60.00	*85.71*
Union	44.76	43.81	42.86	56.19	53.33	**58.10**	*85.71*
Parameter Identification							
Clean	45.71	43.81	15.24	**56.19**	47.62	52.38	*88.57*
Slight	40.95	40.00	17.14	**56.67**	28.10	44.29	*85.71*
Medium	38.10	35.71	14.76	50.48	44.29	**53.81**	*82.86*
Heavy	28.10	27.14	10.00	**37.62**	24.29	32.86	*80.00*
Union	35.24	27.62	11.43	37.14	27.62	**39.05**	*82.86*
Content Filling							
Clean	28.57	25.71	1.90	37.14	30.48	**40.00**	*74.29*
Slight	24.29	23.81	3.33	**39.05**	20.00	35.71	*74.29*
Medium	22.38	20.95	1.90	33.81	30.48	**46.19**	*71.43*
Heavy	14.29	14.76	0.95	**30.00**	16.19	25.24	*68.57*
Union	16.19	16.19	1.90	22.86	18.10	**30.48**	*71.43*

Table 2: Performance of various LLMs in different environments, with the best performance in each environment highlighted in **bold**. "Human" signifies the average level of human performance.

LLMs are tasked with breaking down the user-provided information for populating the content of these parameters. Upon accomplishing this step, LLMs formally conclude the entire tool usage cycle, paving the way to receive the tool's output phase and initiate a new interaction. For each test case, we define a content filling score as follows:

$$s_{CF} = s_{PI} \cdot \prod_{i=1}^{N} \mathbb{I}(c_i = \hat{c}_i) \tag{3}$$

Here, N represents the total number of parameters required to be filled. c_i is the content filled by LLMs for the ith parameter, and $\hat{c}_i$ refers to the correct content for that parameter.

4 Experiments

4.1 Model Selection

To evaluate the robustness of widely-used LLMs with tool-use capabilities, we opt for testing four open-source models (i.e., ToolLLaMA-2-7B-v1 (Qin et al., 2023b), ToolLLaMA-2-7B-v2 (Qin et al., 2023b), NexusRaven-13B-v1 (team, 2023a), NexusRaven-13B-v2 (team, 2023b)) and two closed-source models (i.e., GPT-3.5-turbo[1], GPT-4 (OpenAI, 2023)).[2]

[1] https://platform.openai.com/docs/models/gpt-3-5

[2] The details of LLMs can be found in Appendix A.

Source	Models	F Statistic	P Value
Open-Source	**ToolLLaMA-2-7B-v1**	2.47	4.36×10^{-2}
	ToolLLaMA-2-7B-v2	3.28	1.10×10^{-2}
	NexusRaven-13B-v1	0.76	5.55×10^{-1}
	NexusRaven-13B-v2	6.01	9.13×10^{-5}
Closed-Source	**GPT-3.5-turbo**	6.76	2.33×10^{-5}
	GPT-4	5.31	3.19×10^{-4}
Human	–	*0.04*	*1.00*

Table 3: Welch's ANOVA for s_{CF} across the five enviroments for various LLMs. A p-value below 0.05 indicate significant differences in the data.

4.2 Main Results

As tool learning involves multiple turns of interaction between LLMs and the environment (Qin et al., 2023a; Ye et al., 2024), with intricate intermediate trajectories that cannot be easily compared, our emphasis lies on evaluating the robustness of various LLMs during their initial use of the tool and present the results in Table 2.[3] The resulting data reveals intriguing observations.

The robustness of current LLMs in tool learning presents considerable scope for enhancement. While human performance remains relatively stable across different environments,

[3] The results presented are averages across various scenarios, with specific outcomes for each scenario detailed in Appendix C.

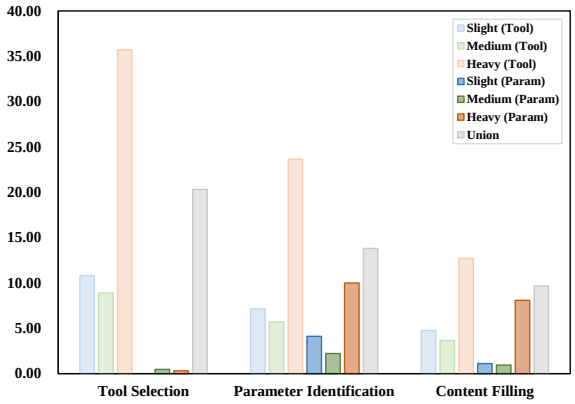

Figure 3: Absolute difference between the average performance of LLMs in various noisy environments and their average performance in Clean-level environment.

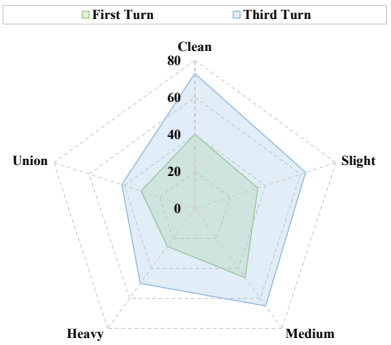

Figure 4: The performance of GPT-4 during the content filling phase in the first and third rounds of interaction.

Models	Tool Selection	Parameter Identification
GPT-3.5-turbo	33.72	33.85
GPT-4	29.17	22.83

Table 4: The percentage of error caused by noise correction at different stages in GPT family of models.

the performance of LLMs exhibits significant fluctuations. For instance, when transitioning from Clean-level environment to Union-level, human performance in tool selection only decreases by 2.86 points, whereas the average performance of all LLMs decreases by approximately 20.32 points. To gain a clearer understanding, we employ Welch's ANOVA (Bl, 1947) to analyze the significance of LLMs' performance during the content-filling stage across various environments. As illustrated in Table 3, our findings underscore the consistency of human performance and the noteworthy disparities in LLMs' performance across different environments. Consequently, enhancing the robustness of LLMs in tool learning is an area that requires significant attention.

Noise affecting tool names has a more pronounced impact on LLM performance than noise introduced to parameters. We compute the absolute difference in average LLMs performance for each type of noise added to tool names or parameters, relative to their performance in the Clean-level environment, respectively. The results depicted in Figure 3 show that tool name noise significantly affects LLMs' tool learning performance throughout the entire process. In contrast, noise in the parameters has minimal impact on the robustness of LLMs during the tool selection stage and exerts less influence on subsequent stages compared to tool name noise. Notably, LLMs exhibit greater robustness in the Union-level environment than in the Heavy (Tool) environment, underscoring the substantial impact of tool naming on model robustness.

Offering LLMs interactive examples enhances their tool learning performance, yet it does not bolster their robustness. As tool learning entails multiple turns of interaction between LLMs and external environments, we initially provide the first two turns of interactions for the test cases in each environment to evaluate LLMs' performance during the third turn of interactions. Upon comparing GPT-4's results in the first and third turns of interactions (Figure 4), it becomes evident that the provision of two turns of interaction examples leads to a consistent performance boost for GPT-4, resulting in an average performance improvement of 22.91 points across various environments. However, when examining the performance variation values, it is noteworthy that the standard deviation of its performance across environments increased from 8.14 in the first turn to 12.56 in the third turn. This observation suggests that while its performance improves, its robustness does not see a corresponding enhancement.

4.3 Why do GPT family of models NOT perform well in Slight-level environment?

A particularly intriguing finding is that, in contrast to other LLMs, the GPT family of models exhibits a lower performance in Slight-level environment compared to Medium-level, despite the limited validity of the information provided by the latter. Our thorough investigation into the model outputs has revealed that this phenomenon can be attributed to the inherent noise correction capability of the GPT family of models. For instance, when the

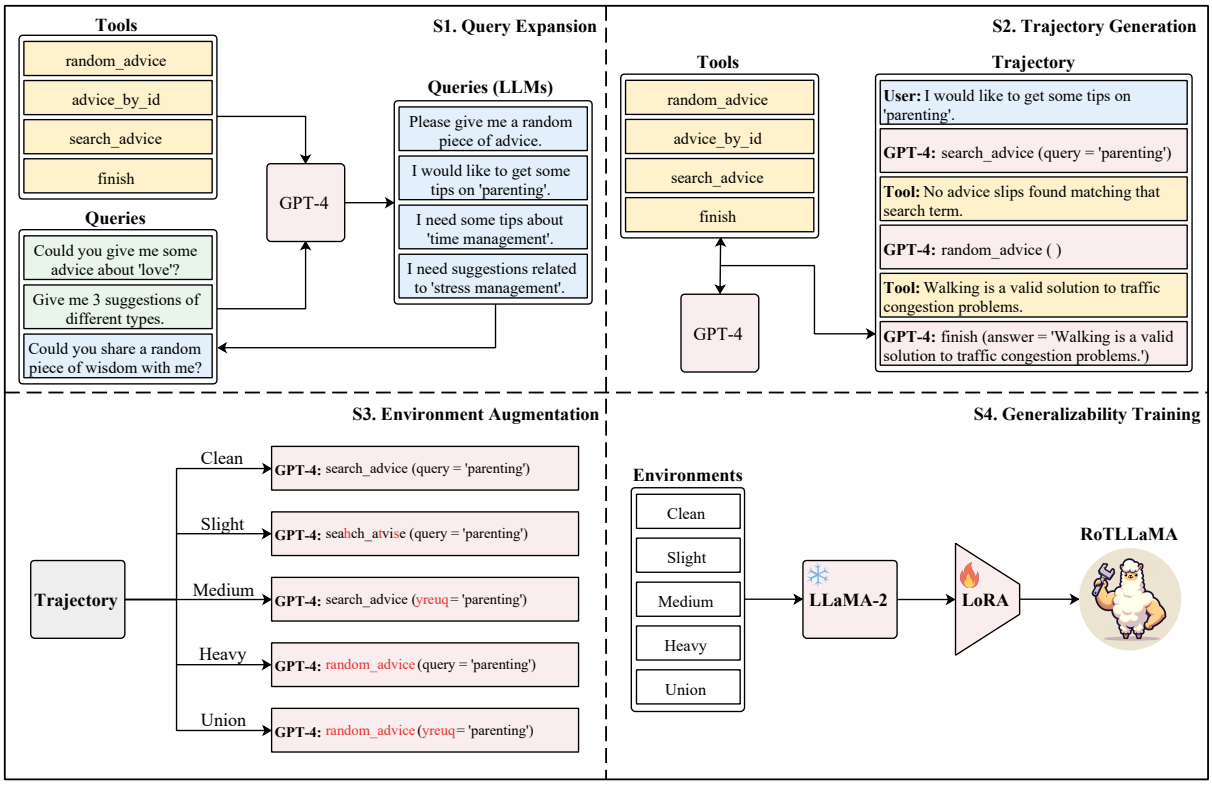

Figure 5: Illustration of RoTTuning. RoTTuning encompasses four phases, aiming at bolstering the robustness of LLMs in tool learning through increased environmental diversity.

GPT family of models selects the tool labeled as "predOict_aTge," it automatically corrects the noise within it and generates "predict_age" as the output, consequently leading to an error. [4]

Table 4 illustrates the proportions of total error attributed to noise correction for the tool selection and parameter identification phases of the GPT family of models within the Slight-level environment. Notably, these proportions are exceptionally high, exceeding one-third for GPT-3.5-turbo. Consequently, addressing the challenge of mitigating capability degradation stemming from the model's inherent characteristics remains a pressing research concern.

5 RoTTuning

It is evident that enhancing the robustness of LLMs in tool learning is imperative. To tackle this issue, we introduce RoTTuning, a novel approach aimed at bolstering the robustness of LLMs through increased environmental diversity.

5.1 Method

RoTTuning encompasses four phases: query expansion, trajectory generation, environment augmentation, and generalizability training (Figure 5).

Query Expansion To efficiently generate high-quality user queries on a large scale, we employ the self-instruct (Wang et al., 2023b) technique, drawing from the 105 existing user queries.[5] Specifically, we instruct GPT-4 to create seven fresh user queries within the context of a subset of tools, accompanied by three existing user queries and two model-generated queries. To ensure diversity in our dataset, we scrutinize the new data for redundancy in relation to each provided example and eliminate queries with Rouge-L values surpassing 0.55. This process yields a total of 4,077 new user queries.

Trajectory Generation Upon obtaining high-quality user queries, we employ GPT-4 to produce tool learning trajectories. To ensure the accuracy of the generated trajectories, we leverage the specifically designed function call feature of GPT-4. Simultaneously, we guide GPT-4 in generating the associated thought process by incorporating a system prompt.[6] Furthermore, we specify that GPT-4's tool usage is limited to a maximum of nine turns. By considering each turn of interaction as a distinct data point, this process results in a total of

[4]For more detailed examples, please refer to Appendix D.

[5]The specific prompt can be found in Appendix G.

[6]The specific prompt can be found in Appendix H.

Level	Clean	Slight	Medium	Heavy	Union
s_{TS}	76.19	72.38	70.48	65.24	63.81
s_{PI}	55.24	50.00	50.48	39.05	44.76
s_{CF}	42.86	36.19	34.29	28.10	28.57

Table 5: The score in different stages (%) of RoTLLaMA in various Environments.

12,247 pieces of training data.

Environment Augmentation To enhance the variety of environments, we modify the trajectories generated in the Clean-level environment to align with the characteristics of noisy environments. This strategy ensures data quality while addressing the challenges of working in noisy settings. To mitigate the potential drawbacks of data coupling, we introduce randomness by augmenting 3000 trajectories for each of the Slight-, Medium-, and Heavy-level environments, along with 1500 trajectories for Union-level environments. When combined with the data from the Clean-level environment, this approach yields a total of 22,747 trajectories, representing a diverse range of environmental conditions.

Generalizability Training Utilizing the diversity trajectories generated, we proceed with the fine-tuning of LLaMA-2-7B-base (Touvron et al., 2023b) and implement a position interpolation (Chen et al., 2023a) technique to extend its context length to 8096. Based on previous research indicating that fine-tuning with LoRA (Hu et al., 2022) achieves superior generalization compared to full parametric fine-tuning (Zeng et al., 2023), we opt for the LoRA fine-tuning approach. We conduct 5 epochs of training to derive the ultimate model, RoTLLaMA, which exhibits robust generalization across multiple environments.

5.2 Experimental Results

We carry out a series of experimental analyses with RoTLLaMA on RoTBench to verify its advantages when facing various noise environments.

Performance We analyze the performance of RoTLLaMA in various environments, and the results are presented in Table 5. The results reveal that RoTLLaMA's performance stability across different environments significantly surpasses that of GPT-4. Specifically, in the tool selection phase, the extreme performance difference is only 12.38, whereas GPT-4 demonstrates a much higher extreme difference of 21.90. Furthermore, in the parameter recognition and content filling phases, the extreme performance differences are 16.19 and 14.76, respectively, both of which are smaller than GPT-4's corresponding values of 20.95 and 20.95.

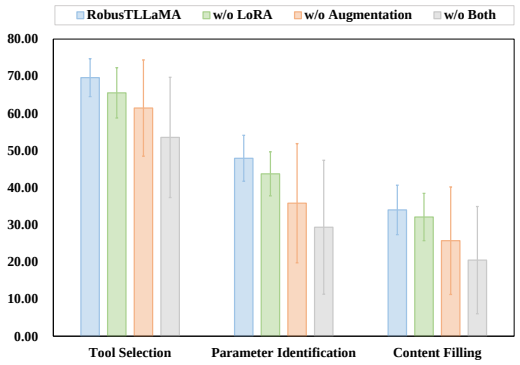

Figure 6: The means and standard deviations of our model's performance in the five environments.

Ablation Study To evaluate the effectiveness of various components within our approach, we conducted ablation studies on RoTLLaMA. As shown in Figure 6, when substituting full-parameter fine-tuning for LoRA fine-tuning (i.e., w/o LoRA), there is a slight decrease in model performance, and standard deviations across environments remain largely unchanged. This suggests that employing LoRA enhances model performance without significantly impacting its robustness. On the other hand, if we omit environment augmentation (i.e., w/o Augmentation), there is a notable decrease in both mean performance and a significant increase in standard deviation within each environment. This underscores the crucial role of environment augmentation in enhancing both model performance and robustness. Furthermore, exclusively utilizing full-parameter fine-tuning on the model (i.e., w/o Both) leads to a degradation of 16.10 points in model performance.

6 Conclusion

In this paper, we introduce RoTBench, a multi-level benchmark for evaluating the robustness of LLMs in tool learning. RoTBench contains five environments, each characterized by varying noise levels, shedding light on the pressing need to bolster the robustness of LLMs. Furthermore, we present RoTTuning, an innovative approach that significantly improves the robustness of LLMs in tool learning by increasing the diversity of environments during the training phase.

Limitations

While we introduce a multi-level benchmark for evaluating the robustness of LLMs in tool learning and a training method aimed at increasing environmental diversity, our work does have some limitations. On one hand, our primary focus is on assessing the robustness of LLMs in a single tool-use round, and we do not delve into whether LLMs are able to self-correct their behavior in response to environmental feedback. However, we analyze the performance of GPT-4 based on the interaction trajectories in the first two rounds and find that this does not enhance model robustness. On the other hand, While tool descriptions are undoubtedly crucial for understanding tools, our analysis centers on the noise present in tool names and parameters. This choice is driven by our discovery that LLMs' comprehension of tools primarily relies on tool and parameter names rather than a nuanced understanding of the meanings conveyed in tool documentation. Within this framework, evaluating LLMs through RoTBench can effectively measure their tolerance to noise in these additional details, thus propelling research endeavors aimed at improving LLMs' tool learning capabilities.

References

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosiute, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemí Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. 2022. Constitutional AI: harmlessness from AI feedback. *CoRR*, abs/2212.08073.

Welch Bl. 1947. The generalisation of student's problems when several different population variances are involved. *Biometrika*, 34(1-2):28–35.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023a. Extending context window of large language models via positional interpolation. *CoRR*, abs/2306.15595.

Xiuying Chen, Guodong Long, Chongyang Tao, Mingzhe Li, Xin Gao, Chengqi Zhang, and Xiangliang Zhang. 2023b. Improving the robustness of summarization systems with dual augmentation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 6846–6857. Association for Computational Linguistics.

Xuanting Chen, Junjie Ye, Can Zu, Nuo Xu, Rui Zheng, Minlong Peng, Jie Zhou, Tao Gui, Qi Zhang, and Xuanjing Huang. 2023c. How robust is GPT-3.5 to predecessors? A comprehensive study on language understanding tasks. *CoRR*, abs/2303.00293.

Zehui Chen, Weihua Du, Wenwei Zhang, Kuikun Liu, Jiangning Liu, Miao Zheng, Jingming Zhuo, Songyang Zhang, Dahua Lin, Kai Chen, and Feng Zhao. 2023d. T-eval: Evaluating the tool utilization capability step by step.

Guanting Dong, Tingfeng Hui, Zhuoma Gongque, Jinxu Zhao, Daichi Guo, Gang Zhao, Keqing He, and Weiran Xu. 2023. Demonsf: A multi-task demonstration-based generative framework for noisy slot filling task. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 10506–10518. Association for Computational Linguistics.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. *CoRR*, abs/2305.11554.

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

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and Lichao Sun. 2023. Metatool benchmark for large language models: Deciding whether to use tools and which to use. *CoRR*, abs/2310.03128.

Zuxin Liu, Zijian Guo, Zhepeng Cen, Huan Zhang, Yihang Yao, Hanjiang Hu, and Ding Zhao. 2023. Towards robust and safe reinforcement learning with benign off-policy data. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 21586–21610. PMLR.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ramakanth Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. 2023. Augmented language models: a survey. *CoRR*, abs/2302.07842.

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

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bowen Li, Ziwei Tang, Jing Yi, Yuzhang Zhu, Zhenning Dai, Lan Yan, Xin Cong, Yaxi Lu, Weilin Zhao, Yuxiang Huang, Junxi Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Zhiyuan Liu, and Maosong Sun. 2023a. Tool learning with foundation models. *CoRR*, abs/2304.08354.

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

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton-Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code llama: Open foundation models for code. *CoRR*, abs/2308.12950.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *CoRR*, abs/2302.04761.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, and Le Sun. 2023. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *CoRR*, abs/2306.05301.

Nexusflow.ai team. 2023a. Nexusraven: Surpassing the state-of-the-art in open-source function calling llms.

Nexusflow.ai team. 2023b. Nexusraven-v2: Surpassing gpt-4 for zero-shot function calling.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Jindong Wang, Xixu Hu, Wenxin Hou, Hao Chen, Runkai Zheng, Yidong Wang, Linyi Yang, Haojun Huang, Wei Ye, Xiubo Geng, Binxing Jiao, Yue Zhang, and Xing Xie. 2023a. On the robustness of chatgpt: An adversarial and out-of-distribution perspective. *CoRR*, abs/2302.12095.

Xiao Wang, Qin Liu, Tao Gui, Qi Zhang, Yicheng Zou, Xin Zhou, Jiacheng Ye, Yongxin Zhang, Rui Zheng, Zexiong Pang, Qinzhuo Wu, Zhengyan Li, Chong Zhang, Ruotian Ma, Zichu Fei, Ruijian Cai, Jun Zhao, Xingwu Hu, Zhiheng Yan, Yiding Tan, Yuan Hu, Qiyuan Bian, Zhihua Liu, Shan Qin, Bolin Zhu, Xiaoyu Xing, Jinlan Fu, Yue Zhang, Minlong Peng, Xiaoqing Zheng, Yaqian Zhou, Zhongyu Wei, Xipeng Qiu, and Xuanjing Huang. 2021. Textflint: Unified multilingual robustness evaluation toolkit for natural language processing. In *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL 2021 - System Demonstrations, Online, August 1-6, 2021*, pages 347–355. Association for Computational Linguistics.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023b. Self-instruct: Aligning language models with self-generated instructions.

In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13484–13508. Association for Computational Linguistics.

Sherry Yang, Ofir Nachum, Yilun Du, Jason Wei, Pieter Abbeel, and Dale Schuurmans. 2023. Foundation models for decision making: Problems, methods, and opportunities. *CoRR*, abs/2303.04129.

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.

Junjie Ye, Xuanting Chen, Nuo Xu, Can Zu, Zekai Shao, Shichun Liu, Yuhan Cui, Zeyang Zhou, Chao Gong, Yang Shen, Jie Zhou, Siming Chen, Tao Gui, Qi Zhang, and Xuanjing Huang. 2023. A comprehensive capability analysis of GPT-3 and GPT-3.5 series models. *CoRR*, abs/2303.10420.

Junjie Ye, Guanyu Li, Songyang Gao, Caishuang Huang, Yilong Wu, Sixian Li, Xiaoran Fan, Shihan Dou, Qi Zhang, Tao Gui, and Xuanjing Huang. 2024. Tooleyes: Fine-grained evaluation for tool learning capabilities of large language models in real-world scenarios. *CoRR*, abs/2401.00741.

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

Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Weirong Ye, Neil Zhenqiang Gong, Yue Zhang, and Xingxu Xie. 2023. Promptbench: Towards evaluating the robustness of large language models on adversarial prompts. *ArXiv*, abs/2306.04528.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2023. Toolqa: A dataset for LLM question answering with external tools. *CoRR*, abs/2306.13304.

A Details of LLMs

To evaluate the robustness of widely-used LLMs with tool-use capabilities, we opt for testing four open-source models and two closed-source models.

A.1 Open-Source LLMs

Among open-source LLMs, we have chosen four models that have undergone dedicated training for tool learning.

ToolLLaMA-2-7B-v1 ToolLLaMA-2-7B-v1, developed by Tsinghua University, is a tool-oriented LLM that harnesses the power of 126,000 data samples, including more than 16,000 APIs, through supervised fine-tuning on LLaMA-2-7B-base. This enables ToolLLaMA-2-7B-v1 to effectively utilize various tools to meet diverse user requirements.

ToolLLaMA-2-7B-v2 ToolLLaMA-2-7B-v2 has undergone fine-tuning from LLaMA-2-7B-base, by assimilating an expansive dataset comprising over 120,000 solution paths and annotated chains of thought. To the best of our knowledge, this model stands as the most extensively trained tool-oriented LLM, utilizing the largest dataset and the broadest spectrum of tools among all available options.

NexusRaven-13B-v1 NexusRaven-13B-v1 is a tool-oriented model that underwent fine-tuning based on CodeLLaMA-13B. Distinguishing itself from prior models, NexusRaven-13B-v1 employs code nesting to invoke tools, generating the entire inference path simultaneously instead of following a step-by-step approach.

NexusRaven-13B-v2 NexusRaven-13B-v2 enhances the performance of NexusRaven-13B-v1 by generating single, nested, and parallel function calls in various complex scenarios. Additionally, NexusRaven-13B-v2 can generate inference paths for the function calls it creates, thereby improving overall generalization.

A.2 Closed-Source LLMs

Among closed-source LLMs, we have opted for two of the most representative models from the GPT family.

GPT-3.5-turbo GPT-3.5-turbo stands out as the most potent and cost-efficient model within the GPT-3.5 series. Tailored for conversations, it excels in comprehending and generating natural language. Furthermore, it exhibits strong tool invocation capabilities.

GPT-4 GPT-4 represents OpenAI's most robust LLM, surpassing its predecessor in delivering safer and more beneficial responses. Additionally, GPT-4 offers formal support for multimodal inputs and has an expanded capability to address a broader spectrum of social requirements.

B Experimental Setup

Inference In accordance with Ye et al. (2024), we adopt the ReAct (Yao et al., 2023) format for inference, employing a consistent prompt template for both the ToolLLaMA-2-7B family of models and the GPT family of models. However, as NexusRaven-13B fmaily of models utilize nested functions for output, we adhere to the guidelines outlined on their official website, which necessitate the use of a distinct set of template.[7] Meanwhile, to evaluate human performance across environments with different noise levels, we enlist three university students. Each student receives identical tool documentation and task descriptions. Independently, they completes the questions and the average score derived from their responses served as the human performance benchmark.

Evaluation We score the performance of LLMs and Human using the evaluation methods defined in Section 3.3. In this system, each data point is scored as 0 or 1 at each stage. This is because, in the context of tool learning, tool calls either succeed or fail, and even small errors can cause the entire process to fail. In particular, In the tool selection phase, an error in tool selection can lead to overall failure, independent of parameter accuracy. In the parameter identification phase, missing necessary parameters or wrong parameter selection can lead to failure. In the content filling phase, incorrect content input can lead to undesirable tool execution results.

C Results in Different Scenarios

We show the performance of each model in different scenarios and document the results from Table 6 to Table 12. From the results, we have the following observations.

The variance in average performance of LLMs across various study scenarios can be attributed to the relevance of specific features of available tools to each scenario. For instance, in both application operations and personal life scenarios, LLMs may err due to the strict sequential order in which tools are called (e.g., obtaining parameter values for "list_properties" necessitates prior execution of "search_locations").

It's notable that the model's perception of environmental complexity may diverge from human intentions. For instance, in information retrieval scenarios, LLMs exhibit inferior average performance in the slight-level environment compared to the medium-level and heavy-level environments, primarily due to limitations in noise correction capabilities (Section 4.3).

Regarding the model itself, variations in training methods and data can lead to unexpected performances in certain scenarios. For instance, ToolLLaMA-7B-v1 demonstrates a performance discrepancy between the clean-level and union-level environments in the application manipulation scenario, scoring 20 and 40, respectively. This disparity arises from its ability to perform better when only two tools are available alongside "ask_to_user" and "finish," whereas GPT4 consistently prompts for API keys even when unnecessary.

D Examples for Noise Correction

In Table 13, we present instances of noise correction observed during the tool selection and parameter identification phases of the GPT family of models.

E Further Studies about RoTTuning

We conduct additional comparative analysis to further validate the effectiveness of RoTTuning in improving the stability of LLMs in noisy environments.

The Number of Tool Hallucinations We compare the number of tool hallucinations for each LLM in all environments and find that our model has significantly fewer hallucinations compared to the GPT family of models (Table 14). This demonstrates the effectiveness of our method in mitigating interference from various sources of noise while accurately acquiring environmental information. It's worth noting that the NexusRaven family of models, which relies on CodeLLaMA (Rozière et al., 2023) as a base, also exhibits low tool hallucinations, suggesting that utilizing code-based approaches for tool learning is a viable direction.

Performance of RoTToolLLaMA To confirm the robustness of our method for enhancing established tool-oriented LLMs, we proceed to fine-tune ToolLLaMA-2-7B using our generated trajectories and obtain RoTToolLLaMA. The corresponding results presented in Table 15 illustrate that our method's fine-tuning significantly

[7] The specific prompt can be found in Appendix F.

Models	Open-Source LLMs				Closed-Source LLMs	
	ToolLLaMA-2-7B-v1	ToolLLaMA-2-7B-v2	NexusRaven-13B-v1	NexusRaven-13B-v2	GPT-3.5-turbo	GPT-4
Tool Selection						
Clean	60.00	73.33	20.00	53.33	**86.67**	**86.67**
Slight	46.67	60.00	30.00	56.67	73.33	**83.33**
Medium	36.67	50.00	30.00	70.00	73.33	**90.00**
Heavy	36.67	43.33	20.00	40.00	53.33	**70.00**
Union	40.00	26.67	26.67	46.67	**60.00**	46.67
Parameter Identification						
Clean	60.00	60.00	6.67	40.00	60.00	**73.33**
Slight	40.00	46.67	13.33	40.00	36.67	**53.33**
Medium	33.33	40.00	10.00	50.00	40.00	**63.33**
Heavy	36.67	30.00	6.67	13.33	23.33	**40.00**
Union	**40.00**	13.33	13.33	**40.00**	26.67	33.33
Content Filling						
Clean	26.67	26.67	6.67	33.33	60.00	**73.33**
Slight	16.67	13.33	10.00	33.33	36.67	**53.33**
Medium	13.33	10.00	6.67	36.67	40.00	**63.33**
Heavy	16.67	13.33	3.33	13.33	20.00	**36.67**
Union	20.00	0.00	6.67	**33.33**	26.67	**33.33**

Table 6: Performance of various LLMs in the text generation scenario, with the best performance in each environment highlighted in **bold**.

Models	Open-Source LLMs				Closed-Source LLMs	
	ToolLLaMA-2-7B-v1	ToolLLaMA-2-7B-v2	NexusRaven-13B-v1	NexusRaven-13B-v2	GPT-3.5-turbo	GPT-4
Tool Selection						
Clean	80.00	80.00	80.00	80.00	**86.67**	**86.67**
Slight	63.33	80.00	70.00	**83.33**	63.33	73.33
Medium	60.00	73.33	66.67	80.00	83.33	**93.33**
Heavy	46.67	56.67	50.00	**60.00**	56.67	56.67
Union	40.00	53.33	46.67	60.00	60.00	**86.67**
Parameter Identification						
Clean	60.00	40.00	26.67	33.33	40.00	**66.67**
Slight	50.00	43.33	26.67	36.67	26.67	**60.00**
Medium	50.00	46.67	16.67	30.00	40.00	**66.67**
Heavy	33.33	**40.00**	10.00	26.67	13.33	26.67
Union	20.00	46.67	6.67	20.00	13.33	**60.00**
Content Filling						
Clean	46.67	33.33	0.00	20.00	26.67	**53.33**
Slight	33.33	40.00	0.00	23.33	16.67	**53.33**
Medium	30.00	40.00	0.00	16.67	30.00	**56.67**
Heavy	13.33	20.00	0.00	**23.33**	10.00	20.00
Union	13.33	40.00	0.00	13.33	6.67	**46.67**

Table 7: Performance of various LLMs in the data understanding scenario, with the best performance in each environment highlighted in **bold**.

Models	Open-Source LLMs				Closed-Source LLMs	
	ToolLLaMA-2-7B-v1	ToolLLaMA-2-7B-v2	NexusRaven-13B-v1	NexusRaven-13B-v2	GPT-3.5-turbo	GPT-4
Tool Selection						
Clean	66.67	60.00	40.00	86.67	73.33	**93.33**
Slight	60.00	50.00	36.67	**80.00**	60.00	**80.00**
Medium	63.33	46.67	43.33	76.67	73.33	**90.00**
Heavy	46.67	36.67	36.67	**73.33**	46.67	56.67
Union	53.33	46.67	26.67	66.67	60.00	**73.33**
Parameter Identification						
Clean	60.00	46.67	6.67	**73.33**	53.33	53.33
Slight	53.33	43.33	6.67	**66.67**	36.67	40.00
Medium	46.67	40.00	10.00	**60.00**	53.33	53.33
Heavy	30.00	30.00	6.67	**43.33**	16.67	23.33
Union	**40.00**	33.33	6.67	**40.00**	33.33	**40.00**
Content Filling						
Clean	**33.33**	20.00	0.00	**33.33**	20.00	**33.33**
Slight	**30.00**	20.00	0.00	**30.00**	20.00	**30.00**
Medium	16.67	10.00	0.00	26.67	30.00	**40.00**
Heavy	6.67	20.00	0.00	**26.67**	10.00	20.00
Union	13.33	13.33	0.00	6.67	26.67	**40.00**

Table 8: Performance of various LLMs in the real-time search scenario, with the best performance in each environment highlighted in **bold**.

Models	Open-Source LLMs				Closed-Source LLMs	
	ToolLLaMA-2-7B-v1	ToolLLaMA-2-7B-v2	NexusRaven-13B-v1	NexusRaven-13B-v2	GPT-3.5-turbo	GPT-4
Tool Selection						
Clean	**86.67**	73.33	73.33	66.67	80.00	73.33
Slight	**80.00**	**80.00**	73.33	70.00	66.67	73.33
Medium	83.33	80.00	73.33	66.67	80.00	**86.67**
Heavy	60.00	50.00	**70.00**	66.67	**70.00**	63.33
Union	**80.00**	53.33	73.33	66.67	66.67	53.33
Parameter Identification						
Clean	40.00	40.00	6.67	**60.00**	53.33	46.67
Slight	56.67	46.67	10.00	**60.00**	36.67	46.67
Medium	53.33	46.67	6.67	53.33	**56.67**	46.67
Heavy	36.67	20.00	13.33	**50.00**	40.00	43.33
Union	**73.33**	40.00	13.33	53.33	40.00	33.33
Content Filling						
Clean	**20.00**	13.33	0.00	**20.00**	**20.00**	**20.00**
Slight	**33.33**	20.00	0.00	20.00	16.67	13.33
Medium	**40.00**	26.67	0.00	16.67	26.67	23.33
Heavy	20.00	6.67	0.00	**26.67**	16.67	13.33
Union	**40.00**	26.67	0.00	13.33	20.00	6.67

Table 9: Performance of various LLMs in the application manipulation scenatio, with the best performance in each environment highlighted in **bold**.

Models	Open-Source LLMs				Closed-Source LLMs	
	ToolLLaMA-2-7B-v1	ToolLLaMA-2-7B-v2	NexusRaven-13B-v1	NexusRaven-13B-v2	GPT-3.5-turbo	GPT-4
Tool Selection						
Clean	53.33	60.00	40.00	66.67	**73.33**	66.67
Slight	46.67	63.33	43.33	**73.33**	50.00	70.00
Medium	50.00	53.33	50.00	63.33	60.00	**73.33**
Heavy	23.33	40.00	43.33	**50.00**	**50.00**	**50.00**
Union	40.00	**53.33**	**53.33**	46.67	40.00	46.67
Parameter Identification						
Clean	26.67	40.00	13.33	**53.33**	26.67	40.00
Slight	30.00	26.67	13.33	**53.33**	10.00	26.67
Medium	26.67	26.67	13.33	36.67	**40.00**	**40.00**
Heavy	6.67	16.67	3.33	**30.00**	16.67	26.67
Union	26.67	20.00	6.67	26.67	26.67	**40.00**
Content Filling						
Clean	20.00	26.67	0.00	**40.00**	13.33	33.33
Slight	16.67	20.00	0.00	**43.33**	10.00	23.33
Medium	13.33	23.33	0.00	33.33	30.00	**40.00**
Heavy	6.67	10.00	0.00	**26.67**	10.00	**26.67**
Union	6.67	20.00	0.00	**26.67**	6.67	**26.67**

Table 10: Performance of various LLMs in the personal life scenario, with the best performance in each environment highlighted in **bold**.

Models	Open-Source LLMs				Closed-Source LLMs	
	ToolLLaMA-2-7B-v1	ToolLLaMA-2-7B-v2	NexusRaven-13B-v1	NexusRaven-13B-v2	GPT-3.5-turbo	GPT-4
Tool Selection						
Clean	60.00	**80.00**	73.33	73.33	46.67	73.33
Slight	50.00	63.33	66.67	**83.33**	43.33	73.33
Medium	43.33	56.67	63.33	**76.67**	53.33	73.33
Heavy	50.00	53.33	53.33	**80.00**	53.33	56.67
Union	26.67	33.33	46.67	**53.33**	40.00	40.00
Parameter Identification						
Clean	26.67	33.33	26.67	**53.33**	40.00	40.00
Slight	16.67	20.00	23.33	**60.00**	30.00	36.67
Medium	16.67	16.67	30.00	**60.00**	43.33	50.00
Heavy	23.33	26.67	16.67	**56.67**	33.33	36.67
Union	20.00	13.33	20.00	**40.00**	**40.00**	**40.00**
Content Filling						
Clean	20.00	26.67	0.00	**46.67**	26.67	33.33
Slight	13.33	16.67	6.67	**56.67**	23.33	30.00
Medium	16.67	13.33	3.33	**53.33**	33.33	46.67
Heavy	23.33	16.67	3.33	**53.33**	26.67	30.00
Union	13.33	6.67	0.00	**33.33**	**33.33**	**33.33**

Table 11: Performance of various LLMs in the information retrieval scenario, with the best performance in each environment highlighted in **bold**.

Models	Open-Source LLMs				Closed-Source LLMs	
	ToolLLaMA-2-7B-v1	ToolLLaMA-2-7B-v2	NexusRaven-13B-v1	NexusRaven-13B-v2	GPT-3.5-turbo	GPT-4
Tool Selection						
Clean	46.67	53.33	53.33	**73.33**	66.67	66.67
Slight	43.33	50.00	43.33	**73.33**	43.33	**73.33**
Medium	46.67	43.33	40.00	66.67	50.00	**70.00**
Heavy	26.67	36.67	36.67	**53.33**	50.00	**53.33**
Union	20.00	26.67	26.67	**46.67**	33.33	**46.67**
Parameter Identification						
Clean	33.33	33.33	20.00	**66.67**	60.00	40.00
Slight	26.67	40.00	23.33	**66.67**	20.00	46.67
Medium	26.67	23.33	16.67	**56.67**	36.67	50.00
Heavy	16.67	16.67	13.33	**33.33**	26.67	23.33
Union	13.33	13.33	13.33	**33.33**	13.33	26.67
Content Filling						
Clean	33.33	33.33	6.67	**60.00**	46.67	33.33
Slight	26.67	36.67	6.67	**60.00**	16.67	46.67
Medium	26.67	23.33	3.33	**46.67**	23.33	**46.67**
Heavy	13.33	16.67	0.00	**33.33**	20.00	23.33
Union	6.67	6.67	6.67	**26.67**	6.67	**26.67**

Table 12: Performance of various LLMs in the financial transactions scenario, with the best performance in each environment highlighted in **bold**.

enhances the model's tool learning capability across all stages, while also bolstering its overall robustness. For instance, across the three stages, our method demonstrates performance extremes of 12.33/13.33/9.53 in various environments, compared to ToolLLaMA-2-7B-v2's 26.67/16.67/10.95. This further underscores the efficacy of our proposed approach.

F Prompt Template for Inference

In the context of inference, both the ToolLLaMA-2-7B family of models and the GPT family of models utilize the same prompt (See Table 16), whereas NexusRaven-13B-v1 and NexusRaven-13B-v2 employ distinct prompts (See Table 17 and Table 18).

G Prompt Template for Query Expansion

We use GPT-4 for query expansion based on prompt in Table 19.

H Prompt Template for Trajectory Generation

We use GPT-4 for trajectory generation based on prompt in Table 20.

Models	Stage	Query	Noisy Part	Model Output
GPT-3.5-turbo	**Tool Selection**	I have a list of names: Maria, Juan, and Carlos. Can you predict their ages?	Tool: predOict_aTge Description: Predicts the ages of one or more people given their names. Parameters: ...	Tool: predict_age
GPT-3.5-turbo	**Parameter Identification**	I want to know what will be the output if we run these commands sequentially in bash: 'cd /home/user/documents', 'ls -a.'	Tool: execute_bash_code Description: ... Parameters: Nommands (Required) Param Description: The command string to be executed.	Parameters: commands
GPT-4	**Tool Selection**	Is there any social event available which requires high accessibility and is free of cost?	Tool: get_activty_by_ye Description: Find a random activity with a given type. Parameters: ...	Tool: get_activity_by_type
GPT-4	**Parameter Identification**	Get me quotes for symbols AAPL, MSFT, and GOOGL from US.	Tool: get_quotes Description: ... Parameters: ymbols (Required) Param Description: The value of symbol field returned in auto-complete endpoint. Separated by comma for multiple entities.	Parameters: symbols

Table 13: Examples for noise correction of GPT family of models.

ToolLLaMA-2-		NexusRaven-		GPT-		RoTLLaMA
7B-v1	**7B-v2**	**13B-v1**	**13B-v2**	**3.5-turbo**	**4**	
53	65	6	0	50	23	3

Table 14: The number of tool hallucinations for each LLM in all environments.

Level	Clean	Slight	Medium	Heavy	Union
s_{TS}	69.52	69.05	70.95	64.76	56.19
s_{PI}	52.38	45.24	50.95	40.95	39.05
s_{CF}	38.10	32.38	34.76	31.43	28.57

Table 15: The score in different stages (%) of RoTToolLLaMA in various Environments.

System

You are an expert in using tools to handle real-time queries from users.
First I will give you the task description, and your task start.
At each step, your task is to give your thought to analyze the current state, decide the next step, with a function call to actually execute your step.
After the call, you will get the call result, and you are now in a new state.
Then you will analyze your status now, then decide what to do next...
After many (Thought-call) pairs, you finally perform the task, then you can give your final answer.

Desired format:
Thought: ⟨ The thought⟩
Action: ⟨ The tool you decide to use⟩
Action Input: ⟨ The parameters for the tool⟩

Remember:
1. You should ALWAYS think about what to do, but all the thought is short, at most in 3 sentences.
2. The action to take should be one of the given tools below.
3. The "Action Input" needs to provide a dict similar to {parameter_1: value_1, parameter_2: value_2} to call action.
4. Always use the "finish" tool upon task completion. The final answer should be comprehensive enough for the user. If the task is unmanageable, use the "finish" tool and respond with "I cannot handle the task."

Task description: You should use tools to help handle the real time user queries. Specifically, you have access of the following tools:
{Tool Document}

Let's Begin!

User

{Query}
Begin!

Table 16: The prompt used for ToolLLaMA-2-7B family of models and GPT family of models, where "{Tool Document}" represents the tool documentation given to LLMs and "{Query}" represents the query given by the user.

User

{Tool Document}

User Query: Question: {Query}

Please pick a function from the above options that best answers the user query and fill in the appropriate arguments.

Table 17: The prompt used for NexusRaven-13B-v1, where "{Tool Document}" represents the tool documentation given to LLMs and "{Query}" represents the query given by the user.

User

{Tool Document}

User Query: {Query}

Table 18: The prompt used for NexusRaven-13B-v2, where "{Tool Document}" represents the tool documentation given to LLMs and "{Query}" represents the query given by the user.

System

As an expert, your assignment is to utilize the comprehensive documentation of various tools to develop a series of problem scenarios that these tools can resolve. Ideally, each scenario should necessitate the sequential use of multiple tools for its resolution.

Remember:
1. The tools employed to address a problem should be a subset of the tools detailed in the provided documentation; ideally, each problem should require the use of more than one tool.
2. The parameter values needed by each tool can either be directly extracted from the query or obtained by invoking the specified other tool.
3. The problem scenario should be expressed in a way that is understandable to humans, while also showcasing the diverse functions of the provided tools and their interrelationships.

Here is the documentation of various tools: {Tool Document}

User

Please generate 12 diverse queries according to the documentation.

Examples:
{Examples}

Table 19: The prompt for query expansion, where "{Tool Document}" represents the tool documentation given to LLMs and "{Examples}" represents the examples for LLMs.

System

You are an expert in using tools to handle real-time queries from users.
At each step, your task is to give your thought to analyze the current state, decide the next step, with a function call to actually execute your step.
After the call, you will get the call result, and you are now in a new state.
Then you will analyze your status now, then decide what to do next...
After a series of these thought-action pairs, you will complete the task and provide the final answer.

Remember:
1. You must ALWAYS select a specific function to execute your idea at each step.
2. Before calling any function, you should ALWAYS give your thought, but limit it to a maximum of three sentences.
3. ALWAYS use the "finish" tool upon task completion. The final answer should be comprehensive enough for the user. If the task is unmanageable, use the "finish" tool and respond with "I cannot handle the task".

Let's begin!

User

{Query}
Begin!

Table 20: The prompt for trajectory generation, where "{Query}" represents the query given by the user.