What Affects the Stability of Tool Learning? An Empirical Study on the Robustness of Tool Learning Frameworks

Anonymous ACL submission

Abstract

Tool learning methods have enhanced the ability of large language models (LLMs) to interact with real-world applications. Many existing works fine-tune LLMs or design prompts to enable LLMs to select appropriate tools and correctly invoke them to meet user requirements. However, it is observed in previous works that the performance of tool learning varies from tasks, datasets, training settings, and algorithms. Without understanding the impact of these factors, it can lead to inconsistent results, inefficient model deployment, and suboptimal tool utilization, ultimately hindering the practical integration and scalability of LLMs in real-world scenarios. Therefore, in this paper, we explore the impact of both internal and external factors on the performance of tool learning frameworks. Through extensive experiments on two benchmark datasets, we find several insightful conclusions for future work, including the observation that LLMs can benefit significantly from increased trial and exploration. We believe our empirical study provides a new perspective for future tool learning research.

1 Introduction

Tool learning aims to augment LLMs with external tools, teaching them how to select appropriate tools, generate correct parameters and ultimately parse execution results to produce accurate responses (Qin et al., 2023b; Li et al., 2023a; Schick et al., 2023). By learning to use various tools, LLMs can better assist users in completing practical tasks, such as planning itineraries (Xie et al., 2024) and controlling physical robots (Wang et al., 2023a) and accessing the Web (Qin et al., 2023a). This capability is crucial for enhancing the interaction between LLMs and real-world applications, enabling them to provide more comprehensive and useful assistance (Lu et al., 2023; Liu et al., 2023; Tian et al., 2024).

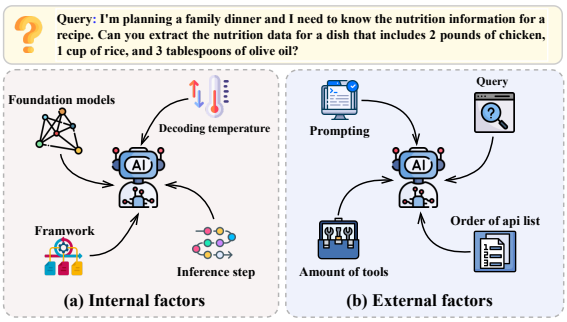

Figure 1: Illustration of various factors that may affect the robustness of tool learning methods.

In tool learning tasks, most of the previous work focuses on improving the performance of LLMs in successfully solving complex tasks, including designing chain-of-thought framework (Lu et al., 2023), employing multi-agent algorithms (Shi et al., 2024b; Qiao et al., 2024) or tuning models on specific datasets (Tang et al., 2023a), Numerous empirical studies are also proposed to evaluate the tool-use capability of LLMs, such as when to use, how to use, and which tool to use (Xu et al., 2023; Huang et al., 2023). Despite their progress, we find that stability, a crucial dimension to reflect the performance variation of LLMs under volatile scenarios (Li et al., 2023b; Gu et al., 2022), is less investigated. In real-world applications, various factors can affect the performance of tool learning models, and sometimes even produce different responses to identical user queries, *a.k.a.*, instability. For example, Ye et al. (2024b) show that even simple perturbations can cause models to use entirely incorrect tools or generate incorrect tool-calling parameters. These seemingly unrelated perturbations can lead to the failure of the task. Therefore, comprehensively exploring the factors related to the stability issue and quantitatively analyzing their impact becomes necessary for practical scenarios.

In this work, we provide the first empirical study on systematically analyzing the stability of tool-use models. To achieve this, we first categorize the diverse factors into two categories: **internal** and **external** factors.

The **internal** factors indicate uncertainties during the development of tool-use models from the developers' perspective. As shown in Figure 1, we consider the decoding temperature, the maximum inference steps, and the selection of different foundation models. Given the numerous works that guide LLMs to automatically use external tools (Yao et al., 2023; Yang et al., 2023b; Qin et al., 2023c), we also analyze the impact of different tool-use frameworks on the model's performance. Exploring these internal factors will help enhance the performance of the framework during development. Different from the internal factors, **external** factors primarily involve diverse prompt engineering when interacting with established tool-use models, which are beyond the control of developers once a tool-use model is deployed. Specifically, these factors includes different styles of user queries, customized system prompts for tool-use models, and the candidate toolset used to solve a query. For a holistic investigation, we change the candidate toolset by reordering it or expanding its scale, respectively. Investigating these external factors will help framework developers understand the stability in user-facing scenarios, thereby improving the overall user experience.

To quantitatively validate the impact of the aforementioned internal and external factors on the tool learning process, we conduct extensive experiments on the most commonly used ToolBench dataset. We employ several commonly used metrics to measure the performance under multiple perspectives and derive a series of interesting findings. We highlight the following:

- Existing tool-use workflow exhibits obvious instability towards various internal and external factors. Even the state-of-the-art methods still exhibit instability with inessential perturbations.

- Among the internal factors, the proper hyper-parameter settings may boost the LLMs to generate diverse solutions. However, it also leads to instability.

- Among the external factors, the LLMs are sensitive to the change of candidate toolset (*i.e.,* order or scale) and the system prompts.

- The advanced tool selection algorithms (*i.e.,* tree-based search) can improve the accuracy, but they may suffer from accumulated hallucination with less stability, as well as substantial inference costs.

2 Related work

Tool learning with LLMs. Tool learning aims to augment LLMs with real-world tools, extending their utility and empowering them as agents to automatically solve practical tasks (Qin et al., 2023b; Tang et al., 2023a; Shi et al., 2023; Gao et al., 2024). Pioneering work like Toolformer (Schick et al., 2023) and ToolkenGPT (Hao et al., 2023) teaches LLMs to utilize tools by training on specific tool-use datasets (Patil et al., 2023; Wang et al., 2024). Recent work leverages the inherent in-context learning capability of LLMs to use various tools, where the demonstration and usage are taken as the prompt (Yang et al., 2023b; Shi et al., 2024a; Guo et al., 2024). Despite the progress of recent tool-use models in successfully solving complex tasks, their stability is less investigated. In this work, we provide a comprehensive empirical study on the stability of tool-use models across diverse scenarios.

Evaluation of tool-use LLMs. In tool learning tasks, previous work primarily evaluates the success rate of LLMs in completing tasks, such as Success Rate (Yang et al., 2023a; Song et al., 2023) and Win Rate (Qin et al., 2023c). Recently, the ToolSword (Ye et al., 2024a) has also proposed to unveil safely-related issues of LLMs during the tool learning process. However, stability, a crucial dimension related to practical applications (Wang et al., 2023b), has been less investigated. Although some work, like RotBench (Ye et al., 2024b), proposes evaluating the robustness of tool-use LLMs, they only consider the different types of noise injected into original candidate toolsets. To the best of our knowledge, a thorough stability evaluation of tool-use LLMs remains under-explored. In our work, we fill this gap by providing a systematic evaluation of the stability of tool-use models, quantitatively their drawbacks under different scenarios.

# Dataset	# Amount	# Category	# APIs	# Avg. APIs
I1-inst.	200	36	995	5.34
I1-tool	200	33	548	4.79

Table 1: Statistics of the experimental datasets, including the count of task, tool category and APIs. The *Avg. APIs* indicates the average number of candidate toolset per task.

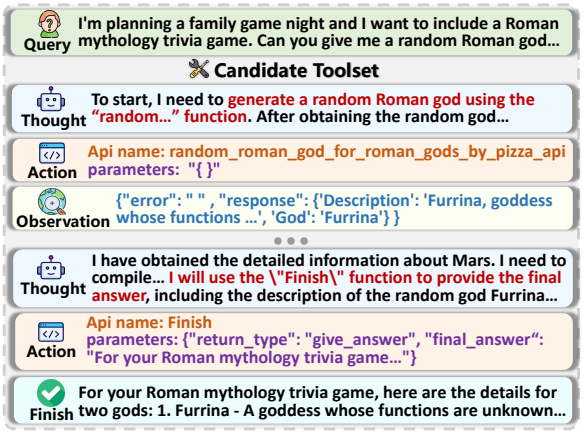

Figure 2: The tool-use framework in our empirical study. The LLM is guided to iteratively decide which tool to use (*Thought*, execute the selected tool (*Action*), and incorporate the intermediate (*Observation*) for the next iteration prediction.

3 Experimental Settings

3.1 Dataset

We conduct experiments on the subset of widely-used ToolBench (Qin et al., 2023c) benchmark, including *I1-instruction* and *I1-tools*. Each dataset contains 200 tasks involving various real-world applications, which evaluates the tool-use model under practical scenarios. The detailed statistics can be found in Table 1.

The original ToolBench only provides a task-solving trajectory of GPT-3.5 as an evaluation reference, which includes both ground truth and irrelevant tools. However, commonly used evaluation metrics require computing the overlap between model-selected tools and the ground truth. Therefore, we repurpose ToolBench to support our evaluation. For each task, we extract the tools used in the original solution. Next, we invite three well-educated experts with relevant research backgrounds to manually select the correct tools for solving the task. Several strategies are employed to ensure the quality of this process, which can be found in Appendix A.2.

3.2 Evaluation Metrics

Following previous work (Ye et al., 2024b; Song et al., 2023), we use the *Success Rate* and *T-test* as evaluation metrics. We also consider the *Give Up Rate*, *Invalid Selection Rate* as metrics for a comprehensive evaluation.

Success Rate (Success%). This metric intuitively evaluates the capability of tool-use LLMs in correctly selecting tools and generating corresponding arguments for execution. It calculates the proportion of tasks that the model can complete successfully within limited inference steps. The success rate is 1 if and only if all the required tools are used to solve a task.

T-test. To analyze the stability of tool-use LLM towards diverse factors, we use a two-tailed paired t-test (Student, 1908) following previous work (Ye et al., 2024b). This metric calculates the statistical significance of the model's performance difference between *vanilla* and *changed* experimental conditions. The significance level α is set to 0.05. Results are marked with ▲ if they are statistically significance are observed; otherwise, they are marked with ▼.

Invalid Selection Rate (Invalid%). We use the Invalid Selection Rate to compute the percentage of instances where the LLM selecting non-existent tool, *i.e.,* generating incorrect tool names. It reflects the ability of the model in tool selection, a crucial phase in the overall tool-use workflow, especially when the candidate toolset is large-scale.

Give Up Rate (Give up%). This metric computes the percentage of tasks that LLMs give up answering after trial and error. In practical scenarios, the model may fail to provide a correct solution for a complex task due to their limited ability. Therefore, it is crucial to build a confident model that is aware of its limitations, referred to as its capability boundary (Ren et al., 2023), allowing it to adaptively and faithfully inform the user instead of providing incorrect answers.

3.3 Tool Learning Framework

For a fair evaluation, we mainly employ the most commonly used ReAct framework (Yao et al., 2023), enabling LLMs to interact with tools in our experiment. In the ReAct framework, the LLM is guided to iteratively perform `Thought`, `Action`, and `Observation` steps. As shown in Figure 3, the `Thought` is to generate tool-use planning in the nature language while the `Action`

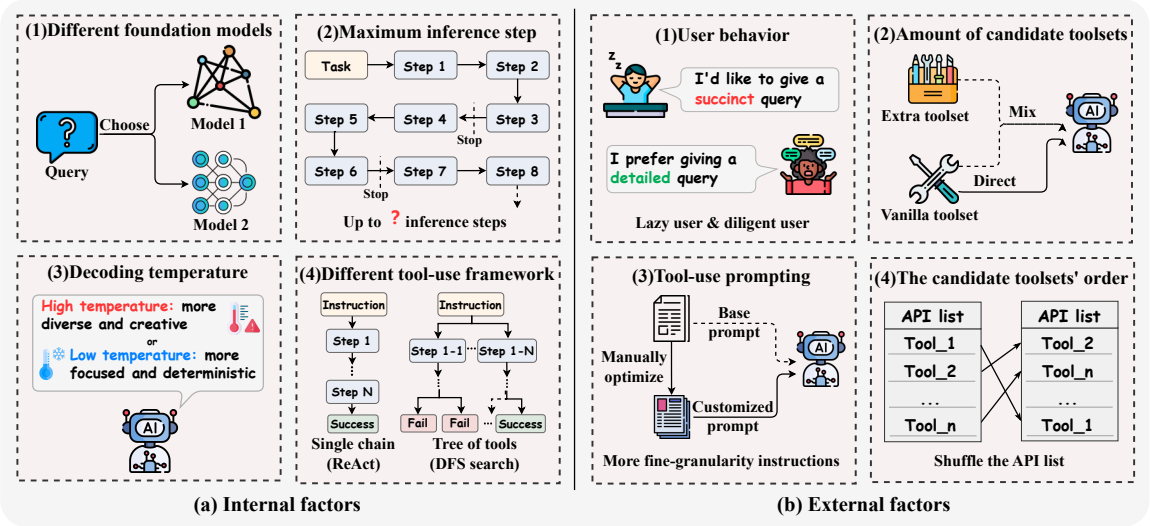

Figure 3: The overall framework of our work, which benchmarks tool-use models under various scenarios to investigate the *internal* and *external* factors that potentially affect their stability.

is to select an appropriate tool and formulate corresponding parameters. The `Observation` step is to incorporate the execution results of tools in the current context. For a comprehensive comparison, we also employ the ToolLLM (Qin et al., 2023c) framework, which augments LLMs with the Depth First Search-based Decision Tree (DFSDT) to select relevant tools in solving a task (§ 4.3).

3.4 Implementation Details

For the closed-source models, *e.g.,* GPT-3.5, we mainly enable them to utilize tools through OpenAI's function-call format[1]. For the open-source models, we use the prompt from Qin et al.. We also analyse the impact of different tool-use prompts in § 4.3. All the prompts in our work can be found in Appendix A.4.

4 Analysis of Internal Factors

We first investigate the influence of internal factors, which indicate the uncertainties in developing a tool learning framework, such as the selection of foundation LLMs and decoding temperature.

4.1 Impact of Foundation LLMs

The foundation LLM is the main component in the overall tool learning framework, which takes the user query as input and automatically executes external tools to generate an answer as a response. We comprehensively evaluate 9 off-the-shelf LLMs, including both close-source model, *i.e., GPT-3.5* and *GPT-4*, and open-source models such as *Mistral* (Jiang et al., 2023). For deterministic generation, the decoding temperature is set to 1 and 0.5 for closed-source and open-source models, respectively, following previous work (Zhuang et al., 2023; Ruan et al., 2024). More details about these models can be found in Appendix A.

Model	Success%↑	Give up%	Invalid%↓
gpt-3.5-turbo-16k	54.00%	29.50%	1.48%
gpt-3.5-turbo-0125	55.50%	36.50%	1.45%
gpt-3.5-turbo-1106	48.00%	50.50%	0.88%
gpt-4o	58.00%	38.00%	0.54%
deepseek-chat	40.50%	34.00%	0.56%
llama-3-70b	8.00%	4.50%	42.16%
llama-3-8b	3.50%	2.00%	28.56%
mixtral-8x7b-inst.	12.00%	14.00%	41.66%
mixtral-8x22b	25.00%	19.00%	10.76%

Table 2: The results with different foundation models on I1-instruction dataset of ToolBench (Qin et al., 2023c).

As shown in Table 2, we find that ***closed-source models substantially outperform open-source models*** in Success Rate while achieving a lower Invalid Selection Rate. For example, GPT-4 achieves a 58% Success Rate with only a 0.54% Invalid Selection Rate. In addition, for the remaining 42% of uncompleted tasks, it can adaptively give up on 38%, illustrating its confidence in the evaluation task.

We also observe the ***scaling law in tool learning*** where the performance of LLMs, including stability and effectiveness, increases along with the scaling up of their parameters. This indicates that the inherent capability of foundation LLMs correlates

[1] https://platform.openai/function-call

Inference step	I1-instruction			I1-tool		
	Success%↑	Give up%	Invalid%↓	Success%↑	Give up%	Invalid%↓
gpt-3.5-turbo-16k						
step $s \rightarrow$10 (vanilla)	49.50%	31.50%	0.99%	50.00%	27.00%	0.71%
step $s \rightarrow 6$	32.50%▲	21.00%▲	0.79%▼	32.50%▲	16.50%▲	0.51%▼
step $s \rightarrow 8$	46.50%▼	22.50%▲	1.04%▼	41.50%▲	23.00%▼	0.89%▼
step $s \rightarrow 12$	54.00%▼	29.50%▼	1.48%▼	55.00%▲	25.00%▼	0.81%▼
step $s \rightarrow 14$	51.50%▼	32.50%▼	1.58%▼	57.00%▲	25.00%▼	1.12%▲
deepseek-chat(21B)						
step $s \rightarrow 10$ (vanilla)	39.00%	37.50%	0.46%	38.00%	40.50%	0.68%
step $s \rightarrow 6$	5.50%▲	12.00%▲	0.68%▼	4.00% ▲	14.00%▲	0.93%▼
step $s \rightarrow 8$	24.50%▲	34.00%▼	0.71%▼	23.00%▲	32.00%▲	0.51%▼
step $s \rightarrow 12$	40.00%▼	34.00%▼	0.56%▼	41.00%▼	39.00%▼	0.63%▼
step $s \rightarrow 14$	43.50%▼	42.50%▼	0.84%▼	43.00%▼	39.50%▼	0.76%▼

Table 3: The results with different setting of the maximum inference step s (Section 4.2). We conduct the experiment using both *GPT-3.5* and *Deepseek* model for a holistic comparison. For each experiment, we mark the values with ▲ to indicate that they are statistically significant compared to the vanilla setting; otherwise, we use ▼.

Temperature		Success%↑	Give up%	Invalid%↓
$t = 1.0$ (vanilla)		54.00%	29.50%	1.48%
$t = 0.2$	↓0.8	48.00%▲	26.50%▼	1.04%▼
$t = 0.6$	↓0.4	49.50%▲	24.50%▼	1.04%▼
$t = 1.4$	↑0.4	54.50%▼	34.50%▼	1.93%▼

Table 4: The results with different decoding temperature on the I1-instruction of ToolBench benchmark.

with their tool-learning abilities.

4.2 Impact of Hyper-parameters

In the development of tool-use LLMs, there are several hyper-parameters need to be considered. We investigate two common hyper-parameters that may affect the stability of tool-use LLMs, including the decoding temperature t and the maximum step of inference s. Generally, lower temperature generations are more focused and deterministic while higher temperature generations are more random (Chen and Ding, 2023). We vary the decoding temperatures t from 0.2 to 1.4 with increments of 0.4. The s indicates the maximum inference steps to conduct tool-use actions, *i.e.,* `Thought`, `Action` or `Observation`, which is alternated in {6, 8, 10, 12, 14}. We allow the LLMs to stop early if they complete or give up on a task within s steps.

We first discuss the influence of temperature. As illustrated in Table 4, with the increase in temperature, the Success Rate improves from 48% to 54.5%, and the Invalid Selection Rate shows a slight increase (0.89%). Significant differences are also observed in the Success Rate metric at different temperatures (*e.g.,* $t = 1.0$ and $t = 0.2$). These results indicate that *(1) LLMs exhibit unstable performance towards decoding temperature, and (2)* ***higher temperatures can potentially improve performance with a slightly increased error in tool selection***. A reason for this phenomenon is that higher temperatures boost the LLM to generate more diverse actions during inference (Peeperkorn et al., 2024; Zhu et al., 2024), thereby expanding the generated solution space. We observe a relatively increasing trend in the Give Up Rate when t shifts from 0.6 to 1.4. We look at the poorly performing cases, where we find the reason is that the LLM generates diverse solutions but still fail to derive a correct answer, thereafter adaptively give up the tasks.

Next, we examine the influence of the inference step. As shown in Table 3, we find that the GPT-3.5 only achieves 32.50% in success rate on the I1-inst. dataset when it allowed inference up to 6 steps. However, its Success Rate increases to 49.00% when the maximum inference step is extended to 10. A more obvious trend can be also observed in the Deepseek model, *e.g.,* shifting from 5.50% to 39.00%. These results show that ***the LLM can benefit more trial and exploration step to complete a task correctly***. We also find a relatively stable performance when the inference steps s keeps increasing, *i.e.,* from 10 to 14. In our experimental setup and dataset, setting the inference step to 14 makes a tradeoff for consideration of effectiveness and efficiency for GPT-3.5.

Method	I1-instruction				I1-tool			
	Success%↑	Give up%	Invalid%↓	Cost (tokens)	Success%↑	Give up%	Invalid%↓	Cost (tokens)
gpt-3.5-turbo-16k								
ReAct (vanilla)	54.00%	29.50%	1.48%	15032	55.00%	25.00%	0.81%	16270
DFSDT	69.50%▲	30.00%▲	1.65%▲	37328	67.50%▲	31.50%▲	2.39%▲	45443
deepseek-chat(21B)								
ReAct (vanilla)	40.00%	34.00%	0.56%	17815	41.00%	39.00%	0.63%	17211
DFSDT	55.00%▲	42.50%▼	16.87%▲	47382	54.00%▲	42.50%▼	19.15%▲	49095

Table 5: The results on two datasets with different tool-use frameworks. We mark the results of DFSDT that significantly outperform the vanilla framework (ReAct) with ▲; otherwise, we use ▼.

Method	I1-instruction		I1-tool	
	Success%↑	Give up%	Success%↑	Give up%
gpt-3.5-turbo-16k				
vanilla task	54.00%	29.50%	55.00%	25.00%
- *w/ shorten*	49.50%▼	33.00%▼	52.50%▼	25.50%▼
- *w/ lengthen*	50.50%▼	33.00%▼	50.50%▼	30.00%▼
deepseek-chat(21B)				
vanilla task	40.00%	34.00%	41.00%	39.00%
- *w/ shorten*	38.00%▼	40.50%▲	39.00%▼	43.50%▼
- *w/ lengthen*	37.00%▼	38.00%▼	32.00%▲	47.50%▲

Table 6: The results on two dataset with different user behaviours, *i.e.,* giving a succinct (*w/ shorten*) or a detailed (*w/ lengthen*) task description.

4.3 Impact of Tool-use Framework

The tool-use frameworks indicate the specific techniques or methods to teach the LLM tool usage, automatically guiding them to interact with tools and solve a practical task. We compare two frameworks that are commonly used in previous work, including the ReAct (Yao et al., 2023) and DFSDT (Qin et al., 2023c). ReAct is the default framework in our experiment mentioned in § 4.3, which grounds the tool-use process into `Thought-Action-Observation` format. In contrast, DFSDT (Qin et al., 2023c) augments the LLM with Depth First Search-based Decision Tree to select tools.

The tree-based framework generally performs better but with substantial costs. As illustrated in Table 5, we find that the DFSDT significantly achieves a higher Success Rate on both two datasets with an average of 30.76% point improvement. These results validate the superiority of the tree-based search algorithm in recalling required tools to solve a task. However, it comes up with substantial inference cost, *i.e.,* consuming nearly triple tokens, which may limits its effectiveness in low-resource scenarios or low-latency applications.

We also observe that the Deepseek model, when equipped with the DFSDT method, shows a substantial increase in Invalid Selection Rate. It indicates that open-source models suffer from relatively severe hallucinations to generate non-existing tool names, especially when intensively selecting tools. Thus, we advocate the optimization of LLM to reduce its hallucination in generating correct tool names, thereby leveraging tree-based tool-use frameworks.

5 Analysis of External Factors

External factors involve the practical prompts to enable tool-use models, including diverse user prompts, customized system prompts, and the input candidate toolset.

5.1 Impact of User Prompts

In real-world applications, users exhibit diverse behaviors when interacting with the tool-use model. Therefore, we first simulate two practical behaviors of users, including: (1) succinct: a user provides a short instruction; and (2) detailed: a user provides a lengthy and comprehensive instruction. To achieve this, we employ *gpt-3.5-turbo-0125* to compress or elaborate the description for each task in our experimental datasets, respectively, without changing the semantics and key information. The details for this rewriting operation can be found in Appendix A.1.

As shown in Table 6, LLMs are relatively stable towards user behaviors. Since LLMs are trained on a massive web corpus, ***they developed strong abilities to capture key information of a task despite the diverse styles of descriptions from various users.***

5.2 Order of Candidate Toolsets

Given a task, the LLM first selects a series of tools from a candidate toolset $\mathcal{S}$ in a step-by-step manner and then executes the selected tools to obtain the final answer. Since the LLM suffers from the position bias (Liu et al., 2024) in a broad range

Method	I1-instruction			I1-tool		
	Success%↑	Give up%	Invalid%↓	Success%↑	Give up%	Invalid%↓
gpt-3.5-turbo-16k						
vanilla toolset	54.00%	29.50%	1.48%	55.0%	25.00%	0.81%
randomly shuffle	51.00%▼	31.50%▼	1.80%▼	52.50%▼	25.50%▼	0.77%▼
expand *w/ intra-category*	51.00%▼	22.50%▲	0.77%▼	47.50%▲	23.50%▼	1.38%▲
expand *w/ cross-category*	47.00%▲	28.00%▼	0.53%▼	52.50%▼	18.00%▲	1.00%▼
deepseek-chat(21B)						
vanilla toolset	40.00%	34.00%	0.56%	41.00%	39.00%	0.63%
randomly shuffle	32.50%▼	35.50%▼	0.52%▼	27.00%▲	32.50%▼	0.76%▼
expand *w/ intra-category*	33.00%▲	34.00%▼	0.05%▼	32.50%▲	38.50%▼	0.63%▼
expand *w/ cross-category*	37.50%▼	32.00%▼	0.26%▼	32.00%▲	37.00%▼	0.51%▼

Table 7: The results on two datasets where we change the candidate toolset provided to tool-use model using different strategies, including randomly shuffle (Section 5.2), relevant sampling and noise sampling (Section 5.3).

of downstream tasks like document ranking (Tang et al., 2023b), we analyze whether the order of the tools in $\mathcal{S}$ can influence its performance in the tool-use workflow. We randomly shuffle the original toolset (vanilla) for each task in our experiment dataset and evaluate the model's performance.

Open-source model suffers from the positional bias of tools. As shown in Table 7, we find the weak open-source model, *i.e.,* Deepseek, suffers from pronounced positional bias. For example, when we shuffle the original order of the toolset, its success rate decreases from 41.00% (*original*) to 27.00% (*shuffle*) on the I1-tool dataset. The 7.5% decrease is also observed in the I1-instruction dataset. A similar phenomenon is also observed in other tasks, such as text summarization (Chhabra et al., 2024), and code search (Li et al., 2023c). In addition, we find that the GPT-3.5 is nearly insensitive to the order of the toolset, and only a 3% point difference in success rate is observed. These results indicate that powerful models with higher Success Rate are more skillful in solving tasks, thereby showing less instability toward positional bias, and vice versa.

5.3 Scale of Candidate Toolsets

Beyond the ground truth tools to solve an input task, the toolset $\mathcal{S}$ is typically large-scale in real-world scenarios, inevitably containing irrelevant or plausible-looking tools (*a.k.a.,* noise). Therefore, we further benchmark the stability of models under the different scale of toolset $\mathcal{S}$. For a more practical evaluation, we expand the scale of the toolset using two sampling strategies for each task: (1) Intra-category sampling: we augment the original toolset with tools sampled from the same category as the ground truth tools. These tools are related to the current task but not useful. (2) Cross-category sampling: we sample irrelevant tools from different categories than the ground truth tools.

Prompt	Success%↑	Give up%	Invalid%↓
gpt-3.5-turbo-0125			
Vanilla (func. call)	55.00%	36.50%	1.45%
Customized	40.00%▲	35.00%▼	14.93%▲
gpt-3.5-turbo-1106			
Vanilla (func. call)	48.00%	50.5%	0.88%
Customized	47.50%▼	41.00%▲	6.73%▲
llama-3-70b			
Vanilla (naive)	8.00%	4.50%	42.16%
Customized	30.00%▲	10.00%▼	3.15%▲

Table 8: The results on various LLMs with different system prompts. (func.: function)

Unstable performance is observed with change of toolset scale. We summarize the results in Table 7. We observe that both closed-source and open-source LLMs exhibit substantial performance degradation with the increase of toolsets scale. These results indicate the instability of LLMs towards irrelevant or relevant but useless tools. We also find a decreased trend in Give Up Rate. Thus, we dive into specific cases, where we find that with more candidate tools, the LLM tends to be stubborn and stuck in continuously selecting useless rather than adaptively stopping. These findings motivate us to carefully design the tool selection module in developing tool-use LLM systems or applications.

5.4 Impact of System Prompts

The system prompts indicate the input prompt demonstrating LLMs how to use tools, which pre-

defines the format of the model's generation. Our vanilla setting implements the system prompt of closed-source LLMs with *function call*, which is an API interface exclusively supported by OpenAI. For open-source LLMs, the system prompt is a zero-shot instruction $\mathcal{P}$ (§ 3.4). Here, we consider human efforts in optimizing the prompts, where we formulate a detailed instruction $\mathcal{P}^{\dagger}$ on top of $\mathcal{P}$ by supplementing fine-granularity description to specify usage specifications. We evaluate both closed-source and open-source model with $\mathcal{P}^{\dagger}$ to analyze their stability towards diverse prompts. We provide all prompts in Appendix A.4.

Table 8 presents the experimental results. The gpt-3.5-0125 suffers from a 15% decrease in Success Rate and a 13.48% increase in Invalid Selection Rate when we swap the official function-call prompt with manually customized prompts. This result intuitively demonstrates ***the LLMs are sensitive to different system prompts.***.

We also observe that the performance of the Deepseek model substantially improves (*e.g.,* 22% higher Success Rate) when equipped with customized prompt $\mathcal{P}$. This result illustrates that the LLM can understand tool-use instructions in a zero-shot manner, aligning with previous work (Hsieh et al., 2023). Therefore, ***directly providing clear rules and instructions in system prompts*** is a potential alternative to enhance the tool-use ability of open-source models without cost-intensive supervised fine-tuning.

6 Discussion

The self-consistency of tool-use models. We further explore the self-consistency of tool-use models. Specifically, we repeatedly prompt a model to solve the tasks in the *I1-inst.* dataset N times with the same settings as in Table 2. We then count the percentage of completed tasks that can be solved in the first run, which reflects the consistency of the model through the discrepancies between different runs. In our implementation, we set N to 3. We find that the Mixtral-8x7B model can solve 57 tasks in the first run, but 20 of the initially failed tasks can be solved during the second and third runs. Similar phenomena are also observed in other LLMs, such as GPT-3.5. These results directly indicate that the stability of LLMs still needs to be improved. More details can be found in Appendix A.3.

Case study. We compare the output of tool-use models for the same task under different experimental settings, such as different prompts, inference steps, and candidate toolsets, showing their instability intuitively. More details can be found in Appendix A.5 for further explanation.

Takeaways. Since the tool-learning frameworks still suffer from instability due to various factors, we summarize our findings as several useful takeaways to boost the performance of tool-learning frameworks: (1) Decoding temperature can significantly affect the stability of tool-use LLMs (§ 4.2). In solving complex tasks, users can set relatively higher temperatures to boost LLMs to generate more diverse actions, thereby expanding the solution space. (2) Users can augment LLMs with tool selection algorithms, *e.g.,* Depth-First-Search, which effectively improve the success rate through more trial and error. However, one should also consider the associated disadvantages, such as increased inference costs and the accumulated hallucination of tool selection errors over extended workflows. (3) Different system prompts result in varied performance. The closed-source models are trained to access tools through specialized function-call prompts, thereby showing fewer errors in workflow. Thus, we advocate tuning models with specific tool-use datasets or supplementing fine-granularity descriptions in prompts, aligning their generation with pre-defined usage specifications. (4) The LLMs are sensitive to the order and scale of the toolset.

7 Conclusion

We present a comprehensive empirical study on the stability of tool-use models. Specifically, we explore the impact of both internal and external factors on the tool-learning frameworks. Internal factors include uncertainties during the development of the tool-use model, while external factors primarily involve diverse input prompts. Our quantitative analysis demonstrates that even powerful models such as GPT-3.5 exhibit significant instability in response to these factors. We also provide valuable findings and practical insights to facilitate further research in this area. Our future work includes: (1) extending our evaluation to tool-use agents empowered by multi-modal LLMs; and (2) exploring the model's stability in more intricate environments, such as dynamic interactions with users.

Limitations

The main limitation is that we only investigate the stability of widely used LLM-based agents. These agents are limited when tackling multi-modal tasks. In the future, we plan to extend our method to agents empowered by multi-modal foundation models. Additionally, our empirical study does not involve dynamic interactions between the user (or user simulator) and the tool-use model for the sake of reproducibility. We plan to extend our work to more intricate environments, such as conversational and user-centered scenarios, further exploring the stability of tool-use models.

Ethics Statement

The research conducted in this paper centers on investigating the stability of tool-use systems. Our work systematically benchmarks LLMs under various real-world scenarios and evaluates their potential instability.

In the process of conducting this research, we have adhered to ethical standards to ensure the integrity and validity of our work. All the tasks as well as tools used in our experiment were obtained from existing benchmarks, thus ensuring a high level of transparency and reproducibility in our experimental procedure.

To minimize potential bias and promote fairness, we use the prompts following existing works, which are publicly accessible and freely available. We have made every effort to ensure that our research does not harm individuals or groups, nor does it involve any form of deception or potential misuse of information.

References

Honghua Chen and Nai Ding. 2023. Probing the creativity of large language models: Can models produce divergent semantic association? *Preprint*, arXiv:2310.11158.

Anshuman Chhabra, Hadi Askari, and Prasant Mohapatra. 2024. Revisiting zero-shot abstractive summarization in the era of large language models from the perspective of position bias. *arXiv preprint arXiv:2401.01989*.

Shen Gao, Zhengliang Shi, Minghang Zhu, Bowen Fang, Xin Xin, Pengjie Ren, Zhumin Chen, Jun Ma, and Zhaochun Ren. 2024. Confucius: Iterative tool learning from introspection feedback by easy-to-difficult curriculum. In *Proceedings of the AAAI Conference on Artificial Intelligence: AAAI*.

Jiasheng Gu, Hongyu Zhao, Hanzi Xu, Liangyu Nie, Hongyuan Mei, and Wenpeng Yin. 2022. Robustness of learning from task instructions. *arXiv preprint arXiv:2212.03813*.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*.

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

Cheng-Yu Hsieh, Si-An Chen, Chun-Liang Li, Yasuhisa Fujii, Alexander Ratner, Chen-Yu Lee, Ranjay Krishna, and Tomas Pfister. 2023. Tool documentation enables zero-shot tool-usage with large language models. *arXiv preprint arXiv:2308.00675*.

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

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

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023a. API-bank: A comprehensive benchmark for tool-augmented LLMs. In *Association for Computational Linguistics: EMNLP*.

Moxin Li, Wenjie Wang, Fuli Feng, Yixin Cao, Jizhi Zhang, and Tat-Seng Chua. 2023b. Robust prompt optimization for large language models against distribution shifts. In *Association for Computational Linguistics*.

Zongjie Li, Chaozheng Wang, Pingchuan Ma, Daoyuan Wu, Shuai Wang, Cuiyun Gao, and Yang Liu. 2023c. Split and merge: Aligning position biases in large language model based evaluators. *arXiv preprint arXiv:2310.01432*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Zhongkun Liu, Zheng Chen, Mengqi Zhang, Zhaochun Ren, Zhumin Chen, and Pengjie Ren. 2024. Zero-shot position debiasing for large language models. *arXiv preprint arXiv:2401.01218*.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models. *Neural Information Processing Systems: NeurIPS*.

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

Max Peeperkorn, Tom Kouwenhoven, Dan Brown, and Anna Jordanous. 2024. Is temperature the creativity parameter of large language models? *arXiv preprint arXiv:2405.00492*.

Shuofei Qiao, Ningyu Zhang, Runnan Fang, Yujie Luo, Wangchunshu Zhou, Yuchen Eleanor Jiang, Chengfei Lv, and Huajun Chen. 2024. AUTOACT: Automatic Agent Learning from Scratch via Self-Planning. *arXiv preprint arXiv:2401.05268*.

Yujia Qin, Zihan Cai, Dian Jin, Lan Yan, Shihao Liang, Kunlun Zhu, Yankai Lin, Xu Han, Ning Ding, Huadong Wang, Ruobing Xie, Fanchao Qi, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2023a. WebCPM: Interactive web search for Chinese long-form question answering. In *Association for Computational Linguistics: ACL*.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, et al. 2023b. Tool learning with foundation models. *arXiv preprint arXiv:2304.08354*.

Yujia Qin, Shi Liang, Yining Ye, Kunlun Zhu, Lan Yan, Ya-Ting Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Marc H. Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023c. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. *International Conference on Learning Representations: ICLR*.

Ruiyang Ren, Yuhao Wang, Yingqi Qu, Wayne Xin Zhao, Jing Liu, Hao Tian, Hua Wu, Ji-Rong Wen, and Haifeng Wang. 2023. Investigating the factual knowledge boundary of large language models with retrieval augmentation. *arXiv preprint arXiv:2307.11019*.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. 2024. Identifying the risks of lm agents with an lm-emulated sandbox. *Preprint*, arXiv:2309.15817.

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. *Neural Information Processing Systems: NeurIPS*.

Zhengliang Shi, Shen Gao, Xiuyi Chen, Yue Feng, Lingyong Yan, Haibo Shi, Dawei Yin, Zhumin Chen, Suzan Verberne, and Zhaochun Ren. 2024a. Chain of tools: Large language model is an automatic multi-tool learner. *arXiv preprint arXiv:2405.16533*.

Zhengliang Shi, Shen Gao, Xiuyi Chen, Lingyong Yan, Haibo Shi, Dawei Yin, Zhumin Chen, Pengjie Ren, Suzan Verberne, and Zhaochun Ren. 2024b. Learning to use tools via cooperative and interactive agents. *arXiv preprint arXiv:2403.03031*.

Zhengliang Shi, Shen Gao, Zhen Zhang, Xiuying Chen, Zhumin Chen, Pengjie Ren, and Zhaochun Ren. 2023. Towards a unified framework for reference retrieval and related work generation. In *Association for Computational Linguistics: EMNLP*.

Yifan Song, Weimin Xiong, Dawei Zhu, Chengzu Li, Ke Wang, Ye Tian, and Sujian Li. 2023. RestGPT: Connecting Large Language Models with Real-World Applications via RESTful APIs. *arXiv*.

Student. 1908. The probable error of a mean. *Biometrika*, pages 1–25.

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

Raphael Tang, Xinyu Zhang, Xueguang Ma, Jimmy Lin, and Ferhan Ture. 2023b. Found in the middle: Permutation self-consistency improves listwise ranking in large language models. *arXiv preprint arXiv:2310.07712*.

Shubo Tian, Qiao Jin, Lana Yeganova, Po-Ting Lai, Qingqing Zhu, Xiuying Chen, Yifan Yang, Qingyu Chen, Won Kim, Donald C Comeau, et al. 2024. Opportunities and challenges for chatgpt and large language models in biomedicine and health. *Briefings in Bioinformatics*, 25(1):bbad493.

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

Jindong Wang, Xixu Hu, Wenxin Hou, Hao Chen, Runkai Zheng, Yidong Wang, Linyi Yang, Haojun Huang, Wei Ye, Xiubo Geng, et al. 2023b. On the robustness of chatgpt: An adversarial and out-of-distribution perspective. *arXiv preprint arXiv:2302.12095*.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024. Executable code actions elicit better llm agents. *arXiv preprint arXiv:2402.01030*.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024. Travelplanner: A benchmark for real-world planning with language agents. *arXiv preprint arXiv:2402.01622*.

Qiantong Xu, Fenglu Hong, Bo Li, Changran Hu, Zhengyu Chen, and Jian Zhang. 2023. On the tool manipulation capability of open-source large language models. *arXiv preprint arXiv:2305.16504*.

Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. 2023a. GPT4Tools: Teaching Large Language Model to Use Tools via Self-instruction. *Neural Information Processing Systems: NeurIPS*.

Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. 2023b. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*.

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 *International Conference on Learning Representations: ICLR*.

Junjie Ye, Sixian Li, Guanyu Li, Caishuang Huang, Songyang Gao, Yilong Wu, Qi Zhang, Tao Gui, and Xuanjing Huang. 2024a. Toolsword: Unveiling safety issues of large language models in tool learning across three stages. *arXiv*.

Junjie Ye, Yilong Wu, Songyang Gao, Sixian Li, Guanyu Li, Xiaoran Fan, Qi Zhang, Tao Gui, and Xuanjing Huang. 2024b. Rotbench: A multi-level benchmark for evaluating the robustness of large language models in tool learning. *arXiv preprint arXiv:2401.08326*.

Yuqi Zhu, Jia Li, Ge Li, YunFei Zhao, Zhi Jin, and Hong Mei. 2024. Hot or cold? adaptive temperature sampling for code generation with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2023. Toolqa: A dataset for llm question answering with external tools. *arXiv*.

A Appendix

A.1 Implement details

Details of Foundation Models in our experiment. We provide the source of the LLMs used in our experiment.

1. **gpt-3.5-turbo-16k-0613** https://platform.openai.com/docs/models/gpt-3-5-turbo
2. **gpt-3.5-turbo-0125** https://platform.openai.com/docs/models/gpt-3-5-turbo
3. **gpt-3.5-turbo-1106** https://platform.openai.com/docs/models/gpt-3-5-turbo
4. **gpt-4o-2024-5-13** https://platform.openai.com/docs/models/gpt-4o
5. **deepSeek-chat** https://github.com/deepseek-ai/DeepSeek-V2
   Weight download: https://huggingface.co/deepseek-ai/DeepSeek-V2-Chat
6. **Llama-3-8b** https://github.com/meta-llama/llama3/tree/main
   Weight download: https://huggingface.co/meta-llama/Meta-Llama-3-8B
7. **Llama-3-70b** https://github.com/meta-llama/llama3/tree/main
   Weight download: https://huggingface.co/meta-llama/Meta-Llama-3-70B
8. **mixtral-8x7b-instruct** https://docs.mistral.ai/getting-started/models/
   Weight download: https://huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1
9. **mixtral-8x22b** https://docs.mistral.ai/getting-started/models/
   Weight download: https://huggingface.co/mistralai/Mixtral-8x22B-v0.1

Method	BertScore		
	Precision	Recall	F1 score
I1-instruction			
shorten	0.767	85.79%	80.96%
lengthen	0.827	76.17 %	79.22 %
I1-tool			
shorten	0.731	85.35 %	78.66%
lengthen	0.832	76.81%	79.8%

Table 9: The BertScores of rewriting queries and vanilla queries.

Details of Rewriting the Task description. In order to verify whether the information and semantics remain consistent before and after rewriting, we invite two well-educated master students to evaluate the similarity of queries rewritten by gpt-3.5-turbo-0125. Despite the differing lengths of the rewritten dataset and the original dataset, the results show that the information in these two datasets has high semantic similarity. The similarity for the two different methods is 98%. We also compute their semantic similarity using the *bertscore* in Table 9 to further validate the reliability of our setting.

We provide the prompts for the rewriting operation as follows.

Prompt for query shorten

```
You are a helpful assistant which can
make a query shorter but remain the
meaning.

please shorten the query to one sentence
: {query}
Just give me the final answer.

Your output:
```

Prompt for query lengthen

```
You are a helpful assistant. What you
have to do is making a query longer to
generate more information in the query's
 scenario but remain the meaning. It's
also a query, but longer than before,
remember it! Do NOT answer any question,
 but rewrite it longer!

please lengthen the query: {query}
Just give me the final answer.

Your output:
```

User prompts. Our experiment is built upon existing publicly available datasets for high transparency and to minimize potential bias. Therefore, we do not change the task description from the semantic level due to the possible misalignment between the changed tasks and the original ground truth. An ideal benchmark scenario is conversational applications, where the tool-use model can interact with more diverse users. We take it as our future work.

A.2 Repurpose existing dataset

The original ToolBench only provides a task-solving trajectory of GPT-3.5 as an evaluation reference, which includes both ground truth and irrelevant tools. However, commonly used evaluation metrics require computing the overlap between model-selected tools and the ground truth. Therefore, we repurpose ToolBench to support our evaluation. For each task, we extract the tools used in the original solution. Next, we invite three well-educated experts with relevant research backgrounds to manually select the correct tools for solving the task. During the annotation, we provide experts with the documentation of candidate tools and detailed solution trajectories for each task to minimize their ambiguity.

We employ the following strategies to ensure the quality of the above process.

- **Detailed annotator training.** We held regular meetings and pre-annotation tests to ensure that each expert undergoes detailed training to familiarize themselves with our annotation task.

- **Tackling discrepancies.** We ask at least two experts to annotate the same task repeatedly. If there is a discrepancy between the two experts, *i.e.,* two experts give different solutions for the same task, we ask a third expert to recheck it. We also filter the task with ambiguity to improve the reliability of our repurposed datasets.

A.3 The self-consistency of tool-use models.

We further explore the self-consistency of tool-use models. Specifically, we repeatedly prompt a model to solve the tasks in the *I1-inst.* dataset N times with the same settings as in Table 2. We then count the percentage of completed tasks that can be solved in the first run, which reflects the consistency of the model through the discrepancies between different runs. In our implementation, we set N to 3.

Model	Successfully finished tasks		
	First run	Second run	Third run
gpt-3.5-turbo-4o	101	8	7
gpt-3.5-turbo-1106	79	12	5
gpt-3.5-turbo-0125	83	20	8
gpt-3.5-turbo-16k	79	18	11
deepspeek-chat	47	21	13
Llama-3-70B	6	6	4
Llama-3-8B	3	3	1
Mixtral-8x22B	28	14	8
Mixtral-8x7B	10	8	6

Table 10: Statistics for the number of successfully finished tasks during the *first*, *second*, and *third* run, respectively.

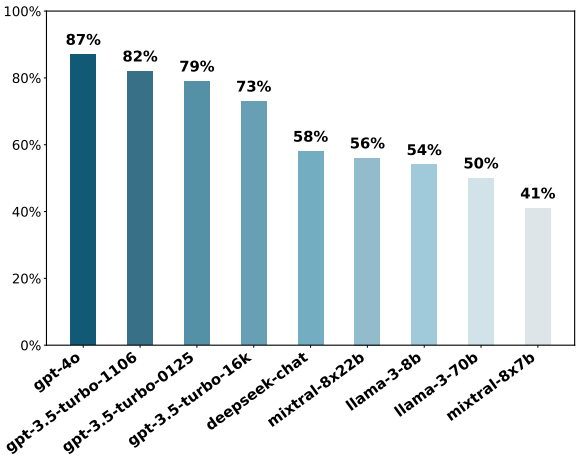

Figure 4: Self-consistency Success Rate of different models.

As illustrated in Table 10, we find that the Mixtral-8x7B model can solve 57 tasks in the first run, but 20 of the initially failed tasks can be solved during the second and third runs. Similar phenomena are also observed in other LLMs, such as GPT-3.5. These results directly indicate that the instability of LLMs still needs to be improved. We also show their consistency percentage in Figure 4 for an intuitive explanation.

A.4 Examples of Instructions

Base system prompt

```
You are AutoGPT, you can use many tools(
functions) to do the following task.
First I will give you the task
description, and your task start.
At each step, you need to give your
thought to analyze the status now and
what to do next, with a function call to
```

```
 actually excute your step. Your output
should follow this format:
Thought:
Action
Action Input:

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 finial answer.
Remember:
1.the state change is irreversible, you
can't go back to one of the former state
, if you want to restart the task, say "
I give up and restart".
2.All the thought is short, at most in 5
 sentence.
3.You can do more then one trys, so if
your plan is to continusly try some
conditions, you can do one of the
conditions per try.
Let's Begin!
Task description: {task_description}
Specifically, you have access to the
following APIs:
{API list}
```

Changed system prompt

```
You are AutoGPT, you can use many tools(
functions) to do the following task.
First I will give you the task
description, and your task start.
At each step, you need to give your
thought to analyze the status now and
what to do next, with a function call to
 actually excute your step. Your  EVERY
output should follow this format:
Thought:{there is your reason for
choosing one api}
Action:{there is the api name you
choosing from the given ones}
Action Input:{there are the inputs for
the chosed api using'{}', and each
parameter should using '\"\"'}
RULEs:
Once after giving one Action Input, stop
 your answer\nDoing this step by step,
ONE TIME ONE ACTION.
If one api is not access, you can choose
 another one.
You had better to give an action each
time.
One step just give one function call,
and you will give ONE step each time I
call you.
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 finial answer.
Remember:
1.the state change is irreversible, you
can't go back to one of the former state
, if you want to restart the task, say "
I give up and restart".
2.All the thought is short, at most in 5
 sentence.
3.You can do more then one trys, so if
your plan is to continusly try some
conditions, you can do one of the
conditions per try.
Let's Begin!
Task description: {task_description}
Specifically, you have access to the
following APIs:
{api list}
```

System prompt while using function call

```
You are AutoGPT, you can use many tools(
functions) to do the following task.
First I will give you the task
description, and your task start.
At each step, you need to give your
thought to analyze the status now and
what to do next, with a function call to
 actually excute 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 finial answer.
Remember:
1.the state change is irreversible, you
can't go back to one of the former state
, if you want to restart the task, say "
I give up and restart".
2.All the thought is short, at most in 5
 sentence.
3.You can do more then one trys, so if
your plan is to continusly try some
conditions, you can do one of the
conditions per try.
Let's Begin!
Task description: {task_description}
```

Input while using function call

```
{query}
"function":[{
    "name":{function name},
    "description":{function description
    },
    "parameters":{function parameters}
},
{function},
...
]
```

A.5 Case study

We conduct a comprehensive cases study to investigate the instability of tool-use models and provide the following cases for intuitive explanations.

The impact of different foundation models. As illustrated in Figure 5, we find that the closed-model, *i.e.,* gpt-4o-2024-05-13 can successfully finish the task with no redundant steps. However, the commonly used open-source models, *i.e.,* Mixtral-8x7B and deepseek-chat, struggle to generate correct arguments and fail to solve the task. This case indicates the varied performance among different backbone LLMs and the open-source models still lay behind the closed-source models in the tool learning tasks.

The impact of different decoding temperature. Figure 6 presents the output of GPT-3.5 under different decoding temperatures. We find that when the decoding temperature is set to 0.2, it becomes stubborn to repeat the same incorrect actions instead of generating new ones. In contrast, when the temperature is increased to 1.4, the LLM can adaptively correct its mistakes in response to error messages and generate new actions. This case demonstrates that the LLM exhibits varied performance at different temperatures, with higher temperatures encouraging the generation of more diverse actions, thereby validating our findings in § 4.2.

The impact of the maximum inference step. We present a concrete task-solving trajectory of GPT-3.5 on the I1-inst. dataset in Figure 7. We find that the LLM fails to solve a complex task within 6 steps. However, when the maximum inference step is increased to 12, the LLM benefits from more trials and exploration, thereby completing the task. This case indicates that it is crucial to adapt the maximum inference step according to the task complexity.

The impact of the tools scale. Figure 9 compares the model's output with different scales of candidate tools. We find that when we add more irrelevant tools in the original toolset (vanilla), the LLM is misled to select inappropriate tools and generate incorrect arguments, thereafter failing the task. This case indicates that despite the powerful LLMs, *e.g.,* GPT-3.5, they are sensitive to the scale of tools, which is aligned with our analysis in § 5.3.

The impact of the tools order. Figure 9 compares the model's output with different order of candidate tools.

Query: I'm looking to buy a house in the Hampshire county area. Can you provide me with a list of zip codes in the county? Also, I'd like to know the location information for zip code 01003.

====gpt-4o-2024-05-13 (Success)====

Action 1: get_zipcode_by_county_for_redfin_base
Action Input 1: {"county":"Hampshire""
Observation 1: {"response": {'data': [{'zipCode': '01002', 'regionId': None, 'city': 'Amherst', 'county': 'Hampshire', 'state':
Action 2: get_location_infomation_by_zipcode_for_redfin_base
Action Input 2: {"zipcode":"01003""
Observation 2: {"response": {'data': [{'zipCode': '01003', 'regionId': None, 'city': 'Amherst', 'county': 'Hampshire', 'state': …
Action 3: Finish
Action Input 3: {"return_type":"give_answer","final_answer":"The list of zip codes in Hampshire county includes: 01002, 01003, 01004, 01007, and others. For the specific zip code 01003, the location information is as follows: City: Amherst, County: Hampshire, State: Massachusetts, Country: United States.""
Observation 3: {"response":"successfully giving the final answer.}

====mixtral-8x7b (Fail)====

Thought 1: I need to get the zip codes in Hampshire county first. After that, I will get the location information for zip code 01003.
Action 1: get_zipcode_by_county_for_redfin_base
Action Input 1: {\n"county":"Hampshire"\n} (After receiving the result from the above action)\n\nThought:\nNow I have the zip codes in Hampshire county. Next,I will get the location information for zipcode 01003.\nAction:....."
Observation 1: {"error": "Tool input parse error...\\n", "response": """
Thought 2: I need to use the function "get_zipcode_by_county_for_redfin_base" to get the zip codes in Hampshire county.
Action 2: get_zipcode_by_county_for_redfin_base
Action Input 2: {"county":"Hampshire""
Observation 2: {"error": "No such function name: get_zipcode_by_county_for_redfin_base", "response": """
…
Action 4: ""
Action Input 4: {}
Observation 4: "The function name can not be none, please give a valid function name by 'Thought:\nAction:\nAction Input:\n' format"

====deepseek-chat (Fail)====

Thought 1: I need to first retrieve a list of zip codes for Hampshire county. After that, I can then get the location information code 01003.
Action 1: get_zipcode_by_county_for_redfin_base
Action Input 1: "{'county': 'Hampshire'}"
Observation 1: {“error”:“Tool input parse error...\\n”, “response”: “”}"
Thought 2: "“
Action 2: get_zipcode_by_county_for_redfin_base
Action Input 2: "{"county": "Hampshire"}“
Observation 2: "{"error": "", "response": "{'data': [{'zipCode': '01002', 'regionId': None, 'city': 'Amherst', 'county': 'Hampshire', 'state':..."
…
Action 4: get_zipcode_by_county_for_redfin_base
Action Input 4: "{zipcode: 'Hampshire'}"
Observation 4: {"error": "", "response\": \"{'data': [{'zipCode': '01002', 'regionId': None, 'city': 'Amherst', 'county': 'Hampshire', 'state':..."

Figure 5: Impact of different foundation models.

Query: I'm planning a family vacation to Santa Monica and I need some assistance. Can you help me find available rental properties in this area? I would like to know the address, monthly rent, number of bedrooms and bathrooms, and any additional details for each property. Also, please provide me with the contact information of agents who specialize in rental properties in Santa Monica.

====gpt-3.5-turbo-16k-0613 T=1.4 (Success)====

Action 1: search_properties_for_rent_for_realtor
Action Input 1: {"location": "Santa Monica\"}
Observation 1: {"error":"", "response": "{'count': 50, 'results': [{'advertisers': [{'email': None, 'fulfillment_id': None, 'href': None…
Action Input 2: {"zipcode":"01003""
Observation 2: {"response": {'data': [{'zipCode': '01003', 'regionId': None, 'city': 'Amherst', 'county': 'Hampshire', 'state': …
Thought 3: I found some rental properties in Santa Monica. I have retrieved the contact information for the agents as well.
Action 3: Finish
Action Input 3: {return_type": "give_answer","final_answer": "I found 50 rental properties in Santa Monica. Here are a few of them:1. Property Address: [123 Main Street], Monthly Rent: $2000, Bedrooms: 2, Bathrooms: 1Agents specializing in rental properties in Santa Monica:1. Agent Name: John Doe, Contact: john.doe@example.com2. Agent Name: Jane Smith, Contact: jane.smith@example.comPlease let me know if you would like more details about any specific property."}
Observation 3: {"response":"successfully giving the final answer.}

====gpt-3.5-turbo-16k-0613 T=0.2 (Fail)====

Action 1: search_properties_for_rent_for_realtor
Action Input 1: {"location": "Santa Monica"}
Observation 1: "{"error": "", "response": "{'count': 50, 'results': [{'advertisers': [{'email': None, 'fulfillment_id': None, 'href': None,…
Thought 2: I found 50 available rental properties in Santa Monica. However, the response from the API is not in a readable format.
Action 2: search_properties_for_rent_for_realtor
Action Input 2: {"location": "Santa Monica"}
Observation 2: "{\"error\": \"\", \"response\": \"{'count': 50, 'results': [{'advertisers': [{'email': None, 'fulfillment_id': None, 'href': …
…
Action 5: search_properties_for_rent_for_realtor
Action Input 5: {"location": "Santa Monica"}
Observation 5: {\"error\": \"\", \"response\": \"{'count': 50, 'results': [{'advertisers': [{'email': None, 'fulfillment_id': None, 'href': …

Figure 6: Impact of different temperature

Query: I want to explore the world of cinema and discover interesting movie news articles. Can you provide me with the top trending movie articles? Additionally, I would like to search for articles about drama movies published in the last month.

====gpt-3.5-turbo-16k-0613 step = 12 (Success) ====

Action 1: trending_for_kubric_the_comprehensive_movie_news_api
Action Input 1: {}
Observation 1: {"error": "", "response": "{'message': 'This endpoint is disabled for your subscription'}"}
Action 2: advanced_search_for_kubric_the_comprehensive_movie_news_api
Action Input 2: {"q": "drama movies","maxdate": "2021-12-31","mindate": "2021-12-01"}
Observation 2: {"error": "", "response": "{'message': 'This endpoint is disabled for your subscription'}"}
Action 3: advanced_search_for_kubric_the_comprehensive_movie_news_api
Action Input 3: {"q": "drama movies","maxdate": "2021-12-31","mindate": "2021-12-01"}
Observation 3: {"error": "", "response": "{'message': 'This endpoint is disabled for your subscription'}"} **<=== step = 6 (Fail) ====**
Action 4: basic_search_for_kubric_the_comprehensive_movie_news_api
Action Input 4: {"q": "drama movies","limit": 5}
Observation 4: {"error": "", "response": "{'success': True, 'body': {'totalArticles': 12991, 'results': [{'_id': '6455399b7294daad5a9a5010'…
Action 5: Finish
Action Input 5: {"return_type": "give_answer","final_answer": "Here is one of the latest drama movie news articles Title: 5 …
Observation 5: "{"response":"successfully giving the final answer."}

Figure 7: Impact of inference step.

Query: I'm planning to attend a football match next week and I want to know the current standings and the top scorers in the league. Can you give me the latest league standings and the leading goal scorers?

====gpt-3.5-turbo-16k-0613 original API list(Success)====

Action 1: competitions_for_zeus_api
Action Input 1: {}
Observation 1: {"error": "", "response": "{'competitions': [{'id': 'PRL', 'name': 'Premier League', 'description': "England's first division…
Action 2: competitions_standings_for_zeus_api
Action Input 2: {"is_id": "PRL"}
Observation 2: {"error": "", "response": "{'name': 'Premier League', 'area': 'ENG', 'yearStart': 2020, 'yearEnd': 2021, 'standings': [{'position"…
…
Action 4: Finish
Action Input 4: "{"return_type": "give_answer","final_answer": "The current standings in the Premier League are as follows:1….
Observation 4: {"response":"successfully giving the final answer."}

====gpt-3.5-turbo-16k-0613 shuffled API list (Success)====

Action 1: competitions_for_zeus_api
Action Input 1: {}
Observation 1: {"error": "", "response": "{'competitions': [{'id': 'PRL', 'name': 'Premier League', 'description': "England's first division….
Action 2: competitions_standings_for_zeus_api
Action Input 2: {"is_id": "PRL"}
Observation 2: {"error": "", "response": "{'name': 'Premier League', 'area': 'ENG', 'yearStart': 2020, 'yearEnd': 2021, 'standings': …
…
Action 4: Finish
Action Input 4: "{"return_type": "give_answer","final_answer": "The current standings in the Premier League are as follows: …
Observation 4: {"response":"successfully giving the final answer."}

Figure 8: Impact of candidate toolsets' order.

Query: I am writing an article about the top tourist destinations in the United States. Can you provide me with the names and locations of the cities in the country?

====gpt-3.5-turbo-16k-0613 vanilla toolset(Success)====

Action 1: cities_in_a_country_for_referential_db
Action Input 1: {"countrycode": "US"}
Observation 1: {"error": "", "response": \"[{'id': '110965', 'name': 'Abbeville', 'state_code': 'GA', 'country_code': 'US', 'latitude': …
Thought 2: Based on the data from the referential database, here are some cities in the United States:1. Abbeville, Georgia 2. Abbeville…
Action 2: competitions_standings_for_zeus_api
Action Input 2: {"is_id": "PRL"}
Observation 2: {"error": "", "response": "{'name': 'Premier League', 'area': 'ENG', 'yearStart': 2020, 'yearEnd': 2021, 'standings': [{'position':…
Action 3: Finish
Action Input 3: {"return_type": "give_answer","final_answer": "Here are some cities in the United States: Abbeville, Georgia…
Observation 3: {"response":"successfully giving the final answer."}

====gpt-3.5-turbo-16k-0613 expand w/ cross-category (Fail)====

Action 1: cities_in_a_country_for_referential_db
Action Input 1: {"countrycode": "US"}
Observation 1: {"error": "", "response": "[{'id': '110965', 'name': 'Abbeville', 'state_code': 'GA', 'country_code': 'US', 'latitude':….
Thought 2: "Here are some cities in the United States:1. Abbeville, Georgia2. Abbeville, Louisiana3. Abbeville, South Carolina…
Action 2: states_for_referential_db
Action Input 2: {"countrycode": "U"}
Observation 2: {"error": "", "response": "[{'id': '1398', 'name': 'Howland Island', 'code': 'UM-84', 'country_code': 'US', 'type': 'islands …
…
Thought 4: Here are some countries:1. Afghanistan2. Aland Islands3. Albania4. Algeria...There are many more countries.
Action 4: cities_in_a_country_for_referential_db
Action Input 4: {"countrycode": "US"}
Observation 4: {"error": "", "response": "[{'id': '110965', 'name': 'Abbeville', 'state_code': 'GA', 'country_code': 'US', 'latitude' …

Figure 9: Impact of the amount of candidate toolsets.